Neural Network Applications in Seismology

Stephen Glenn Mosher

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
Doctor of Philosophy degree in Earth and Environmental Sciences

Ottawa-Carleton Geoscience Centre
Department of Earth and Environmental Sciences
Faculty of Science
University of Ottawa

Summary

Neural networks are extremely versatile tools, as evidenced by their widespread adoption into many fields in the sciences and beyond, including the geosciences. In seismology neural networks have been primarily used to automatically detect and discriminate seismic signals within time-series data, as well as provide location estimates for their sources. However, as neural network research has significantly progressed over the past three decades, so too have its applications in seismology. Such applications now include earthquake early warning systems based on smartphone data collected from large numbers of users, the prediction of peak ground acceleration from earthquake source parameters, the efficient computation of synthetic seismograms, providing probabilistic estimates of solutions to geophysical inverse problems, and many others. This thesis contains three components, each of which explore novel uses of neural networks in seismology. In the first component, a previously established earthquake detection and location method is supplemented with a neural network in order to automate the detection process. The detection procedure is then applied to a large volume of seismic data. In addition to automating the detection process, the neural network removes the need for several user-defined thresholds, subjective criteria otherwise necessary for the method. In the second component, a novel approach is developed for inverting seafloor compliance data recorded by ocean-bottom seismometers for the shallow shear-wave velocity structure of oceanic tectonic plates. The approach makes use of mixture density networks, a type of neural network designed to provide probabilistic estimates of solutions to inverse problems, something that standard neural networks are incapable of. In the final component of this thesis, the mixture density network approach to compliance inversion is applied to a group of ocean-bottom seismometers deployed along the continental shelf of the Cascadia Subduction Zone in order to investigate shelf sediment properties.

Deus Scientiarum Dominus Est

Acknowledgments

Undertaking research with the goal of earning a Ph.D was much harder than I ever anticipated it would be, but harder in ways that I did not expect. I think at the start I thought that most of the difficulties would have to do with learning and research. The thought of *becoming an expert* on a narrow range of topics seemed daunting. Developing deep understandings of the relevant physical and mathematical concepts, and performing computations and calculations endlessly, until satisfactory results emerged seemed quite tough. To be sure, those are difficult aspects of life as Ph.D candidate in a quantitative field, but, I eventually learned that the hardest parts had to with my internal processes. Imposter syndrome is real and quite common in academia. Constantly feeling like I'd be found out or that I didn't really know what I was doing was a real struggle. Moreover, constantly submitting pieces of your hard work for intense scrutiny by your peers and strangers can really take a toll. It can be hard to dissociate criticism, both helpful and necessary for great science, from feelings of personal rejection or judgement.

That's where I found the battle was, and that's why I'm quite confident that I would not have ever completed this thesis without the love, support, and encouragement from the following people. Firstly, my wife, Leah. You're the best, you've been a constant source of encouragement and have uplifted me throughout the process, I love you for that. Truly, I would not have come this far without you. Second, my parents and siblings. You've also been great encouragers and supports to me during my Ph.D candidacy, giving wise counsel when I needed it. Third, my supervisor, Pascal Audet. I'm not sure if you fully appreciate how much you shaped the course of my life. At the start of my B.Sc I never thought I'd pursue a M.Sc, and at the start of my M.Sc I never thought I'd pursue a Ph.D. We met at just the right time, you took me on as a student, and I discovered that I really enjoyed research in geophysics and seismology, and that I was capable of it. You've mentored and guided me professionally for over seven years and I think it's safe to say that it has really paid off.

Finally, I can't end this section without naming a few more people. I'd like to thank Zach Eilon at UCSB for working with me for three months last year. The time I spent working with you was one of tremendous professional growth and bore much fruit in my research life. I'd like to thank the members of our small geophysics group at Ottawa U for all the good times. I thoroughly enjoyed grabbing coffee with each of you and talking science and life in general over the past four years. Jeremy, our semi-regular phone calls during the pandemic months toward the end really helped to keep me sane and motivated. I'd like to thank Natalia Poiata for sending me portions of code when I was working on my first Ph.D project, that really helped me out. I'd like to thank all the researchers who took time to chat with me while I was working on mixture density networks, Andrew Valentine, Andrew Curtis, and Stephanie Earp. The conversations I had with each of you were crucial in helping me work out the finer points of neural network inversion. And lastly, I need to thank Tyler Shendruk, who generously shared his Ph.D thesis LaTeX template(s) with me. That has made compiling and writing this thesis orders of magnitude easier than it could have been otherwise. Thank you, all!

Publication List

1. *Automatic Detection and Location of Seismic Events From Time-Delay Projection Mapping and Neural Network Classification*, **S.G. Mosher** and P. Audet, Journal of Geophysical Research: Solid Earth, *125*, e2020JB019426 (2020).

2. *Probabilistic Inversion of Seafloor Compliance for Oceanic Crustal Shear Velocity Structure Using Mixture Density Neural Networks*, **S.G. Mosher**, Z. Eilon, H. Janiszewski, and P. Audet, Submitted to Geophysical Journal International, November, 2020 (GJI-S-20-1152).

3. *Shear-Wave Velocity Structure of Sediments on Cascadia's Continental Margin From Probabilistic Inversion of Seafloor Compliance Data*, **S.G. Mosher**, P. Audet, and J.M. Gosselin, Submitted to Geochemistry, Geophysics, Geosystems, February, 2021 (2021GC009720).

Statement of Originality

I hereby declare that this thesis results from my own investigations and represents original work performed while I was a doctoral candidate at the University of Ottawa under the supervison of Dr. Pascal Audet.

Statement of Contribution

The following researchers contributed to chapter 2: Natalia Poiata provided code for computing local cross-correlations between pairs of seismometers. I conceived of the project, obtained and processed all the data, performed the analysis, and wrote the additional code necessary for doing so. The manuscript was written with input from Pascal Audet.

The following researchers contributed to chapter 3: I conceived of the project while working with Zach Eilon during a research term abroad at UCSB. I wrote all the code necessary for creating testing and training models. Wayne Crawford's code was used to forward model synthetic compliance signals. I wrote the remaining code to build and train a MDN using an open-source Python package. Helen Janiszewski and Pascal Audet computed the compliance and coherence curves for station A02W. I conducted the inversion and analyzed the results. The manuscript was written with input from Zach Eilon, Helen Janiszewski, and Pascal Audet.

The following researchers contributed to chapter 4: This project is largely a logical extension of components developed in chapter 3. Jeremy Gosselin suggested the use of Bernstein polynomials as a continuous parameterization of V_S. I improved and expanded upon the code I had previously written for conducting MDN inversion by translating Wayne Crawford's code into my own Python and Fortran versions. I downloaded and processed all station data. I used components from the open-source OBStools package to compute compliance and coherence curves. I trained the MDNs, performed the inversions, and interpreted the results. The manuscript was written with input from Pascal Audet and Jeremy Gosselin.

Thesis Committee

Advisor: Pascal Audet

Committee Members: Glenn Milne, John Crowley, and Mareike Adams

External Member: James B. Gaherty

Contents

A computer program is said to ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

(Mitchell, 1997)

1

Introduction

Contents

1.1 Foreword

This is not a thesis on computer science, nor is it a thesis on machine learning. This is a *seismology* thesis. It investigates problems of earthquake detection and location as well as geophysical inversion, two areas I consider to be pillars of seismology, a statement with which I suspect others would agree. After all, not knowing whether an earthquake occurred or where it occurred is a lot like looking for your keys in the dark when it comes to seismic imaging methods, and trying to determine what it is *down there* that gave rise to various signals observed *up here* occupies a large portion of seismologist's efforts. That being said, although this is a *seismology* thesis, the methodological core of the work herein contained is the neural network, an entity shrouded in mystery and misconception, perhaps the blackest of all black boxes for many. Indeed, there was a time when I felt that way. However, I've since come to believe that it's not actually the mechanics of neural networks that perplex people, for the essential elements themselves are really quite simple, rather, it's their *versatility* that is perplexing! When one does finally take the time to learn what these seemingly nebulous objects are and how it is that they work, the question becomes, "How is it possible that these things can be used for so much?", it's almost unreasonable, and at best it's non-intuitive. For example, neural networks, in their various incarnations, find application in self-driving cars, signal separation, image processing, fraud detection, financial analysis, e-mail filtering, facial recognition, photonics, optical computing, and user-targeted advertising just to name a few. Of course, neural networks are also used for applications within the geosciences.

Therefore, one of the goals of this introductory chapter is to provide a primer on neural networks and supervised machine learning. In my opinion, the best way to work toward an understanding of neural networks is to proceed in three steps. First, we'll begin with a different machine learning algorithm altogether, linear regression. Although not truly a machine learning algorithm, linear regression can easily be cast into the language of supervised machine learning, and doing so helps to set the stage for important machine learning concepts. Second, following linear regression one should consider logistic regression, a simple algorithm for classifying data that helps to further bridge the conceptual gap with respect to neural networks. Third, after a discussion of logistic regression, we'll be primed to take a look at neural networks in the context of classification type problems. We'll then mention what changes are required to apply neural networks to regression type problems, and briefly mention mixture density networks (MDNs), which are described in greater detail in Chapter 3 of the thesis. Following the primer on neural networks and machine learning, I include a brief historical outline of these topics and mention some resources for further study. Or-

dinarily, a historical outline of a subject precedes any discussion of that subject. However, in this case I feel that discussing the history of machine learning and neural networks after encountering a few algorithms and their base concepts helps to better contextualize the historical discussion.

Finally, this introduction ends with a short description of the seismological problems addressed by this thesis using neural networks, and a summary of the contents of the main thesis chapters.

1.2 A Primer on Machine Learning and Neural Networks

1.2.1 Linear Regression

We begin our exploration of machine learning by considering the problem of simple linear regression, that is, linear regression in one variable x. The traditional statement of the problem is something like: given a set of observations in x, determine the best fit to those observations. What exactly constitutes the *best* fit is a subjective choice; however, whatever metric we choose will be an objective measure of fit. An example of a randomly generated univariate dataset is shown in Figure 1.2.1, along with several linear trends through that data, and the line of best fit according to the L_2-norm.

In the parlance of machine learning, the set of observed data are the training examples. The x's are the inputs, or features of that data, and the y's are the outputs, or targets. Crucially, the training examples are a set of data for which the outputs of given features are *known*. A single training example is therefore a pair (x, y) and we'll refer to the i^{th} training example as $(x^{(i)}, y^{(i)})$. In the context of machine learning, the linear regression problem statement could be phrased as follows: given a set of training examples, determine the best fitting function to those examples, such that future predictions can be made for features with unknown outputs. Because the goal of the problem is to learn how to generalize a relationship from features with known outputs, the problem is a supervised problem.

To solve the problem, one first proposes a hypothesis function. Moving forward I'll use a superscript letter to distinguish between different types of algorithms. In this case, a superscript ^{R} will be used to denote various linear regression specific functions. For the type

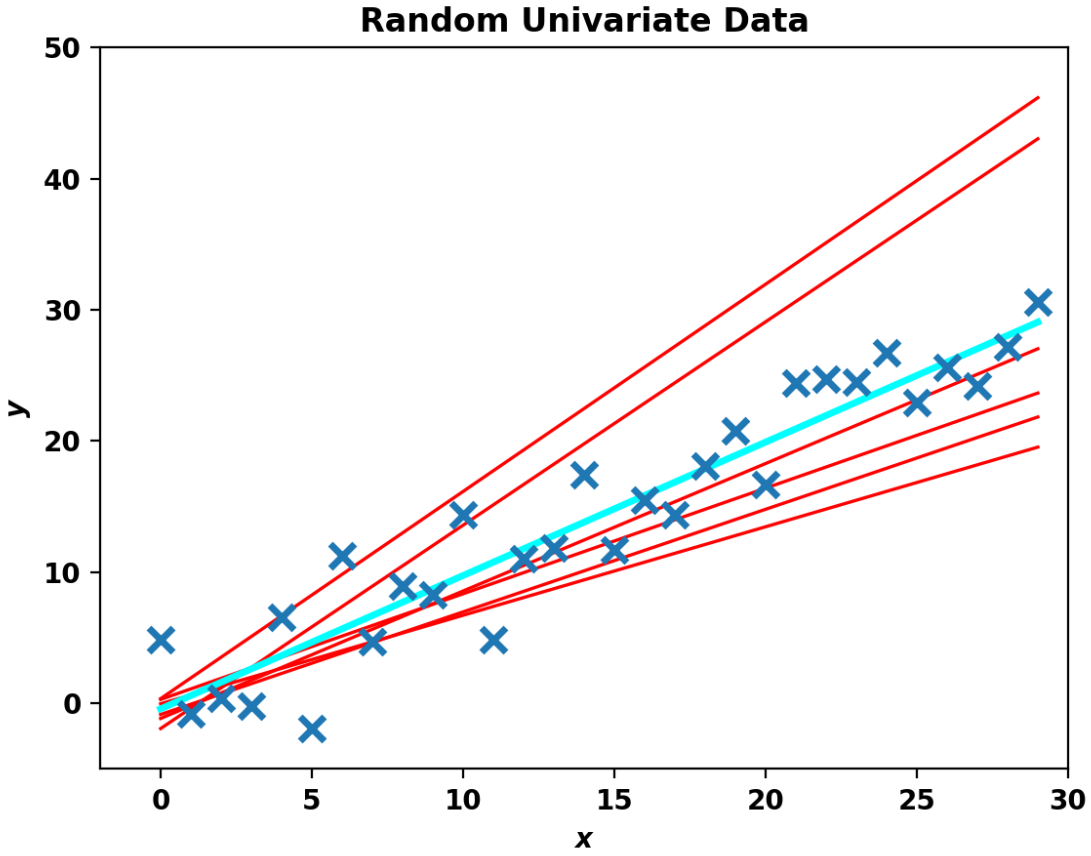

Figure 1.2.1: Simple linear regression. The blue crosses represent the observed data (or training data), the red lines correspond to various linear trends (or hypotheses), and the cyan line is the line of best fit.

of simple linear regression problem depicted in Figure 1.2.1, it is obvious that the hypothesis function should be a linear function of input feature x with parameters θ_0 and θ_1

$$h_\theta^R(x) = \theta_0 + \theta_1 x. \tag{1.1}$$

Thus, the various linear trends shown in red in Figure 1.2.1 represent a collection of different hypothesis functions. The goal then, is to determine the set of hypothesis parameters that *best fit* the training data. In this case we determine the line of best fit by minimizing the following cost function J, which is a function of the hypothesis parameters

$$J^R(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta^R(x^{(i)}) - y^{(i)} \right)^2 = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2. \tag{1.2}$$

The m in equation 1.2 denotes the total number of training examples, and recall that the superscript (i) is used to denote the i^{th} training example. Looking at equation 1.2, it is clear that the linear regression cost function $J^R(\theta_0, \theta_1)$ is the traditional sum of squared errors between the observed outputs $y^{(i)}$ and the outputs predicted by the hypothesis function $h^R_\theta(x^{(i)})$, summed over all m training examples (i.e. the L_2-norm). Note that the $1/2$ term is included for mathematical convenience.

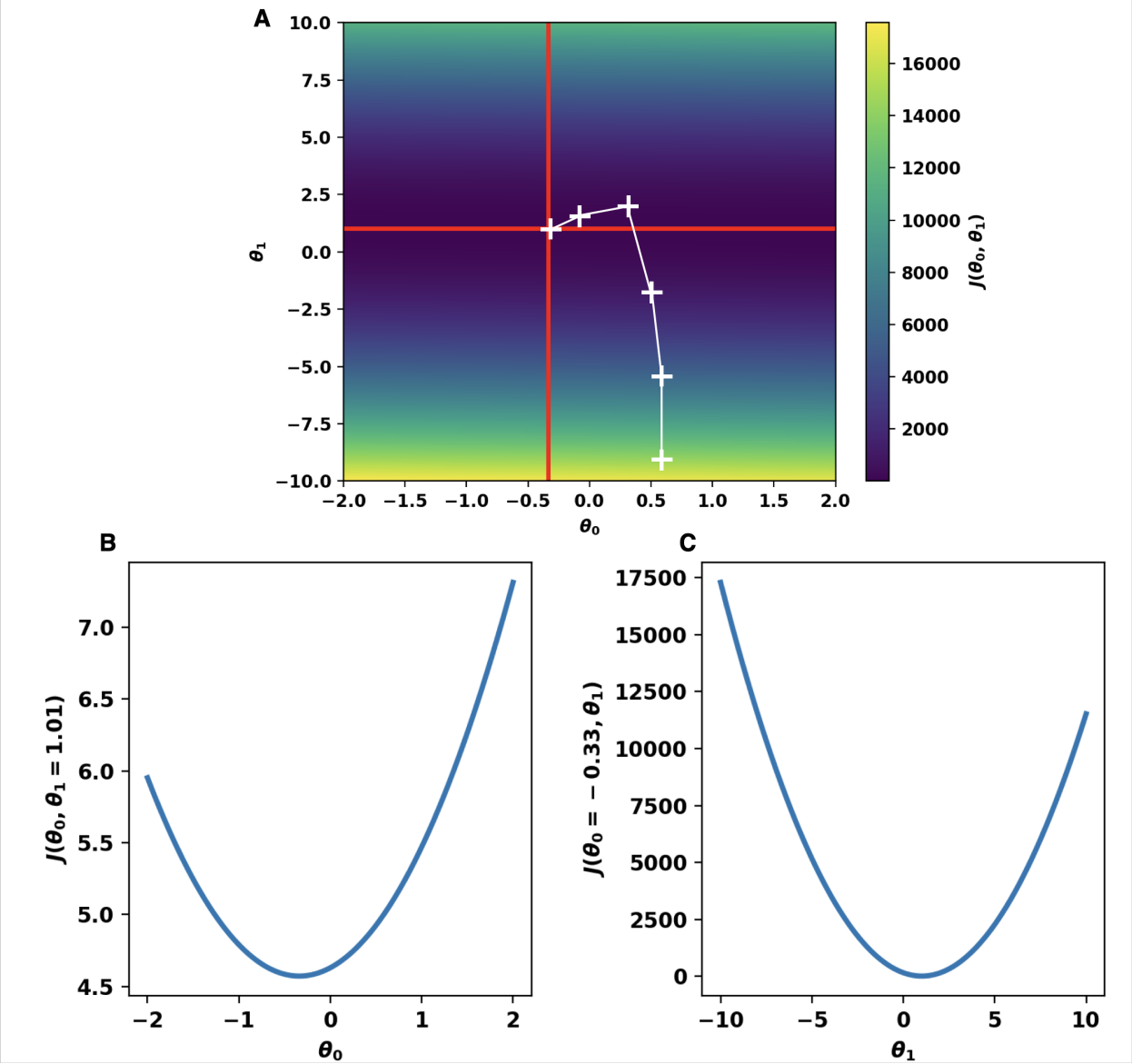

Figure 1.2.2: Panel A: The simple linear regression cost function $J^R(\theta_0, \theta_1)$ for the training data shown in Figure 1.2.1. The red lines denote the parameter combination that minimizes the cost function (as determined from the figure). The trajectory of the white crosses represents a gradient descent run. Panel B: A cross section of $J^R(\theta_0, \theta_1)$ through $\theta_1 = 1.01$. Panel C: A cross section of $J^R(\theta_0, \theta_1)$ through $\theta_0 = -0.33$.

Before we discuss how to determine the unique set of parameters that minimize the linear regression cost function, it's instructive to visualize $J^R(\theta_0, \theta_1)$ for different values of θ_0 and θ_1 with respect to the training data in Figure 1.2.1. While it's difficult to see, Figure 1.2.2 demonstrates that the error surface defined by the linear regression cost function takes on the shape of a parabaloid, albeit with walls of significantly different steepness. The cross sections in Figure 1.2.2 help to make this more evident. Clearly, changes in θ_1 (the slope) are a more significant control on the total cost than changes in θ_0 (the y-intercept) over the parameter ranges shown. Although it's beyond the scope of this introduction to discuss in detail, in general, the linear regression cost function defines a convex function, even in multivariate and higher dimensional cases. Consequently, the linear regression cost function always possesses a well-defined global minimum. It's worth noting that this is perhaps a miraculous property of equation 1.2.

So, how does one go about *training* a machine to *learn* the best fitting parameters for a linear regression fit to a set of training examples? The answer: using gradient descent. Although gradient descent is a widely applicable algorithm for estimating minima of functions, the miraculous property of equation 1.2 that I previously mentioned makes the implementation of gradient descent in linear regression especially elegant, precisely because it's always guaranteed to find a solution close to the global minimum. Gradient descent proceeds to find a minimum of a function by starting at an arbitrary point on the function's domain. At that point, partial derivatives are computed with respect to each of the function's parameters to determine the direction in which the gradient is locally maximized. The algorithm then updates to a new point by taking *a step* in the opposite direction of the maximum gradient (i.e. in the *downhill* direction). The procedure continues for a fixed number of iterations, or until the changes between the function values associated with the previous point and the new point are less than a specified tolerance. In the case of equation 1.2, the partial derivatives are

$$\frac{\partial}{\partial \theta_0} J^R(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta^R(x^{(i)}) - y^{(i)} \right) \tag{1.3}$$

and

$$\frac{\partial}{\partial \theta_1} J^R(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta^R(x^{(i)}) - y^{(i)} \right) x^{(i)}. \tag{1.4}$$

Using these partial derivatives, we can mathematically describe how gradient descent moves from a point $(\theta_{0,n}, \theta_{1,n})$ to a new point $(\theta_{0,n+1}, \theta_{1,n+1})$ as

$$\theta_{0,n+1} = \theta_{0,n} - \alpha \frac{\partial}{\partial \theta_0} J^R(\theta_0, \theta_1) \tag{1.5}$$

and

$$\theta_{1,n+1} = \theta_{1,n} - \alpha \frac{\partial}{\partial \theta_1} J^R(\theta_0, \theta_1), \tag{1.6}$$

where the *learning rate* α controls the step size and n indicates the iteration of the algorithm. An example of a trajectory taken by a gradient descent run is shown in Figure 1.2.2 for a moderate value of α.

While there is much more that could be said regarding linear regression and gradient descent, our brief discussion here is sufficient for our purposes. In this section we translated a familiar algorithm, linear regression, into the language of supervised machine learning, and in doing so several important machine learning concepts have been introduced. These concepts are summarized below:

- Supervised machine learning problems involve datasets of training examples in which input features x are paired with *known* output values y. More generally, these can be vector quantities $\mathbf{x}$ and $\mathbf{y}$.
- The supervised problem consists of selecting a hypothesis function $h_\theta(x)$, which is a function of the input features x, and contains parameters θ_0, θ_1, ...
- A training objective is determined by choosing a cost function $J(\theta_0, \theta_1, ...)$, a function of the hypothesis parameters, that provides a cost for each training example.
- The training objective is accomplished by selecting a suitable method to minimize $J(\theta_0, \theta_1, ...)$, such as gradient descent, and it is through such an iterative minimization method that the algorithm is said to *learn*.

1.2.2 Logistic Regression

In this section we shift our focus from linear regression to logistic regression. Whereas linear regression is concerned with learning an optimal fit to a set of training examples in order to make future predictions for unobserved data, logistic regression is concerned with learning how to classify data. The name logistic regression is perhaps a misnomer, in the sense that logistic regression does not predict values of continuous variables, but instead, predicts values of discrete variables. To that end, our discussion of logistic regression will focus on binary classification, that is, the prediction of output variables of the form $y = \{0, 1\}$. We'll build up an understanding of logistic regression for binary classification by working through a simple motivating example.

Consider the problem predicting whether a segment of seismic data (i.e. a velocity time-series) is simply a noise signal or contains a signal of interest from a seismic event. We'll refer to these as our negative ($y = 0$) and positive ($y = 1$) classes, respectively. Based on measuring the amplitude of velocity spectra at 1 Hz for a collection of representative examples in each class, you determine that it should be possible to use this feature to classify unobserved data as noise or signal. An example of such training data is shown in Figure 1.2.3. The problem then, is how to *train* an algorithm to perform the classification for new data *automatically* based on the set of training examples? Just as was the case in linear regression, we'll need to choose an appropriate hypothesis function, define a cost function, and determine a suitable training strategy in order to figure out the parameter values that minimize that cost function with respect to the training data.

Looking at the left-hand side of Figure 1.2.3, it might be tempting to use the linear regression hypothesis function to determine a decision boundary between the classes. For example, one might classify examples where $h_\theta^R(x) < 0.5$ as belonging to the negative class, and examples where $h_\theta^R(x) \geq 0.5$ as belonging to the positive class. For the case shown on the left-hand side of Figure 1.2.3 this seems reasonable enough. However, suppose that we later incorporate another example into our positive class, which has a much larger feature value (Figure 1.2.3-right). It is clear that the new example doesn't really change the training data. Yet, if we re-train our original linear regression classifier with the augmented dataset, we soon discover that all of our previous positive class examples become mis-classified. This is because linear regression is sensitive to outliers, and this simple example serves to show that the linear regression hypothesis function is not useful for classification. Moreover, the linear regression hypothesis function predicts continuous outputs, whereas the outputs in our training data are discrete. Thus, in order to train a logistic regression classifier, we want

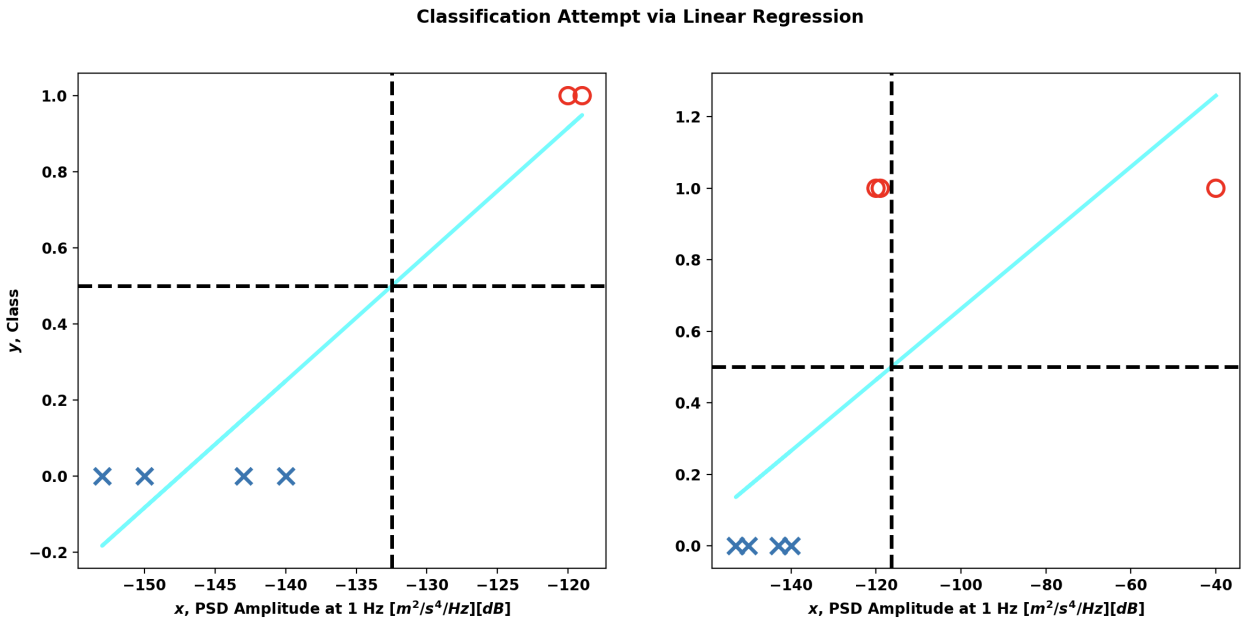

Figure 1.2.3: An example of attempting to train a classifier using linear regression. Left: Blue crosses denote training data belonging to the negative class. Red circles denote training data beloning to the positive class. The cyan line is the linear regression fit to the training data $h_\theta^R(x)$. The horizontal black dashed line shows where $h_\theta^R(x) = 0.5$ and the vertical line shows the class decision boundary as determined by $h_\theta^R(x) = 0.5$. Right: Same as the left, but with the addition of a new example to the positive class.

a hypothesis function that more closely resembles our target data. In particular, we want the hypothesis function we choose to satisfy $0 \leq h_\theta(x) \leq 1$. While a number of potential functions satisfy this criterion, the classic choice is based on the standard logistic equation (also known as the sigmoid function)

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{1.7}$$

A plot of equation 1.7 is shown in Figure 1.2.4. In order to use equation 1.7 as the basis of a logistic regression classifier, we make a subtle modification

$$h_\theta^C(x) = \sigma(\theta_0 + \theta_1 x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}, \tag{1.8}$$

where our hypothesis parameters in this simple 1D example are θ_0 and θ_1, and our single feature is x, the spectral amplitude of a seismic velocity time-series at 1 Hz. I'll use the superscript ^{C} to denote logistic regression specific functions throughout.

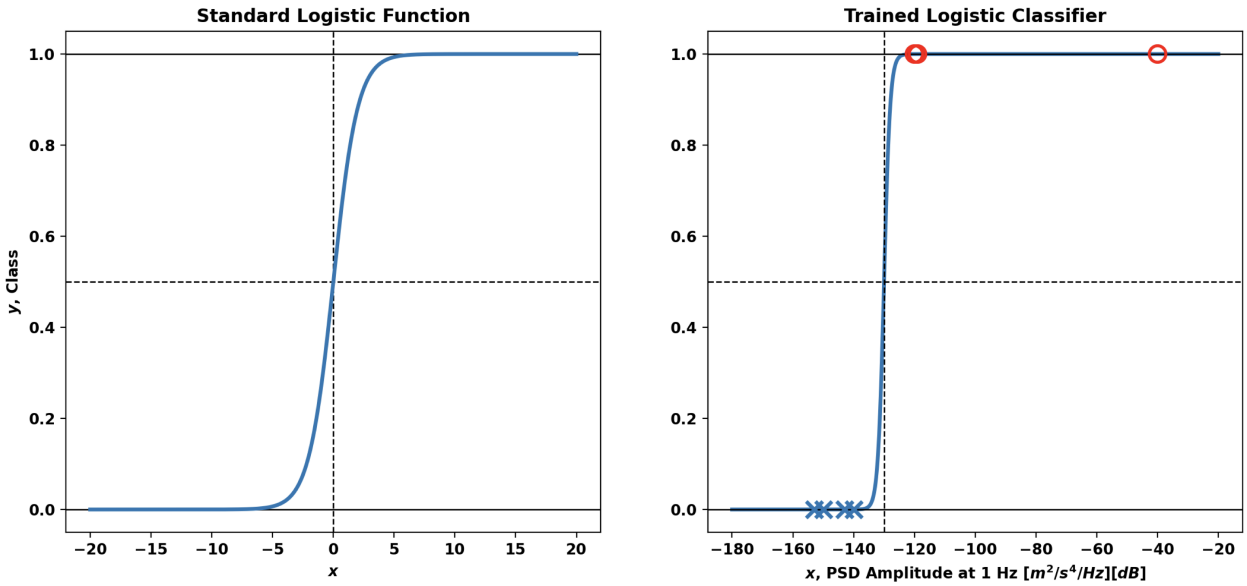

Figure 1.2.4: Left: The standard logistic function (equation 1.7) on the interval $[-20, 20]$. At $x = 0$, $\sigma(x) = 0.5$, for $x < 0$, $\sigma(x) < 0.5$, and for $x > 0$, $\sigma(x) > 0.5$. Right: A logistic regression fit to the augmented dataset in Figure 1.2.3. The decision threshold is -130 dB.

Before discussing the logistic regression cost function and training strategy, let's first consider a logistic regression fit to the data in Figure 1.2.3 (shown in Figure 1.2.4). In this simple 1D case, the best fitting parameters are $\theta_0 = 130$ and $\theta_1 = 1$. That these are the best fitting parameters comes from the fact that these numbers minimize the logistic regression cost function (to be discussed momentarily) with respect to the training data. The decision boundary is determined by the point where $h_\theta^C(x) = 0.5$ (equivalently, where $\theta_0 + \theta_1 x = 0$). This means that future data will be classified as noise ($y = 0$) if $h_\theta^C(x) < 0.5$ and as signal ($y = 1$) if $h_\theta^C(x) \geq 0.5$. Furthermore, this means that $h_\theta^C(x)$ can be interpreted as the probability that $y = 1$ for a given input x

$$h_\theta^C(x) = P(y = 1|x; \theta_0, \theta_1). \tag{1.9}$$

Evidently, the inclusion of the new data point in our classification problem does not significantly impact the decision boundary *learned* by the logistic regression algorithm, and intuitively, the algorithm learns the best set of parameters that shift or modify the logistic function such that the decision boundary aligns with the transition between the classes.

So, now that we've discussed the logistic regression hypothesis function, what is the logistic regression cost function $J^C(\theta_0, \theta_1)$, and how do we proceed to minimize it? A naïve approach

might be to substitute equation 1.8 into equation 1.2 and use gradient descent. However, doing so does not define a convex cost function and is therefore less than ideal. An alternative is to use the following piecewise cost function, based on the principle of maximum likelihood and shown in Figure 1.2.5,

$$J^C(\theta_0, \theta_1, y) = -\frac{1}{m}\sum_{i=1}^{m} \begin{cases} ln\big(h_\theta^C(x^{(i)})\big) & if \quad y^{(i)} = 1 \\ ln\big(1 - h_\theta^C(x^{(i)})\big) & if \quad y^{(i)} = 0. \end{cases} \tag{1.10}$$

Although it's beyond the scope of this thesis to account for the following fact, the cost function in equation 1.10 is a convex function, and therefore can be minimized efficiently using gradient descent. Essentially, when calculating the cost associated with a particular training example, if the training example belongs to a given class, but the current parameters of $h_\theta^C(x)$ cause it to predict that the example belongs to the opposite class, then the cost associated with that example will be prohibitively large. The converse is also true, meaning that predictions of $h_\theta^C(x)$ that are close to the actual class value of a particular training example will have small costs. Because the training examples always assume discrete values (either 0 or 1 in this case), equation 1.10 can be combined into a single logistic regression cost function

$$J^C(\theta_0, \theta_1, y) = -\frac{1}{m}\sum_{i=1}^{m} \Big(y^{(i)} \cdot ln\big(h_\theta^C(x^{(i)})\big) + (1 - y^{(i)}) \cdot ln\big(1 - h_\theta^C(x^{(i)})\big) \Big). \tag{1.11}$$

Thus concludes our tour of logistic regression, though our brief discussion does leave much to be desired. For example, logistic regression can be naturally extended to multivariate cases, and can be used to determine highly non-linear decision boundaries for training data which exhibit complex relationships. Yet, what we have discussed so far, with respect to both linear regression and logistic regression, enables us to describe neural networks in sufficient detail to understand the work presented in this thesis.

1.2.3 Neural Networks

The goal of this section is to give a short description of neural networks and how they work. We'll start by considering a single neuron, the fundamental building block of neural networks,

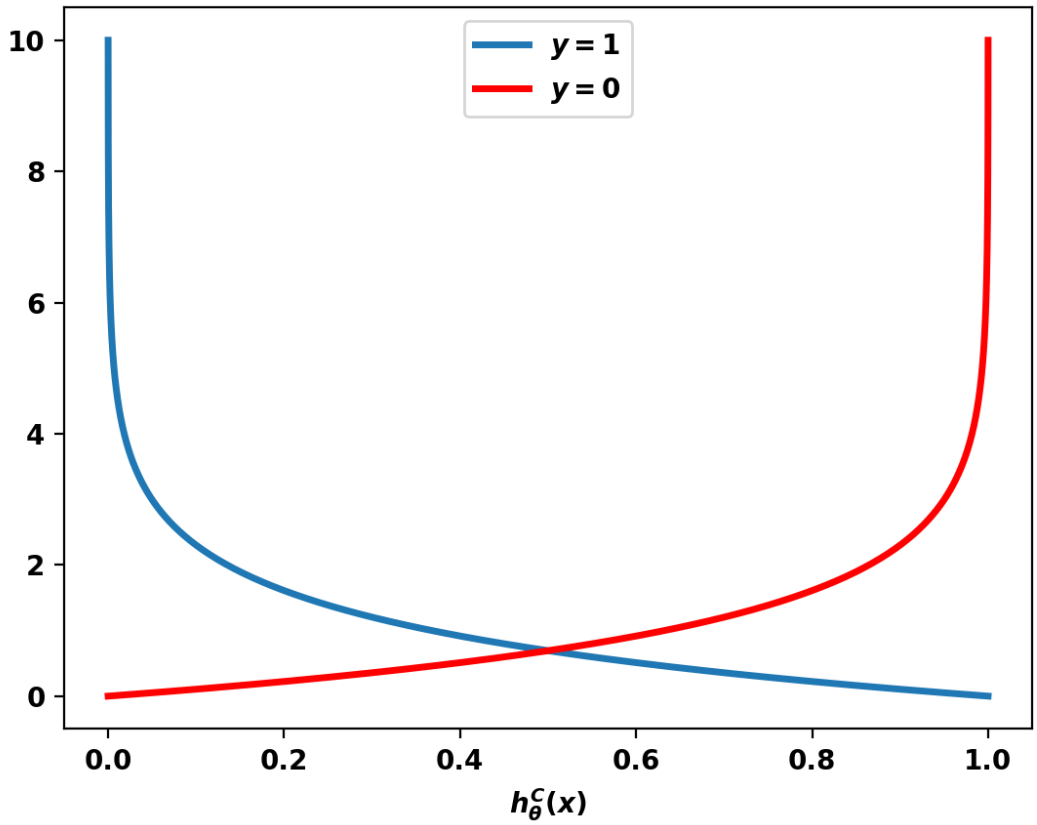

Figure 1.2.5: The piecewise cost function $J^C(\theta_0, \theta_1, y)$ given by equation 1.10.

and then proceed to describe multilayer perceptrons (MLPs), the simplest class of neural network. Following a general description of MLPs, we'll discuss the application of neural networks to binary classification problems. We'll then conclude by briefly discussing what changes need to be made to networks in order to apply them to regression type problems, and mention MDNs. I remind the reader that a brief historical sketch of machine learning and neural networks follows this section so I'll postpone any historical comments until then.

Neurons

A neuron is a simple computational unit that takes input from one or several sources and applies a function to that input to generate an *activation*. The activation of the neuron is simply the output of the unit (Figure 1.2.6). Thus, neurons themselves are functions. In theory, any function can be used to compute the activation of the neuron, but because

neurons were originally biologically motivated, typical activation functions include things like step functions, ramp functions, or even the sigmoid function. The motivating idea is that the sum of the inputs to the neuron need to reach a certain threshold in order for the neuron to *fire* (more on that analogy in the historical account). Mathematically, the activation a of a single neuron, using a sigmoid activation function for example, can be expressed as

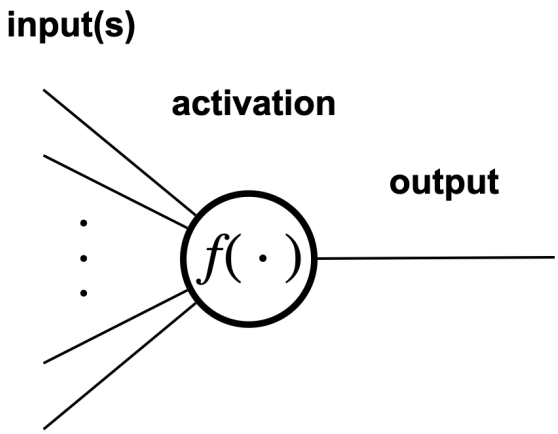

Figure 1.2.6: Schematic representation of a single neuron.

$$a = \sigma(\theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n), \tag{1.12}$$

where the x_i terms refer to individual inputs or features that each have their own associated weight θ_i. By convention the x_0 term equals unity and is called the *bias unit*. Thus, equation 1.12 can also be expressed as

$$a = \sigma(\theta_1 x_1 + ... + \theta_n x_n + b), \tag{1.13}$$

where $b = \theta_0$ is the *weight* associated with the bias unit, which itself is often simply referred to as the bias unit. The function of the bias unit is to shift the activation threshold of a given neuron. An insight that will be helpful later is to notice that a single neuron with a sigmoid activation function is equivalent to a logistic regression classifier, in particular, $h^C_\theta(x_1, x_2, .., x_n)$.

Multilayer Perceptrons

MLPs are simply networks of neurons. Specifically, they are networks of neurons organized into sequences of layers that are fully connected, and in which computation flows in one

direction (i.e. they are feed-forward networks). Naturally, the first layer is called the input layer, the final layer is called the output layer, and any layers in between are called hidden layers. I'll point out that although the first layer is distinguished, in that it provides the initial input to entire network, all subsequent layers act as input for the layer that immediately follows. To get a feel for how a neural network behaves computationally, let's explicitly write down an activation using the network in Figure 1.2.7 as our reference and assuming sigmoid activation functions. With respect to Figure 1.2.7, the activation of the first neuron in the hidden layer (the second layer) $a_1^{(2)}$ will be given by

$$a_1^{(2)} = \sigma(\theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + b_1^{(1)}), \tag{1.14}$$

where the bracketed superscripts refer to the different layers and $\theta_{1k}^{(1)}$ is the weight between the neuron at $a_1^{(2)}$ and the k^{th} neuron in the previous layer. While explicitly writing out network activations is useful pedagogically, it is otherwise quite cumbersome. We can save ourselves quite a bit of trouble by leveraging linear algebra to express things more efficiently as vectors and matrices. To do so, first note that we can express the input to a given neuron as a vector of activations $\mathbf{a}^{(i)} = [1, a_1^{(i)}, a_2^{(i)}, ...]^T$ from the previous layer i. Note that this applies to the first layer as well. Thus, when explicitly referencing the vector of activations between the first and second layer of the network, we may write $\mathbf{a}^{(1)} = [1, a_1^{(1)}, a_2^{(1)}, ...]^T$ or simply $\mathbf{x} = [1, x_1, x_2, ...]^T$.

Second, we can express the network weights between layers i and $i+1$ as a matrix $\mathbf{\Theta}^{(i)}$ with entries $\theta_{jk}^{(i)}$. Third, we can express the bias weights in layer i as a vector $\mathbf{b}^{(i)} = [b_1^{(i)}, b_2^{(i)}, ...]^T$. Therefore, the activation vector of any layer beyond the input layer can be expressed as

$$\mathbf{a}^{(i+1)} = \sigma(\mathbf{\Theta}^{(i)} \mathbf{a}^{(i)} + \mathbf{b}^{(i)}). \tag{1.15}$$

Using this notation, we can explicitly write the full activation vector for the final layer of the network shown in Figure 1.2.7 with a single equation, however unwieldy it might be

$$\mathbf{a}^{(3)} = \sigma(\mathbf{\Theta}^{(2)} \mathbf{a}^{(2)} + \mathbf{b}^{(2)}) = \sigma(\mathbf{\Theta}^{(2)} \sigma(\mathbf{\Theta}^{(1)} \mathbf{a}^{(1)} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}), \tag{1.16}$$

where

$$\mathbf{a}^{(1)} = \mathbf{x} = [x_1, x_2]^T; \quad \mathbf{b}^{(1)} = [b_1^{(1)}, b_2^{(1)}, b_3^{(1)}]^T; \quad \mathbf{b}^{(2)} = [b_1^{(2)}, b_2^{(2)}]^T,$$

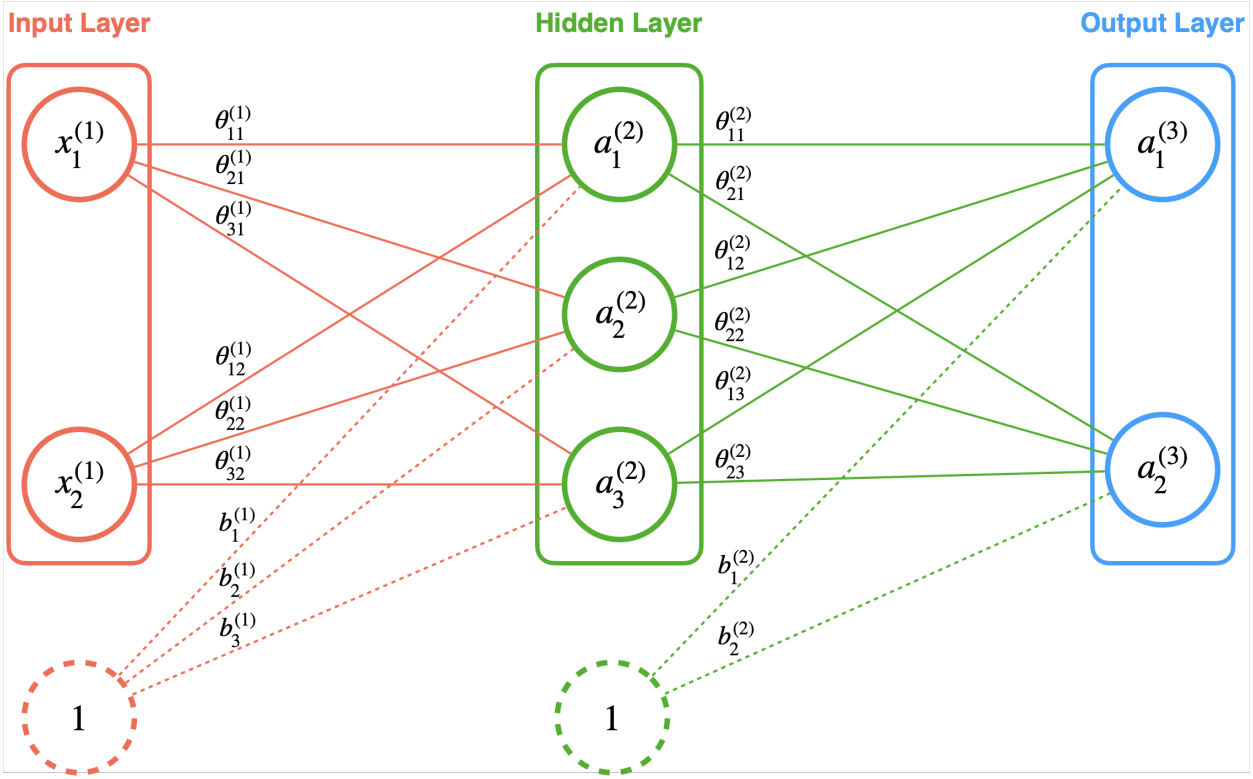

Figure 1.2.7: Schematic representation of a 3-layer MLP with explicitly labelled activations $a_j^{(i)}$, weights $\theta_{jk}^{(i)}$, and bias units $b_k^{(i)}$. The network has 17 total parameters. The bias units are denoted by dashed circles and lines. The superscript i denotes the layer and $\theta_{jk}^{(i)}$ is the weight between node k in layer i and node j layer $i+1$.

$$\boldsymbol{\Theta}^{(1)} = \begin{pmatrix} \theta_{11}^{(1)} & \theta_{12}^{(1)} \\ \theta_{21}^{(1)} & \theta_{22}^{(1)} \\ \theta_{31}^{(1)} & \theta_{32}^{(1)} \end{pmatrix} \quad \boldsymbol{\Theta}^{(2)} = \begin{pmatrix} \theta_{11}^{(2)} & \theta_{12}^{(2)} & \theta_{13}^{(2)} \\ \theta_{21}^{(2)} & \theta_{22}^{(2)} & \theta_{23}^{(2)} \end{pmatrix},$$

and I've coloured the substitution in equation 1.16 to make it more readable.

Our discussion of neural networks so far has been purely descriptive. We've seen that a single neuron is a computational unit that takes input from various sources. It then applies an activation function to a weighted sum of the inputs to produce a single output. We've also encountered MLPs, which are networks of neurons arranged into sequences of layers. A network takes a vector of initial values (or features) and then feeds that vector forward to a layer of neurons that compute a set of activations from those features. The resulting activations are passed on as the inputs to the next layer, and so on until an activation vector is computed in the final layer. The activation vector of the final layer is the output of the

network.

Binary Classification

At this point, if I were to provide you with a MLP and a set of weights and biases for that network, you'd be able to compute the deterministic output of any input by hand (or with the help of a computer). So how exactly can a MLP be used to perform binary classification, or any other task for that matter? Hearkening back to our previous discussions on linear regression and logistic regression, we might wish to ask: 1) What is the hypothesis function for a MLP with respect to binary classification? 2) What is the cost function? 3) How do I *train* such a network? We finally have all the ingredients we need to answer these questions.

Before we answer the first of these, let's remind ourselves that in a binary classification problem our training data belong to one of two classes, hence, our output variables are of the form $y = \{0, 1\}$. Therefore, whatever the structure of the MLP we use to perform the classification, we'll fix the output layer to have a single node. We'll also apply a sigmoid activation function to that node so that whatever its output, it will be a value between 0 and 1. Thus, the output of the MLP will represent the probability that an input belongs to the class $y = 1$. If the probability is greater than or equal to 50% we'll classify the input as belonging to class $y = 1$, and class $y = 0$ otherwise. So, what is the hypothesis function in this case? Well, the neural network is the hypothesis function. This shouldn't be too surprising for the following reasons.

First, single neurons are functions themselves, and, as we saw in our previous example, a MLP is itself a function, albeit a very complex non-linear function, as evidenced by equation 1.16. That being said, the example MLP shown in Figure 1.2.7 is relatively simple since it only contains 17 tuneable parameters. In practice, your garden variety neural network often contains several layers, each with perhaps hundreds of neurons, and therefore the number of network parameters can be as large as several thousand or even a million.

Second, you might recall that I remarked earlier that a single neuron with a sigmoid activation function is equivalent to a logistic regression classifier. Let's now take that idea one step further by considering a general MLP set up for binary classification (Figure 1.2.8). The MLP takes an input layer with n features and produces a single output which is a number between 0 and 1. Now consider just the hidden layer $l - 1$, containing p neurons,

and the activation in the output layer l

$$a_1^{(l)} = \sigma(\mathbf{\Theta}^{(l-1)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l-1)}) = \sigma(\theta_{11}^{(l-1)}a_1^{(l-1)} + \theta_{12}^{(l-1)}a_2^{(l-1)} + ... + \theta_{1p}^{(l-1)}a_p^{(l-1)} + b) = h_{\mathbf{\Theta}}^{C}(\mathbf{a}^{(l-1)}). \tag{1.17}$$

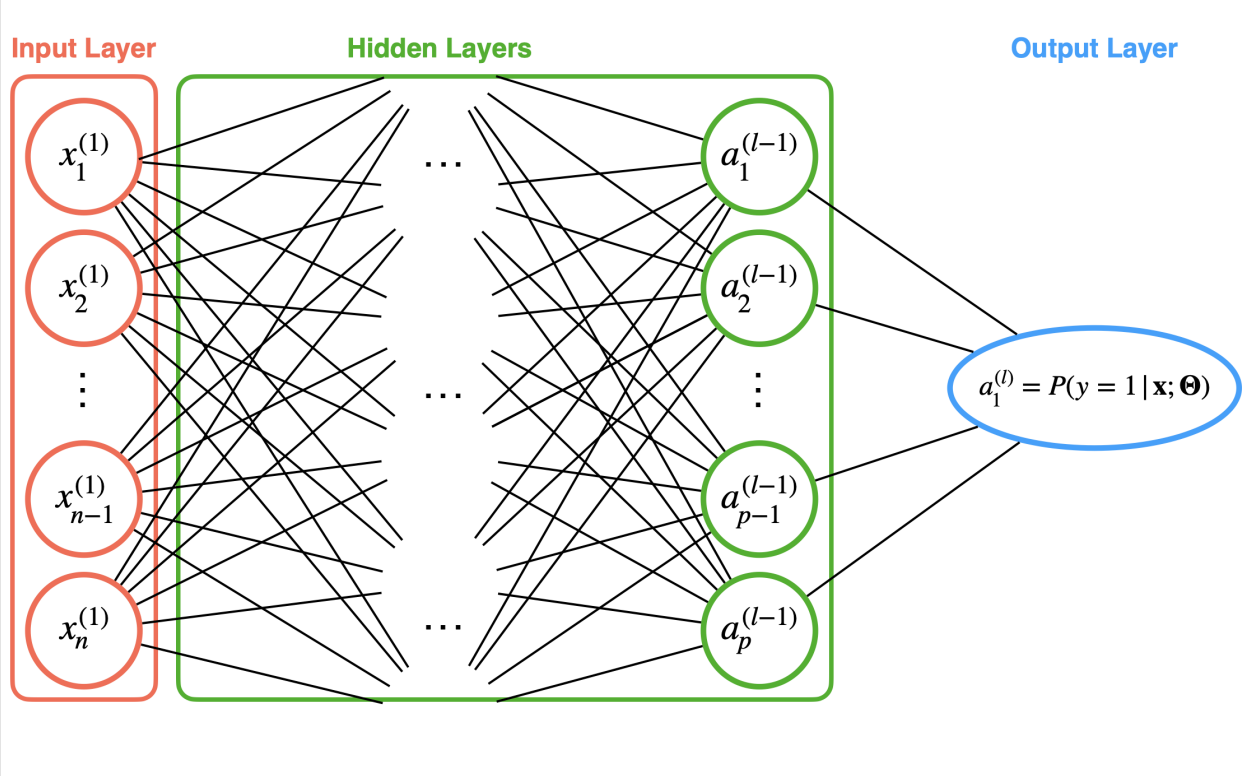

Figure 1.2.8: A general MLP for binary classification. The MLP has l layers, the input layer is a vector of n features, and layer $l-1$ has p neurons.

Once again, the expression in equation 1.17 is equivalent to a logistic regression classifier (equation 1.8) that takes the activation vector $\mathbf{a}^{(l-1)}$ as input features and applies the sigmoid function with parameters $\mathbf{\Theta}^{(l-1)}$ and $\mathbf{b}^{(l-1)}$. The subtle difference, however, is that components of $\mathbf{a}^{(l-1)}$ have been computed from the activations and network weights in the previous layers. Therefore, a MLP with sigmoid activation functions is something like a compounded linear regression classifier that contains a high degree of non-linearity. Beyond that there is also a sense in which the MLP has a chance to *learn* its own features. This is because the network shapes and molds the original input vector into a different set of activations by the time the data reach the penultimate layer. Not surprisingly, the logistic regression cost function (equation 1.11) is used as the cost function for this type of problem, where the network output $a_1^{(l)}$ is used for $h_{\mathbf{\Theta}}(\mathbf{x})$.

Now that we've discussed the MLP hypothesis and cost functions for binary classification, only one question remains. How does one *train* a MLP for binary classification? In other words, how does one determine the best set of network parameters $\mathbf{\Theta}$ and $\mathbf{b}$ that minimize the cost function with respect to the training data?

Back Propagation

The general strategy for training a neural network with a given cost function is to use gradient descent to minimize that cost function with respect to the network weights and biases, averaged over all the training examples. Although the use of gradient descent is an elegant minimization strategy for convex cost functions, it can still be used for non-convex cost functions. Moreover, several sophisticated variations of gradient descent exist that can be used as general minimization techniques for complicated functions. That being said, all variants of gradient descent work in essentially the same way: they rely on the partial derivatives of the function with respect to its parameters to seek out minima. Back propagation is the algorithm by which the partial derivatives of a neural network cost function can be computed with respect to the various network weights and biases. Computing these partial derivatives does require some work, but, a comprehensive intuition for back propagation can be obtained by working through a relatively simple example. And, once acquired, you'll be in the position of knowing how neural networks essentially function, from front to back!

Let's work toward an understanding of the back propagation algorithm by considering the simplistic MLP in Figure 1.2.9. Our example MLP has four layers and each layer contains a single neuron. Therefore, the network contains a total of 6 parameters $(\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, b^{(1)}, b^{(2)}, b^{(3)})$. This example allows us to drop the subscript indices, since the activations of each layer are scalars. In what follows, I'll be using a shorthand for the weighted sum of inputs to a neuron to make the notation nicer, namely, $z^{(l)} = \theta^{(l)}a^{(l)} + b^{(l)}$. Also note that when a network is first instantiated, its weights and biases are randomly initialized. With that out of the way, let's examine what happens when we *push* a single training example through this network, say, the first example. The final activation for that example will be

$$a^{(4)} = \sigma(\theta^{(3)}a^{(3)} + b^{(3)}) = \sigma(z^{(3)}) = h_\theta(x^{(1)}), \tag{1.18}$$

and the cost of that example will be

$$J_1 = y^{(1)} \cdot ln(a^{(4)}) - (1 - y^{(1)}) \cdot ln(1 - a^{(4)}), \tag{1.19}$$

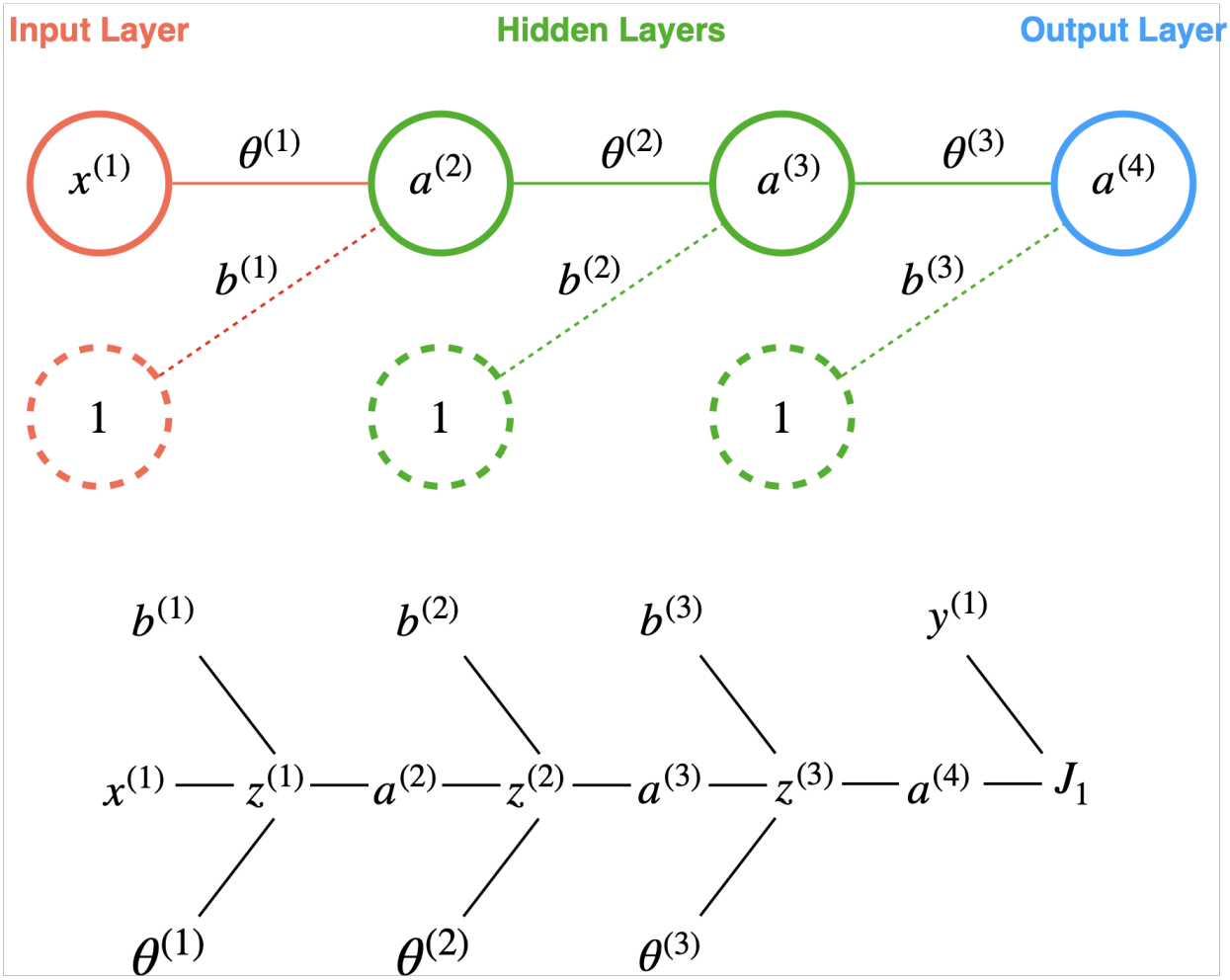

Figure 1.2.9: Top: A simple MLP with 4 layers, each with a single neuron. Bottom: A graphical representation of the chain of contributions to the cost J_1 of training example $(x^{(1)}, y^{(1)})$.

where $y^{(1)}$ is the target value of the training example. Recall that the cost of a training example is a measure of how different the network estimate is from the target value. Intuitively, what we'd like to do is tweak the network parameters to decrease the cost associated with the current example. We might imagine that this particular network is a machine with 6 different dials and knobs that we can adjust. We could randomly adjust those knobs and hope for the best, but what we're really after is a way to determine which adjustments will be the most effective changes to make in order to decrease the cost. In essence, that's exactly what the gradient of $J(\boldsymbol{\Theta})$ is, a vector of changes to the network parameters required to most effectively decrease the cost. The challenge is that the network parameters are *nested*. So, figuring out the partial derivatives of $J(\boldsymbol{\Theta})$ is really an exercise in multivariable calculus and the chain rule. I should also state that the back propagation algorithm is not trivial, in that it took early neural network researchers several years to discover.

Let's figure out a few of the partial derivatives of J_1 by using the graph shown in Figure 1.2.9 as a reference. First, notice that the cost J_1 depends on the final network activation $a^{(4)}$ and the target value $y^{(1)}$. Also note that the activation in the output layer $a^{(4)}$ depends on $z^{(3)}$, which in turn depends on $\theta^{(3)}$ and $b^{(3)}$. For now let's operate as if the chain of contributions to J_1 ends with these components. If we string these contributions to the cost together, then we can compute the partial derivative of J_1 with respect to θ_3 for instance

$$\frac{\partial J_1}{\partial \theta^{(3)}} = \frac{\partial z^{(3)}}{\partial \theta^{(3)}} \frac{\partial a^{(4)}}{\partial z^{(3)}} \frac{\partial J_1}{\partial a^{(4)}}. \tag{1.20}$$

We can expand on this further by computing the following

$$\frac{\partial J_1}{\partial a^{(4)}} = \frac{y^{(1)}}{a^{(4)}} - \frac{(1-y^{(1)})}{1-a^{(4)}} = \frac{y^{(1)} + a^{(4)}}{a^{(4)}(1-a^{(4)})} = \delta,$$

$$\frac{\partial a^{(4)}}{\partial z^{(3)}} = \frac{\partial}{\partial z^{(3)}} \left(\sigma(z^{(3)}) \right) = \sigma'(z^{(3)}),$$

and

$$\frac{\partial z^{(3)}}{\partial \theta^{(3)}} = a^{(3)},$$

where we've introduced the shorthand δ in the first of the previous three terms. Combining these terms, we now have an expression for the first of our partial derivatives in terms of quantities from the previous layer and the δ term based on the cost

$$\frac{\partial J_1}{\partial \theta^{(3)}} = a^{(3)} \sigma'(z^{(3)}) \delta. \tag{1.21}$$

In general, the partial derivative of the cost function with respect to $\theta^{(3)}$ will be averaged over all the training data such that

$$\frac{\partial J}{\partial \theta^{(3)}} = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial J_i}{\partial \theta^{(3)}}, \tag{1.22}$$

of course, this fact applies to the other partial derivatives as well.

So far we've figured out how to compute one component of the gradient of our cost function, let's now compute the partial derivative of J_1 with respect to $b^{(3)}$. Once again, using the graph in Figure 1.2.9 helps to keep straight the various components according to the chain rule, and we're only considering terms as far as $z^{(3)}$ for now. Notably, only one term changes from equation 1.20

$$\frac{\partial J_1}{\partial b^{(3)}} = \frac{\partial z^{(3)}}{\partial b^{(3)}} \frac{\partial a^{(4)}}{\partial z^{(3)}} \frac{\partial J_1}{\partial a^{(4)}}. \tag{1.23}$$

The new term is simply

$$\frac{\partial z^{(3)}}{\partial b^{(3)}} = 1,$$

so we have

$$\frac{\partial J_1}{\partial b^{(3)}} = 1 \cdot \sigma'(z^{(3)})\delta = \sigma'(z^{(3)})\delta. \tag{1.24}$$

So what have we accomplished? We've determined how to express the partial derivatives for the network parameters in the final layer, in terms of quantities from the previous layer, and a term that depends on the current cost, and really, that's all one needs to do. At this point we simply continue to iterate the procedure backward through the network, until all the partial derivatives of J have been computed. At that point we can use $-\nabla J(\boldsymbol{\Theta})$ with gradient descent to minimize J and tune the network parameters accordingly. To make this concrete, let's compute one more partial derivative, that of J_1 with respect to $\theta^{(2)}$. We start as before, but now we extend further along the chain to get

$$\frac{\partial J_1}{\partial \theta^{(2)}} = \frac{\partial z^{(2)}}{\partial \theta^{(2)}} \frac{\partial a^{(3)}}{\partial z^{(2)}} \left(\frac{\partial z^{(3)}}{\partial \theta^{(3)}} \frac{\partial a^{(4)}}{\partial z^{(3)}} \frac{\partial J_1}{\partial a^{(4)}} \right), \tag{1.25}$$

where last three terms are precisely equation 1.21 so

$$\frac{\partial J_1}{\partial \theta^{(2)}} = \frac{\partial z^{(2)}}{\partial \theta^{(2)}} \frac{\partial a^{(3)}}{\partial z^{(2)}} \left(\theta^{(3)} \sigma'(z^{(3)}) \delta^{(4)} \right). \tag{1.26}$$

From there it's straightforward to work out the other terms and show that

$$\frac{\partial J_1}{\partial \theta^{(2)}} = a^{(2)} \sigma'(z^{(2)}) \theta^{(3)} \sigma'(z^{(3)}) \delta^{(4)} = a^{(2)} \sigma'(z^{(2)}) \frac{\partial J_1}{\partial \theta^{(3)}}. \tag{1.27}$$

Finally, although we looked at back propagation using an incredibly simple MLP whose layers each had a single neuron, the types of expressions we derived for the final partial derivatives are not really that different from those in the general case. The partial derivative expressions in the general case simply contain more indices and further involve summations over the vector of activations in a given layer. It's beyond the scope of this introduction to derive anything else, so I'll simply show the general rule for calculating the partial derivatives of a MLP cost function for binary classification as a conclusion to this section

$$\boxed{\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{jk}^{(l)}} = a_k^{(l)} \sigma'(z^{(l)}) \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_j^{(l+1)}}}. \tag{1.28}$$

Beyond Binary Classification

In this introductory primer on machine learning we covered the basic concepts of supervised learning and discussed three different learning algorithms: linear regression, logistic regression, and simple MLPs for binary classification. My purpose here was not to provide an in-depth explanation of machine learning and neural networks, but rather, to provide you with enough background on these topics to sufficiently understand the content of this thesis. However, as this section concludes I'd like to add a few more comments to our discussion.

First, MLPs can be used for much more than just binary classification, or K-class classification for that matter. Simple MLPs can also be used in regression type problems such as fitting non-linear functions. Whatever the application of the MLP, the essential factors to consider are: 1) What type of variable is the network trying to predict? 2) What type of activation functions should I use to allow the network to predict that variable? And 3) What is an appropriate cost function to use for the problem? For example, a MLP will never be able to approximate a continuous function if a sigmoid activation is applied to its output.

Second, MLPs are really just the beginning of neural networks and are effectively an old technology (as we'll see shortly). Modern variations such as convolutional neural networks, recurrent neural networks, and adversarial networks are much more sophisticated and capable than their predecessors.

Third, we saw that neural networks are ultimately highly non-linear functions that are trained (or minimized) to approximate a target function. With this realization in mind, it's natural to ask just what kinds of functions neural networks are capable of approximating.

Are there limitations? In short, several so-called universal approximation theorems have been established for neural networks that have determined the conditions under which a given function can be approximated by a neural network. One such theorem states that "a standard multilayer feed-forward network with a locally bounded piecewise continuous activation function can approximate any continuous function to any degree of accuracy if and only if the network's activation function is not a polynomial" (Leshno et al., 1993). Such powerful theorems have helped to propel the adoption of neural networks into many fields.

Finally, in anticipation of Chapter 3, I'll briefly state here that whereas MLPs approximate functions between two domains of arbitrary dimension, MDNs are a type of neural network that approximates a probability distribution between two domains. As such, they have a unique cost function and network architecture.

1.2.4 A Brief History

Many might find it surprising to learn that machine learning, or at least the seeds from which modern machine learning germinated were planted several decades ago. In 1949, psychologist Donald Hebb published a highly influential model of brain cell interaction in which he postulated that, "when one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell" (Hebb, 1949). Hebb essentially thought that neurons repeatedly acting together would develop a strong association over time, and conversely, that associations between neurons acting independently would decay over time. Ostensibly, this is where the colloquialism "neurons that fire together, wire together" comes from. Concurrently, in 1948 IBM researcher Arthur Samuel began working on a computer program to play checkers on IBM 701 machines (Weiss, 1992). Samuel worked on his checkers program for over 11 years, and while his program received a great deal of publicity through IBM, none of his work was published until 1959 (Samuel, 1959; Weiss, 1992). Samuel's seminal work in 1959 was the first demonstration that a computer could be programmed to perform a task better than the programmer (Samuel, 1959). Incidentally, Samuel's checkers program became famous when it won a game against America's number four ranked player in 1962 (Knuth, 1990).

Drawing on the model proposed by Hebb, and the computational insights gained by Samuel, research psychologist Frank Rosenblatt designed the Mark I Perceptron in 1958, the first computational tool designed to perform pattern recognition and classification tasks

(Rosenblatt, 1958; Seising, 2018; Van Der Malsburg, 1986), and the predecessor of the modern neural network. At first the possibilities presented by Rosenblatt for tackling pattern recognition problems using perceptrons seemed quite promising and subsequent research boomed. However, Rosenblatt's initial ideas generated some highly optimistic claims, and, eventually drew heavy criticism (Wasserman & Scwhartz, 1988). In particular, Marvin Minsky and Seymour Papert wrote a book that outlined severe problems and limitations concerning the capabilities of Rosenblatt's perceptrons. Chief among them was the fact that early perceptrons were capable of some classification tasks, yet not others, such as predicting whether simple images had connected domains (Minksy & Papert, 1969). Furthermore, although researchers at that time had been investigating MLPs, a method for training MLPs was not yet known (Minksy & Papert, 1969). The net effect of the criticisms levied by Minsky and Papert was that research progress in the area was significantly halted (Wasserman & Scwhartz, 1988), leading to what has been termed the AI Winter. However, some research on perceptrons and MLPs continued, and in the 1980s the field underwent a resurgence when the back propagation algorithm for training MLPs became widely known (Rumelhart et al., 1986).

Following the resurgence of neural network research in the 1980s, the first uses of neural networks in seismology occurred in the 1990s. Dowla et al. (1990) and Dysart and Pulli (1990) both trained neural networks to discriminate between natural earthquakes and underground nuclear blasts using the spectral amplitude ratio between the Lg/Pg crustal phases. Dai and MacBeth (1995) used a neural network to automatically detect and pick seismic arrivals generated from local events. Upon comparing the results of the trained network against human analysts, they found that the network accurately detected 93.9% of the P-arrivals and 90.3% of the S-arrivals. Ultimately, the rapid uptake of machine learning techniques into the geosciences post-2000 had to wait for advances in computational capabilities and the availability of easy-to-use machine learning toolboxes on various platforms (Bergen et al., 2019).

1.3 Learning Resources

In writing the preceding sections I drew upon notes that I created when taking Andrew Ng's online *Machine Learning* course (currently available at https://www.coursera.org/learn/machine-learning). I also took inspiration from Grant Sanderson's online video series on neural networks, which gives an excellent description of back propagation (currently available at

https://www.youtube.com/c/3blue1brown). In general, there is a large variety of free content online enabling one to learn about machine learning and neural networks at a high level. Ian Goodfellow, Yoshua Bengio, and Aaron Courville's freely available *Deep Learning* book is another great example.

1.4 Seismological Scope

Having discussed the methodological core of this thesis, I now briefly discuss the seismological problems that are addressed. The first of these is the detection and location of seismic sources (or events), and the second is geophysical inversion. Note that the specifics of these problems are fleshed out in greater detail in the introductory sections of the subsequent chapters.

1.4.1 Detection and Location

Broadly speaking, seismology is the study of earthquakes (or seismic sources, more generally) and the elastic waves that they generate. Importantly, such a simple paradigm allows for two basic arenas of study. First, there is the study of seismic sources, which asks questions such as: "Under what stress conditions do seismic sources occur?", "Where do seismic sources occur?", and "What distinguishes different types of seismic sources?". Second, as the elastic waves generated by seismic sources propagate throughout the Earth, they interact with, and are influenced by the structures they encounter. In particular, the velocity of seismic waves depend on material properties such as the density, shear-rigidity, compressibility, and temperature of the media through which they travel. Therefore, because Earth structure influences elastic wave propagation, various physical properties within the Earth can be inferred from observations of seismic waves at the surface. Crucially however, in order to make accurate inferences regarding Earth structure from observations of seismic waves, origin times and locations of seismic sources need to be known. It is for precisely this reason that the detection and location of seismic events are among the most fundamental problems in seismology. For without the origin time and location of seismic sources, one cannot determine the travel-time taken by elastic waves from the source to the receiver. Without the travel-time, one cannot measure the velocity of seismic waves, and therefore, in the absence of these pieces of information, no inferences regarding the structure between the source and receiver can be made. Thus, catalogues of detected earthquakes, their locations, and origin times, are indispensable tools in most seismic studies.

Although the act of seismic detection is logically prior to that of location, techniques for locating seismic sources were developed first, assuming detection. The first earthquake location algorithm was pioneered by L. Geiger in 1910. Geiger proposed that epicentral latitudes, longitudes, and origin times of sources could be found by minimizing the least-squares error between observed and theoretical P-wave travel times among groups of stations (Geiger, 1912). Indeed, many modern location techniques are simply modifications of Geiger's original algorithm and focus on minimizing travel time residuals between P and S seismic phases of detected events. Naturally, the precise identification (or "picking") of phase arrivals due to seismic events is crucial to the success of these methods, and prior to the 1970s, this task was performed manually. However, from the 1970s onward, when computers were finally capable, and as seismology was becoming digitized, work on automated Geiger-type detection and location algorithms began to develop (Stewart, 1977). In addition, several alternative array-based automatic detection and location methods, based on exploiting the multistation coherence of seismic waveforms, have since been developed (see Li et al., 2020, for a review). Such methods are often applicable to a greater variety of seismic source types beyond typical earthquakes.

Ultimately, research into ever more sophisticated detection and location methods has been a large focus in seismology over the past 50 years. The need for such research is made clear when one recognizes that the rate of seismic data available to researchers is increasing exponentially (Kong et al., 2018), vastly outpacing the capabilities of human analysts. Therefore, the use of neural networks for automatic detection and location tasks in seismology was almost an inevitability, simply because neural networks were developed to perform these kinds of tasks.

1.4.2 Inversion

While it is the case that catalogues of detected earthquakes, and their locations and origin times are required for a great many seismic studies, it could be argued that what seismologists are more interested in, is geophysical inference. That is, inferring the causes of observations. This is precisely the domain of inverse theory, and geophysical inversion may well lie at the heart of seismology. Whereas other scientific fields are capable of performing direct measurements of various phenomena (physics, chemistry, biology, etc.), seismologists are forced to live with indirect measurements. We cannot see what is inside the Earth, nor touch it. Instead, seismologists rely on the physics of elastic wave propagation to make educated guesses as to the probable range of structures that could generate a given set of

observations. I should state that classical inverse theory is a broad subject in its own right, and is not exclusive to seismology.

An inverse problem has three simple ingredients. First, there is the forward operator, which is essentially any theory that allows one to make predictions. Second, there are the models. The models are theoretical constructs to which the forward operator can be applied to make predictions. Third, there are the data. The data are the real-world pieces of information that one is able to observe. Generally speaking, the goal of any inverse problem is to move *backward* from the set of observed data, in order to determine the model responsible for those data, assuming a particular forward operator. Ideally, one would be able to determine a unique model for a given problem, but in reality, inverse problems are notoriously difficult in that they most often do not possess unique solutions. To illustrate this point, consider the following toy problem. Suppose I wish to determine the exact route you take to work each given morning. I know the start and end points of your route (your home and work, respectively), and I know how long the trip takes (the data). In general it will be impossible for me to determine the exact route you take in the absence of any additional information. The best I can do is make reasonable guesses by perhaps ruling out certain routes that would likely be prohibited by the data, or by compiling routes that could fit the data. Such a situation is fairly typical in geophysical inverse problems, which are most often under-determined or mixed-determined problems. It is also worth noting that the simple example I provided is essentially no different from the problem of seismic tomography.

Inverse theory is a rich and important topic, and although most geophysical inverse problems are non-unique, several sophisticated solution strategies exist for determining the most probable range of models for a given problem (MCMC for example Sambridge & Mosegaard, 2002). While such techniques have been used to great effect in seismology, the use of neural networks to approach inverse problems represents a relatively new and unique prospect.

1.5 Summary of Thesis Chapters

This thesis is comprised of three components. In the first of these, I supplement an existing earthquake detection and location algorithm developed by Poiata et al. (2016) with a neural network for binary classification. The method of Poiata et al. (2016) operates by exploiting the seismic waveform coherence observed between multiple stations. Essentially, coherent waveform information recorded by multiple stations is computed over a given time

and is turned into a 3D image. The resulting image transforms the earthquake detection and location problems into image processing problems. I extend this work by training a neural network to perform the earthquake detection task automatically, independently of user defined thresholds. The network uses spectral features from the station waveforms and features derived from the 3D image to classify whether the stations observe seismic signals or noise.

In the second component of this thesis I investigate a neural network approach to seafloor compliance inversion. The phenomenon of seafloor compliance is inherently interesting. Long-period ocean waves with long wavelengths and small displacements induce a loading and deformation of the seafloor over which they propagate. Because the elastic properties of the seafloor subsurface govern the response to such loading, the compliance signal is sensitive to subsurface structure. In particular, the compliance signal is most sensitive to the subsurface shear modulus, and can therefore be used to estimate the shear-wave velocity structure of the subsurface. The situation thus sets up a classic geophysical inverse problem. While previous methods have been developed to invert compliance signals for shallow shear-wave velocity structure of oceanic plates, in this component of my thesis I propose an approach based on neural networks. Specifically, I implement an inversion scheme based on mixture density networks, which I argue has certain advantages over previous methods.

The final component of this thesis is an extension of the second. Whereas the second component is a proof-of-concept, the third component is an application of the MDN compliance inversion technique. In this component, I apply the MDN inversion technique to a group of ocean-bottom seismometers (OBSs) deployed along the continental margin of the Cascadia subduction zone. Several studies have investigated the shallow sediment properties of Cascadia using compliance inversion at OBSs deployed in the Cascadia basin. By contrast, this component investigates the shallow sediment properties of the shelf using a passive imaging method.

Finally, the components of this thesis have either been published or submitted for publication in the following journals. Chapter 2 was published in the Journal of Geophysical Research: Solid Earth (October, 2020). Chapter 3 has been submitted to Geophysical Journal International (November, 2020). Chapter 4 has been submitted to Geochemistry, Geophysics, Geosystems (February, 2021).

Where were you when I laid the earth's foundations? Tell me, if you understand. Who marked off its dimensions? Surely you know! Who stretched a measuring line across it? On what were its footings set, or who laid its cornerstone–

Job 38:4-6

2

Article: Automatic Detection and Location of Seismic Events From Time-Delay Projection Mapping and Neural Network Classification

Contents

2.1 Abstract

The past several decades have seen an exponential increase in the volume of available seismic data, and with it has come the need to develop fast, automatic earthquake detection, and location algorithms. Some of the most recent and promising tools come from the field of machine learning. In this study, we combine a recent seismic detection and location method with neural network classification and analyze 4 months of continuous data recorded by a network of 76 stations in northern California. While these approaches have been used separately, our implementation is unique in that it is not constrained by source templates and avoids user-defined detection thresholds. In particular, we partition our data set into 234,240, 3-min long time windows with 75% overlap. For each time window, we create a 3D image that captures information about the coherence of the seismic wavefield. We then devise four features as input and train a neural network classifier to predict which time windows in the data set are likely to contain regional seismic events. These features include the second and fourth Hu image moments computed from 2D cross sections of our 3D images and statistical p values that quantify the probability of observing network-wide power-spectral density values at 0.2 and 0.5 s. Our neural network model predicts that 2,522 time windows contain seismic events, from which we locate 1,192 unique events.

2.2 Introduction

The detection and location of seismic sources are fundamental problems in seismology, as any seismological investigation requires knowledge related to either seismic sources or the elastodynamic waves that they generate. While the act of detection is logically prior to that of location, algorithms for locating earthquakes were developed first, assuming detection. The first earthquake location algorithm was pioneered by L. Geiger in 1910, wherein he suggested that epicentral latitudes, longitudes, and origin times of sources could be found by minimizing the least-squares error between observed and theoretical P-wave travel times among groups of stations (Geiger, 1912). In fact, many modern location techniques are simply modifications of Geiger's original algorithm and focus on minimizing travel time residuals between P and S seismic phases of detected events. Naturally, the precise identification (or "picking") of phase arrivals due to seismic events is crucial to the success of these methods, and prior to the 1970s, this task was performed manually. However, motivated by significant changes in the observational capabilities of seismic networks, and made possible by the increasing availability of computational resources, automated Geiger-type detection and location algorithms began to develop during the following decades (Stewart, 1977).

The strategy of automated Geiger-type (i.e., phase detection and picking) schemes is to replace raw signals with characteristic functions (CFs), which more easily allows a machine to mimic the mental transformations that human analysts unconsciously apply to data when analyzing them (Stewart, 1977). Since earthquakes produce changes in the character of a seismogram, such as energy, amplitude, polarization, or frequency content, relative to some background (Allen, 1982; Lomax et al., 2012), CFs are carefully constructed to enhance such changes. One of the most common operators used to construct early CFs was the squaring operator $CF(i) = Y^2(i)$, which enhances the amplitude change associated with transient signals (Allen, 1982). In another study, Allen (1978) used an envelope CF built from the squared amplitude of an input signal plus the weighted square of its first derivative in order to enhance both frequency and amplitude changes. Often within this framework, automated Geiger-type methods accomplish detections by comparing running short-term averages (STAs) to long-term averages (LTAs) of CFs designed to enhance P and S arrivals. If at any point the STA/LTA ratio exceeds a predefined threshold, then a phase detection is declared, and arrival times are estimated based on the point at which the threshold was crossed (e.g. Allen, 1978; Lomax et al., 2012; Withers et al., 1998). For a review of such methods in the context of microseismic sources, see Cesca and Grigoli (2015).

In contrast to Geiger-type location and detection methods that rely on single-station

phase picking, array-based detection and location techniques instead exploit the multistation coherence of seismic waveforms or their CFs (for a review see Li et al., 2020). Such methods are often applicable to a greater variety of seismic source types beyond typical earthquakes. For example, using the cross-spectral method, Métaxian et al. (2002) compute time delays between station pairs and invert them for slowness vectors. They then use a probabilistic approach to identify significant back azimuths and ray parameters between groups of sensors in order to locate sources of volcanic tremor. Kao and Shan (2004) use the source-scanning algorithm to locate sources of tectonic (or nonvolcanic) tremor in the Cascadia subduction zone. In these studies, alternative detection and location methods are pursued because Geiger-type methods are ineffective when applied to tremor-type events as they often lack clear P- and S-wave arrivals (Kao & Shan, 2004; Nadeau & Dolenc, 2005; Obara, 2002). For similar reasons, Grigoli et al. (2013) modify the typical STA/LTA Geiger-type approach in order to account for the difficulties encountered when trying to use S waves to automatically locate regional, small-magnitude, crustal events. In these cases (distances on the order of 10–100 km), S waves are often obscured by P-wave codas, and thus, their travel times are not easily identified. Specifically, Grigoli et al. (2013) discretize a seismic search volume and locate regional events by identifying the element in the grid with the highest coherence between uniquely designed P- and S-wave CFs. Using the method of Frank and Shapiro (2014), Frank et al. (2014) compute a beamformed network response function over potential source locations within a 3D grid in order to locate low-frequency earthquake (LFE) source templates and use the newly identified LFE templates to find LFE multiplets via a template matching strategy. Although proven to be efficient, most of the above approaches still require some input from the analyst, such as specifying detection threshold parameters.

The adoption of machine learning algorithms in seismology has also significantly impacted seismic detection, location, and classification algorithms (see Dai & MacBeth, 1995; Dowla et al., 1990; Dysart & Pulli, 1990, for some of the earliest examples). Recent examples include Reynen and Audet (2017), who use a logistic regression classifier, in conjunction with frequency- and polarization-based features to distinguish between blasts and earthquakes in southern California, as well as detect 25 times more seismic events in Oklahoma than in previously available earthquake catalogues. Seismogram classification has also been performed using a support vector machine by Tang et al. (2020) to distinguish between earthquakes and blasts in the Tianshan orogenic belt in China. Alternatively, Perol et al. (2018) use an artificial neural network to detect and locate induced seismic events in Oklahoma and show that their classifier detects 17 times more earthquakes than are available in local catalogues. Mousavi et al. (2019) develop and train a deep neural network to learn the time-

frequency characteristics of the dominant phases in an earthquake from three-component data recorded by individual stations. They then use the network to locate over 800 induced microearthquakes with magnitudes as small as 1.3 ML. Meier et al. (2019) train a series of neural networks to discriminate between seismic events and noise in real-time, in order to improve the reliability and speed of earthquake early warning systems. An example of a recent study that uses a neural network to perform automatic arrival-time picking is that of W. Zhu and Beroza (2019). For recent reviews on machine learning in the geosciences and artificial neural networks in seismology, consult Bergen et al. (2019); Rojas et al. (2019).

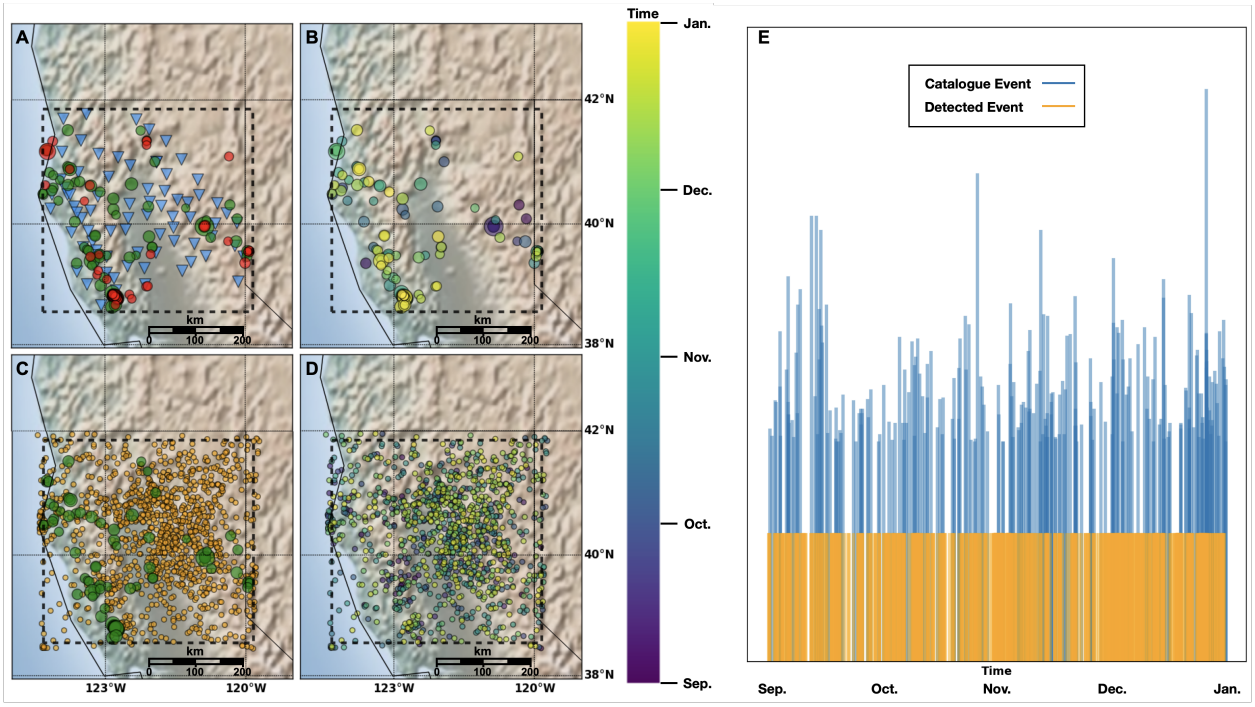

Figure 2.2.1: Panel A: Map showing the 76 stations (blue inverted triangles) and 233 catalogue earthquakes (circles) used in this study. Note that all catalogue earthquake symbols are scaled by magnitude. Red denotes catalogue earthquakes not used for training. Green denotes catalogue earthquakes used for training. Panel B: Spatiotemporal distribution of all catalogue earthquakes. Panel C: Spatial distribution of catalogue earthquakes used for training (green) and final detections in this study (orange). Panel D: Spatiotemporal distribution of final detections in this study. The black dashed box in Panels A–D shows the spatial catalogue bounds. The actual search volume used in this study includes a 20-km buffer beyond the catalogue search bounds. Panel E: Temporal distribution of all catalogue and detected earthquakes. Catalogue earthquakes are scaled by magnitude.

Of particular relevance to our study is recent work by Poiata et al. (2016), who proposed a multiscale, flexible framework for simultaneous event detection and location which makes use of array-based techniques exploiting waveform coherence. While their study is not the first to treat detection and location simultaneously, it is unique in that it presents a method-

ology for detecting and locating multiple seismic source types occurring at different dynamic and spatial scales. In their work, Poiata et al. (2016) demonstrate the ability of the method to detect and locate the energy release associated with overlapping seismic radiation from earthquakes and low-frequency tectonic tremors on the Nankai subduction zone in south-western Japan, and in a subsequent study, they apply their method to analyze and image a 9-day long tremor sequence beneath Shikoku (Poiata et al., 2018).

In this work, we propose supplementing a variation of the method presented by Poiata et al. (2016) with a machine learning classifier. Although similar neural network approaches have been considered (see for example Beaucé et al., 2019), the distinct advantage of our implementation is that it does not depend on templates and is unconstrained by user-defined detection thresholds.

2.3 Data

In this study, we analyze vertical-component data recorded by 76 broadband stations located in northern California belonging to the Mendocino Experiment. This region encompasses the northern San Andreas Fault system as well as the forearc of the southern Cascadia subduction zone. The geographic footprint of these stations is approximately 400 km^2 (see Figure 2.2.1) and the group yields 2,850 unique station pairs. The average interstation separation is 162 km. From this group of stations, we download all available data recorded between September and December 2008, (date range chosen arbitrarily) as day-long seismograms.

Our supervised machine learning approach further requires a set of known labels in the form of previously detected and located earthquakes. We therefore consult a regional earthquake catalogue acquired from the USGS and obtain a set of 233 unique seismic events that occurred during the period between September and December 2008 and within the geographic bounds shown in Figure 2.2.1. The only restrictions we impose on the catalogue are to limit event depths to 40 km and magnitudes to those above 2.0. Event magnitudes for this set range from 2.0 to 4.5 (according to differing magnitude types).

2.4 Methods

The method proposed in this study involves two main components. The first of these is a variation on the method proposed by Poiata et al. (2016) and is detailed in section 2.4.1. We refer to this component as time-delay projection mapping. For our purposes, the essential feature of time-delay projection mapping is that it allows one to create a 3D image that captures information about the coherence of the seismic wavefield during a given time window. The second component of our methodology involves training a neural network classifier to predict periods of time in our data set that contain seismic events and is detailed in section 2.4.2 To make such predictions, our neural network uses features calculated from the 3D images computed using time-delay projection mapping, as well as features derived from raw waveform data.

Note that the order of individual data processing elements within section 2.4.1, as presented, is not the same as the order in which they occur in practice. The order in which these operations occur in practice is as follows: (1) we first download day-long seismograms; (2) we spectrally whiten them then decimate them to 1 Hz; (3) we window seismograms and build CFs over each window; (4) we perform all possible local cross-correlations (LCCs) between station pairs for every window in our data set; (5) lastly, we map all station pair LCCs into space to create several thousand 3D seismic images for each window. Each of these operations are detailed in the following sections.

2.4.1 Time-Delay Projection Mapping

Overview

Many array-based seismic detection and location methods often require calculating the cross-correlation between simultaneously recorded seismograms (e.g., template matching techniques, Gibbons & Ringdal, 2006; Ross et al., 2019; Shelly et al., 2006) (for an example of a non template matching study see Dales et al., 2017). These cross-correlation functions (CCFs) are most commonly thought of as measures of similarity between signals as a function of lag time. One may also think of CCFs as sets of probabilistically weighted travel-time differences (TTDs) or time-delay estimates (TDEs), where it is implicitly assumed that a common source has been recorded by sensors. For example, given an observed TTD between two sensors, and knowledge of the velocity structure, one may identify spatial positions from

which a source would produce that TTD (e.g. Dales et al., 2017). In homogeneous media, constant TTDs will be spatially associated with hyperbolic surfaces. This is illustrated in Figures 2.4.1A and 2.4.1B, in which values of an idealized, synthetic CCF between two sensors have been mapped to hyperbolic trajectories, a process that we refer to as time-delay projection mapping. This procedure generalizes to heterogeneous media, where the hyperbolas are distorted by various wave effects. Furthermore, the functions being used for the mapping need not be CCFs, as any type of TDE function can be used.

Mapping a single TDE function between a pair of sensors into space is insufficient to constrain the location of a source. The utility of the method comes from the fact that mapping several station pair TDEs exploits the coherency of station signals caused by a common source. Following Poiata et al. (2016), we refer to the combined mapping of all pairwise TDE functions as the spatial likelihood function (SLF). In Figures 2.4.1C and 2.4.1D, we show an example of a multipair SLF, under idealized conditions, and using CCFs as our TDE functions. In general, if any pairwise set of TDE functions satisfy, (1) their range is over the interval [0, 1], and (2) their respective domains occur over the maximum possible time delay between each station pair, then their SLF will represent a conditional likelihood, normalized in the interval [0, 1] (Poiata et al., 2016). In other words, the SLF represents the likelihood of observing a pairwise set of TDEs, given a source occurred at some position in a search volume of interest.

Several steps are involved in computing SLFs from individual seismograms, and the computation is performed over a given time window. Once computed, the location of a source occurring during that time window can be extracted from the peak of the SLF, and the origin time can be extracted from the TDE functions used to create it (see Poiata et al., 2016, for details).

In this overview, we have left out many important details. For instance, in practice, one must take into account station sampling rates, imperfect knowledge of the velocity structure, and implement the method within a 3D grid; each of these elements affects SLFs. In the next three sections, we describe key elements borrowed from Poiata et al. (2016) for computing SLFs; these are kurtosis-based CFs, TDE functions, and the projection of TDE functions onto a velocity space. At the end of section 2.4.1 (Spectral Whitening and Methodological Differences), we discuss modifications to the method of Poiata et al. (2016).

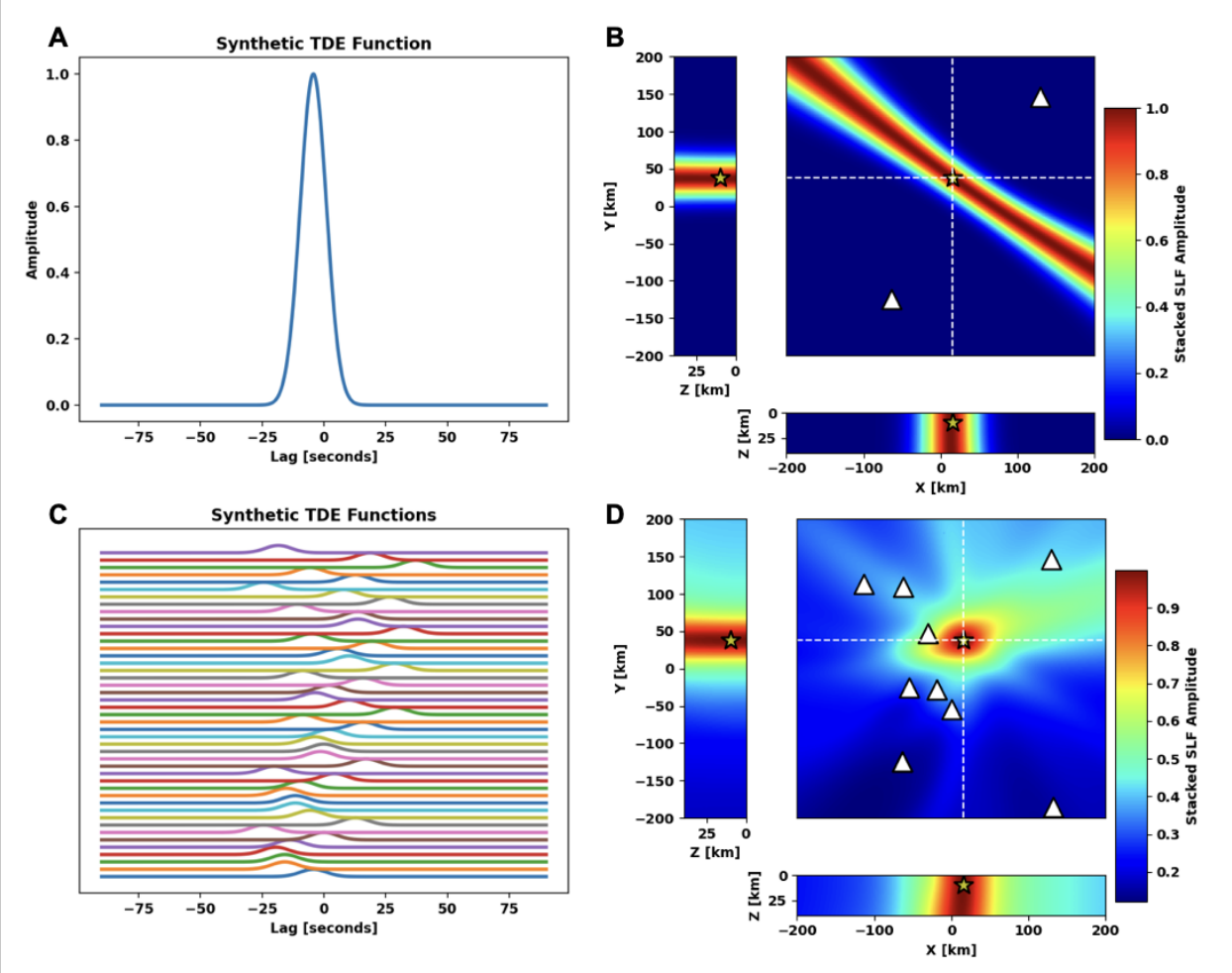

Figure 2.4.1: Synthetic SLFs in the single and multipair cases. Panel A: Synthetic TDE function between N = 2 stations. Panel B: Projection of the TDE function (SLF) in Panel A onto a 3D spatial grid surrounding the stations. Panel C: Synthetic TDE functions between N = 10 stations, including the two stations in the single pair case. Panel D: Aggregated projections of the TDE functions in Panel C onto a 3D spatial grid surrounding the stations. Stations are plotted as white triangles. The source location is plotted as a yellow star. In Panels B and D cross sections of the SLFs are plotted using the coordinates of the source.

Kurtosis-Based Characteristic Functions

In this study, we focus on P-wave arrivals from impulsive (i.e., earthquake) events and restrict ourselves to vertical-component data. Therefore, we choose to construct CFs using a recursive kurtosis operation. Kurtosis, defined as the fourth standardized moment, is

expressed mathematically as

$$Kurt[X] = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right], \tag{2.1}$$

where E denotes the expectation operator, μ is the mean defined by $\mu = E[X]$, and σ is the standard deviation of the random variable X. It is a suitable basis for constructing CFs in this context because the presence of transient changes in the nonstationary statistics of signals generates rapidly increasing values of the kurtosis function (Saragiotis et al., 2002). In other words, the kurtosis-based CF of a seismic signal will highlight direct P-wave arrivals. However, since kurtosis functions are vulnerable to amplifying any transient causing significant changes in signal statistics, such as spurious spikes, they alone cannot be used as the basis of seismic detection. Instead, the coherence of kurtosis CFs across multiple stations must be utilized when examining potential detections.

In constructing kurtosis-based CFs, we employ a recursive computation scheme (see Chassande-Mottin, 2002; Langet et al., 2014, for details). Within this scheme, the kurtosis at each sample t_i of a time series $u(t)$ is computed as

$$K(t_i) = \frac{m_4(t_i)}{(m_2(t_i))^2} \tag{2.2}$$

where $m_2(t_i)$ and $m_4(t_i)$ are estimates of the second and fourth central moments (i.e. $m_{p=2,4}$), respectively, which are themselves estimated using the recursive exponential average $\hat{\mu}(t_i)$ (Baillard et al., 2013; Langet et al., 2014):

$$\hat{\mu}(t_i) = C \cdot u(t_i) + (1-C) \cdot \hat{\mu}(t_{i-1}), \tag{2.3}$$

$$m_p(t_i) = C \cdot (u(t_i) - \hat{\mu}(t_{i-1}))^p + (1-C) \cdot m_p(t_{i-1}). \tag{2.4}$$

Thus within this framework, one must specify a decay constant C that controls how strongly the recursion depends on previous input samples, chosen such that

$$0 < C - 1 < 1, \tag{2.5}$$

and which can be expressed as

$$C = \delta T / T_{decay}, \tag{2.6}$$

where δT is the station sampling rate (in seconds) and T_{decay} is the time length of the averaging scale. Thus, C controls how quickly the kurtosis decays after spiking in response to a short transient. These restrictions on C also impose upper and lower bounds on the choice of T_{decay}, namely, the total signal length and the station sampling rate, respectively. In this study, we fix C by choosing a T_{decay} of 4 s as this choice leads to the best overall performance of the method.

After computing recursive kurtosis functions for signals in our data set, the next step in the process of building suitable CFs is to take positive derivatives of these functions in order to minimize delays between the onsets of transients and their corresponding maxima. Finally, to account for timing uncertainty, the positive derivatives of kurtosis functions are convolved with Gaussian windows of width $T_{decay}/2$ (2 s in our case). Thus, the T_{decay} parameter both determines the decay rate of kurtosis estimates and smooths the final CFs to account for onset uncertainties. Example kurtosis CFs are shown in Figure 2.4.2.

Time Delay Estimate Functions

Having replaced seismograms in our data set with kurtosis-based CFs, we then use them to compute pairwise TDE functions. In the overview of this section 2.4.1, we presented an example of a SLF built using cross-correlation as the basis of our TDE functions (Figure 2.4.1). An attractive alternative to cross-correlation is the operation of LCC, which provides a time-dependent measure of the similarity between two signals by estimating the cross-correlation in the neighborhood of each data point (Fomel, 2007; Hale, 2006a). Mathematically, the operation is expressed as the convolution of a Gaussian function with the product of shifted signals

$$LCC(t', l) = \int_{-\infty}^{\infty} f(t, t') g(t + l, t') dt, \tag{2.7}$$

with

$$f(t', t) = f(t) w(t' - t), \tag{2.8}$$

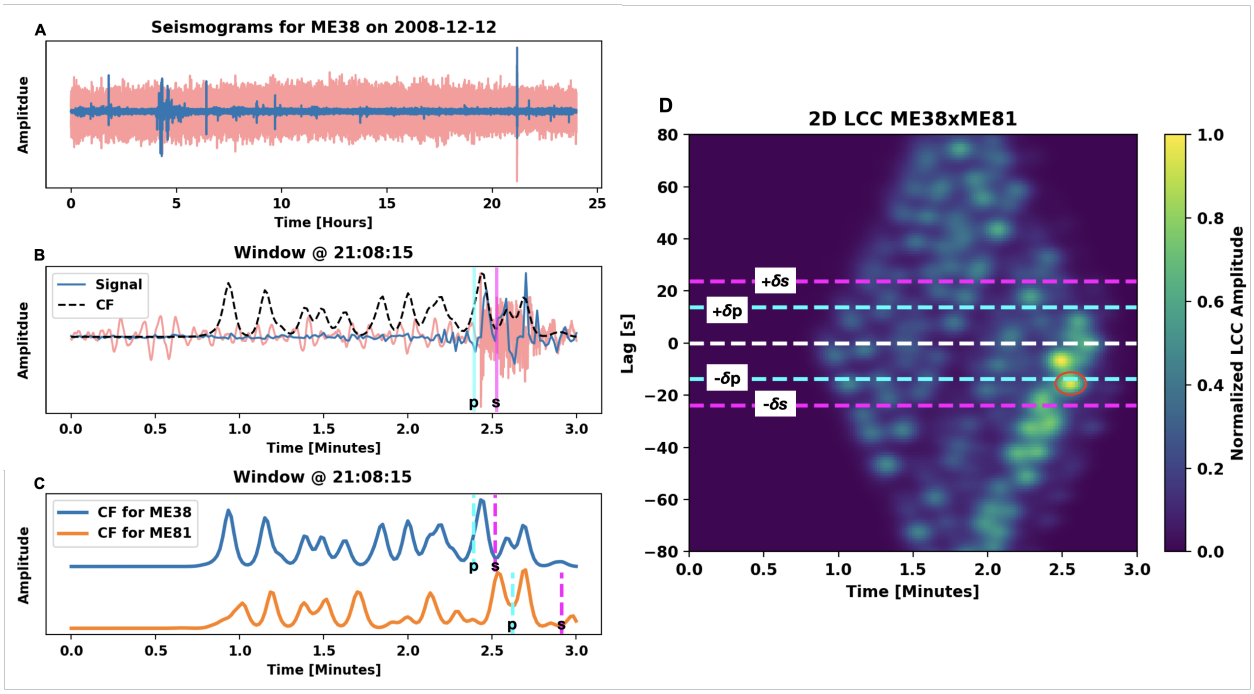

Figure 2.4.2: An example of CFs and 2D LCC. Panel A shows the raw (pink) day-long seismogram recorded by station ME38 on 12 December 2008, as well as the whitened signal (blue). Panel B shows the raw signal (pink) for the 3-min window commencing at 21:08:15 UTC, the whitened signal (blue), the kurtosis CF built from the whitened signal (black, dashed), and P and S arrivals computed from IASP91. Signal amplitudes have been normalized. Panel C shows the CF from Panel B as well as the CF built during the same window from station ME81. P- and S-wave arrivals computed from IASP91 are also shown. Panel D shows the 2D LCC between the CFs in Panel C. There is a peak in the 2D LCC corresponding to the theoretical P delay between the two stations at a lag value of approximately -17 s (circled in red). While nondirect P-delay peaks in LCCs such as the one at approximately -10 s are not uncommon, only peaks coherent between multiple LCCs contribute to the SLF.

$$g(t', t) = g(t)w(t' - t), \tag{2.9}$$

where $w(t' - t)$ is a Gaussian function with half-width σ, centered at $t = t'$, and $f(t)$ and $g(t)$ are the signals to be correlated. The parameter σ accounts for uncertainties in the assumed velocity model. In this study, we set $\sigma = 2$ s. For further details on how to compute LCCs efficiently, see Hale (2006a, 2006b) and Young and van Vliet (1995).

While LCCs are 2D functions of lag time and time, they can be reduced to 1D functions of lag time in a manner that maintains the useful properties of TDE functions. This reduction is accomplished by applying a maximum operator to the 2D LCC function over its lag domain. Furthermore, LCC functions need not be computed over all possible lag times, since

both the group of stations used and the velocity model constrain the maximum possible lag between each station pair. Therefore, an upper bound can be set for the lag domain of LCC computations, corresponding to the greatest maximum lag time among all station pairs in the group. In our case, the greatest lag among stations is 80 s. An example 2D LCC is shown in Figure 2.4.2 between kurtosis CFs for two stations used in this study.

Projecting Time Delay Estimates Into Space

The construction of SLFs occurs by projecting pairwise TDE functions into space and then aggregating the results. To do so, one must work with a discrete spatial grid and velocity model for a given seismic phase of interest. Theoretical travel times from each grid point to each station and therefore TTDs between each pair of stations can be precomputed, which allows one to associate a set of theoretical TTDs to each grid point. In order to map a single TDE function into space, we perform a grid search over the theoretical TTDs associated with the grid points for the corresponding station pair. Once the grid points with the closest theoretical TTD are found, they are assigned the value of the TDE function (LCC) at that TTD (lag value). The final SLF is constructed from the summation of all such TDE mappings and normalized by the number of contributing stations pairs. Note that theoretical travel-time calculations need only be calculated once for the entire search volume, given the station geometry and grid parameters. In this study, we compute travel times using a simple ray-tracing algorithm and a 1D P-wave velocity model composed of a regional crustal velocity model for California and the AK135 reference model at greater depths (Kamer, 2015; Kennett et al., 1995). The velocity model is summarized in Table 2.4.1.

In order to use time-delay projection mapping to locate sources within a large, temporally contiguous data set, one needs to partition the data set into smaller time segments. SLFs are then computed for each segment, which involves two steps: (1) computing pairwise TDE functions and (2) projecting those TDE functions onto a spatial grid. In general, several factors will control the computational demand of this process, including the total time interval spanned by the data set, the size of subsegments, the number of station pairs, the sampling rate of the stations, and the number of grid points in the search volume of interest.

We choose to partition our data set into 234,240, 3-min long segments with 75% overlap. In general, overlapping windows should be used for such tasks because they mitigate against

abruptly cutting earthquake signals. We choose a window length of 3 min based on (1) the maximum observable time lag between stations in this study according to our assumed velocity model (Table 2.4.1) (∼80 s); (2) in order to accommodate the full domain of LCCs between signals, we require the length of our time windows to be twice as long as the maximum time lag (∼160 s); and (3) the extra 20 s act as a short buffer. Thus, the spatial scale of the network imposes a limit on the size of the windows that can be used.

Table 2.4.1

P-Wave Velocity Model

Z (km)	V_p (km/s)
0	5.28
2	5.68
4	6.05
6	6.13
8	6.21
10	6.30
12	6.36
15	6.53
18	6.59
23	6.83
30	7.84
50	8.20
80	8.045
120	8.051

Note. Entries at depths shallower than 80 km come from (Kamer, 2015). Entries at depths of 80 km or greater come from AK135 (Kennett et al., 1995).

In mapping LCC functions to space, we choose to use a spatial grid that covers the geographic footprint of our stations and which extends deep enough to coincide with the depth range of events from the earthquake catalogue used in this study. Therefore, the dimensions of our grid are approximately $400 \times 400 \times 40$ km in X, Y, and Z, respectively, and the origin $(0, 0, 0)$ corresponds to the geographic center of our stations (ignoring elevation differences). Since mapping LCC functions to space involves a grid search over a large number of points, the number of grid points, as well as the number of station pairs (2,850), are significant in determining the computational speed of the method. We choose to use a grid cell size of 5 km^3 , which leads to 48,048 grid points in our search volume (according to our exact dimensions). The rationale for this choice will be discussed later.

Spectral Whitening and Methodological Differences

Impulsive seismic sources are, by definition, broadband signals (i.e., they excite energy across a wide range of frequencies), and in general, broadband signals are best characterized through

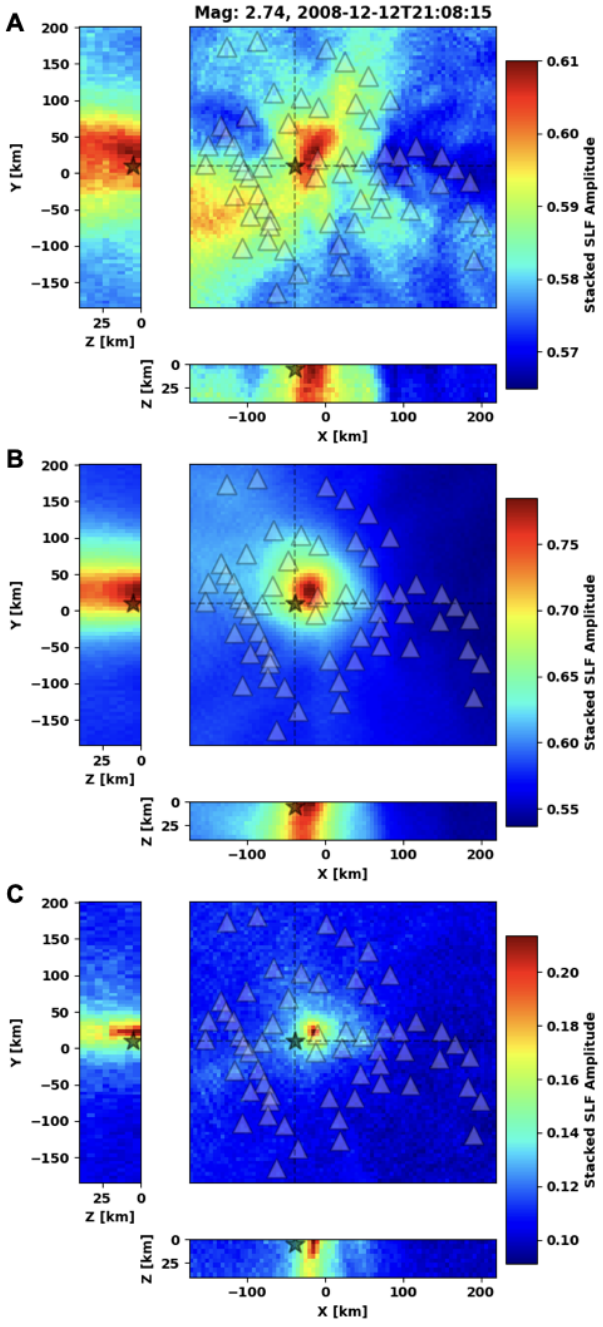

Figure 2.4.3: A comparison of SLFs computed for the same time-window during 12 December 2008 and commencing at 21:08:15 UTC using three different methodologies. The black star denotes the location of a known Md 2.74 event. Note that the decay constant for the construction of kurtosis CFs is the same across examples. The SLF in Panel A is computed using the standard pre-processing approach. The SLF in Panel B is computed using our preferred method, based on spectral whitening. The SLF in Panel C is computed following Poiata et al., (2016).

some form of time-frequency analysis (Tary et al., 2014). For this reason, when constructing kurtosis-based CFs to highlight P waves from impulsive sources, as a prior step Poiata et al. (2016) multiband filter raw seismograms into several logarithmically spaced narrow frequency bands. They then compute recursive kurtosis functions in each filter band and ultimately combine these to create a broadband CF for each seismogram under consideration. Instead of multiband filtering, we choose to incorporate signal frequency characteristics through a spectral whitening procedure, which gives us the flexibility to heavily decimate our signals from 40 to 1 Hz. Normally, the act of decimation severely limits which frequency components a signal is capable of resolving, in accordance with the Nyquist-Shannon sampling theorem. Thus, to counteract the loss of frequency information due to decimation, we spectrally whiten signals before decimating by reconstructing them entirely from their phase. While

spectral whitening (or spectral normalization) is not a common preprocessing operation in seismic detection and location methods, it is a widely used operation in the context of ambient noise interferometry (see for example Bensen et al., 2007; Brenguier et al., 2007; Mosher & Audet, 2017).

The main advantages of spectral whitening are that (1) it is an effective pre-processing step. Mosher and Audet (2017) show that when whitening raw signal data the result is comparable to applying the typical preprocessing steps of detrending, demeaning, and high-pass filtering, except that whitening further achieves frequency normalization. (2) When the main data processing operation of a study involves a cross-correlation-like operation, then there is no need to deal with instrument responses as long as instruments all have the same phase response (this is indeed the case for stations in this study). (3) Whitening through phase reconstruction equalizes disparate signal amplitudes, making low magnitude events more easily visible in whitened signals. (4) Finally, a signal's phase information is the essential component needed for obtaining accurate travel-time measurements, hence preserving raw signal phase aids in acquiring unbiased travel-time measurements from cross-correlations (Seats et al., 2012).

Therefore, to construct CFs for our signals, we simply whiten day-long seismograms, decimate them, and then compute kurtosis functions for individual windows. An example of this process is shown in Figure 2.4.2. Because the decimation occurs after whitening and the relative shapes of CFs are the only important part when correlating signals, we are still able to create high-quality SLF images with significant improvements in computational speed. In Figure 2.4.3, we compare SLFs created using three different methodologies for the time window during 12 December 2008 and commencing at 21:08:15 UTC. Note that this time window contains a known Md 2.74 earthquake whose location is denoted by the black star. Figure 2.4.3A displays a SLF created by first detrending, demeaning, and high-pass filtering day-long seismograms. Kurtosis CFs are then computed for each station signal in the window, cross-correlated locally, and then mapped into space. Figure 2.4.3B displays a SLF created by simply whitening day-long seismograms then decimating 40 times (our preferred method). Again, station signals in the window are correlated locally and then mapped into space. Lastly, Figure 2.4.3C shows a SLF created by first detrending and demeaning day-long seismograms. Then, for each signal in the window, a broadband kurtosis CF is created through a multiband filter approach with 12 logarithmically spaced frequency bands between 1 and 40 Hz (method of Poiata et al., 2016). The rest of the processing is the same as the other cases. Figure 2.4.3 demonstrates that a significant improvement is achieved in the overall clarity of the SLF image produced in b versus a (whitening vs. nonwhitening), as

well as a consistency in the imaged source location across all three panels. However, while Panel c does yield a more precise location estimate than either a or b, it takes 1,000 times as long to compute as the SLF in Panel b because signals in c sample at 40 times the rate of signals in b. Thus, given the size of the data set from which we wish to detect and locate earthquakes, the increase in speed brought about by decimating, which is facilitated by whitening, is extremely useful.

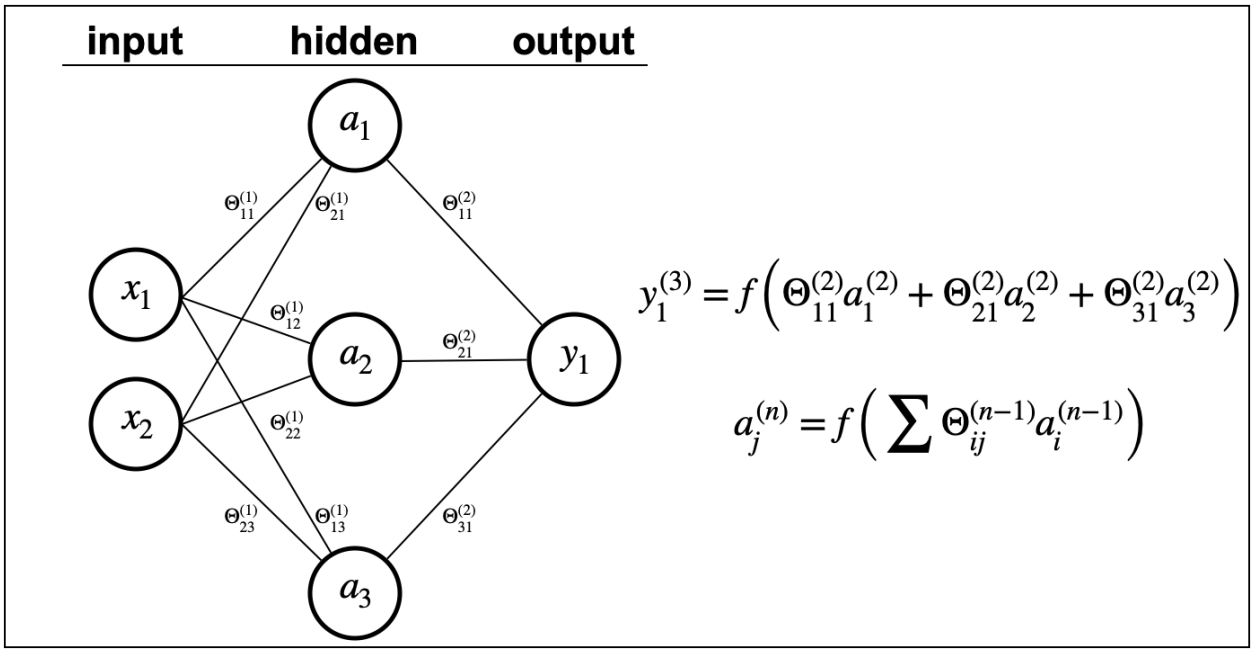

Figure 2.4.4: Schematic of a three-layer feed-forward neural network with input units x_1 and x_2, hidden activation units a_1, a_2, and a_3, and single output unit y_1. Network weights in the ith layer are designated by $\Theta^{(i)}$. $f(\cdot)$ symbolically represents any activation function.

2.4.2 Event Detection Via Neural Network Classification

Overview

Using time-delay projection mapping, we create a large set SLFs from which we wish to robustly identify regional earthquakes. Thus, we are presented with a choice: either manually search through SLFs and flag candidates for further inspection or attempt to find events through an automated procedure. Since we have a catalogue of earthquakes known to occur within our data set, we formulate the detection problem as a supervised machine learning task. Given that we are interested in distinguishing between earthquake and noneartquake classes, we seek to use a binary algorithm. Several supervised machine learning algorithms

exist for binary classification, and many have been used in the geosciences (for reviews, see Bergen et al., 2019; Rojas et al., 2019). Among the most popular classification algorithms are neural networks, logistic regression, and support vector machines. In this study, we choose to train a neural network to predict which time windows in our data set contain earthquake signals.

A neural network is a machine learning algorithm loosely based on models of interconnected neurons in biological brains (Bishop, 1995), and such models can be trained to learn arbitrarily complex functions given a set of examples as inputs with known outputs (Lapedes & Farber, 1988; Valentine & Woodhouse, 2010). While the body of literature on the theory, principles, and architecture of neural networks is vast, one of the simplest forms of neural networks, the multilayer perceptron, is well-suited to simple classification tasks (MacKay, 1992).

A multilayer perceptron is a feed-forward network with threshold or sigmoidal activation functions (Bishop, 1995). Feed-forward networks refer to networks in which the information processing flows unilaterally (i.e., this class of networks excludes feedback elements such as in recurrent neural networks). A simple multilayer perceptron with three layers, each of which contain individual nodes (or neurons) is shown in Figure 2.4.4. The first layer in such a network is called the input layer, and the final layer is simply the output layer. All layers in between the input and output layers are referred to as hidden layers. The specific network in Figure 2.4.4 takes two features as input in the form of a 2D vector $\mathbf{x} = (x_1, x_2)$. The activation at node a_i in the hidden layer is computed by applying an activation function $f(\cdot)$ to the linear combination of weights Θ_{ij} and inputs from the previous layer ($\mathbf{x}$ in this case). Likewise, the output, or equivalently the activation in the final layer, is computed in the same way. Typical sigmoidal activation functions include the standard logistic function $L(x) = 1/(1 + e^{-x})$ or the hyperbolic tangent $tanh(x)$. Network training is accomplished by first randomly initializing network weights and then pushing known inputs through the network. Once the output for a given input is computed, the loss between the known output and the result computed by the network can be calculated. In a process known as backpropagation, the network then uses any gradient-descent style algorithm to make adjustments to its weights in order to reduce the error. This training process is then run for a large number of iterations until the network is sufficiently trained.

Feature Selection

Prior to training a neural network, we must calculate features that will be used as input into the model. For each of the 234,240 windows in our data set, we compute 14 features: seven of which come directly from SLFs and seven of which come from raw waveform data. From the SLFs, we compute the seven Hu image moments from 2D cross sections of our 3D SLFs as features. Hu image moments were first devised by Ming-Kuei Hu (1962). Formally, the 2D $(p+q)th$-order moment of a continuous density function $f(x,y)$ is defined as

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy, \tag{2.10}$$

and the 2D central moments of order $p+q$ are

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q f(x,y)dxdy, \tag{2.11}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}. \tag{2.12}$$

If the image density function $f(x,y)$ is assumed to be piecewise continuous, then moments of all orders exist, and the set of moments $\{m_{pq}\}$ for $f(x,y)$ are uniquely determined, and $f(x,y)$ is uniquely determined by its moments (Ming-Kuei Hu, 1962; Zhihu Huang & Jinsong Leng, 2010). Hu moments are combinations of various normalized 2D central moments

$$\eta_{pq} = \frac{\mu_{pq}}{(m_{00})^\gamma}, \quad \gamma = (p+q+2)/2, \quad p+q = 2,3,\ldots \quad . \tag{2.13}$$

Explicitly, they are

$$\phi_1 = \eta_{20} + \eta_{02},$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2,$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2,$$

$$\phi_5 = (\eta_{30}-3\eta_{12})(\eta_{30}+\eta_{12})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2]+(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2],$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}),$$

$$\phi_7 = (3\eta_{21}-\eta_{03})(\eta_{30}+\eta_{12})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2]-(\eta_{30}-3\eta_{12})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2].$$

The aspects of the Hu image moments that make them excellent image-based features are that they are invariant under transformations of scaling, translation, and rotation. Thus images that are *essentially* the same, apart from these transformations, will have *similar* Hu moments. In our application, Hu moments are especially useful as features since SLFs containing well-located earthquakes will be similar under these transformations, whereas SLFs containing only noise will be significantly different. We compute Hu moments from our 3D SLFs by taking 2D cross sections at the maximum value of the SLF. In Figure 2.4.5, we compare Hu moments computed from four of our SLFs. Figure 2.4.5 demonstrates that the Hu moments of the earthquake examples are nearly identical, as expected, given that the images are *essentially* the same.

The remaining seven waveform-derived features are based on the frequency content expected for earthquake signals since, when compared to noise, regional earthquake signals typically contain greater power at lower frequencies. To encapsulate this observation into seven network-wide features, we first compute probabilistic power spectral densities (PPSDs) from hour-long noise segments recorded by each of our stations. This is done following the method of McNamara and Buland (2004), using as much data as possible recorded by each station (up to 4 months). Knowing the noise characteristics of each station over the entire period of time considered, we can assess the probability of observing a given power, at a given frequency, during any time segment in our data set. Under the assumption of observing noise, PPSDs enable us to calculate statistical p values. In other words, stations observing statistically unlikely powers at certain periods highly suggest the presence of signals other than noise within the time window in question. We choose to compute station p values at seven periods (0.1, 0.2, 0.5, 1, 2, 5, and 10 s) for each time window. A portion of this process is illustrated in Figure 2.4.6 where the PPSD for station ME38 is shown in Figure 2.4.6A. In Figure 2.4.6B, we show Gaussian kernel density estimates of the probability distributions of PSD values computed at 0.2, 0.5, and 1.0 seconds. These distributions are used as the basis for computing the probability of observing specific PSD values at these periods during any time window in the data set. Finally, to aggregate statistical p values into network-wide features, we take the mean p values of the 10 stations that have the lowest overall p val-

ues across all frequencies considered. The rationale for such averaging is that we require a detectable earthquake to be observed by at least 10 stations in the network.

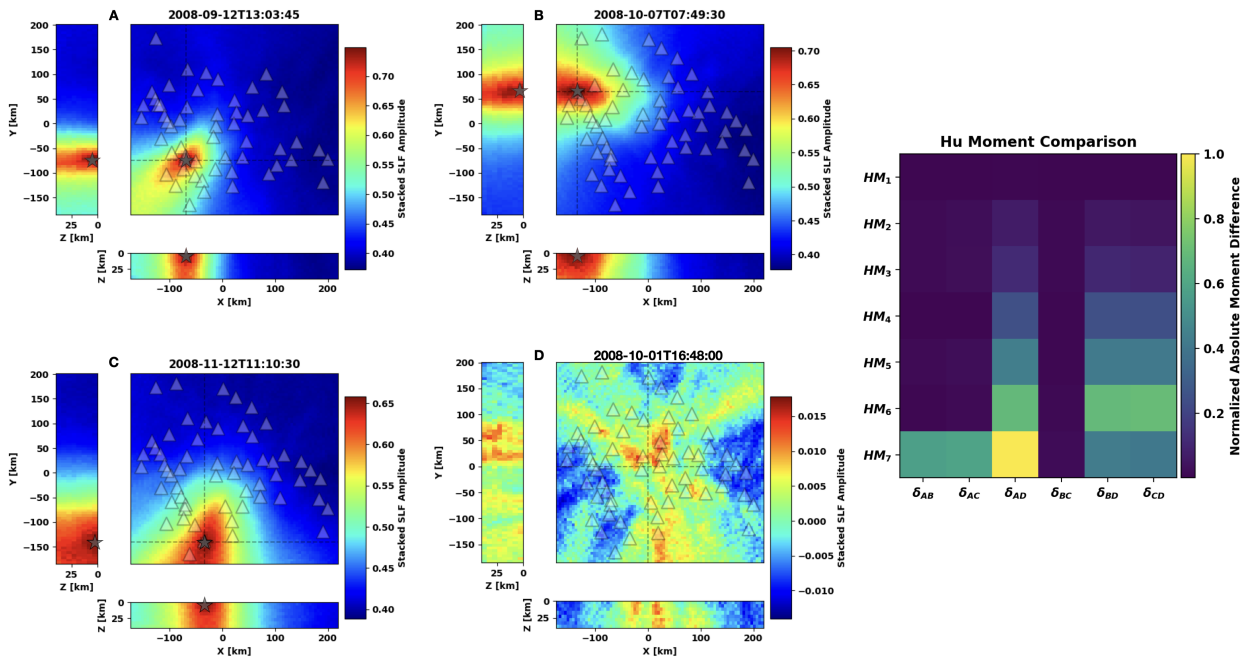

Figure 2.4.5: Panels A–C of this figure show examples of three different SLFs in this study either known or determined to contain a seismic event. Panel D is a SLF of a noise window. Hu image moments have been computed for each of these SLFs, and the differences between each moment of each figure are shown. Clearly, Hu moments for the earthquake examples are nearly identical.

Feature Evaluation and Training

Any set of examples we use to train a neural network classifier must include features from time windows that contain earthquakes and those that do not. We refer to these as our earthquake and null classes, respectively. In order to create a set of null examples, we view waveforms and SLFs associated with several thousand time windows in our data set chosen randomly. To be confirmed as a null example, we require that neither the waveforms nor the SLF associated with the time window in question shows any visually obvious seismic signal. We repeat this process until a set of 1,000 nulls are confirmed.

The seismic catalogue provides us with 233 potential earthquake examples; however, not all of the events listed in the catalogue are suitable for our needs. The catalogue contains all known events of magnitude 2.0 or greater within our search volume, but not all of these events are large enough to produce coherent energy across multiple stations and many are therefore

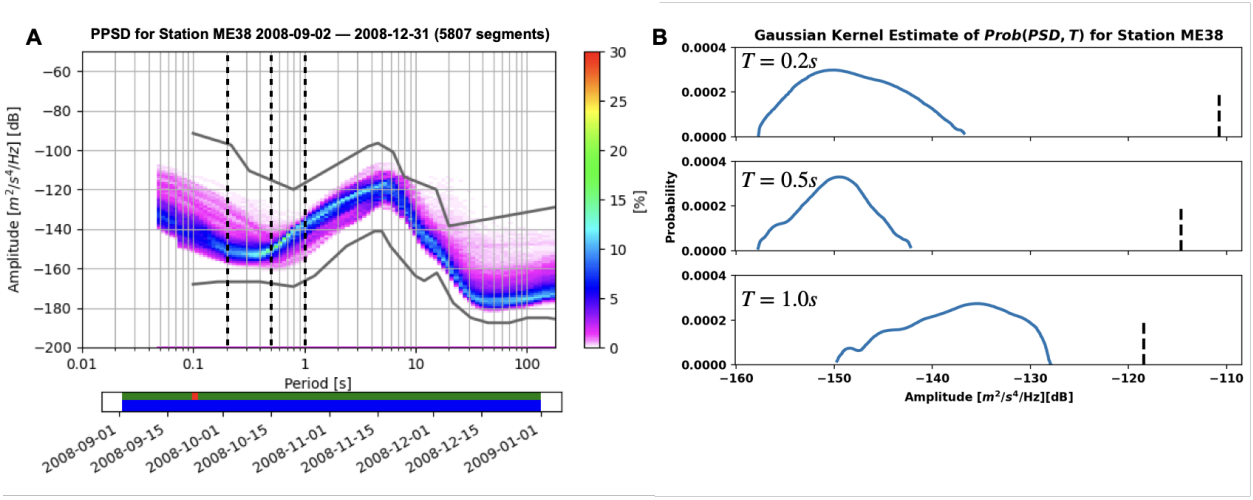

Figure 2.4.6: Panel A: PPSD for station ME38 computed from 5,807 noise recordings between 2 September and 31 December 2008. Panel B: Gaussian kernel density estimates of the distribution of PSD values along three discrete periods in the PPSD (0.2, 0.5, and 1.0 s). These periods are indicated by the black dashed lines in A. Black dashed lines in B indicate the observed power at these periods during the 3-min time window commencing at 21:08:15 UTC during 12 December 2008. Because this window contains an Md 2.74 earthquake, the probability that the power observed at these frequencies is due to noise is essentially zero (i.e., the statistical p value = 0).

unobserved in our SLF images. Indeed, the average interstation spacing of our network is 162 km, and the mode is 142 km. Furthermore, it is difficult to assess an observational magnitude limit since the catalogue contains multiple magnitude types. We find that none of the events with listed magnitudes of 2.5 or greater are unobserved in the SLFs produced by our network, but below this range, some events are observed in our SLFs and some are not. Considering these facts, the best we can do is empirically guess that our method is capable of reliably detecting events of magnitude "2.5" or greater. That being said, we note that the average magnitude uncertainty of catalogued events that are unobserved in our SLF images is 0.163. Evidently, the scale of the network being used impacts the magnitude detection threshold. However, beside mitigating against abruptly cutting earthquake signals when windowing data, the use of highly overlapping windows in our work allows us to bolster our set of earthquake examples. Since many of the earthquake examples in the catalogue are visible across multiple windows, we end up with a set of 249 earthquake examples to use during training and testing (corresponding to 113 of the catalogued events). Thus, our machine learning data set consists of 249 earthquake examples and 1,000 null examples.

Not all of the features we compute are equally useful for discriminating between time windows in our data set that contain either noise or earthquakes. Before training, we first assess which features are the most useful from pair-plot distributions of the features computed

for examples in our training set. From this exercise, we settle on four features to use as input to train a neural network: the second and fourth Hu moments and the aggregated network p values at 0.2 and 0.5 s (Figure 2.5.1). These features are selected because their distribution between our two classes shows the clearest separation between them.

2.5 Results

In this study, we train an ensemble of 15 neural network classifiers to predict which time windows in our data set contain earthquakes. Each network is run for 20,000 training epochs, and we select for our final classifier the network that achieves the best overall score. We quantify total network score by averaging 4 metrics that we use to quantify individual network performance (discussed below). We use an ensemble of networks to more effectively find a favorable final state of the network since training involves the use of stochastic gradient descent. When training each network, we use the same set of earthquake and null examples. Our network architecture includes a 4D input layer which takes the second and fourth Hu image moments as features as well as the aggregated network p values at periods 0.2 and 0.5 s, a single hidden layer with 14 nodes and an output layer with two nodes. Between the input and hidden layer, we apply a rectified linear unit (ReLU) activation function, and a sigmoid function is applied to the output layer. The sigmoid function maps any input to the interval [0, 1]. Thus, the two output nodes represent probability estimates that an input belongs to each of our respective classes. Note that we define our class labels as $y(earthquake) = 1$ and $y(null) = 0$. Finally, networks are trained using a cross-entropy loss function (MacKay, 1992).

Given our chosen architecture, networks in this study contain a total of 100 parameters (including bias units). Generally speaking, networks with large numbers of tuneable parameters are better able to learn complicated nonlinear relationships but simultaneously require a large number of training examples to learn effectively. In order to avoid overfitting, it is best to keep the number of network parameters as low as possible (Valentine & Woodhouse, 2010). A general rule of thumb is to ensure that there are at least 10 times as many training examples as there are network parameters.

When training each network, we set aside 30% of our machine learning examples as a test set. The test set is used to gauge the performance of trained networks. Various metrics exist for quantifying network performance against a test set, and most are derived from various combinations of the number of true positives (TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs) that result from the test set. We quantify overall network performance by averaging the precision, recall, balanced accuracy (BA), and F1 score. BA is a more appropriate metric to use than traditional accuracy when working with imbalanced data sets, as is the case here, and the F1 score is the harmonic mean of the precision and

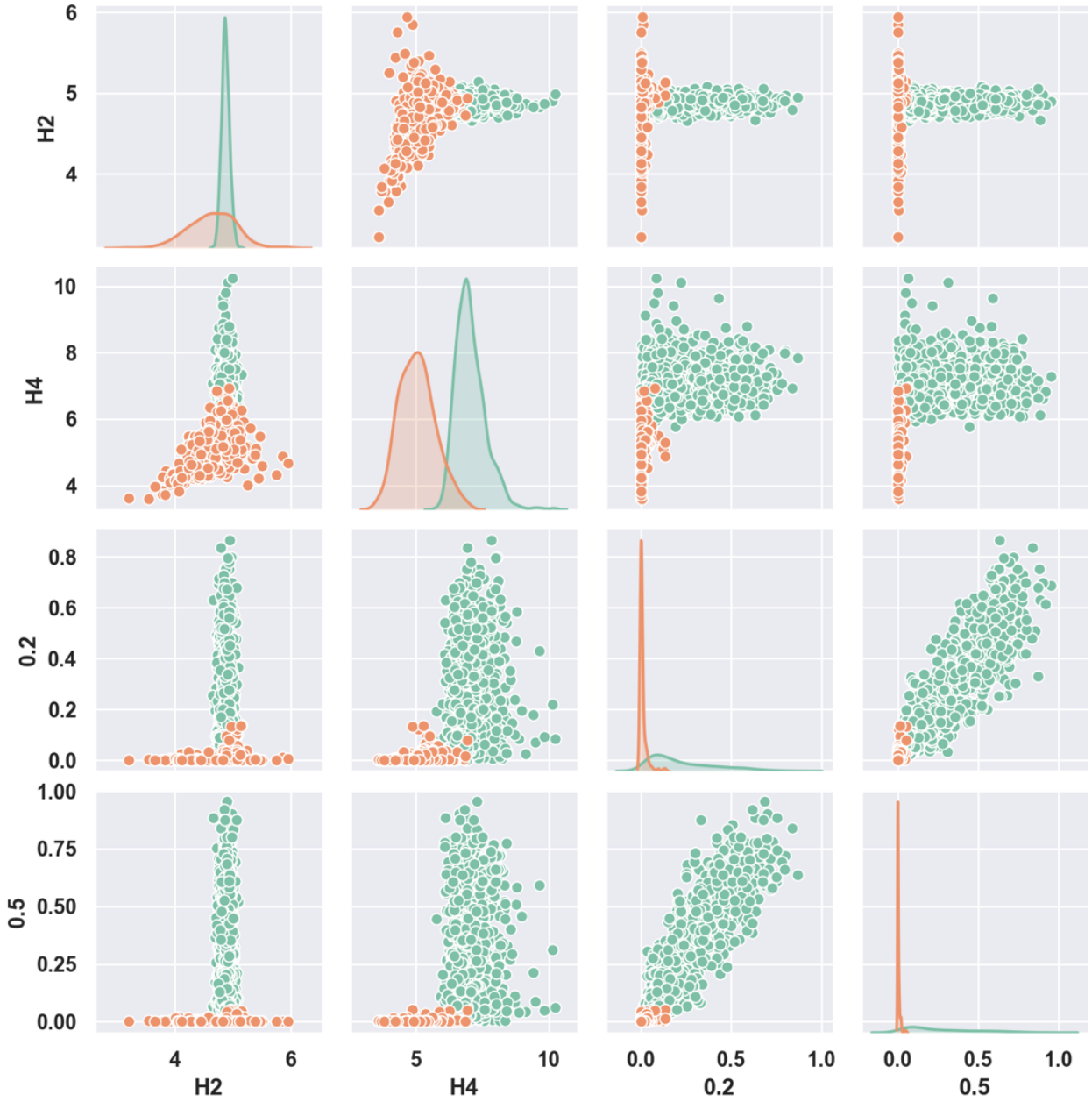

Figure 2.5.1: Pair plots of the distribution of features among our training examples (H2 and H4 are the second and fourth Hu image moments, 0.2 and 0.5 are the networked averaged p values at periods of 0.2 and 0.5 seconds). Orange points represent the earthquake class. Green points represent the null class.

recall. Mathematically, these metrics are defined as

$$Precision = \frac{TP}{TP + FP} \ , \tag{2.14}$$

$$Recall = \frac{TP}{TP + FN} \ , \tag{2.15}$$

$$BA = \frac{1}{2}\left(Recall + \frac{TN}{TN + FP}\right) \ , \tag{2.16}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \ . \tag{2.17}$$

The results of our final classifier against the test set are shown in Figure 2.5.2A in the form of a confusion matrix. This network achieved a precision of 97.4%, a recall of 97.4%, a BA of 98.4%, and an F1 score of 97.4% for a total network score of 97.7%. Having chosen our final classifier, we supply it our four features computed for each window in our 4-month long data set to identify which time windows are likely to contain earthquakes. The network predicts that 2,522 windows contain earthquakes.

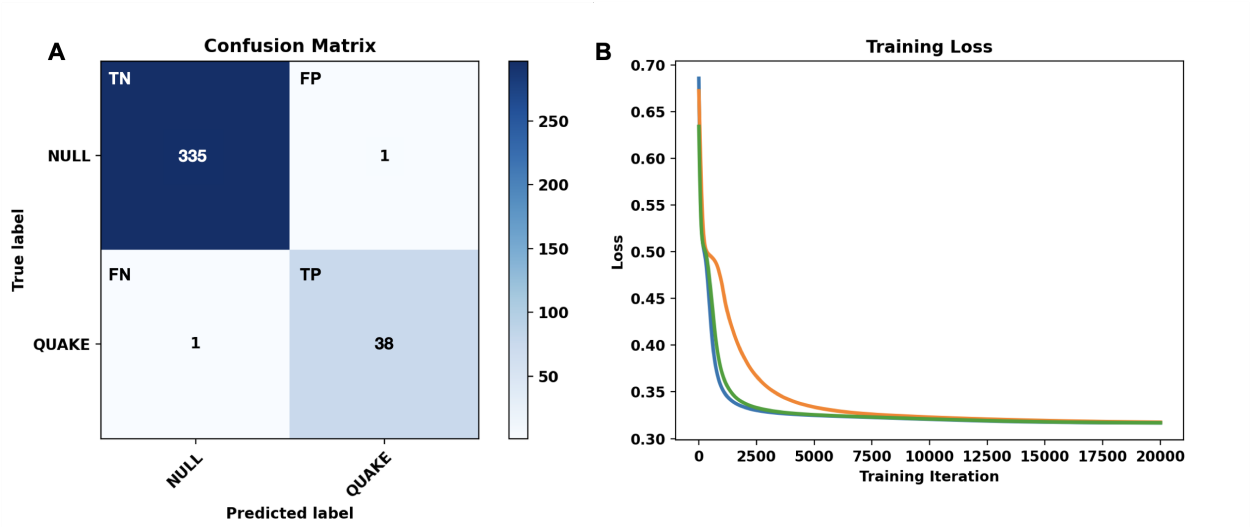

Figure 2.5.2: Panel A: The results of our final neural network against the test set. Panel B: The training loss of each network as a function of training epoch.

2.6 Discussion

2.6.1 Uncertainties and Identification of Unique Events

We evaluate location and origin time estimates for the 2,522 detections predicted by our neural network through bootstrap resampling. That is, for each window detected to contain a seismic event, we recompute 10 SLFs for the same window but randomly exclude 20% of the LCCs between station pairs during each computation. This allows us to collect 10 location and origin time estimates for each detection and average the results to obtain more robust location and origin time estimates. This also allows us to estimate the uncertainty associated with these estimates by computing their standard deviation.

In order to distinguish between uniquely detected events and events detected multiple times within adjacent windows, we use the bootstrap uncertainties computed for each prediction. To distinguish between these two cases, we temporally sort and loop through the 2,522 prediction windows. If the average bootstrap location between one event and the next is within the location uncertainty of either event (which may be different in principle) and if the events are within adjacent windows, then we declare the events to be multiple detections. Otherwise, if events are temporally separated (i.e., not shared by overlapping windows), or if their locations are not within the uncertainty of one another when temporally adjacent, we declare them to be distinct. Proceeding through the prediction windows in this manner, we identify 1,717 unique seismic detections. To obtain location and origin time estimates, as well as their uncertainties for this set, we simply average the locations of multiply detected events and use the original bootstrap statistics for isolated events. Histograms of location and origin time uncertainties for the set of 1,717 unique detections are shown in Figure 2.6.1. Finally, from the set of 1,717 unique detections, we reject events with location uncertainties greater than 5 km and origin time uncertainties greater than 1 s to obtain our final set of 1,192 seismic detections (shown in Figure 2.2.1). Comparing the number of events detected by our network against the number of events in the earthquake catalogue for the same time and region, we detect approximately 5 times as many events.

2.6.2 Limitations

By combining time-delay projection mapping with a neural network classifier to detect and locate earthquakes using a network of stations, this study demonstrates a detection increase

by a factor of 5. However, our application of this joint method has limitations. In the case of SLF construction, the greatest limitation is related to the spatial scale of the network being used. In the case of machine learning, limitations are largely due to the spatiotemporal constraints on earthquakes. Furthermore, these limitations influence one another.

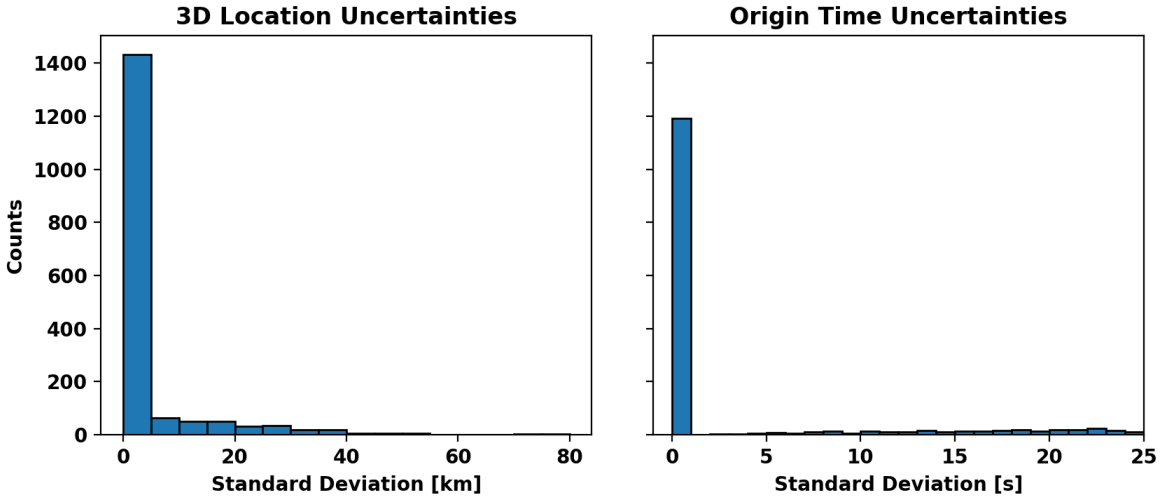

Figure 2.6.1: Uncertainty histograms for the set of 1,717 uniquely detected seismic events in our study.

In principle, SLF computations can be performed using a network of stations at any scale. While several factors decrease the overall precision, many are taken into account by the method. For example, when building kurtosis-based CFs, the final function is convolved with a Gaussian window to account for uncertainties related to the onset of direct P-wave arrivals. Likewise, when calculating the LCC between signals, convolutional smoothing is incorporated in order to account for uncertainties in the velocity model. At large spatial scales, the largest decrease to SLF precision will come from the grid discretization. For example, if we consider event locations to be given by the middle of cells, the intrinsic uncertainty will be at most half the diagonal length of the cell. Additionally, the theoretical minimum grid cell size will be set by the station sampling rates. For instance, station signals sampled at 1 Hz and situated in a medium with homogeneous velocity of 5 km/s will not be able to meaningfully map TTDs onto grids finer than 5 km^3. This is precisely why in this study we chose to use 5-km^3 cells. While in general we are at liberty to make grids as fine as the theoretical minimum, doing so will lead to substantial increases in computational demands when working with signals sampled at higher frequencies, especially in conjunction with large spatial scales. Therefore, when creating SLFs over large spatial networks, there will necessarily be a trade-off between precision and computational feasibility. In our study, the choice of grid size corresponds to an intrinsic minimum SLF location error of 4.33 km

(i.e., assuming perfect knowledge of the relevant velocity structure).

Temporally contiguous seismic data sets are inherently imbalanced due to the fact that earthquakes are, by nature, spatiotemporally constrained. In ideal supervised machine learning problems, one has access to high-quality data sets that can be used to create large training sets. However, the spatiotemporal constraints on earthquakes, in conjunction with incomplete catalogues, make it difficult to acquire large numbers of earthquake examples. Note that by using a machine learning classifier to detect events within a specific search volume, we have implicitly restricted which earthquakes can be used as training examples. Therefore, although in other machine learning contexts one may increase the search volume surrounding a network, either spatially or temporally, to ensure a sufficient number of earthquake examples are acquired for training, these options may not be possible in our case. For example, many arrays are temporary deployments and simply do not record sufficient numbers of regional earthquakes within a given volume to build a usable training set. Another option would ordinarily be to expand the search volume to compensate; however, this then increases the computational demands on constructing SLFs. While it is possible that classifiers trained on one network, in one region, may generalize to other networks and other regions, we suspect that the training process as described in this work must be performed on a case by case basis, especially when the intended use is related to a specific source type.

Finally, we note that the pattern of detected and located events does not appear to follow the long-term pattern of seismicity in this region. However, we speculate that many of the detected events may represent nonvolcanic tremor, which are better expressed at periods of 0.1–1 s, encompassing the spectral features used in our neural network model. LFEs have been reported in northern California during this time period over a similar geographical footprint (Idehara et al., 2014), although no catalogue exists for nonvolcanic tremor to confirm this hypothesis. More work is required to substantiate this claim.

2.7 Conclusions

The main goal of this study was to combine time-delay projection mapping with a supervised machine learning classifier in order to automatically detect earthquakes within a large data set. To accomplish this goal, we partitioned 4 months of data recorded by a group of 76 stations in northern California, into 234,240, 3-min long time segments with 75% overlap. At each time segment, we computed TDE functions between 2,850 station pairs and then used these TDE functions to create a large set of SLFs. In order to train a machine learning classifier, we use four of the 14 features we compute for each window, two of which are derived from SLFs and two of which are derived from raw waveform data. Rather than train a single classifier, we train an ensemble of 15 neural network classifiers and choose the final classifier by selecting the network which demonstrates the best performance against the test set. After analyzing the predictions from our trained classifier, as well as their location errors, we identify and locate a total of 1,192 events in our data set, which is an increase of 5 times the number of catalogued events.

Who shut up the sea behind doors when it burst forth from the womb, when I made the clouds its garment and wrapped it in thick darkness, when I fixed limits for it and set its doors and bars in place

Job 38:8-10

3

Article: Probabilistic Inversion of Seafloor Compliance for Oceanic Crustal Shear Velocity Structure Using Mixture Density Neural Networks

Contents

3.1 Abstract

Measurements of various physical properties of oceanic sediment and crustal structures provide insight into a number of geologic and geophysical processes. In particular, knowledge of the shear-wave velocity (V_S) structure of marine sediments and oceanic crust has wide ranging implications from geotechnical engineering projects to seismic mantle tomography studies. In this study we propose a novel approach to non-linearly invert compliance signals recorded by co-located ocean-bottom seismometers and high-sample-rate pressure gauges for shallow oceanic shear-wave velocity structure. The inversion method is based on a type of machine learning neural network known as a mixture density neural network. We demonstrate the effectiveness of the mixture density neural network method on synthetic models with a fixed deployment depth of 2015 m and show that among 30,000 test models, the inverted shear-wave velocity profiles achieve an average error of 0.025 km/s. We then apply the method to observed data recorded by a broadband ocean-bottom station in the Lau basin, for which a V_S profile was estimated using Monte Carlo sampling methods. Using the mixture density network approach, we validate the method by showing that our V_S profile is in excellent agreement with the previous result. Finally, we argue that the mixture density network approach to compliance inversion is advantageous over other compliance inversion methods because it is faster and allows for standardized measurements.

3.2 Introduction

Accurate measurements of the physical properties (i.e. seismic velocity, density, temperature, thickness, porosity, conductivity, etc.) of oceanic sediment and crustal structures are critical for understanding a variety of geologic and geophysical processes. Measurements of sediment thicknesses and their seismic velocities can constrain marine sedimentation rates, which improve our understanding of paleoclimate and tectonic uplift (Agius et al., 2018). Near surface seismic parameters have provided insight into hotspot volcanism (Doran & Laske, 2019), hydrothermal fluid circulation and crustal formation (Crawford et al., 1991), and the study of oceanic gravity waves (Yamamoto & Torii, 1986). These parameters also inform geotechnical engineering projects (Yamamoto & Torii, 1986) and hazard assessment (Ruan et al., 2014). Furthermore, accurate profiles of shallow oceanic crustal structure have direct bearing on the measurement and interpretation of gravity anomaly residuals (Herceg et al., 2015) and for resolving tradeoffs between crustal structure and deeper anomalies in mantle tomographic models (Marone & Romanowicz, 2007), even at long periods (Montagner & Jobert, 1988).

The shear-wave velocity structure (V_S) of oceanic sediments is of particular interest in marine seismology since, due to their typically low V_S values (Hamilton, 1971), sediments strongly affect shear wave traveltime measurements recorded by ocean-bottom seismometers (OBSs). Additionally, the elastic properties of sediments, on which V_S depends, are also responsible for large site effects that bias seismic amplitudes recorded by OBSs. Therefore, V_S is also required for seismic studies of oceanic structures that rely upon seismic amplitude information (Ruan et al., 2014).

Both active and passive methods exist to measure seismic velocities within oceanic structures. However, since active methods involve the excitation of seismic energy through the use of explosives or air gun shots (e.g. Sauter et al., 1986; Davy et al., 2020), and because subsurface shear-wave energy is difficult to excite acoustically (Sauter et al., 1986; Whitmarsh & Miles, 1991), direct measurements of V_S via active methods are difficult. A passive technique to measure 1D shear modulus profiles within oceanic structures that exploits the phenomenon of seafloor compliance was first pioneered by Yamamoto and Torii (1986). Seafloor compliance is the phenomenon whereby long period ocean infragravity waves propagating along the ocean's surface induce a deformation of the seafloor. Longuet-Higgins (1950) was the first to provide a physical mechanism for this process, which involves non-linear interactions between wind-generated waves (Ardhuin et al., 2014). Such infragravity waves typically propagate in the open ocean with wavelengths up to tens of kilometers, periods of several minutes, and

with wave height displacements ranging from millimeters to centimeters (Aucan & Ardhuin, 2013). Yamamoto and Torii (1986) inferred 1D, layered shear modulus profiles using a linearized inversion of compliance signals recorded by shallow-water OBSs. Since their original work, the most significant developments to the method have been; (1) Its adaptation to deep sites (deployment depths greater than 1000 m) and for V_S rather than shear modulus (Crawford et al., 1991); (2) Its extension to the 2D case involving laterally varying, layered structures (Crawford et al., 1998); (3) Its adaptation for the case when seafloor deformation is induced by Rayleigh waves rather than ocean infragravity waves, which allows the method to be used in a different frequency band (Ruan et al., 2014; Bell et al., 2015). In this study we modify the approach taken by Crawford et al. (1991) and demonstrate the possibility of non-linearly inverting compliance signals for V_S within oceanic sediment and crustal structures through the use of mixture density neural networks (MDNs). The motivation for pursuing MDN inversion over other inverse methods is that MDN inversion is often faster than both linear and non-linear methods (see for example Earp et al., 2020), and allows for easily standardized measurements between researchers. The latter point comes from the fact that even the most sophisticated of trained neural networks are entirely specifiable by only thousands or millions of numbers, which amount to mere kilobytes of data. Thus, neural networks are easily shareable, and because they operate in a deterministic manner, they produce repeatable measurements.

3.3 Theory

3.3.1 The Forward Problem

The compliance $\xi(\omega)$ of a uniform half-space as a function of angular frequency ω due to the forcing caused by ocean infragravity waves was first derived by Sorrells and Goforth (1973) and is given by

$$\xi(\omega) = -\frac{1}{k(\omega)}\left(\frac{V_P^2}{2\rho V_S^2(V_P^2 - V_S^2)}\right), \tag{3.1}$$

where $k(\omega)$ is the wavenumber of the ocean infragravity waves, V_P is the P-wave velocity, V_S is the shear-wave velocity, and ρ is the density of the medium. Following Crawford et al. (1991), we prefer to work with normalized compliance $\eta(\omega)$ in which the filtering effect of ocean infragravity waves is removed

$$\eta(\omega) = k(\omega)\xi(\omega) = -\frac{V_P^2}{2\rho V_S^2(V_P^2 - V_S^2)}. \tag{3.2}$$

The normalized compliance of a 1D layered Earth model described by $V_P(z)$, $V_S(z)$, and $\rho(z)$ can be forward computed numerically by applying the matrix-propagator method (Aki & Richards, 2002) to equation 3.2.

3.3.2 Compliance Measurement

Normalized compliance signals measured by OBSs are computed using the complex vertical displacement $Z(\omega)$ and pressure $P(\omega)$ spectra of these instruments as (Crawford, 2004)

$$\eta(\omega) = k(\omega)\gamma_{PZ}(\omega)\sqrt{\frac{|Z(\omega)|}{|P(\omega)|}}, \tag{3.3}$$

where the coherence between the pressure and vertical displacement spectra $\gamma_{PZ}(\omega)$ is given by

$$\gamma_{PZ}(\omega) = \sqrt{\frac{|C_{PZ}(\omega)|^2}{C_{PP}(\omega)C_{ZZ}(\omega)}}, \tag{3.4}$$

and $C_{XY}(\omega)$ indicates the cross-spectral density between signals X and Y or the auto-spectral density when $X = Y$. Given a normalized compliance signal measured by an OBS, we wish to predict the most probable range of structures (V_P, V_S, and ρ) that correspond to that signal.

3.3.3 Frequency Considerations

The compliance response of the seafloor to ocean infragravity wave forcing is sensitive to different depth ranges at different frequencies. This is analogous to how surface waves at different frequencies have different sensitivities to structures at different depths. However, unlike surface wave dispersion measurements, compliance measurements are limited by the coherence observed between the pressure and vertical channels. Furthermore, not all ocean infragravity waves are physically capable of generating pressure fluctuations on the seafloor. The amplitude P_B of the pressure signal generated on the seafloor at depth H due to an infragravity wave with displacement height ζ, wavenumber k, and wavelength λ is given by (Webb et al., 1991)

$$P_B = \frac{\rho_w g \zeta}{cosh(kH)} = \frac{\rho_w g \zeta}{cosh(2\pi H/\lambda)}, \tag{3.5}$$

where ρ_w is the water density and g is the gravitational acceleration. Critically, P_B depends on the water depth H, and the only infragravity waves that will produce measurable seafloor deformation are those with wavelengths greater than or equal to the water depth (Figure 3.3.1). Moreover, using the dispersion relation for ocean infragravity waves (Apel, 1987)

$$\omega^2 = gk \cdot tanh(kH) \tag{3.6}$$

and the requirement that $\lambda \geq H$ in order to produce compliance effects, it can be shown that the maximum frequency f_c at which infragravity waves will produce measurable compliance signals is

$$f_c \approx \sqrt{\frac{g}{2\pi H}}. \tag{3.7}$$

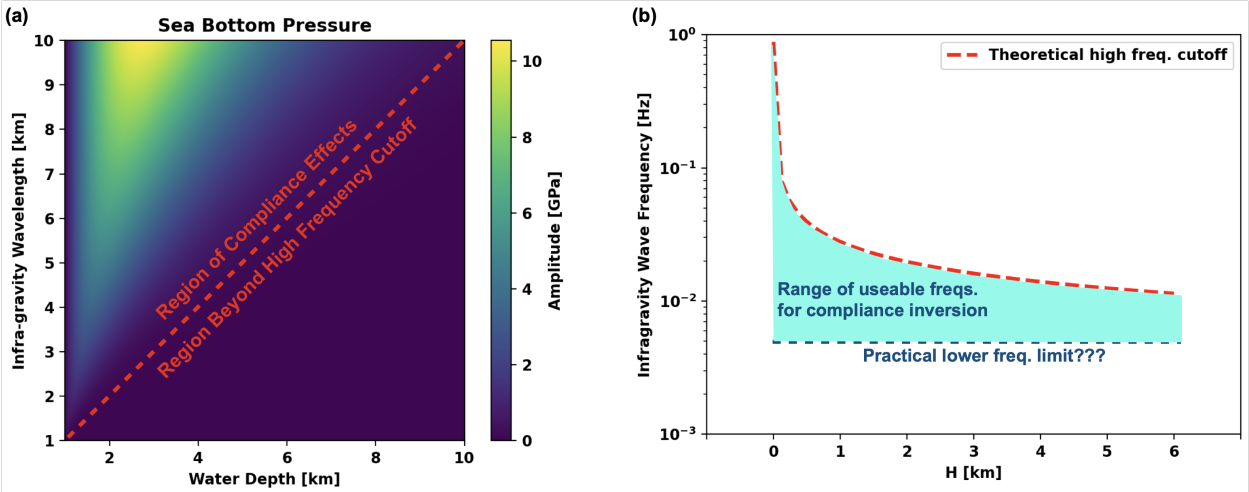

Figure 3.3.1: (a) Sea bottom pressure due to ocean infragravity waves as a function of water depth and wavelength. The region of observable compliance effects is clearly demarcated. (b) Usable frequency band for compliance inversion as a function of OBS deployment depth H.

Thus, the frequency domain over which compliance signals can be inverted is fundamentally limited by the station deployment depth H and the pressure-vertical coherence $\gamma_{PZ}(\omega)$. While there is no theoretical lower frequency bound f_l at which compliance effects can be observed, and while ocean infragravity waves with periods as large as 1000 s are possible (Aucan & Ardhuin, 2013), practical limits are set by instrument sensitivities (Doran & Laske, 2019). We further speculate that the physical mechanism generating ocean surface waves at periods beyond 1000 s and wavelengths greater than 10 km changes so that the preceding analysis is likely no longer relevant (i.e. equation 3.6 no longer holds). Examples of normalized compliance and pressure-vertical coherence functions computed for OBS A02W of the Eastern Lau Spreading Center Seismic Experiment are shown in Figure 3.3.2. The functions shown in Figure 3.3.2 were computed from an ensemble of 75 days of hour-long noise recordings, using the ATaCR package (Janiszewski et al., 2019a; Audet & Janiszewski, 2020).

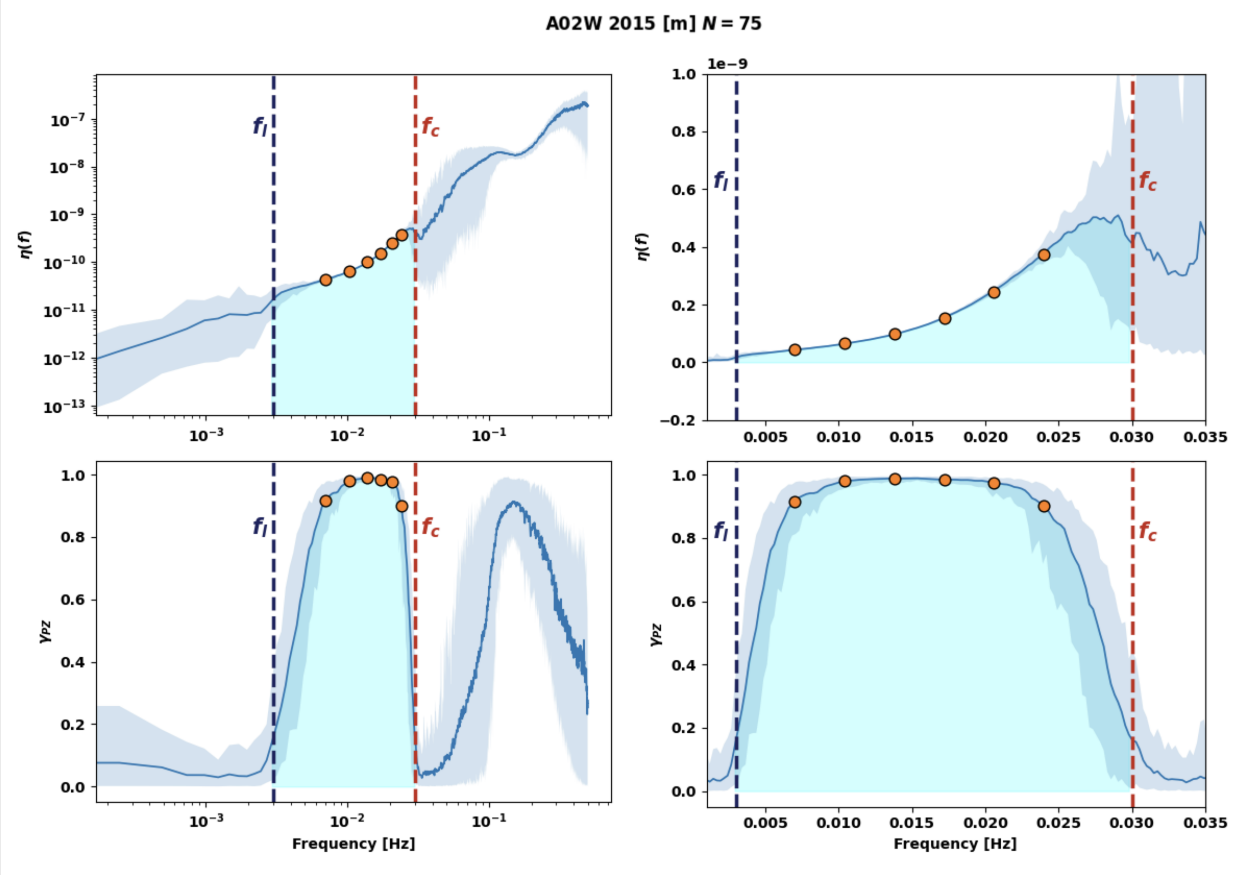

Figure 3.3.2: Normalized compliance (top) and pressure-vertical coherence (bottom) measured for OBS A02W of the Eastern Lau Spreading Center Seismic Experiment, deployed at a depth of 2015 m, and computed from an ensemble of 75 days of hour-long noise recordings. The blue shaded regions denote 95% confidence intervals. The cyan shaded regions denote the compliance frequency band. The theoretical high frequency cutoff f_c and empirical low frequency cutoff f_l are denoted by the red and blue vertical dashed lines, respectively. Signals on the left have been plotted in log-frequency space. Signals on the right have been centered on the compliance frequency band and plotted in linear-frequency space. The orange circles denote the compliance and coherence values used in the inversion.

3.4 Neural Network Inversion

3.4.1 Standard Networks

The adoption of machine learning techniques as tools for solving problems in seismology, particularly neural networks, has had a significant impact on the field (for a recent review of machine learning in the geosciences see Bergen et al., 2019). Originally, neural networks were intended as models of interconnected neurons in biological brains (Bishop, 1995; Valentine & Woodhouse, 2010). A single neuron, the fundamental unit of a neural network, operates

by applying some known function $f(\cdot)$ (referred to as an activation function) to a number of inputs to produce a single output. Several such neurons are thus connected in a given architecture to form a network (Figure 3.4.1). The simplest class of neural networks, known as multi-layer perceptrons (MLPs), are defined as feed-forward networks which have sigmoidal or threshold activation functions (Bishop, 1995). Feed-forward networks refer to networks in which the information processing flows unilaterally (i.e., this class of networks excludes feedback elements such as in recurrent neural networks). Popular activation functions originally included the standardized logistic function $L(x) = 1/(1 + e^{-x})$ or the hyperbolic tangent $tanh(x)$, and were biologically motivated. However, the rectified linear unit (ReLU) activation function, defined as $max(0, x)$ has become the predominant activation function used in most neural networks due to its superior performance over a wider range of problem types (Glorot et al., 2011; Ramachandran et al., 2017).

An example of a simple, three-layer MLP is shown in Figure 3.4.1. The first layer in such a network is called the input layer, the final layer is called the output layer, and any layers between the input and output layers are referred to as hidden layers. The particular MLP in Figure 3.4.1 takes two features as input as a vector $\mathbf{X} = (x_1, x_2)^T$. The activation of neuron i in the hidden layer, a_i, is computed by applying the activation function $f(\cdot)$ to the linear combination of weights Θ_{ij} and inputs from the previous layer ($\mathbf{X}$ in this case). The final output (equivalently the activation in the final layer) is similarly computed from the hidden layer immediately before it. Training a neural network refers to the process of randomly initializing network weights and then pushing inputs with known outputs through the network. Once the output for a given input is computed by the network, a suitably chosen objective function is used to compute the *loss* (the misfit) between the true value of the output and the result computed by the network. In a process known as back-propagation, the network then uses a gradient-descent style algorithm to make adjustments to its weights in order to reduce the error. The training process is then run for a large number of iterations until the network is sufficiently trained.

The utility of neural networks comes from the fact that they can be used to learn arbitrarily complex functions from $\mathbb{R}^m$ to $\mathbb{R}^n$ given a set of examples consisting of inputs with known outputs (Lapedes & Farber, 1988; Valentine & Woodhouse, 2010). Recent applications of neural networks in seismology include signal versus noise discrimination (Meier et al., 2019), automatic arrival-time picking (W. Zhu & Beroza, 2019), and automatic detection and location of seismic events (Mosher & Audet, 2020).

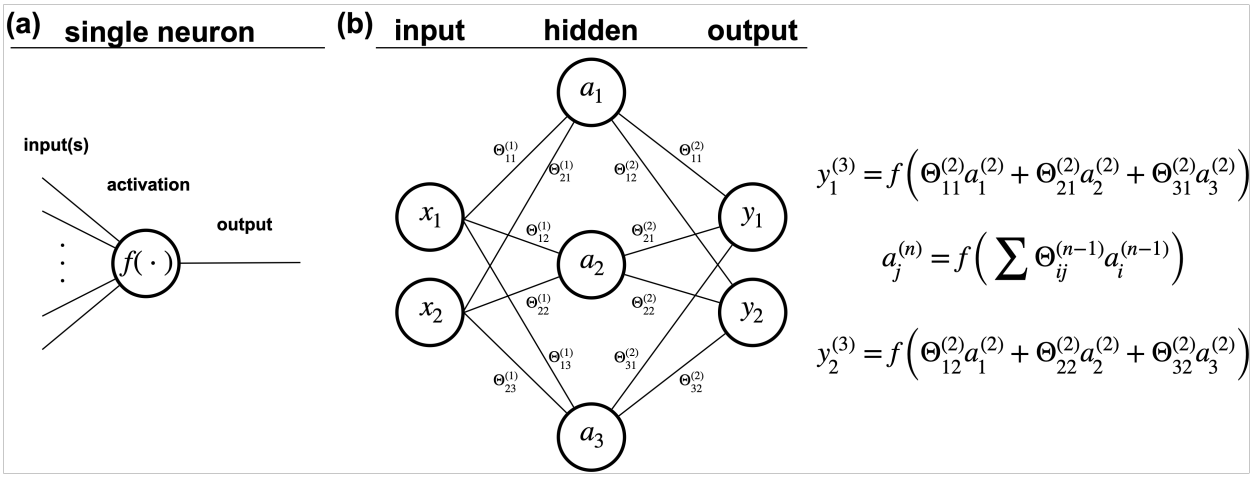

Figure 3.4.1: (a) A graphical representation of a single neuron, the fundamental unit of a neural network. The neuron takes an input vector and applies a known activation function $f(\cdot)$ to generate a single output. (b) A graphical representation of a simple 3-layer MLP. The input to this network is a 2D vector $\mathbf{X} = (x_1, x_2)^T$. The output of the network is the 2D vector $\mathbf{Y} = (y_1, y_2)^T$. The j^{th} activation in the n^{th} layer, $a_j^{(n)}$, is computed by applying the activation function $f(\cdot)$ to the linear combination of weights and activations from the previous layer ($\Theta_{ij}^{(n-1)}$ and $a_i^{(n-1)}$, respectively).

3.4.2 Mixture Density Networks

Mixture density networks were first devised by Bishop (1994). Whereas standard neural networks learn to map a vector from $\mathbb{R}^m$ to $\mathbb{R}^n$, MDNs learn to map a vector from $\mathbb{R}^m$ to an n-dimensional conditional probability distribution. Moreover, the probability distribution learned by the MDN is not restricted to be Gaussian, rather, MDNs learn arbitrary probability distributions by parameterizing them as Gaussian mixture models (GMMs). Mathematically, a multi-dimensional conditional probability distribution parameterized as a GMM can be expressed as (Bishop, 2006)

$$P(\mathbf{t}|\mathbf{X}) = \sum_{k=1}^{K} \Pi_k(\mathbf{X}) \mathcal{N}(\mathbf{t}|\mu_k(\mathbf{X}), \sigma_k(\mathbf{X})), \tag{3.8}$$

where the probability $P(\cdot)$ of observing target vector $\mathbf{t}$, given input vector $\mathbf{X}$, is given by the sum of K n-dimensional parametric Gaussian PDFs $\mathcal{N}(\mathbf{t}|\mu, \sigma)$, each with their own mean $\mu_k(\mathbf{X})$, standard deviation $\sigma_k(\mathbf{X})$, and mixture weight $\Pi_k(\mathbf{X})$. Note that the mixture weights must satisfy the following constraints

$$\sum_{k=1}^{K} \Pi_k(\mathbf{X}) = 1 \qquad 0 \leq \Pi_k(\mathbf{X}) \leq 1. \tag{3.9}$$

A MDN can be understood as a MLP augmented with a final layer whose outputs represent the parameters of an n-dimensional parametric GMM (Figure 3.4.2). In general, if $\mathbf{t}$ is D-dimensional then the final layer of a MDN will contain $3KD - K(D-1)$ units consisting of the means, standard deviations, and weights of each mixture component. To ensure that the units in the final MDN layer correctly represent the components of the GMM, the following operations are applied to the final outputs of the MLP, which we refer to as the vector $\mathbf{Z} = (z_1, z_2, ..., z_i)^T$ (Bishop, 1995, 2006)

$$\Pi_k(\mathbf{X}) = \frac{\exp(Z_k^{\Pi})}{\sum_{l=1}^{K} \exp(Z_l^{\Pi})} \tag{3.10}$$

and

$$\sigma_k(\mathbf{X}) = \exp(Z_k^{\sigma}). \tag{3.11}$$

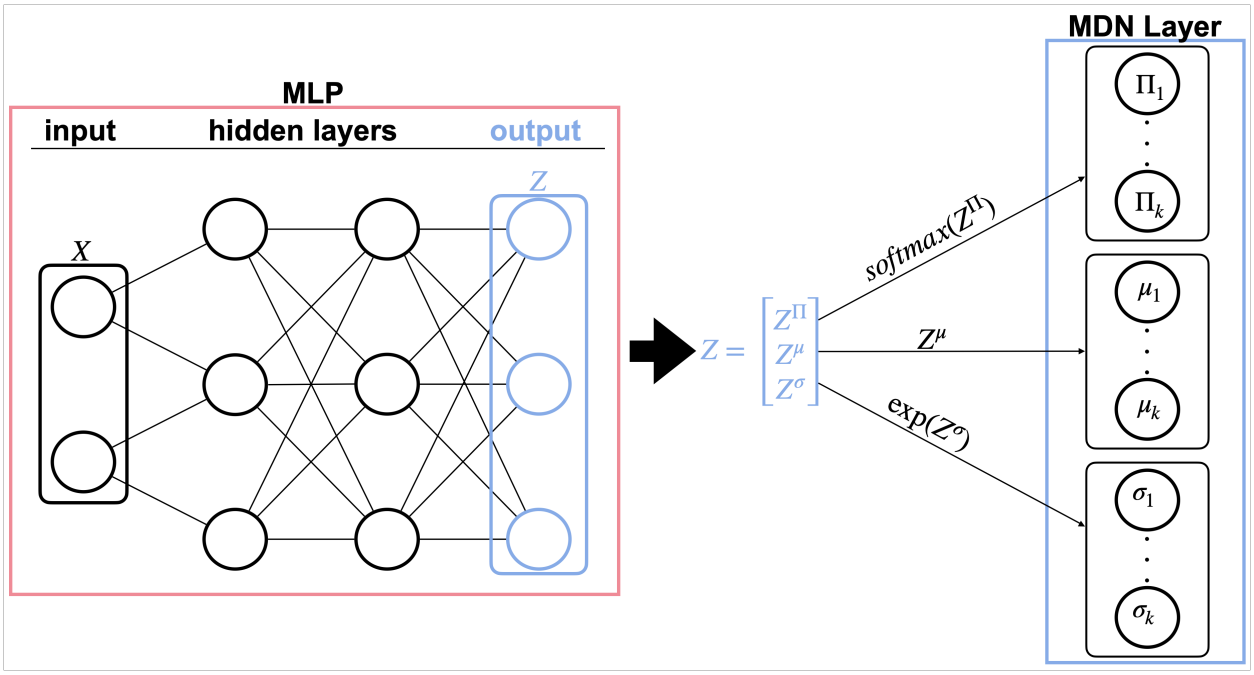

Figure 3.4.2: A graphical representation of a MDN. The entire MDN consists of an MLP augmented with a final MDN layer. The MDN layer applies the transformations indicated to the output $\mathbf{Z}$ of the MLP. The softmax operator is applied to the components of $\mathbf{Z}$ intended to represent GMM mixing coefficients (Π_k). The exponential operator is applied to the components of $\mathbf{Z}$ intended to represent the GMM standard deviations σ_k. No operation is applied to the components of $\mathbf{Z}$ intended to represent the GMM means μ_k.

In equation 3.10 the softmax operation is applied to the components of $\mathbf{Z}$ intended to represent the GMM weights (denoted as $\mathbf{Z}^{\Pi}$) to ensure the weights satisfy the constraints in equation 3.9. In equation 3.11 the exponential is applied to the components of $\mathbf{Z}$ intended to represent the GMM standard deviations (denoted as $\mathbf{Z}^{\sigma}$) so as to ensure that $\sigma_k^2(\mathbf{X}) \geq 0$. Finally, the components of $\mathbf{Z}$ intended to represent the GMM means (denoted as $\mathbf{Z}^{\mu}$) can be represented directly by the final MLP network activations, thus

$$\mu_k(\mathbf{X}) = Z_k^{\mu}. \tag{3.12}$$

Typical training protocols for MLPs include the minimization of loss quantified by least-squares or cross-entropy objective functions. However, Bishop (1994) showed that MLPs trained with either of these protocols approximate conditional averages of target data. For the class of problems in which the probability distribution of the target variable is either Gaussian or unimodal, networks trained with such protocols will give meaningful results, however, if the distribution of the target variable is either non-Gaussian or multi-modal then these training protocols are unsuitable (Meier et al., 2007). In the context of MDNs, training is instead accomplished by minimizing the following negative log likelihood function (assuming N independent pieces of data)

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \Pi_k(\mathbf{X}_n, \mathbf{w}) \mathcal{N}(\mathbf{t}_n | \mu_n(\mathbf{X}_n, \mathbf{w}), \sigma_k(\mathbf{X}_n, \mathbf{w})) \right\}, \tag{3.13}$$

where the vector $\mathbf{w}$ consists of the network parameters, and the explicit dependence of Π_k, μ_k, and σ_k on $\mathbf{w}$ has been included (Bishop, 2006). This objective function, along with the GMM approach, allows MDNs to obtain a more complete description of the target data (Bishop, 1994). In the next section we illustrate these points with a toy problem.

3.4.3 A Toy Problem

Consider the problem of predicting output variable x_2 from input variable x_1 for the data set generated by

$$x_2 = \pm\sqrt{x_1 - 1} + 3 + \epsilon, \tag{3.14}$$

where ϵ indicates the addition of a small amount of Gaussian noise (Figure 3.4.3a). The essential features of this problem are that it is (1) non-linear and (2) multi-modal. In other words, given any x_1 we expect two distinct ranges of possibilities for $P(x_2|x_1)$. The result established by Bishop (1994) is that a typical MLP trained with a least-squares or cross-entropy minimization objective will approximate the conditional average of the target data (x_2 in this case). This result is demonstrated in Figure 3.4.3a, in which predictions of a simple MLP trained on the dataset generated by equation 3.14 uniformly approximate the conditional average of the target variable x_2. Thus, the conditional average of x_2 is an incomplete description and the inappropriateness of attempting to train a MLP in such a manner for this style of problem is evident. By contrast, since we know that this dataset is bi-modal, we should be able to train a simple MDN to learn $P(x_2|x_1)$ using $K = 2$ GMM components. Once $P(x_2|x_1)$ is learned, we can then interrogate it for a complete description of the relationship between x_1 and x_2. Samples taken from the probability distribution learned by a MDN trained on the data generated by equation 3.14 are also shown in Figure 3.4.3a. The MDN adequately learns the relationship between x_1 and x_2 and the components of the GMM ($\mu_k(x_1)$, $\Pi_k(x_1)$, and $\sigma_k(x_1)$) learned by the MDN are shown in Figure 3.4.3a and b. With $K = 2$ components in this example, the two mixture components parameterize the upper and lower branches, respectively, of the parabola. In general, the optimal number of mixture components required for a given problem is not known *a priori*. However, MDNs are parsimonious in that they typically assign zero weight to unnecessary mixture coefficients $\Pi_k(\mathbf{X})$ and use as few mixture components as needed (Bishop, 1995). K is rarely required to be larger than 15 for most problems (Meier et al., 2007; de Wit et al., 2013; Earp et al., 2020).

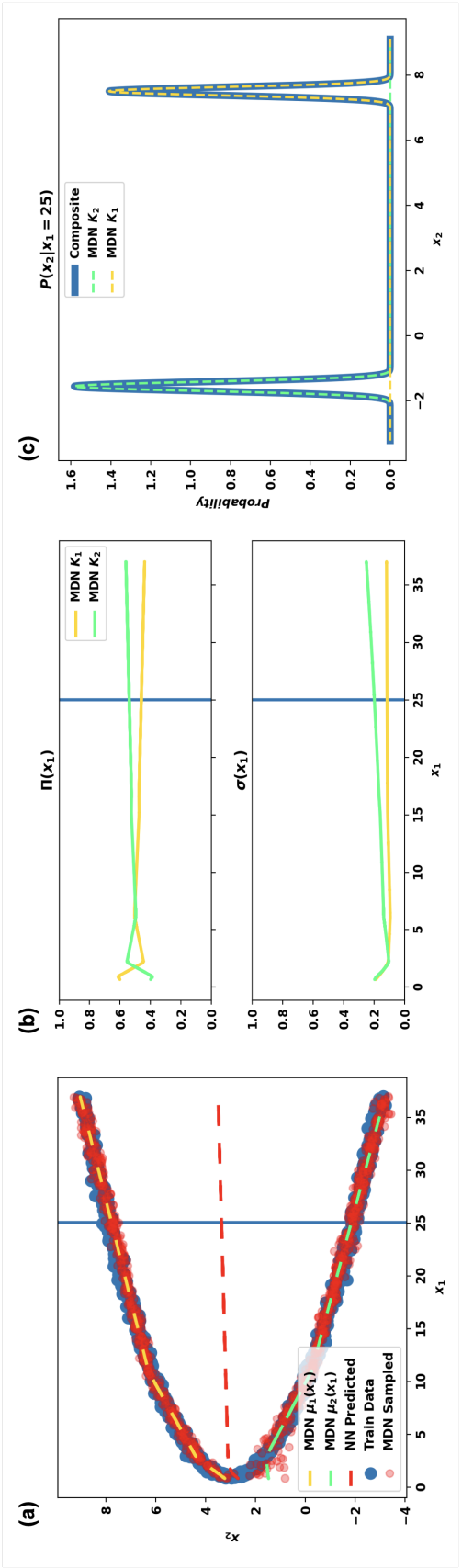

Figure 3.4.3: (a) An example of noisy, multi-modal, quadratic data (the training data), as well as the predictions of a standard neural network, and samples taken from a GMM learned by a MDN, each trained on this data. With $K = 2$ mixture components, the MDN learns to parameterize the i^{th} branch of the parabola using the i^{th} mean $\mu_i(x_1)$ of the GMM. (b) The remaining parameters of the GMM learned by the MDN, namely $\Pi_i(x_1)$ and $\sigma_i(x_1)$. (c) The MDN estimate of the conditional probability $P(x_2|x_1 = 25)$ for the data in (a). The contributions from each mixture component are superimposed on the conditional probability distribution.

3.4.4 Prior Applications of MDNs in Seismology

The motivation for implementing MDNs over MLPs or other types of neural networks is in their superior handling of non-linear inverse problems. Indeed MDNs have been successfully used in this capacity for a number of studies in seismology: Meier et al. (2007) pioneered this technique to invert surface wave data for a model of global crustal thickness; Shahraeeni et al. (2012) predicted porosity, clay content, and water saturation of reservoirs from V_P and V_S; de Wit et al. (2013) trained a MDN to invert P-wave travel time curves for Earth's spherically symmetric V_P structure; de Wit et al. (2014) obtained 1D velocity and density profiles by inverting degree-zero spheroidal mode splitting function measurements with an ensemble of MDNs; Käufl et al. (2016) inverted coseismic displacement observations for point source parameter estimates; and, in a study similar to ours, Earp et al. (2020) used an ensemble of MDNs to invert surface wave dispersion data for shear wave velocity models and their non-linearized uncertainty beneath the Grane field in the Norwegian North Sea.

3.5 Application to Synthetic Data

3.5.1 Validating the Method with a Synthetic Recovery Test

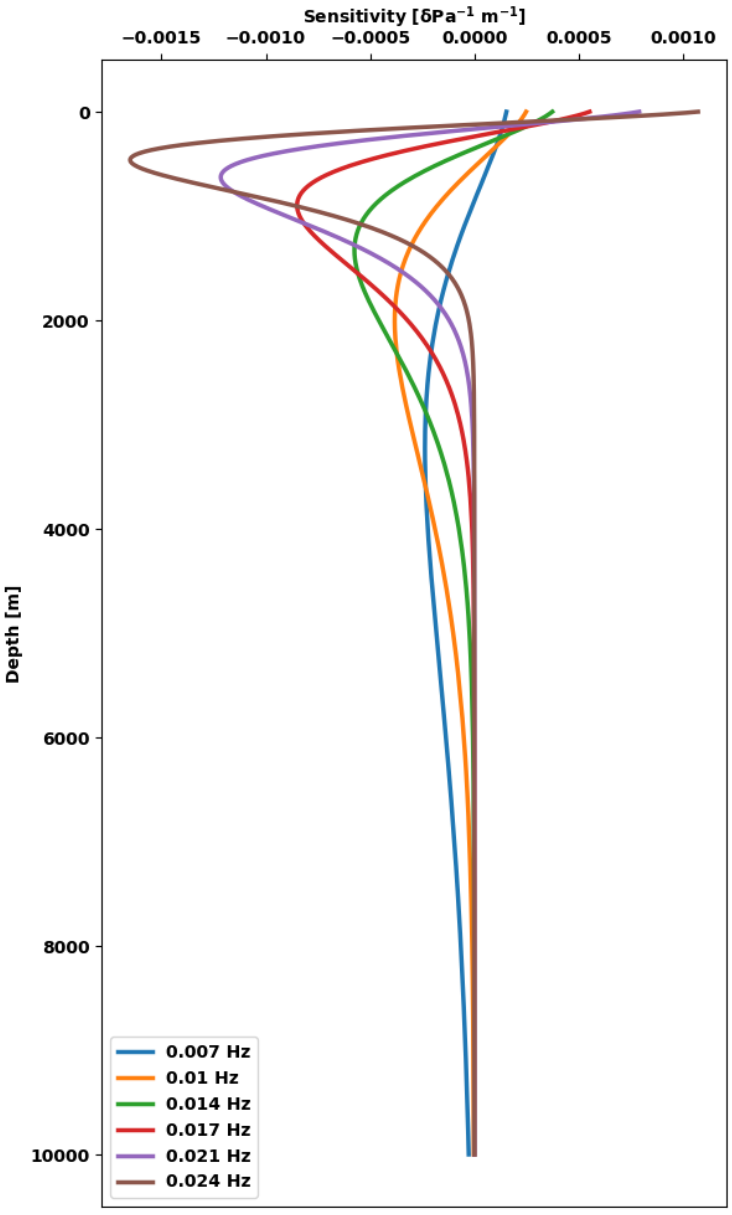

Figure 3.5.1: Theoretical compliance sensitivity kernels computed for the 6 inversion frequencies used in this study for a station deployed at a depth of 2015 m.

To test whether a MDN can invert compliance signals for shallow V_S we must first select a station deployment depth. Recall that the station deployment depth controls the theoretical high frequency limit f_c at which compliance effects will be measurable. Therefore, when modelling compliance signals for an OBS with an arbitrary deployment depth, equation 3.7 may be used to determine f_c, and the low frequency limit f_l may be set according to a particular instrument response. However, because we intend to later invert the compliance signal measured by A02W (Figure 3.3.2), we choose to model synthetic data using the particulars of A02W. Thus, we set the station deployment depth in the synthetic case to be 2015 m, and we determine the range of inversion frequencies available to use by considering the pressure-vertical coherence spectrum of A02W (Figure 3.3.2). Doing so, we choose to invert compliance values measured at 6 frequencies

equally spaced between 0.007 and 0.024 Hz. Having determined our inversion frequencies we then compute theoretical sensitivity kernels for these 6 frequencies in order to get a sense of what structures the inversion will be able to resolve (Figure 3.5.1). Because the theoretical sensitivity kernels inform us that compliance effects for a station deployed at a depth of 2015 m will be most sensitive to earth structure shallower than 2 km, we therefore limit our structural models to a depth of 2 km and include a half-space layer below. Due to this shallow structure focus, we thus choose to parameterize our structural models with the intent of predominantly characterizing oceanic sediment structure.

In many of the previous studies involving MDNs in seismology, structural parameterizations were accommodated using cubic splines, which allow for continuous representations of Earth structures with depth across discontinuities (Meier et al., 2007; de Wit et al., 2013; de Wit et al., 2014). However, in this study we prefer a model parameterization based on cubic Bernstein polynomials. The use of cubic Bernstein polynomials allows for a low number of model parameters (4) and allows for an efficient parameterization of continuous V_S profiles (which are expected in oceanic sediments) without the need to prescribe a layered structure. Furthermore, this representation has optimal stability for a given polynomial order (Farouki & Rajan, 1987; Farouki, 2012). An additional attractive feature of Bernstein polynomials is that, since they sum to unity at all depths, the width of the prior bounds on the coefficients is equivalent to the width of the prior bounds on the geophysical parameter being represented (i.e. $V_S(z)$, Gosselin et al., 2017). Therefore, we generate V_S profiles by randomly selecting cubic Bernstein coefficients from the uniform distribution $U(0.1, 3)$ and we also enforce V_S profiles generated in this manner to be strictly monotonically increasing with depth. Furthermore, we assume a constant V_P profile of 6.0 km/s and a constant density profile of 2.0 g/cm^3 for our structural models. Fixing V_P and ρ helps aid the inversion of V_S, since we do not include these parameters as additional input to the MDN. Although more sophisticated fixed parameterizations could be used for V_P and ρ, compliance is much less sensitive to V_P and ρ at high V_P/V_S (i.e. $V_P/V_S > 2$) (Zha & Webb, 2016). Because V_S is limited to values from 0.1 - 3 km/s in our models, our V_P/V_S is always > 2, and hence, assuming constant values for these quantities is not detrimental.

Although we construct V_S profiles by sampling cubic Bernstein coefficients from uniform distributions, it is important to note that uniform sampling of Bernstein coefficients on a fixed interval does not correspond to uniform sampling of V_S. That being said, the distribution of V_S profiles used in this study adequately covers the range of real compliance signals observed. An example of the distribution of 100,000 V_S profiles generated in the described manner is shown in Figure 3.5.2 along with a single structural model (V_S only).

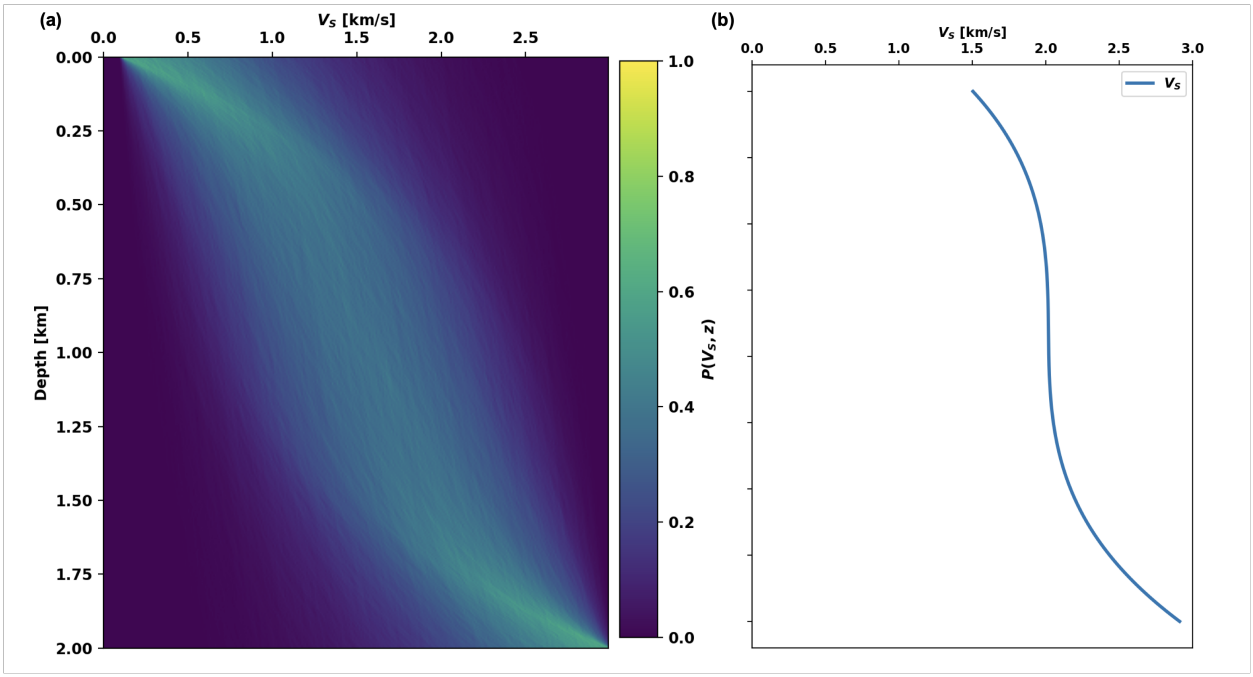

Figure 3.5.2: (a) An example of the prior distribution of V_S models used in this study generated from 100,000 models. (b) A particular V_S profile selected randomly from the prior distribution in (a).

While a higher order polynomial basis could be used to parameterize V_S profiles in principal, we argue that 3^{rd} order (cubic) polynomials are sufficient for parameterizing shallow V_S for two reasons. First, quadratic V_S relations have been determined for marine sediment structures such as those in the Cascadia subduction zone (Bell et al., 2015; Ruan et al., 2014). Second, it is possible to compute the effective number of model parameters M_{eff} required to invert compliance signals at 6 frequencies using the theoretical sensitivity kernels in Figure 3.5.1. Assuming the sensitivity kernels in Figure 3.5.1 as the data kernel $\mathbf{G}$ for a minimum-length inverse problem, we compute M_{eff} by calculating the trace of the model resolution matrix given by $\mathbf{R} = \mathbf{G}^{-\mathbf{g}}\mathbf{G}$, where $\mathbf{G}^{-\mathbf{g}}$ is the generalized inverse. Doing so, we find that as long as we limit our inversion structure to a maximum depth of 2 km, then $M_{eff} = 3.99$, otherwise M_{eff} increases if greater structural depths are considered.

In preparation for training a MDN, we generate 100,000 random structural models to use as a training set and 30,000 models to use as a testing set following the above process. We then use equation 3.2 and the 1D matrix-propagator method (Aki & Richards, 2002; Crawford et al., 1991) to forward model the normalized compliance signals that each of the models would produce at the 6 inversion frequencies. Once the signals have been forward computed, we add random noise to each compliance value using the 95% signal statistics computed for A02W at the appropriate frequencies (Figure 3.3.2). Ensemble averaged compliance signals

such as that computed for A02W typically have low uncertainty and are well constrained at frequencies where γ_{PZ} is large. Thus, the inversion method will perform best for such frequencies.

In other compliance inversion studies, the role of the pressure-vertical coherence has been to simply inform practitioners of the frequency domain over which compliance effects can be observed. Therefore, the multiplication by γ_{PZ} (which is essentially a weighting function) in equation 3.3 is not performed when calculating $\eta(\omega)$ in practice. Instead, the inversion of $\eta(\omega)$ is usually restricted to frequencies where $\gamma_{PZ} \geq 0.8$ (e.g. Doran & Laske, 2019; Zha & Webb, 2016). The reason for this is that if significant noise sources persist on the horizontal OBS components, then even large values of γ_{PZ} may underestimate compliance signals by 40% (Crawford & Singh, 2008). However, Zha and Webb (2016) demonstrate that compliance estimates for frequencies where $\gamma_{PZ} \geq 0.8$ are not significantly biased in this manner. In essence, such approaches end up assuming a pressure-vertical coherence of unity. Rather than follow the threshold approach and assume $\gamma_{PZ} = 1$, however, we prefer to include the coherence multiplication when forward modelling synthetic compliance signals. In this way, we more accurately reproduce the distribution of compliance signals likely to be measured by a given station. In the general synthetic case, this means assuming a coherence function for a hypothetical station and deployment depth. In the current case we assume the coherence function computed for A02W (Figure 3.3.2). The potential impact of such an assumption on the generalizability of the method will be discussed further below.

Because normalized compliance values are typically quite small (values on the order of 10^{-9} Pa^{-1} or lower), rather than work with raw signals, we instead take the log_{10} of our compliance signals and then standardize them before they are fed to the MDN by applying Z-score normalization. This type of feature scaling ensures that the compliance values with respect to each frequency in the training and testing sets have zero mean and unit variance. Note that this type of feature scaling assumes zero co-variance between the compliance values at each frequency being used, which is unrealistic. While in principle the MDN framework can be modified to account for fully co-variant input parameters (Bishop, 2006), MDNs trained under the assumption of a diagonal co-variance matrix have often been found to adequately model target probability distributions even when the input parameters are co-variant (e.g. Meier et al., 2007; de Wit et al., 2013). Moreover, we found that this type of feature scaling provided the best network performance. We refer to these transformed signals to be fed to the MDN as $\hat{\eta}(\omega)$.

We train a MDN to estimate $P(\mathbf{B}|\hat{\eta}(\omega))$ using 5 hidden layers, each with 42 units, and

allowing for $K = 6$ GMM components, where $\mathbf{B}$ is a vector of cubic Bernstein polynomial coefficients (i.e. $\mathbf{B} = (B_0, B_1, B_2, B_3)^T$). Since any given $\mathbf{B}$ in our framework is a 4-dimensional vector, the final MDN layer contains 42 units, and the entire MDN consists of 9,840 tuneable parameters (including bias units). When training neural networks it is best to keep the number of network parameters as low as possible, in order to avoid over-fitting, (Valentine & Woodhouse, 2010). A general rule of thumb is to ensure that there are at least 10 times as many training examples as there are network parameters. In our case this rule was the main principle for deciding upon the network architecture, since the number of hidden layers, as well as the number of units in the hidden layers are not critically important in these types of problems (Meier et al., 2007; de Wit et al., 2013). We confirmed this ourselves by testing several different network architectures, noting that no single architecture significantly outperformed another. To train the MDN, we use mini-batch gradient descent with batch sizes of 32 samples and train the network for a maximum of 1,000 epochs. Rather than train the network for the maximum possible number of epochs, however, we instead employ early stopping, whereupon training ceases if the performance of the network, as measured against a portion of the training set held out for validation, does not improve after 8 epochs. Moreover, we use such a validation set to ensure that the trained MDN will not be prone to over-fitting (Figure 3.5.3). Network training takes approximately 20 minutes on a 2.8 GHz quad-core i7 powered laptop.

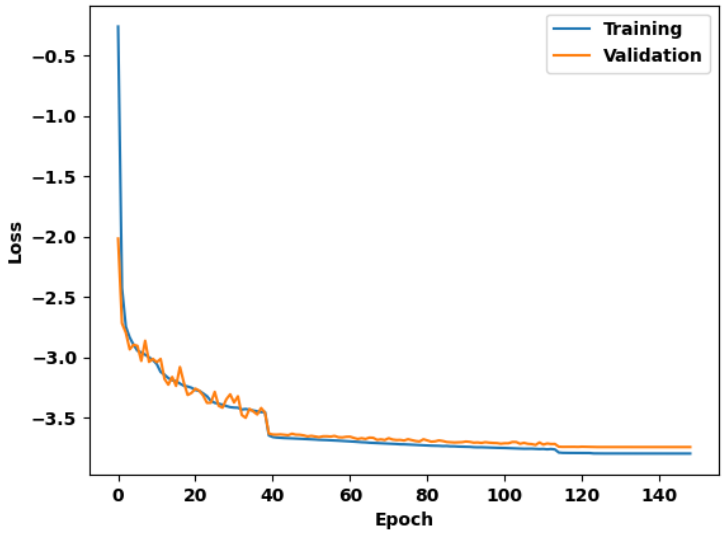

Figure 3.5.3: The training curve of the final trained MDN used in this study. The similar performance of the MDN against the training and validation sets demonstrates that the network is not over-fitting.

Recall that MDNs do not learn a function from one vector space to another, but rather, learn to associate a probability distribution to a vector. In our case we train a MDN to approximate $P(\mathbf{B}|\hat{\eta}(\omega))$, that is, the conditional probability of a set of cubic Bernstein coefficients given an observed compliance signal. Thus, we must decide on a sampling strategy

in order to determine *the* inverted V_S profile estimated from $P(\mathbf{B}|\hat{\eta}(\omega))$. We determine the final inversion result by taking 1000 samples of Bernstein coefficients from $P(\mathbf{B}|\hat{\eta}(\omega))$. We then use the sampled coefficients to construct 1000 V_S profiles and take the mean of these samples as the final inversion result for a particular compliance signal. The process is illustrated in Figure 3.5.4, in which we show a single inversion result for one of our test models. This sampling approach also allows us to compute the 95% confidence intervals on our V_S estimates.

3.5.2 MDN Performance Assessment

To quantitatively assess the overall performance of the MDN against the test set, we compute the following error metrics. First, we compute the L_2-norm between mean predicted Bernstein coefficients and the true Bernstein coefficients for each test model. Doing so, we can assess the overall coefficient error of the MDN across all test models, as well as the error associated with each individual Bernstein coefficient. Second, we compute the absolute difference between the true and inverted V_S profiles across all test models. As with the coefficient errors, the errors associated with the velocity profiles can be assessed as a function of depth or depth-averaged. In Figure 3.5.5 we show these metrics computed for each of the 30,000 test models we created. Looking at Figure 3.5.5, we see that the MDN trained on the synthetic structures has a depth-averaged absolute velocity error of 0.025 km/s and an average coefficient error of 0.2. Moreover, the MDN achieves its lowest errors when predicting the Bernstein coefficients B_0 and B_3, whereas the larger spreads for coefficients B_1 and B_2 are likely indicative of model parameter tradeoffs. Figure 3.5.5 also shows that, as a function of depth, the inversion results obtained from the MDN have the largest errors at depths between 0-0.25 and 1.25-1.5 km, which correlate spatially with areas of low and decreasing compliance sensitivity (Figure 3.5.1). Nevertheless, the metrics in Figure 3.5.5 demonstrate the effectiveness of using MDNs to invert compliance signals for V_S in oceanic sediments and the shallow crust.

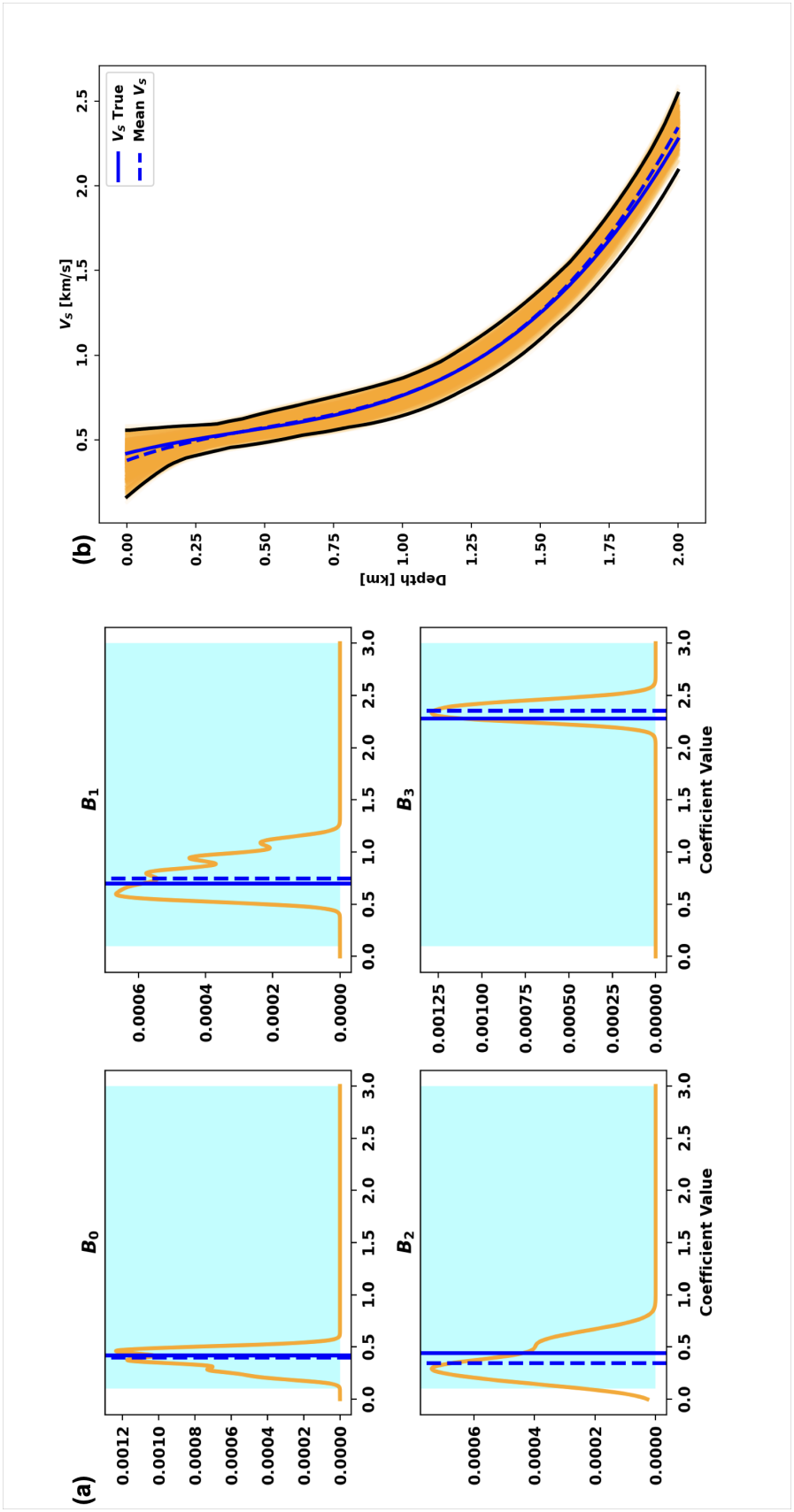

Figure 3.5.4: A single synthetic inversion example. (a) The posterior distributions of each inverted cubic Bernstein polynomial coefficient (orange). Also shown are the prior distributions of Bernstein coefficients (the shaded cyan boxes indicating $U(0.1, 3.0)$). True Bernstein coefficients are indicated by solid vertical blue lines. The recovered coefficients are indicated by dashed vertical blue lines (the means of 1000 samples taken from $P(\mathbf{B}|\hat{\eta}(\omega))$. (b) The V_S profiles generated from the 1000 sets of Bernstein coefficients, sampled from the GMM learned by the MDN (orange). The true V_S profile is shown in solid blue. The mean profile (our estimate of V_S) is shown in dashed blue. The 95% confidence bounds are plotted in black.

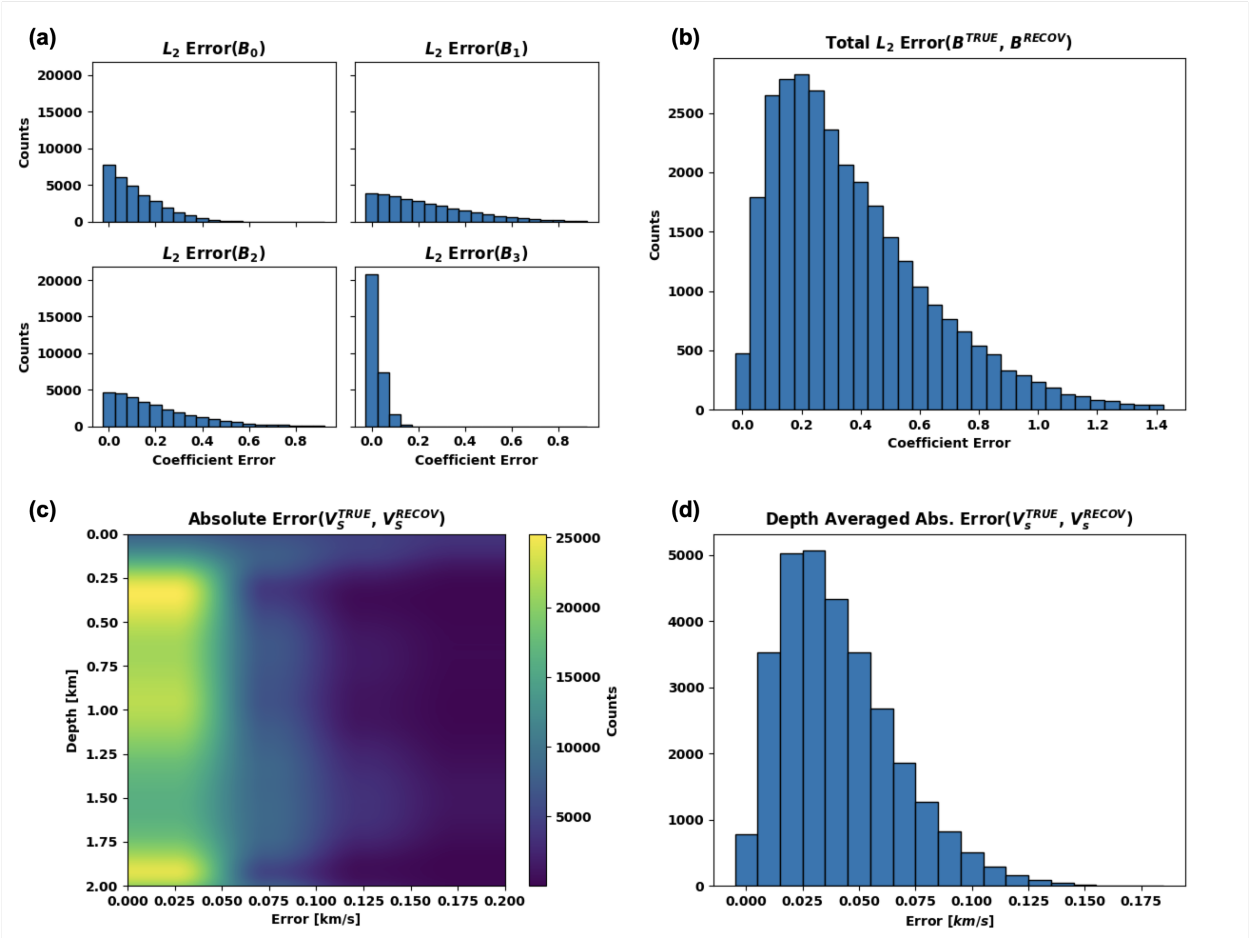

Figure 3.5.5: Compiled error metrics computed for each of the 30,000 synthetic test models. (a) Histograms of the L_2 error between each true and recovered Bernstein coefficient individually. (b) Histogram of the overall L_2 error between true and recovered Bernstein coefficients. (c) The 2D histogram of the absolute error between true and recovered V_S profiles as a function of error and depth. (d) The depth-averaged absolute error between true and recovered V_S profiles.

3.6 Results and Discussion

3.6.1 A02W

In this section, we apply the MDN technique to invert the compliance signal recorded by OBS station A02W of the Eastern Lau Spreading Center Seismic Experiment. This station offers a convenient benchmark result since it was recently analyzed in detail by Zha and Webb (2016) who used a Markov Chain Monte Carlo (MCMC) method to non-linearly invert compliance signals for V_S. As described previously, rather than use the deployment depth of A02W (2015 m) to determine the highest frequency available to us for compliance inversion, and decide the lower limit by considering the instrument response, we do so by considering the pressure-vertical coherence of the station (Figure 3.3.2). Furthermore, it should be noted that we invert compliance using a slightly different frequency band than Zha and Webb (2016), namely 0.007 to 0.024 Hz, rather than 0.006 to 0.02 Hz. Additionally, while we parameterize our synthetic models assuming the same density that Zha and Webb (2016) use for sediment layers in their inversion (ρ=2.0 g/cm^3), Zha and Webb (2016) parameterize V_P using a relation obtained by Dunn et al. (2013), whereas we assume a constant V_P profile of 6.0 km/s. Because our model parameterization remains unchanged from the synthetic scenario, we are able to invert the real compliance signal measured by A02W using the network trained previously on the synthetic data. The inversion result is shown in Figure 3.6.1 as well as the fit between the predicted compliance signal (i.e. that computed from the MDN inverted V_S profile) and the measured signal. Figure 3.6.1 demonstrates an excellent agreement with the inversion result obtained by (Zha & Webb, 2016) for A02W (insofar as the results share the same structural domain), and validates the method. Furthermore, the predicted compliance signal agrees quite well with the measured signal, except for the highest frequency value. This high-frequency discrepancy is likely reflected in the absolute depth-error histogram of Figure 3.5.5c. Otherwise, all predicted compliance values are within the 95% confidence levels of the measured signal.

3.6.2 Limitations

The MDN approach to inverting compliance signals recorded by OBSs for V_S depends principally on the following aspects; the station deployment depth, the measured or assumed pressure-vertical coherence function, the type of structural parameterization, and the noise

assumptions implicit in the training process. Each of these aspects have important implications on the ability of the method to generalize or its limitations, which we in turn discuss here.

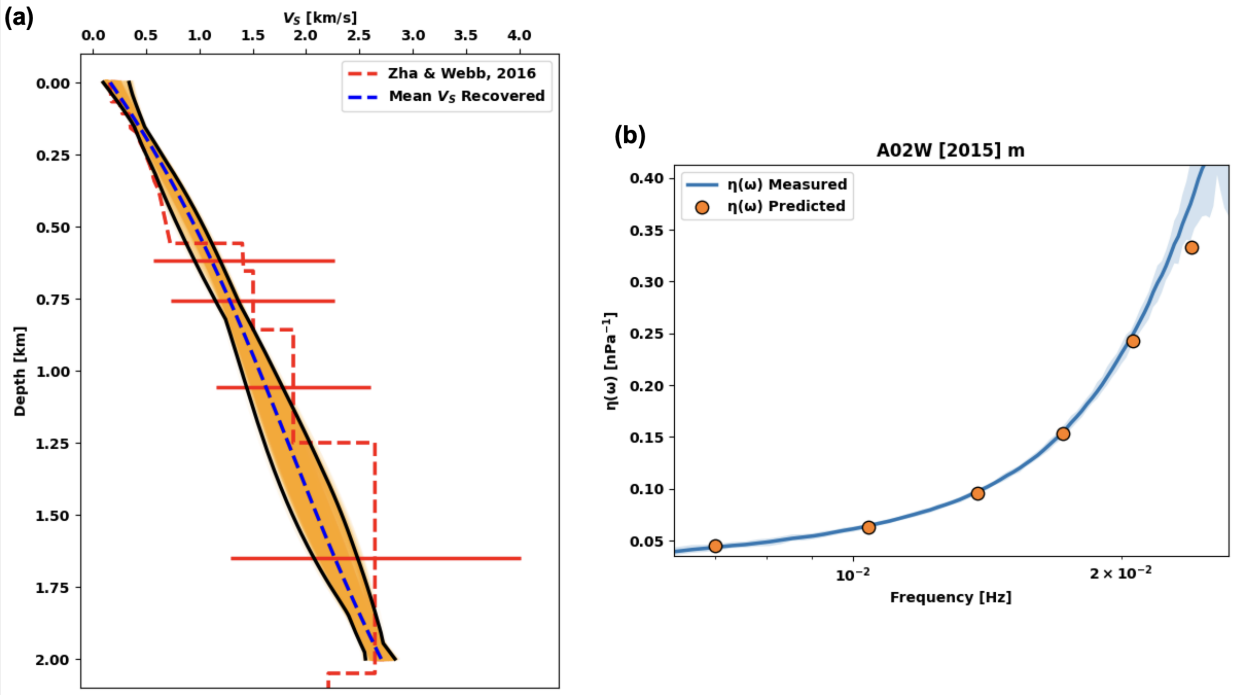

Figure 3.6.1: Inversion of the normalized compliance signal measured by A02W. (a) The V_S profiles generated from the 1000 sets of Bernstein coefficients, sampled from the GMM learned by the MDN (orange). The mean profile (our estimate of V_S) is shown in dashed blue. The 95% confidence bounds are plotted in black. The inversion result obtained by Zha and Webb (2016) is plotted in dashed red along with their corresponding confidence bounds (solid red bars). (b) A comparison of the normalized compliance signal predicted by our result in (a), shown as orange circles, against the signal measured by A02W. The 95% confidence bounds for the measured signal are shown in shaded blue. All values of the predicted signal are within the confidence bounds of the measured signal, apart from the highest frequency value.

Because seafloor compliance is a depth-dependent signal, rather than training a single MDN to invert compliance signals recorded by any OBS, one must train different MDNs to invert compliance signals recorded by OBSs deployed at different depths. Although it would likely be unnecessary to train separate MDNs for stations whose deployment depths differ only on the order of meters, an important test to perform in future analyses will be to determine at what point a difference in deployment depth between stations necessitates the training of a separate MDN. For example, consider that it is found that MDNs trained for a given depth can also satisfactorily invert signals recorded by stations whose deployment depths differ by up to 50 meters. Then the physically possible deployment depths on

Earth could be binned into 50 m intervals, and a suite of MDNs could be trained to invert compliance signals for any depth. In this way the method could be generalized to various depths.

In addition to deployment depth, real compliance signals observed by OBSs depend on the pressure-vertical coherence at a given station. For the purposes of generalizing the method, this is a more subtle issue than the deployment depth as it is not entirely clear whether pressure-vertical coherence is a purely depth dependent function, or additionally, depends upon instrument design or even the localized deployment environment. We speculate that by conducting a statistical analysis of coherence functions observed by instruments deployed at various depths globally, it may be possible at the very least to determine empirical depth dependent coherence distributions which could be used to train general MDNs for various depths. Indeed, this will be the subject of future manuscripts. For example, when generating training samples by forward modelling compliance signals from structural models, rather than multiplying signals by a fixed coherence, as we have done, one could multiply by a randomly selected coherence function chosen from a suitable distribution of coherence functions for the selected depth. If these coherence values are recorded, then, during the training process, they can be fed to the MDN as prior information in conjunction with the forward modelled compliance signals. Using such an approach the method could be made station independent and thus, entirely depth dependent. Similarly, in the implementation presented here we have also effectively assumed the noise statistics for a single station when creating training signals. Again, it may be possible to generalize compliance noise in a station independent way when creating training data for the network. For now, however, the MDNs we have trained in this study are depth and station specific.

Finally, in our implementation of this method, we used a simple parameterization scheme and assumed constant density profiles and V_P profiles. In general, the trained MDN performance will depend on the parameterization scheme used, and more sophisticated parameterization schemes may enable greater utility of the method. Moreover, the focus of this study has been on inverting for V_S in oceanic sediments. However, the method could be adapted for use in more general settings.

3.6.3 Advantages

Despite the limitations above, many of which may be addressed in principle, the main advantage of MDN inversion over other inverse methods is that MDN inversion is a non-linear

method capable of directly estimating Bayesian posterior probability distributions. In fact, the MDN inversion procedure is a rigorously-formulated Bayesian inference procedure and is equivalent to MCMC methods (Sambridge & Mosegaard, 2002) given sufficient training data (Käufl et al., 2016). However, unlike other non-linear inverse techniques such as MCMC methods, which construct posterior probability distributions by sampling from them, MDNs can be understood to estimate posteriors through prior sampling (Käufl et al., 2016). In essence, MDNs evaluate all prior samples without reference to any particular data observations and, for this reason, MDN inversion is orders of magnitude faster than both MCMC approaches and linearized inverse methods (Earp et al., 2020). That being said, the computationally intensive aspect of MDNs lies within their training, which can take up to several hours depending on the exact problem. However, once trained, MDN inversion is on the order of seconds, thus they are especially suited to situations in which the same inverse problem needs to be solved repeatedly at different points in space or time (Käufl et al., 2016). Finally, MDNs can be standardized and are easily shared. For example, in the case of compliance inversion pursued here, the MDN used to invert data from A02W occupies 185 kB on disk and, if used by anyone else, will produce repeatable measurements.

3.7 Conclusions

In this study, we demonstrate the effectiveness of a novel approach to non-linearly invert compliance signals recorded by co-located OBSs and high-sample-rate pressure gauges for V_S in shallow oceanic crustal structures within a probabilistic framework. Rather than use a non-linear inverse technique such as MCMC, which estimates posterior probability distributions through prior sampling, we use a machine learning technique known as a MDN to learn conditional probability distributions between compliance signals and structural V_S models via prior sampling (Käufl et al., 2016; de Wit et al., 2013; Meier et al., 2007). Using this approach, we were able to train a network to adequately invert for oceanic V_S profiles in several thousand synthetic models. Among all 30,000 synthetic inversion tests, the average velocity error was 0.025 km/s. We then applied the trained MDN to invert compliance data recorded by OBS A02W of the Eastern Lau Spreading Center Seismic Experiment, for which a V_S profile was recently estimated by Zha and Webb (2016), using an MCMC approach. The resulting V_S profile obtained using the MDN in this study is in excellent agreement with the result of Zha and Webb (2016).

In the context of V_S inversion from compliance data, the contrast between non-linear sampling techniques (e.g., MCMC) and MDN is that, within the prior sampling framework, the process of computing realizations of data is separate from the actual inversion. Therefore, while the act of training a MDN can be computationally demanding, once a network is trained, the actual inversion process itself is computationally extremely advantageous. Furthermore, we note that the use of MDNs allows for repeatable measurements and therefore helps to standardize geophysical inversions. Finally, while the MDN approach to compliance inversion pursued in this work led to networks that are not able to generalize to other stations, we discussed improvements to the method which would allow it to do so.

All models are wrong, but some are useful

George Edwin Pelham Box

4

Article: Shear-Wave Velocity Structure of Sediments on Cascadia's Continental Margin From Probabilistic Inversion of Seafloor Compliance Data

Contents

4.1 Abstract

Several seismic techniques, both passive and active, exist for estimating the shear-wave velocity V_S structure of shallow sedimentary structures. In particular, passive compliance signals recorded by broadband ocean-bottom seismometers (OBSs) can be used to invert for V_S structure. While compliance-based imaging studies have been carried out at several locations across the Cascadia Subduction Zone, such an approach has not been extensively applied to OBSs deployed on the continental shelf and slope. In this study, we measure compliance and coherence signals at 13 broadband OBSs deployed along Cascadia's continental shelf and upper slope. We then use a recently developed technique to probabilistically invert compliance signals for shallow V_S structure that makes use of mixture density neural networks. Finally, we compare and contrast our inverted V_S profiles and derived properties obtained using this method with previous studies focused on properties of basin sediments.

4.2 Introduction

The structure and composition of shallow sedimentary layers (< 4 km) deposited onto oceanic and continental tectonic plates is a significant control on a number of important geophysical processes that occur at subduction zones. Near the trench, the level of consolidation and the hydration state of subducting sedimentary layers has been shown to influence both the up-dip and down-dip rupture extent of shallow megathrust earthquakes (Han et al., 2017; J. Zhu et al., 2020). This is primarily due to the impact that sediments and their fluid content have on friction and stress near the megathrust (Saffer & Tobin, 2011; Saffer & Wallace, 2015), which influences both megathrust and tsunamigenic potential (Polet & Kanamori, 2000; Gulick et al., 2011; Vannucchi et al., 2017). Landward of the trench, shallow sediments in accretionary prisms and continental slopes may contain significant reservoirs of marine gas-hydrates, whose extent and formation depend in part on sedimentation rates, the thickness of sediment covers, the textural and mineralogical properties of sediments, and their pore water geochemistry (Kvenvolden, 1994; Dickens & Quinby-Hunt, 1997; Clennell et al., 1999). Because these factors control the marine gas-hydrate stability field, they also have the potential to impact both climate change and hydrocarbon resource extraction (Hyndman & Davis, 1992). On continental slopes, marine gas-hydrate dissociation and large deposition rates are examples of processes that can induce excess pore fluid pressure in shallow sedimentary layers, thereby destabilizing slopes and promoting submarine landslides (Canals et al., 2004; Solheim et al., 2006). Such landslides are responsible for approximately 8% of the world's tsunamis globally and pose significant hazards to coastal communities and marine geotechnical engineering projects (Vanneste et al., 2013).

Within the Cascadia Subduction Zone (CSZ), a variety of seismic measurement techniques, both passive and active, have been used to study the physical properties of shallow marine sedimentary structures within the basin and the margin. Yet, the majority of studies focused on imaging shallow marine sediment properties of the continental margin have involved active source methods (e.g. Cochrane et al., 1994; Yuan et al., 1994; MacKay, 1992; Dash & Spence, 2011a; Han et al., 2017) because they provide higher resolution at small scales compared to passive methods. However, because shear-wave energy is difficult to excite acoustically, active source studies, which rely on air gun shots or explosives, are unable to directly estimate shear-wave velocity (V_S) structure (Sauter et al., 1986; Whitmarsh & Miles, 1991; Davy et al., 2020) and are therefore restricted to estimating compressional wave velocity (V_P) structure, or V_S derived from P-to-S conversions (e.g. Rychert et al., 2018; J. Zhu et al., 2020). Alternatively, passive methods that exploit compliance signals recorded

by ocean-bottom seismometers (OBSs) have been successfully used to study V_S and sediment thickness of shallow oceanic structures (e.g. Yamamoto & Torii, 1986; Crawford et al., 1991, 1998; Ruan et al., 2014; Bell et al., 2015; Zha & Webb, 2016; Doran & Laske, 2019; Saikia et al., 2020; Mosher et al., 2021). However, such an approach has not been extensively applied to OBSs deployed on the Cascadia continental shelf.

In a recent study Mosher et al. (2021, in review) developed a novel method to probabilistically invert seafloor compliance signals for V_S using mixture density networks (MDNs), a type of neural network capable of handling non-linear inverse problems (Bishop, 1994). In this study, we apply this method to a group of broadband OBSs deployed along the continental shelf and slope of the CSZ to recover shallow V_S estimates. We then use recovered V_S profiles to estimate V_P/V_S ratios and porosity values, and compare our results to other studies focused on basin sediments. Finally, we also consider implications of the method for marine gas-hydrate exploration.

4.3 Data and Methods

In this study, we use data recorded by 13 broadband OBSs deployed during year 3 of the Cascadia Initiative Community Experiment (Toomey et al., 2014) to calculate compliance signals. The stations were deployed in a quasi-linear fashion, which started offshore near Gray's Harbour, Washington, and extended approximately 80 km west (Figure 4.3.1). The station line spans the continental shelf and a portion of the continental slope. Deployment depths range from 56 - 1764 meters below sea level. In this section we first describe compliance theory before detailing our processing steps to obtain reliable compliance functions that are used as input to the probabilistic inversion. We then outline our inversion strategy and model parameterization to estimate $V_S(z)$ profiles at each OBS station.

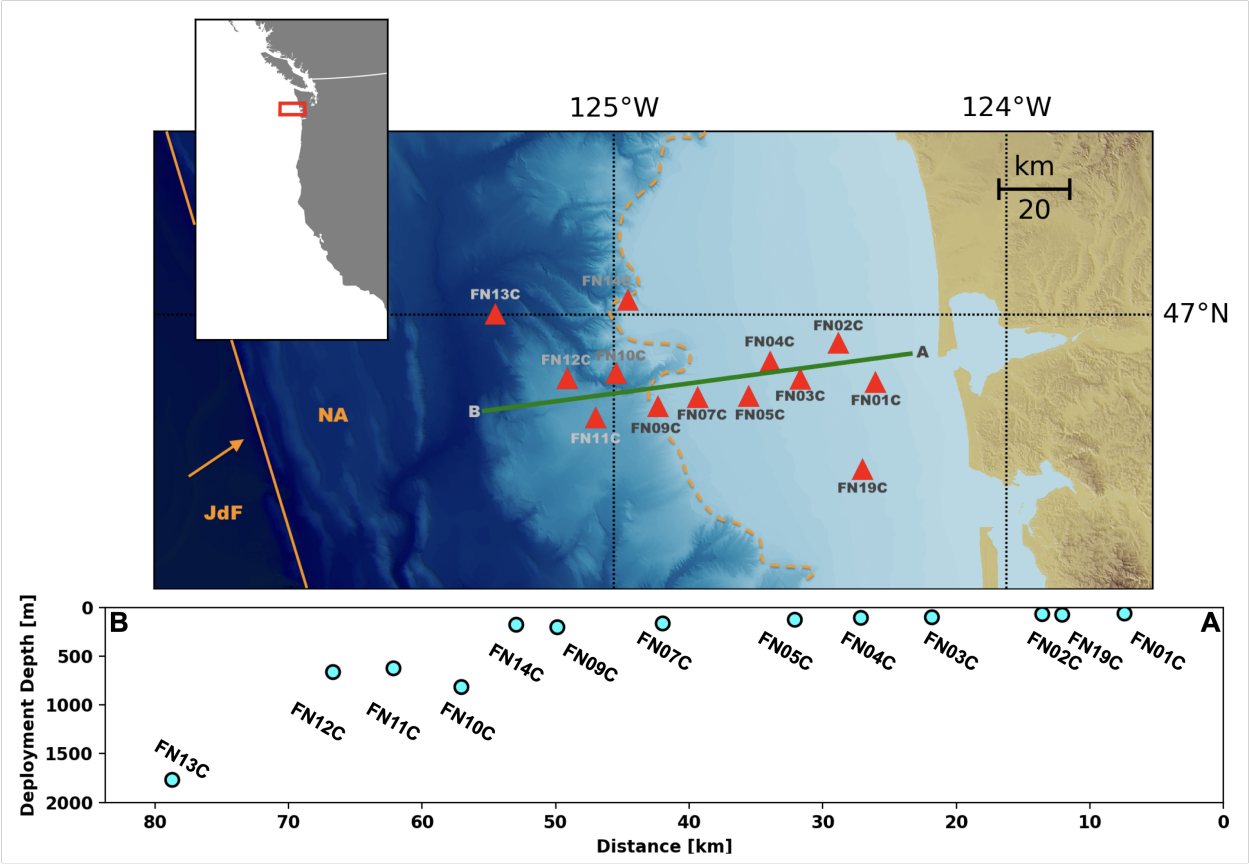

Figure 4.3.1: Top Panel: Map of the study region showing the 13 CICE instruments used in this study (red triangles). The transect A-B (green line) is used for projecting station results. The approximate boundary of the Cascadia deformation front is shown as a solid orange line. The dashed orange line demarcates the continental shelf from the slope. NA and JdF denote the North American and Juan de Fuca tectonic plates. Bottom Panel: The distribution of station deployment depths plotted along the transect A-B (from right to left).

4.3.1 Compliance Theory

Seafloor compliance is the phenomenon whereby long-period ocean infra-gravity waves propagating along the sea surface induce a loading and deformation of the seafloor. Such waves propagate with wavelengths on the order of tens of kilometers, periods of several minutes, and with wave displacements ranging from millimeters to centimeters (Aucan & Ardhuin, 2013). The compliance of a uniform half-space was first given by Sorrells and Goforth (1973), and can be expressed mathematically as

$$\xi(\omega) = -\frac{1}{k(\omega)}\left(\frac{V_P^2}{2\rho V_S^2(V_P^2 - V_S^2)}\right), \tag{4.1}$$

where ρ is the density of the medium and $k(\omega)$ is the wavenumber of the ocean infra-gravity wave as a function of angular frequency ω. It is clear from equation 4.1 that the seismic velocity structure, and the elastic properties on which they depend, govern the response of the seafloor to such loading. The wavenumber $k(\omega)$ is obtained from the infra-gravity wave dispersion relation

$$\omega^2 = gk(\omega)\tanh\left(k(\omega)\,H\right), \tag{4.2}$$

where g is the acceleration due to gravity, and H is the station deployment depth. In practice, solving equation 4.2 for $k(\omega)$ is non-trivial, and either deep or shallow water approximations are used, depending on the context. Yamamoto and Torii (1986) were the first to invert seafloor compliance for shallow oceanic structure (1D shear modulus profiles), which they applied to stations deployed at shallow depths ($<$ 1000 meters below sea level). Crawford et al. (1991) then modified the procedure to invert for V_S at deep water sites ($>$ 1000 meters below sea level). Following Crawford et al. (1991), it is customary to work with normalized compliance, in which the filtering effect of ocean infra-gravity waves is removed,

$$\eta(\omega) = k(\omega)\xi(\omega) = -\left(\frac{V_P^2}{2\rho V_S^2(V_P^2 - V_S^2)}\right). \tag{4.3}$$

Equation 4.3 can be used to predict compliance for 1D layered Earth models ($V_P(z)$, $V_S(z)$, and $\rho(z)$, for depth below the seafloor z) using the matrix propagator method (Aki

& Richards, 2002). Henceforth, whenever we refer to compliance, we implicitly mean normalized compliance.

4.3.2 Compliance Measurement

In this section, we provide a detailed description of how the compliance signals and related quantities used in this study were computed. Compliance is measured by multiplying the wave number $k(\omega)$ of the infra-gravity forcing waves by the transfer function (spectral ratio) between the vertical displacement spectrum $Z(\omega)$ and seafloor pressure spectrum $P(\omega)$ (Crawford et al., 1991, 1998; Zha & Webb, 2016). In practice, ensemble averaged estimates of $Z(\omega)$ and $P(\omega)$ are used in order to avoid potential bias in the presence of noise (Ruan et al., 2014). Note that we differentiate between predicted and measured compliance signals using a superscript "*". Thus, measured compliance is expressed mathematically as

$$\eta^*(\omega) = k(\omega)\frac{\langle Z(\omega)\rangle}{\langle P(\omega)\rangle}, \tag{4.4}$$

where the angled brackets denote ensemble averaging of the spectra. In this study, we solve for $k(\omega)$ using a rational approximation of the hyperbolic tangent function in equation 4.2, namely

$$\tanh(x) \approx \frac{x(27+x^2)}{27+9x^2}. \tag{4.5}$$

Doing so gives a quartic equation for k which will always have solvable roots. Moreover, the rational approximation in equation 4.5 is always more accurate than the shallow water approximation ($tanh(x) \approx x$, for small x), and is also more accurate than the deep water approximation ($tanh(x) \approx 1$, for large x), unless the product of $k(\omega)H > 2.96$ (Figure 4.3.2). Therefore, we chose to solve for $k(\omega)$ using the rational approximation in equation 4.5, unless the product of $k(\omega)H > 2.96$, in which case we use the deep water approximation.

All data processing was carried out using modules from the open-source ATaCR software package that are specifically designed for computing compliance and coherence signals from OBS data (Janiszewski et al., 2019b; Audet & Janiszewski, 2020). To compute the compliance signals recorded by the 13 OBSs used in this study, we download all available

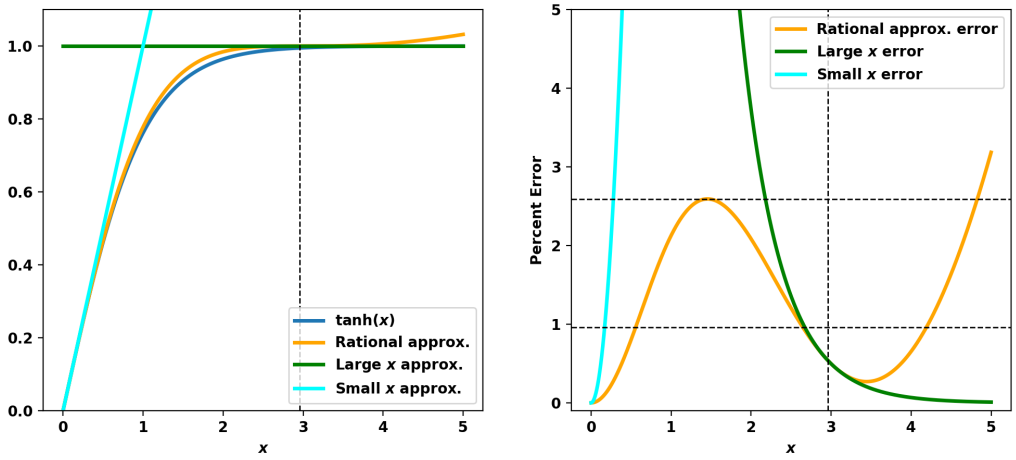

Figure 4.3.2: Left: Approximations to the hyperbolic tangent function. Right: Approximation errors in percent. The vertical dashed lines denote $x = 2.96$, above which the large x (i.e. deep water) approximation outperforms the rational approximation given by equation 4.5. The horizontal dashed lines denote the average error and maximum error of the rational approximation, approximately 1 % and 2.5 %, respectively.

station data in daily segments (approximately 10 months of data per station). We then down-sample data to 1.0 Hz and remove the instrument response. Following this, we calculate daily ensemble averages of $Z(\omega)$ and $P(\omega)$ by cutting the acquired time-series into hour-long windows and applying a 50% overlap, yielding roughly 48 windows per day. Once individual estimates of $Z(\omega)$ and $P(\omega)$ have been computed for each window, ATaCR uses an F-test to conduct an outlier analysis and reject anomalous spectra from daily averages. In addition to compliance, we also use ATaCR to compute daily ensemble averages of coherence estimates between the vertical and pressure channels of each OBS. The coherence estimates follow the same processing and ensemble averaging steps as the compliance estimates. Finally, we collect the daily compliance and coherence estimates produced by ATaCR and compile them into single-station averages. However, in order to contribute to the final single-station average, we require that daily compliance estimates, when averaged over the station's infragravity frequency band, are above a certain pressure-vertical coherence threshold. In practice, the coherence threshold needs to be set uniquely for each station. Typical thresholds may range from 0.8 and above for coherence values in the station's infragravity frequency band (e.g. Doran & Laske, 2019; Zha & Webb, 2016). All of the resulting single-station compliance and coherence estimates are shown in Appendix C and a single station example is shown in Figure 4.3.3.

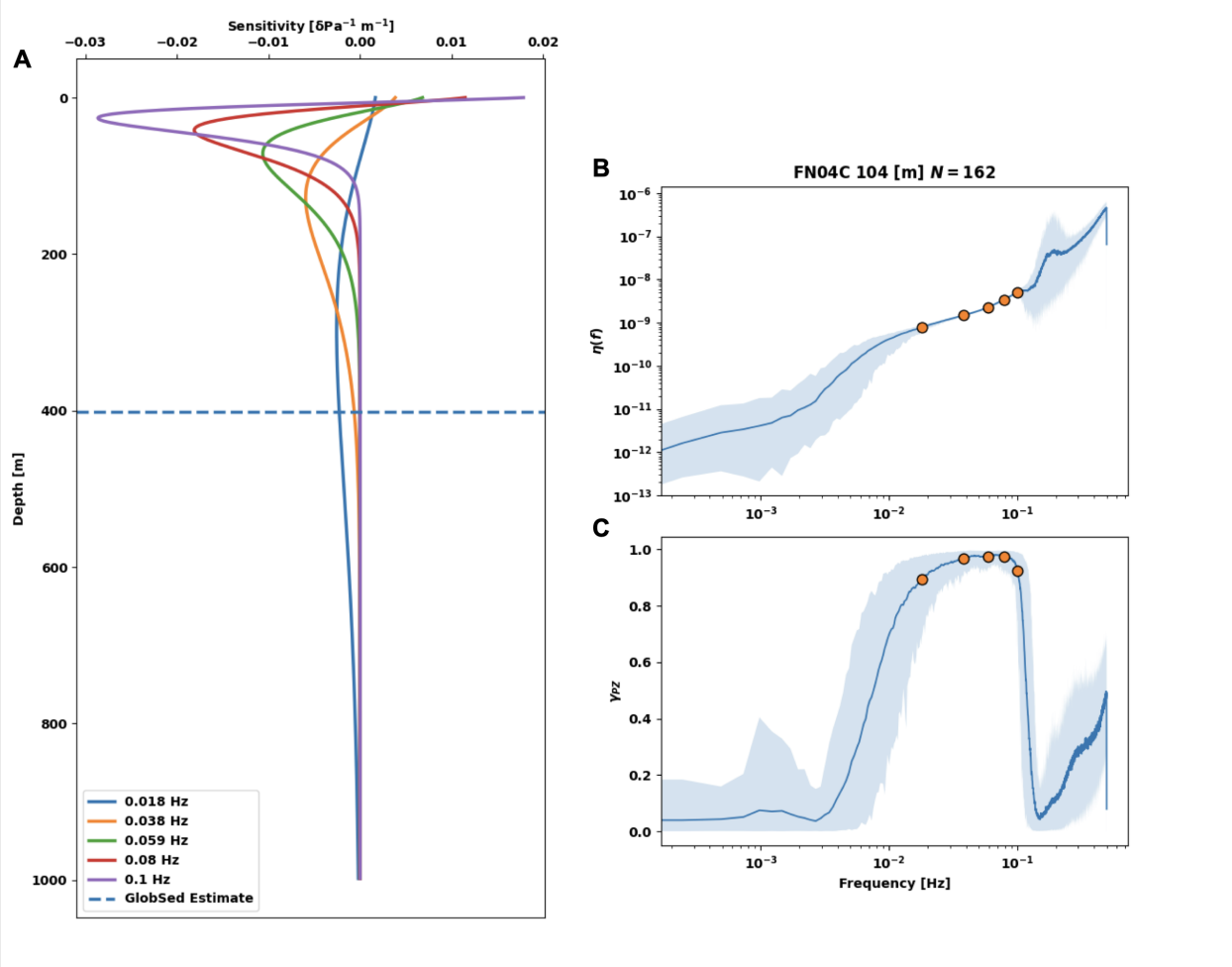

Figure 4.3.3: Panel A: Theoretical normalized compliance sensitivity kernels computed for a station deployed 104 meters below sea level for 5 different frequencies. Also indicated by the dashed line is the sediment thickness estimate from GlobSed (Straume et al., 2019) for station FN04C. Panel B: The average compliance signal computed for CICE station FN04C. Panel C: The average pressure vertical coherence signal computed for CICE station FN04C. Note that the shaded regions around the compliance and coherence curves denote the 95% confidence intervals. The signals in panels B and C were generated from $N = 162$ estimates and the circles denote the frequency-value pairs used in the inversion and forward modelling processes.

4.3.3 Mixture Density Neural Network Inversion

Whereas prior compliance inversion schemes have used traditional linearized or fully non-linear inverse techniques (e.g. Crawford et al., 1991; Zha & Webb, 2016; Doran & Laske, 2019; Saikia et al., 2020), Mosher et al. (2021, in review) implemented a non-linear inversion scheme making use of MDNs. MDNs were first devised by Bishop (1994) and have been adapted and used to study a diverse range of inverse problems in seismology, from inverting surface wave dispersion data for V_S, to inverting degree-zero spheroidal mode splitting func-

tion data for 1D radial velocity and density profiles (e.g. Meier et al., 2007; Shahraeeni et al., 2012; de Wit et al., 2013; de Wit et al., 2014; Käufl et al., 2016; Earp et al., 2020).

MDNs are a type of neural network that learn to approximate arbitrarily complicated multi-dimensional probability density functions (PDFs) between a model space and data space. Once the MDN is sufficiently trained, it can be used to interrogate the target PDF to compute conditional probabilities (Figure 4.3.4). Moreover, there is no inherent assumption that the model and data spaces are normally distributed. Instead, MDNs parameterize the target PDF as a Gaussian mixture model with K mixture components. While the number of mixture components for a particular problem is a hyper-parameter of the method specified in advance, MDNs are parsimonious, using as few mixture components as possible when approximating target PDFs (Bishop, 1995). Furthermore, it has been found that for most physical problems, the number of mixture components is rarely required to be greater than 15, and reasonable approximations of target PDFs can be achieved with far fewer mixture components (Meier et al., 2007; Käufl et al., 2016; Earp et al., 2020; Mosher et al., 2021).

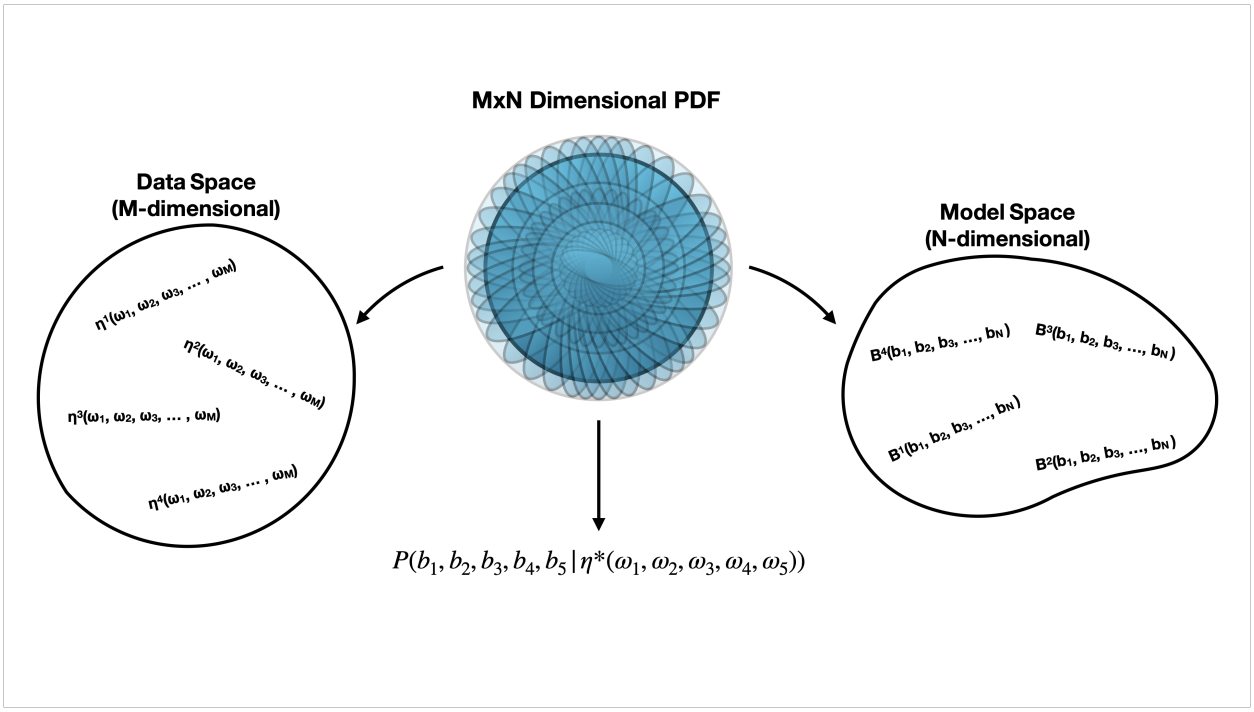

Figure 4.3.4: A conceptual image of a mixture density network as an approximation to a non-Gaussian probability density function between a data space and model space. Our data space consists of values of compliance signals at M discrete frequencies $(\omega_1, ..., \omega_M)$. Our model space consists of sets of N Bernstein polynomial coefficients $(b_1, ..., b_N)$. Particular realizations within these spaces are denoted by superscript values. Once the network is trained, it can be used to compute the conditional probability of a model given a measured compliance signal η^*.

4.3.4 Model Space, Data Space, and Parameterization

In the context of this study, the data space consists of synthetic compliance signals that are forward computed from randomly generated Earth structures, which comprise the basis of our model space ($V_P(z)$, $V_S(z)$, $\rho(z)$). To keep the dimensionality of the problem reasonable, we make the following three decisions. First, rather than training a MDN to learn the association between entire compliance signals and Earth structures, we limit compliance signals to values observed at 5 discrete frequencies, determined uniquely for each station. Second, we fix $V_P(z)$ and $\rho(z)$ using the following established relationships, appropriate for oceanic crustal structures (Hamilton, 1971; Christensen & Shaw, 1970, respectively):

$$V_P(z) = 1.511 + 1.304z - 0.741z^2 + 0.257z^3 \tag{4.6}$$

and

$$\rho(z) = 1.85 + 0.165\, V_P(z), \tag{4.7}$$

where z is depth in km, V_P is in km/s, and ρ is in g/cm^3 (note that equation 4.6 is independent of V_S, which will be important for the Discussion). As a result of fixing V_P and ρ, our inversion only attempts to recover $V_S(z)$, which is the geophysical property that compliance data are most sensitive to (Crawford et al., 1991, 1998). Third, we generate V_S profiles by randomly sampling coefficients for fourth-order Bernstein polynomials. How we determine the polynomial order to use is discussed below. Therefore, the resulting MDNs that we train approximate PDFs between compliance values at 5 frequencies, and 5 Bernstein polynomial coefficients, setting the total dimensionality of the problem we consider at 10 (Figure 4.3.4). In general, the number of data and model parameters need not be equal in the inversion (e.g. Mosher et al., 2021).

The use of Bernstein polynomials to parameterize V_S allows for the efficient parameterization of a continuous depth-function, reducing the total number of model parameters considered. This is in contrast to using a large number of layers for example. Moreover, continuous V_S profiles are expected within oceanic sediments, and this representation has optimal stability for a given polynomial order (Farouki & Rajan, 1987; Farouki, 2012). An additional attractive feature of Bernstein polynomials is that, since they sum to unity at all depths, the width of the prior bounds on the coefficients is equivalent to the width of the

prior bounds on the geophysical parameter being represented (i.e. $V_S(z)$, Gosselin et al., 2017). Using this property, we set upper bounds for our Bernstein coefficients (equivalently, our V_S profiles) using the quadratic V_S relationship determined by Ruan et al. (2014) for Cascadia basin sediments, which we refer to as $R(z)$. Specifically, we generate V_S profiles by randomly selecting 5 fourth-order Bernstein coefficients from the uniform distribution $U\big(0.1, vmax\big)$ where $vmax = R(zmax) + 0.1$ and $zmax$ is the maximum structural depth of our Earth models. The effects that fixing $vmax$ and $zmax$ have on our inversion results are addressed in Section 4.1. Additionally, we also enforce V_S profiles generated in this manner to be strictly monotonically increasing with depth. Note, that while we parameterize V_S profiles using Bernstein polynomials, we forward model synthetic compliance by discretizing randomly generated V_S, V_P, and ρ profiles into 1 m intervals. Finally, although we construct V_S profiles by sampling coefficients from a uniform distribution, it is important to note that uniform sampling of Bernstein coefficients on a fixed interval does not correspond to uniform sampling of V_S. That being said, the distribution of V_S profiles used in this study (our model space) adequately covers the range of real compliance signals observed by our stations. An example of the distribution of 100,000 V_S profiles generated in the described manner is shown in Figure 4.3.5 along with a single structural model for which $zmax = 1$ km.

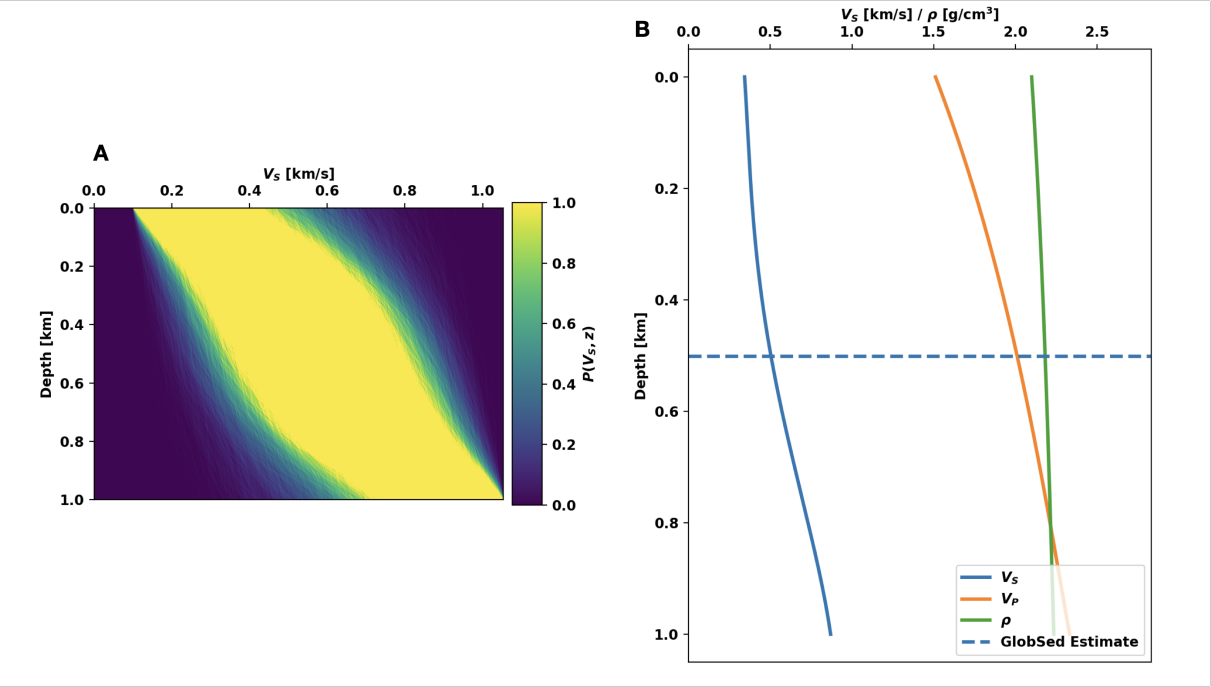

Figure 4.3.5: Panel A: An example of a randomly generated model space by uniformly sampling fourth-order Bernstein coefficients from $U(0.1, R(z = 1.0) + 0.1)$ for 100,000 models. Panel B: An example of a single randomly generated Earth model $(V_S(z), V_P(z), \rho(z))$, including a hypothetical sediment thickness estimate from GlobSed (Straume et al., 2019).

4.4 Results

4.4.1 MDN Architecture and Training

For each of the 13 stations considered in this study, we train a unique MDN to probabilistically invert compliance signals recorded by that station for 5 fourth-order Bernstein polynomial coefficients. Probabilistic estimates of V_S are then obtained by reconstruction using the Bernstein polynomials (basis functions) and the inverted coefficients. The training procedure is accomplished by generating 100,000 training models and network performance is gauged using 30,000 test models. Note that in the context of MDN training, the term model here refers to an Earth structure and its forward computed compliance signal as a pair.

In order to generate 100,000 training models, we first need to specify a maximum structural depth $zmax$ and compliance frequency range for each station. These are determined uniquely by jointly considering theoretical compliance sensitivity kernels and the single-station average coherence curve computed during the deployment. Examples of these quantities are shown for CICE station FN04C (Figure. 4.3.3). In the figure, we see that FN04C has high coherence between its pressure and vertical components and a correspondingly well-constrained compliance curve between 0.018 - 0.1 Hz. Therefore, when building training models for FN04C, we limit forward computed compliance signals to values at 5 frequencies linearly spaced within this frequency band.

To set $zmax$, we visually inspect theoretical compliance sensitivity kernels computed for the current station depth and selected frequencies to determine depths beyond which the compliance data will likely have negligible sensitivity. We choose conservative depth limits that are deeper than the presumed sensitivity of the data. However, while we use theoretical sensitivity kernels to set $zmax$, we restrict the depth of our reported inversion results on the basis of local sediment thickness estimates, which we obtain from GlobSed (Straume et al., 2019). The reason for this limitation is twofold. First, the inverse method we use specifically aims to invert compliance for V_S in oceanic sediments. Second, we parameterize V_S using Bernstein polynomials, which are inherently smooth. Thus, such a parameterization will not be sensitive to velocity discontinuities, such as those associated with a transition from sediment to crystalline oceanic crust. For example, on the basis of the theoretical sensitivity kernels computed for FN04C, we set $zmax = 1$ km but only report results down to 501 m, which is the estimated sediment thickness at FN04C's location obtained from GlobSed, plus an additional buffer of 100 m.

In addition to using the theoretical sensitivity kernels to set $zmax$, the kernels also allow us to compute the effective number of model parameters M_{eff} required to invert compliance signals at 5 frequencies at each station. Assuming the sensitivity kernels computed for each station as the data kernel $\mathbf{G}$ for a minimum-length inverse problem, we compute M_{eff} by calculating the trace of the model resolution matrix given by $\mathbf{R} = \mathbf{G}^{-\mathbf{g}}\mathbf{G}$, where $\mathbf{G}^{-\mathbf{g}}$ is the generalized inverse. Doing so, we find that M_{eff} is required to be 5 for all stations. Thus, the decision to use fourth-order polynomials (with 5 model parameters).

Once $zmax$ and compliance frequency ranges have been chosen for each station, we randomly generate structural models in the manner described above. By definition, forward computed compliance signals assume a pressure-vertical coherence of unity, which is not realistic. We therefore multiply synthetic compliance signals by the average station coherence to mimic a more realistic distribution of signals in data space. As a final step, these compliance signals are transformed and standardized before being fed to a MDN for training. This process is described in further detail by Mosher et al. (2021), and includes the addition of noise to synthetic compliance signals, the amount of which is determined by the 95% confidence intervals of the compliance signals measured at each station.

Each MDN that we train uses the same network architecture. In particular, we allow for up to $K = 6$ Gaussian mixture components, prescribe 5 hidden layers followed by a final MDN layer, set the number of hidden units in each layer to 40, and use swish activation functions. This particular network architecture, in conjunction with the dimensionality of the inverse problem, yields 9,506 tuneable network parameters. By using 100,000 examples during training, we ensure that there are at least 10 times as many training examples as there are network parameters, which is a general rule of thumb for avoiding over-fitting (Valentine & Woodhouse, 2010; Meier et al., 2007; de Wit et al., 2013). Finally, networks are trained using mini-batch gradient descent with batch sizes of 32 samples for a maximum of 1,000 epochs. See Mosher et al. (2021) for further details.

4.4.2 Inverted V_S Profiles

A single station inversion result is shown for FN04C (Figure 4.4.1). Recall that the trained MDN approximates the true PDF between 5 fourth-order Bernstein coefficients and observed compliance values at 5 frequencies, selected uniquely for a given station. Therefore, we obtain a probabilistic estimate of the 1D V_S structure beneath a station by taking repeated sam-

ples from the learned PDF, conditioned on the compliance values observed at that station. Mathematically, we take repeated samples from

$$P\big(b_1, b_2, b_3, b_4, b_5 | \eta^*(\omega_1), \eta^*(\omega_2), \eta^*(\omega_3), \eta^*(\omega_4), \eta^*(\omega_5)\big), \tag{4.8}$$

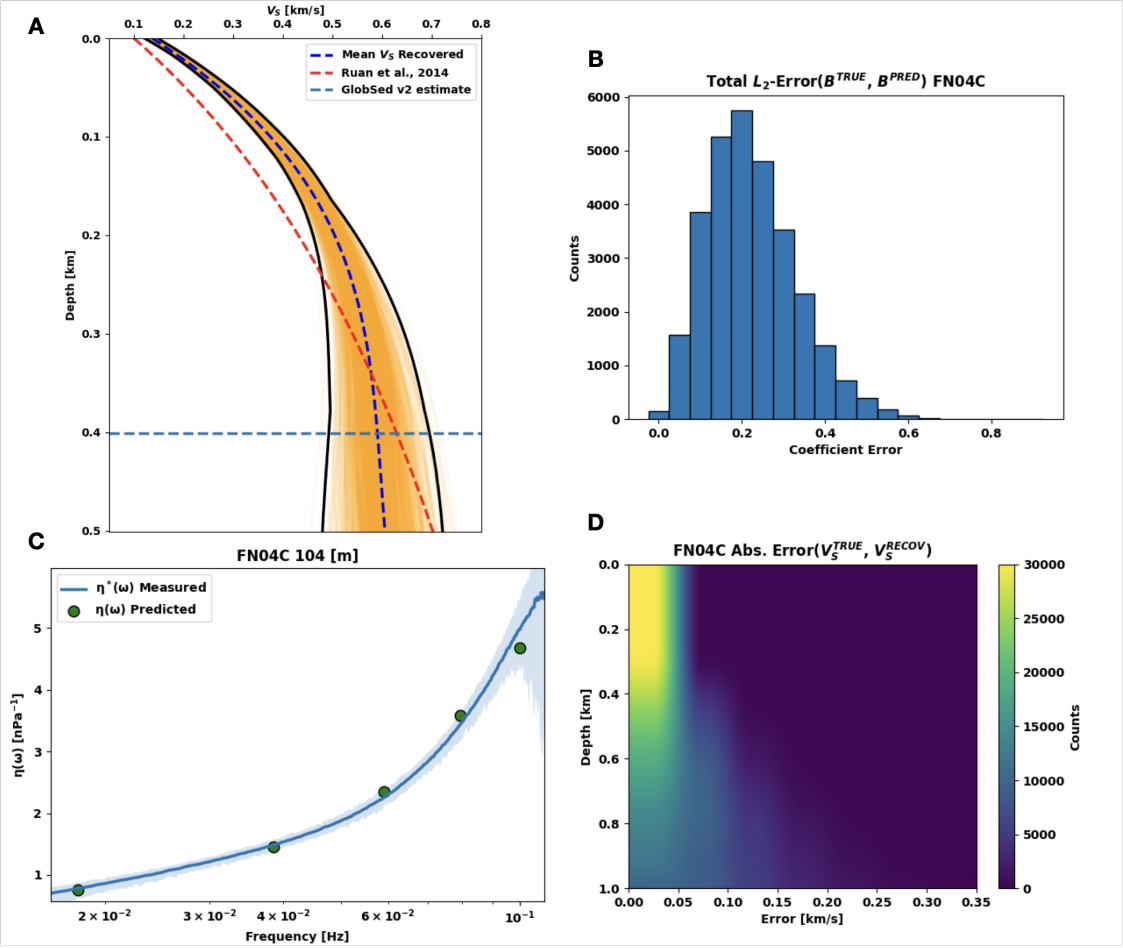

Figure 4.4.1: Panel A: The single station inversion result obtained for FN04C (blue dashed) compared to the reference profile (red dashed) of Ruan et al., 2014. Orange lines represent individual samples from the MDN conditioned on the compliance signal recorded by FN04C, and the black lines represent the 95% confidence intervals computed from 1,000 samples. Panel B: Total L_2-error for recovered Bernstein coefficients compared to the true coefficients from 30,000 test models. Panel C: The forward computed compliance signal from the recovered V_S profile in A ($\eta(\omega)$) compared to the measured signal ($\eta^*(\omega)$) for FN04C. The shaded region indicates the 95% bounds of the measured signal. Panel D: The distribution of absolute errors between recovered and true V_S profiles as a function of depth for the 30,000 test models.

where the b_i's are the Bernstein coefficients and the $\eta^*(\omega_j)$ terms are the components of the measured compliance signal. We obtain our final estimate $V_S^{Recov.}(z)$ by computing

the mean of 1,000 V_S profiles, where each profile is constructed from a particular sample of Bernstein coefficients. This approach allows us to compute the 95% confidence intervals of our velocity estimates.

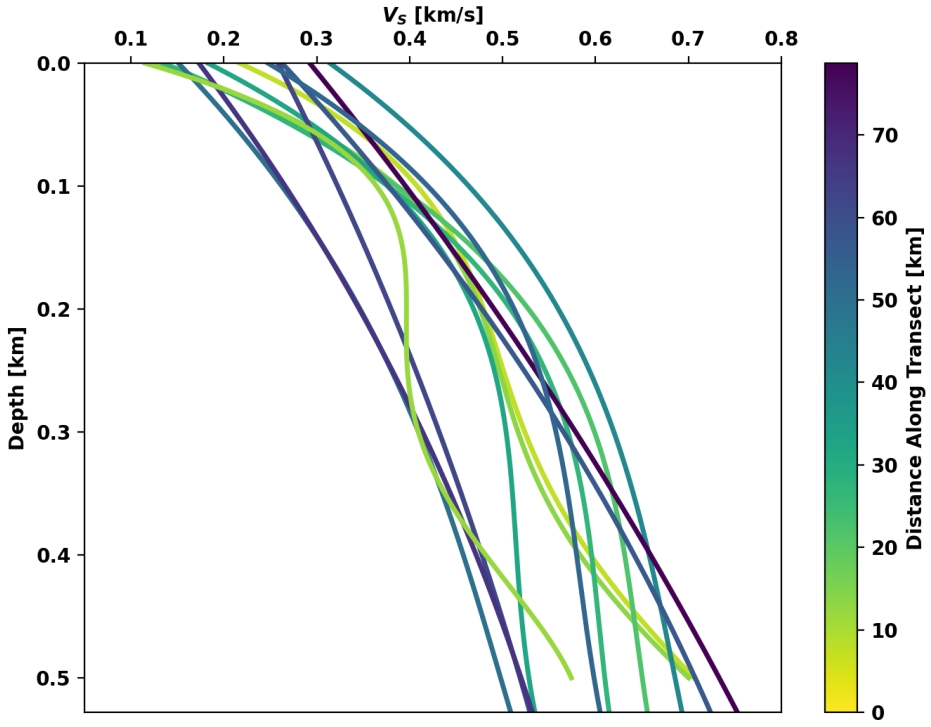

Figure 4.4.2: Final inverted V_S profiles obtained at all 13 stations used in this study reported down to 550 m. The profiles have been colour-coded by distance along the transect shown in Figure 4.3.1. Warmer colours indicate stations closer to the coast. Cooler colours indicate stations closer to the deformation front. The profiles have been plotted without confidence intervals for clarity.

We gauge the performance of a given MDN using the 30,000 test models created for the corresponding station and its specific hyper-parameters (maximum structural depth and chosen compliance frequencies). Note that test models are not included during the network training phase and, in principle, are models that the MDN has not yet encountered. While we are able to evaluate the MDN performance against the test models using a variety of error metrics (see Mosher et al., 2021), we focus on three metrics here. The first is the total L_2-error between the true and recovered Bernstein coefficients for each of the 30,000 test models. The recovered Bernstein coefficients for each of the test models are the mean coefficient estimates computed from 1,000 samples from the trained MDN (conditioned on the synthetic compliance data for the respective test model). For the second error metric,

we consider the absolute error between $V_S^{Recov.}(z)$ and the true profile $V_S^{True}(z)$ for each test model, as a function of depth. For the third error metric, we forward compute the compliance signal that $V_S^{Recov.}$ would generate and compare it to the measured compliance signal $\eta^*(\omega)$. Examples of these error metrics are shown for FN04C (Figure 4.4.1) and figures of the third error metric are shown for all stations in Appendix C. Finally, we also show all inversion results obtained in this study on a single plot, with V_S profiles colour-coded by distance along the transect shown in Figure. 4.3.1, and without confidence intervals for clarity (Figure 4.4.2).

4.5 Discussion

4.5.1 Assessment of Various Factors on Results

In obtaining these inversion results we have made the following assumptions. First, we have assumed that the V_P and ρ profiles given by equations 4.6 and 4.7, respectively, appropriately characterize sediments on Cascadia's continental shelf. Second, we have implicitly assumed that the V_S values we expect for sediments on Cascadia's continental shelf should not be much beyond those given by the V_S profile obtained by Ruan et al. (2014), if at all. This assumption comes from the fact that we set the upper velocity bound on our prior model space $vmax$ according to $vmax = R(zmax) + 0.1$. Third, we have imposed structural depth limits on our inversion results on the basis of theoretical sensitivity kernels. Although these depth limits have been set conservatively, they are an assumption nonetheless.

Therefore, it is important to assess how strongly the inversion results depend upon these assumptions. Ideally, the results should not change significantly if different relationships for V_P and ρ are used. Such a consideration is especially relevant in regional studies such as this one, where there may exist multiple well-supported empirical relationships between geophysical properties. In many cases, empirical relationships inferred in one region are applied in a different region, and the effect of these assumptions is unknown. Likewise, the results should not change significantly if we increase $vmax$, thereby increasing the size of our prior model space. In general, increasing the prior model space will naturally decrease the performance of the MDN, because the domain of the PDF learned by the MDN increases. However, if the MDN is learning a reasonable approximation to the true PDF between compliance and V_S, then the final result should not change significantly. Finally, because we set $zmax$ at each station to depths well below the presumed sensitivity of the compliance data, our inversion results should be robust to reasonable changes to $zmax$.

To assess the impact of these factors on our inversion results, we re-ran the MDN inversion procedure for three test cases, for station FN04C. In each case one of the previously mentioned factors was altered (V_P and ρ, $vmax$, and $zmax$) and we compared the inverted V_S and predicted compliance signals to the original results (Figure 4.4.1). In the first test case, we re-ran the inversion procedure assuming constant V_P and ρ profiles of 6.0 km/s and 2.0 g/cm^3, respectively. In the second case, we increased $vmax$ from $R(zmax) + 0.1$ to $R(zmax) + 0.5$. In the final case, we simply reduced $zmax$ from the original limit of 1 km to 700m. The inverted V_S profiles for each of these tests are plotted in Figure 4.5.1 against

our original result. In Figure 4.5.1 we also compare the compliance signals predicted by each of these V_S profiles against the measured signal. Figure 4.5.1 clearly demonstrates that our final results do not significantly depend on any of these parameters. Although differences manifest between the V_S profiles as depth increases, each of these profiles are well within the error of one another. In particular, at the shallowest depths (approx. 0 - 180 m), the results are most robust, as there is hardly any difference at all. Similarly, each of the predicted compliance signals computed from these V_S profiles fit the observed compliance signal within its confidence intervals.

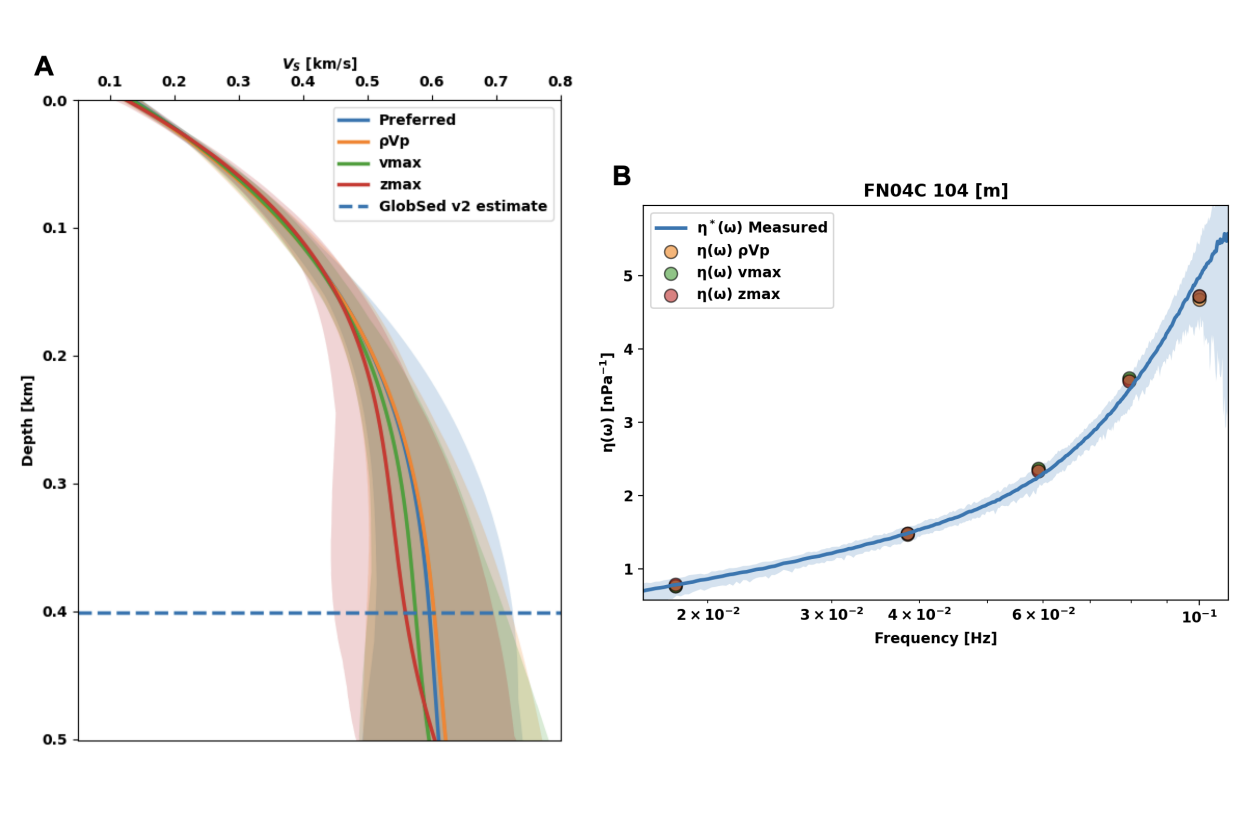

Figure 4.5.1: Panel A: Inversion results obtained at station FN04C from four test cases. Each result and its corresponding error are colour-coded. Panel B: A comparison of the compliance signal measured by FN04C (solid blue), and the 5 compliance values predicted by each of the inverted results in panel A. In each test case, all of the predicted compliance values (colour-coded circles) are within the confidence values of the measured signal (blue shaded region).

4.5.2 Comparisons and Interpretation

In Figure 4.5.2A we compare our average inverted V_S result, computed for depths from 0 - 500 m, to the average inverted result obtained by Ruan et al. (2014) for Cascadia basin

sediments. We find that overall our our average inverted V_S profile is systematically higher than $R(z)$. However, we also note that, with the exception of the shallow-most portion (approx. 0 - 180 m), $R(z)$ is within the 95% confidence region spanned by our average V_S estimate. Possible reasons for such a discrepancy include differing consolidation and stress levels between the basin and the margin sediments (Cochrane et al., 1994; Han et al., 2017; Yuan et al., 1994; J. Zhu et al., 2020), or differences in sediment lithology. Sediments deposited in the basin are composed of two main units, a pre-Pleistocene unit of fine-grained hemipelagic sediments, overlain by a rapidly deposited unit of coarse-grained Pleistocene turbidites (Cochrane et al., 1994; Hyndman & Davis, 1992; Yuan et al., 1994). By contrast, the Columbia river is the dominant sediment source for the Washington continental shelf, which is nearly covered by modern sediment (Gross et al., 1967). Generally, at water depths less than 90 m the shelf sediment consists of fine sand and coarse silt occurs at greater depths (Gross et al., 1967). Moreover, the contribution of terrestrial components to shelf sediments is much larger than in the basin (as much as 60% on the Washington shelf and 30% on the slope), and shelf sediments are generally richer in carbonates (Gross et al., 1967; Prahl et al., 1994). Despite these differences however, uncertainties are not reported for the Ruan et al. (2014) result, so the average V_S measurements in Figure 4.5.2 may yet be comparable.

Taking our average V_S measurement, and the V_P relationship assumed in this study (equation 4.6), we also compute a semi-quantitative V_P/V_S ratio which we compare to the average V_P/V_S ratio obtained by J. Zhu et al. (2020) (Figure 4.5.2B). In their study, J. Zhu et al. (2020) focus on marine sediment properties within the Cascadia basin, which they derive from active source multichannel reflection data from the Ridge-to-Trench experiment (Carbotte et al., 2012). J. Zhu et al. (2020) use their measurements to make inferences about sediment porosity values and fluid properties, and also compute an average V_P/V_S ratio for the basin sediments. With the exception of the near surface portion, there is good agreement between the results obtained in this study and those of J. Zhu et al. (2020). We suspect that the differences between these V_P/V_S estimates may be due to the fact that the J. Zhu et al. (2020) result is defined using an exponential function whereas our inversion of V_S is based on smooth polynomials. Because Bernstein polynomials allow for a more flexible parameterization of V_S compared to an exponential, they are better able to resolve more structure (i.e. curvature) in the near surface. Indeed, this is both precisely where the V_P/V_S results disagree and where our V_S results are most robust.

Several active source studies that image shallow V_P structure and compute V_P-derived porosity measurements have been conducted at a variety of margin-crossing transects within the CSZ (e.g. Yuan et al., 1994; Cochrane et al., 1994; Han et al., 2017). From margin-

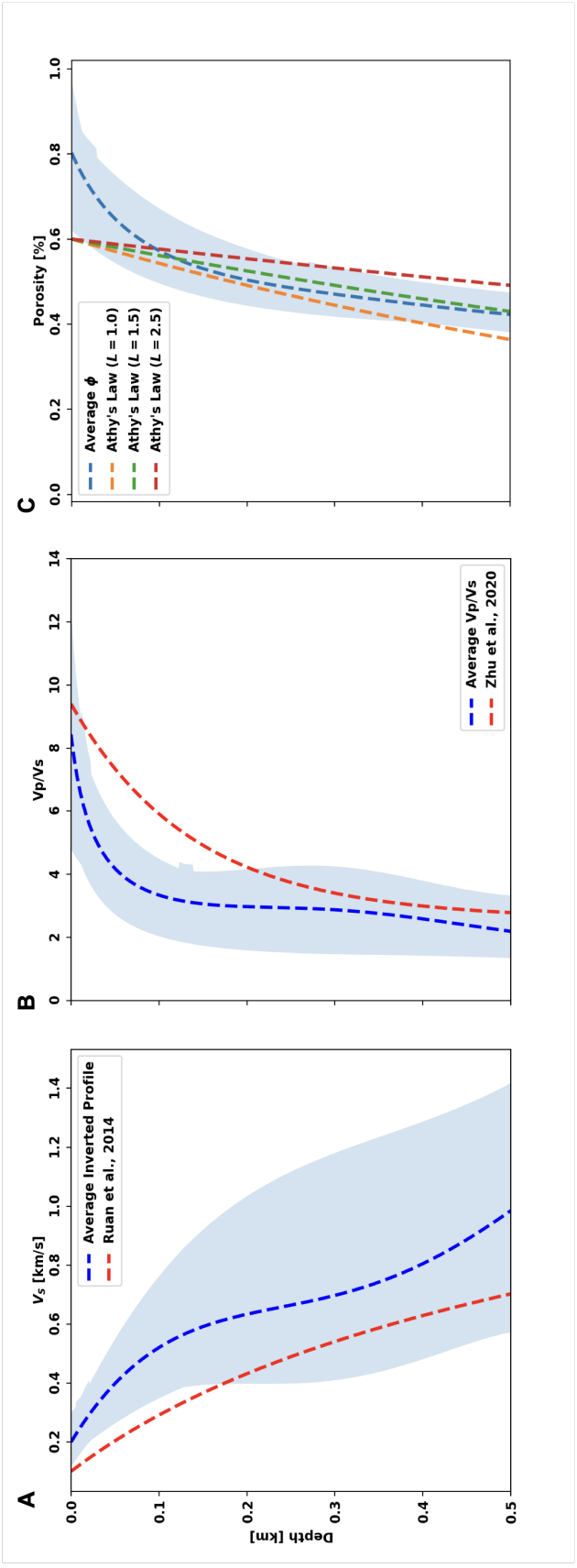

Figure 4.5.2: Panel A: The average MDN V_S inversion result obtained in this study compared to the average V_S profile obtained by Ruan et al., (2014) for Cascadia basin sediments. Panel B: The average V_P/V_S estimate obtained in this study using a semi-quantitative approach, compared to the average V_P/V_S ratio obtained for Cascadia basin sediments by Zhu et al., (2020). Panel C: The average porosity estimate obtained in this study using a semi-quantitative approach, compared to various forms of Athy's Law. The blue shaded regions denote the 95% confidence intervals.

crossing transects in northern Cascadia, Yuan et al. (1994) observe that sediment velocities progressively increase from the basin toward the accretionary prism, where compressive tectonic stresses act to overconsolidate sediments and release large amounts of pore fluids. In the accretionary prism landward of the deformation front, much lower seismic velocities are observed within sediments compared to the basin, indicating high porosity and underconsolidation. Further landward of the deformation front, Yuan et al. (1994) note that slope sediments appear to return to equilibrium as remaining pore fluids are released, restoring sediment porosity to consolidation levels similar to those in the basin. Furthermore, they also note significantly lower sediment velocities on the upper slope. In a similar study, Han et al. (2017) compute V_P and porosity measurements along two margin-crossing transects in Cascadia, one off the coast of Oregon and one off the coast of Washington, close to the line of stations used in this study. Han et al. (2017) observe that V_P and porosity estimates within basin sediments are similar along both transects. However, they observe differences in these properties starting at the deformation front and continuing landward into the accretionary prism.

The difficulty encountered when trying to interpret our results in the context of these studies (Yuan et al., 1994; Ruan et al., 2014; Han et al., 2017; J. Zhu et al., 2020) is that the transects involved typically terminate 20 - 30 km landward of the deformation front, whereas the station closest to the deformation front in our study is approximately 50 km landward. Nonetheless, we use our average V_S result to make one more type of semi-quantitative estimate, this time for an average porosity curve, which we compare to different forms of Athy's Law,

$$\phi(z) = \phi_o e^{-z/L}, \tag{4.9}$$

where the porosity $\phi(z)$ is an exponentially decreasing function of depth z controlled by the porosity at the surface ϕ_o, and L, a depth constant determined by the sediment lithology and deposition regime (Yuan et al., 1994). Yuan et al. (1994) determined L values for sediments in Cascadia to be; $L = 1.5$ km for normally consolidated basin and shelf sediments, $L = 2.5$ km for overconsolidated sediments on the slope, and $L = 1.0$ km for underconsolidated sediments near the deformation front. Currently there is no straightforward means of estimating porosity values from V_S, so we must resort to estimating porosity from V_S by first converting V_S to V_P. We do this using the V_P/V_S relation of Dash and Spence (2011b). The reason for choosing this relation rather than using equation 4.6 is because, as noted earlier, equation 4.6 is independent of V_S. Therefore, once we convert our average V_S estimate to

V_P, we use the empirical relationship derived by Hyndman et al. (1993) to convert from V_P to ϕ,

$$\phi = -1.18 + 8.607/V_P - 17.89/V_P^2 + 13.94/V_P^3. \tag{4.10}$$

The average porosity estimate we obtain is shown in Figure 4.5.2C. As with the previous two estimated quantities, our result shows both good agreement at depth with the comparable quantity (an Athy's Law with L between 1.0 and 1.5 in this case), as well as potentially significant differences in the near surface. We speculate that one potential cause for the significantly larger porosity values derived near the surface from the shelf measurements may be the presence of carbonates. As noted previously, shelf sediments are known to contain significantly higher concentrations of carbonates than basin sediments (Gross et al., 1967), and carbonate sediments are known to have porosities as high as 75% (Choquette & Pray, 1970; Das & Mukherjee, 2020).

Overall, our average V_P/V_S and porosity results suggest that, well into the accretionary prism and progressing further onto the continental shelf, sediments continue to release remaining pore-fluids and eventually return to consolidation levels similar to those of basin sediments, which is consistent with the interpretation of Yuan et al. (1994). However, significant differences between the V_S and derived properties of sediments on the shelf and basin may exist due to differences in the sediment lithology and tectonic forces acting in each of these respective environments. Moreover, the individual station results for V_S lead us to conclude that the inverted V_S profiles do not display systematic trends with distance along a trench perpendicular line (Figure 4.4.2).

Finally, we briefly mention the application of our method to marine gas-hydrate exploration. The use of compliance signals to image sedimentary structure for marine gas-hydrate reservoirs has been extensively studied (e.g. Willoughby & Edwards, 2000; Willoughby et al., 2008). However, the approach taken in these studies has been to forward model compliance signals associated with 3D Earth models of hydrate bearing structures and then compare to observed data. Naturally, a method that probabilistically inverts compliance data for V_S in sedimentary layers is advantageous for exploration.

4.6 Conclusions

In this study, we used a MDN approach to probabilistically invert seafloor compliance signals recorded by a group of 13 broadband OBS stations for shallow V_S structure within the Cascadia margin. The stations were deployed in a quasi-linear geometry and spanned a portion of the continental shelf and upper slope off the coast of Washington. The inverted V_S profiles did not display any systematic trends with distance in a trench-perpendicular direction. Upon comparing our average V_S profile with the average profile obtain by Ruan et al. (2014) for basin sediments, we find the average V_S result for shelf sediments to be systematically larger. However, as uncertainties are not reported for the Ruan et al. (2014) result, the average V_S profiles may be similar yet. Furthermore, using our average inverted V_S profile, we compute semi-quantitative estimates of V_P/V_S and porosity for shelf sediments, which we compare to results obtained for basin sediments. While our average V_P/V_S estimate agrees with the basin result at depth, we observe potentially significant differences in the near surface. However, these differences may be the result of our parameterization method for V_S, which has the ability to capture a high degree of structure (i.e. curvature) in the near surface. Similarly, our average porosity estimate agrees with the basin result at depth, but significant differences occur in the near surface. In this case, we speculate that the higher porosity values of the shelf sediments may be related to the increased presence of carbonates on the shelf. Finally, we interpret these results as suggesting that sediments in accretionary prisms continually release pore-fluids under the action of large compressive tectonic stresses until they eventually re-equilibriate to consolidation levels similar to those found in basin sediments. But, due to changes in lithology, caused by the increased presence of carbonates on the shelf, much higher porosities are observed in the near surface.

The earth is the Lord's and everything in it, the world, and all who live in it; for he founded it upon the seas and established it upon the waters.

Psalm 24:1-2

5

Conclusion

Contents

5.1 Conclusion

In this thesis I have explored different uses of neural networks for solving problems in seismology. Using a simple MLP for binary classification, I supplemented a previous detection and location method in order to automate and simplify the detection process. I then applied the method to four months of continuous seismic data recorded by a large group of stations in California and showed a 5-fold increase in the detection rate within the spatiotemporal study window. Using a neural network approach to inversion, I developed a novel method for inverting seafloor compliance signals for shallow oceanic V_S structure. I validated the method by showing that the results obtained through its use were in agreement with other inverse procedures applied to the same data. I also argue that the primary advantage of compliance inversion using MDNs comes from the fact that neural networks allow for repeatable and consistent measurements between researchers. I further applied the MDN compliance inversion procedure to a group of OBSs deployed on the continental shelf of the CSZ. A systematic study of the sediment properties on the shelf had not been attempted using a passive imaging method. The results I obtained demonstrate the usefulness of the method for studying various geophysical properties of sediments, and that the properties of sediments on the shelf constrast those in the basin in interesting ways.

So what's next for these projects? With regard to the detection and location method presented in Chapter 2, it could be easily adapted for a variety of seismic source types. Poiata et al. (2016) show that their detection method can be tuned for different seismic source types by a suitable choice of CF. For example, by constructing waveform envelope based CFs, Poiata et al. (2018) use their method to detect and image a tremor sequence in southwestern Japan. Thus, by using such CFs, in conjunction with a catalogue of tremor events, one could train a neural network to automatically detect tremor events using the procedure described in Chapter 2. An advantage of such an approach to tremor detection versus earthquake detection, is that a tremor catalogue for a given spatiotemporal window will almost always contain more training events than an earthquake catalogue for the same region. Another case in which the method presented in Chapter 2 could be advantageous is that of locating microseismic sources. The smaller spatial scale of these studies, coupled with the fact that microseismic sources are relatively abundant, mean that the method could significantly benefit microseismic studies.

With regard to the inversion of seafloor compliance for shallow V_S using MDNs, much work can be done. First, as currently implemented, MDNs trained to invert seafloor compliance for V_S are station-specific. The MDNs trained for this kind of inversion would be much

more useful if single MDNs could be trained for unique station deployment depths. That way, a MDN trained to invert compliance for V_S at a particular deployment depth could be used to invert signals from any OBS on the globe deployed at that depth. You'll recall that when creating training models for the MDNs in Chapters 3 and 4, the models needed to be multiplied by the pressure-vertical coherence of the station at which the inversion was to be performed. This guaranteed that the synthetic training data would more realistically approximate the actual distribution of compliance signals recorded by the station. If it is the case that an empirical relationship can be determined to predict the pressure-vertical coherence of an OBS based solely on its deployment depth, then it would be possible to train MDNs to invert compliance signals in a station-independent manner. However, it's currently unclear whether such an empirical relationship exists. Furthermore, if such a relationship exists, then determining how different a station's deployment depth can be from the depth assumed by a MDN used to invert its signals is critical. For example, would a MDN trained to invert compliance signals recorded by a station deployed at 2000 meters below the sea surface be able to reliably predict V_S for a station deployed at 1900 meters below seafloor? If training unique MDNs for different deployment depths is possible, and if a minimal depth discretization is feasible, then in principle, one could train a set of MDNs to invert compliance signals for any OBS on the globe, at any depth. This would be an extremely useful tool to the marine seismic community, because once trained, the inversion time is negligible, and the networks produce repeatable measurements. Second, another promising application of compliance-inverting MDNs would be their use in time-dependent studies. Assessing if and when seismic structures change in time can provide a great deal of insight into many geophysical processes. Therefore, the speed at which MDNs perform compliance inversion for V_S makes them excellent tools for time-lapse imaging of seismic structures.

5.1.1 Final Remarks

In light of the work presented in this thesis, as well as the vast body of machine learning studies conducted in seismology over the past three decades, a natural question to ask is whether machine learning has revolutionized seismology? It's undeniable that machine learning and neural networks have had a *significant* impact on seismology and many other fields in the sciences. However, my personal opinion is that to deem that impact *revolutionary* is too strong. I prefer to hold a healthy skepticism regarding the future of machine learning in seismology. I think better questions to explore, are the extent to which machine learning has impacted the field, and what its continued role will be in the future. The volume of

seismic data available is currently expanding at an exponential rate (Kong et al., 2018) and it's clear that research on tools that will help to automatically sort and process those data will continue to be necessary. Machine learning has become a critical asset in this regard and can even assist in identifying new phenomena and patterns in data that are not observable by human analysts (Bergen et al., 2019). Just as was the case in the 1960s through to the 1980s, the capabilities and expectations of machine learning tend to become inflated and subsequently adjusted over time (Jiao & Alavi, 2020).

While neural networks may be able to recognize what features are most important for a particular analysis, as well as accurately predict various phenomena, they do not provide any insight into the relevant physics behind those phenomena. For example, consider the problem of predicting peak ground acceleration caused by a local earthquake. This is a difficult problem that, as of yet, does not have a universal theoretical solution. Instead, a variety of empirically derived peak ground acceleration equations exist, which are region specific. While neural networks have been successfully developed to predict peak ground acceleration from earthquake source parameters (e.g. Gandomi et al., 2016; Günaydın & Günaydın, 2008; Shiuly et al., 2020), they don't provide any insight into the underlying physics. In other words, while the weights and biases of neural networks that predict peak ground acceleration are completely known, the networks are totally opaque regarding the underlying physics. Ultimately, what this example and others like it demonstrate is that machine learning tools are agnostic regarding the real-world processes that we *truly* wish to understand. Yes, we want to be able to calculate and predict quantities like peak ground acceleration in seismology, but the underlying physics that we're more interested in, well, neural networks have no idea about that. Therefore, whereas machine learning techniques such as neural networks have become indispensable tools of the trade for analyzing seismic data, and perhaps even for understanding seismic data, the role of the seismologist in developing and understanding the physical theories behind phenomena supercedes any data-driven analysis. And I expect that this will be the case for a long time to come.

A

List of Figures

B

List of Tables

C

Coherence and Compliance Curves for Cascadia OBSs

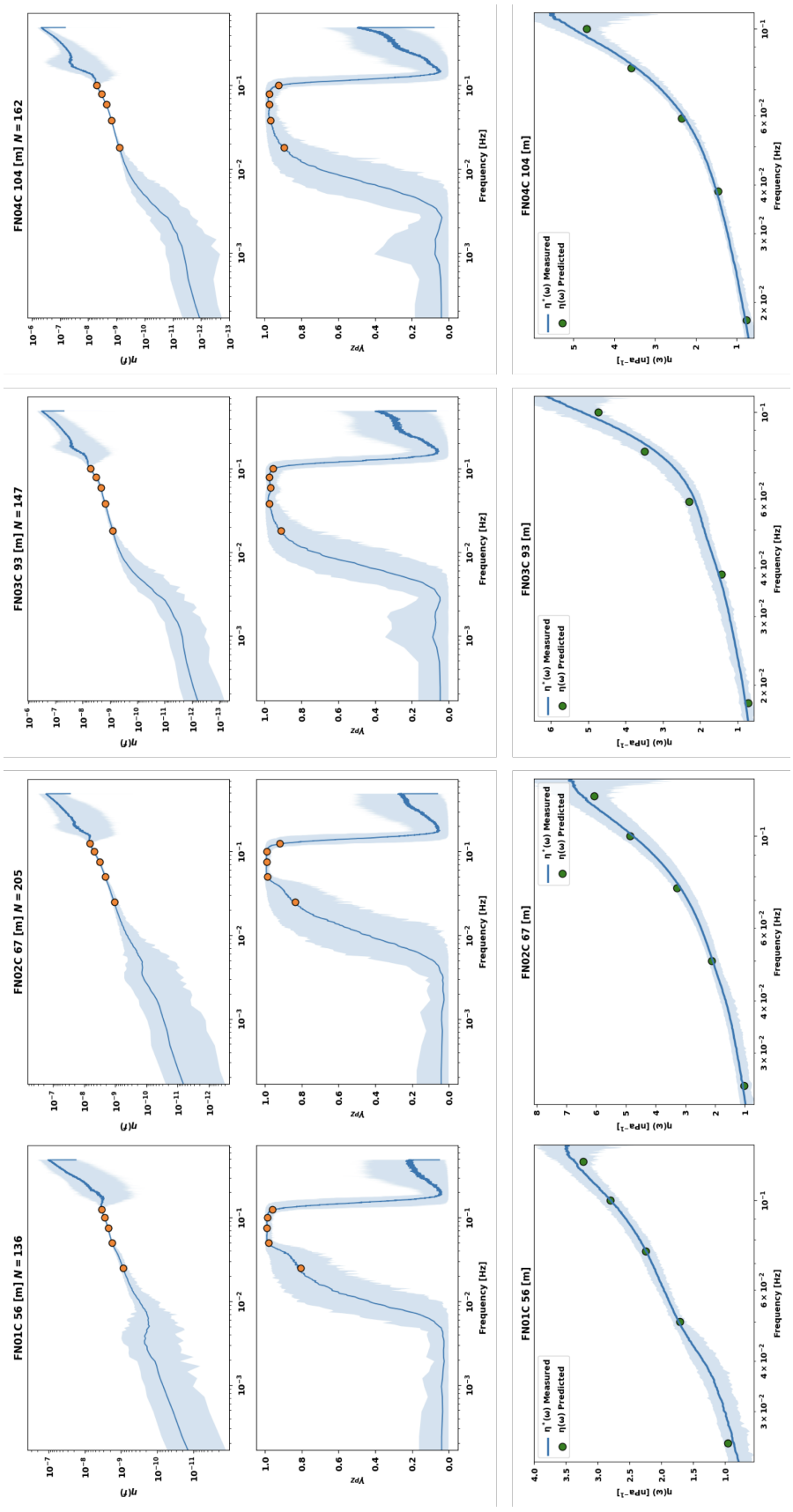

Figure C.0.1: Compliance and coherence curves for stations FN01C, FN02C, FN03C, FN04C

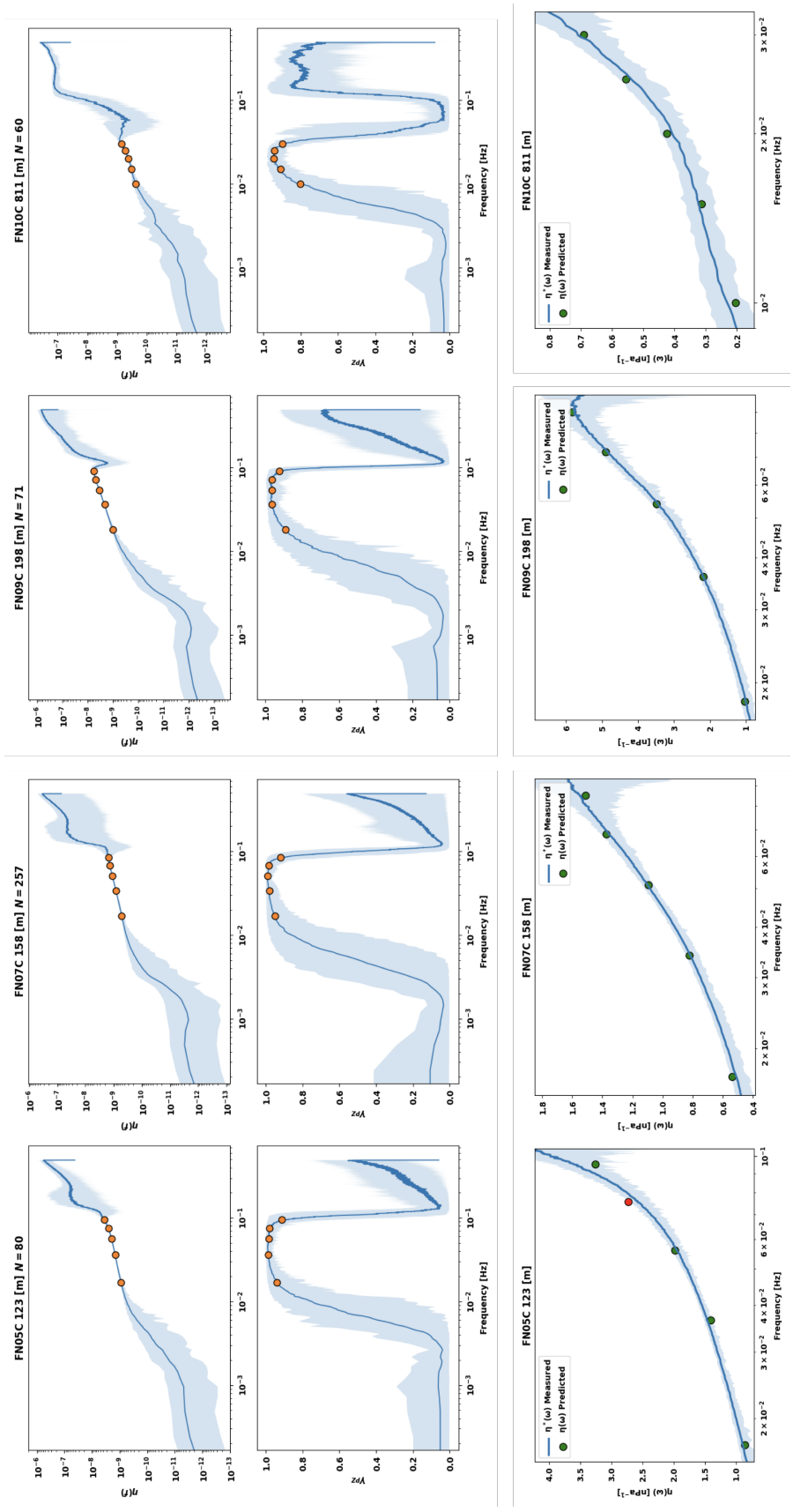

Figure C.0.2: Compliance and coherence curves for stations FN05C, FN07C, FN09C, FN10C

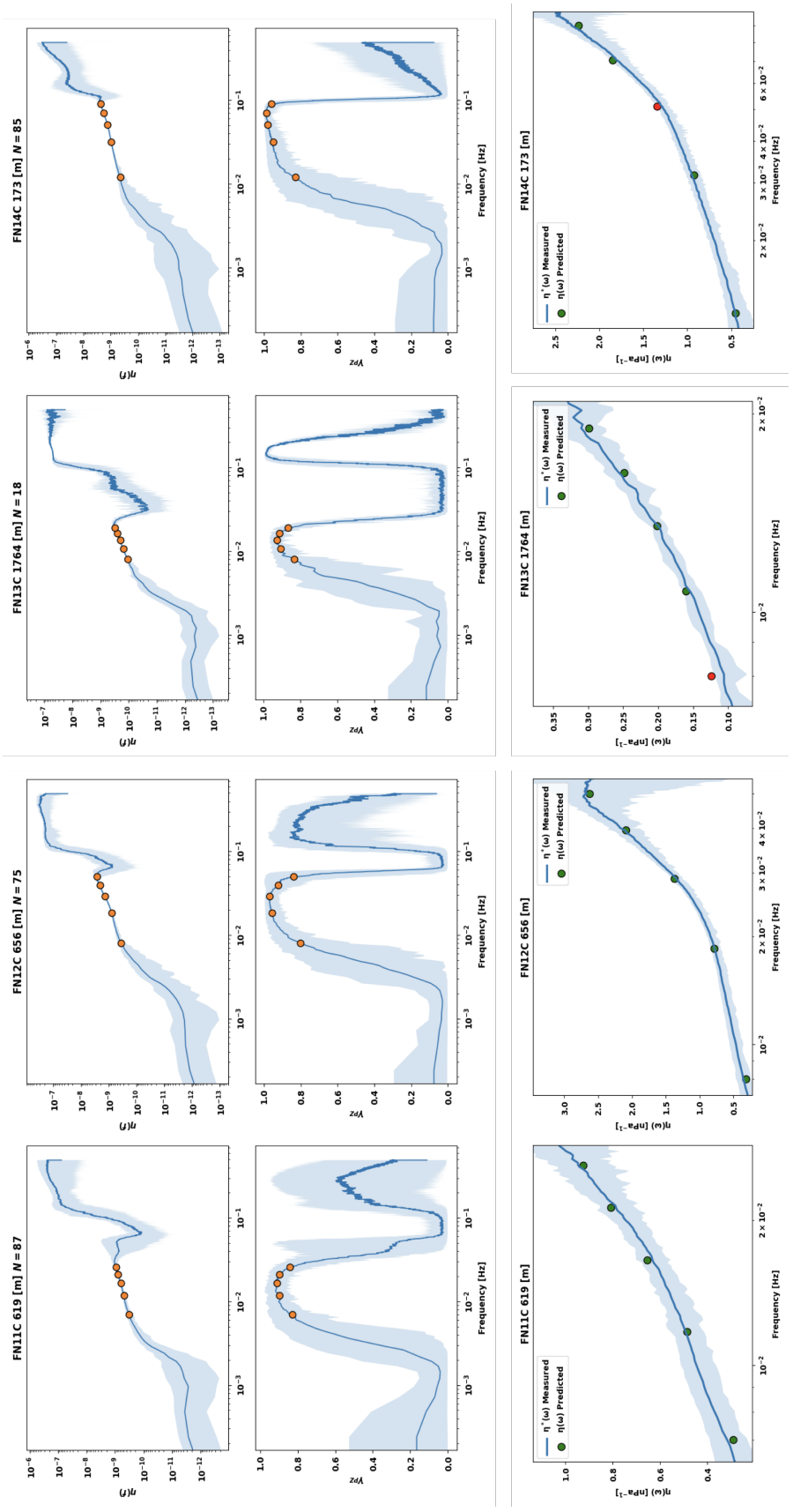

Figure C.0.3: Compliance and coherence curves for stations FN11C, FN12C, FN13C, FN14C

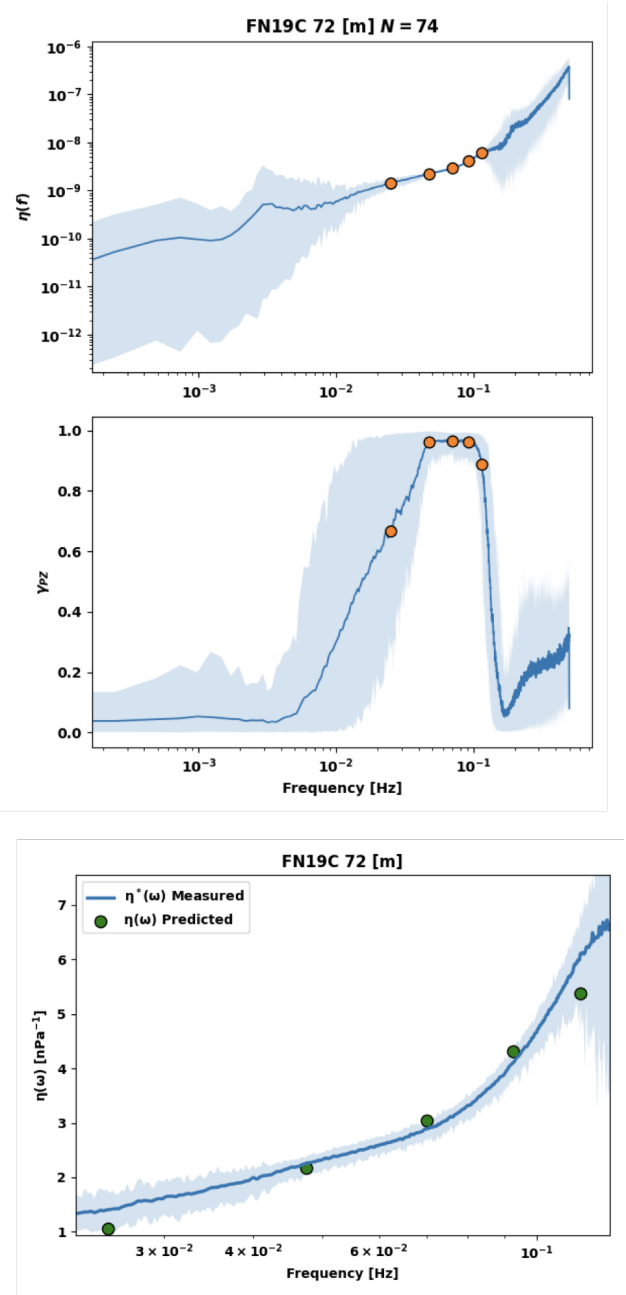

Figure C.0.4: Compliance and coherence curve for station FN19C

D

References

References

Agius, M. R., Harmon, N., Rychert, C. A., Tharimena, S., & Kendall, J.-M. (2018). Sediment characterization at the equatorial mid-atlantic ridge from p-to-s teleseismic phase conversions recorded on the pi-lab experiment. *Geophysical Research Letters*, *45*(22), 12,244-12,252.

Aki, K., & Richards, P. G. (2002). *Quantitative seismology* (2nd ed.). University Science Books. Hardcover.

Allen, R. (1978). Automatic earthquake recognition and timing from single traces. *Bulletin of the Seismological Society of America*, *68*(5), 1521-1532.

Allen, R. (1982). Automatic phase pickers: Their present use and future prospects. *Bulletin of the Seismological Society of America*, *72*(6B), S225-S242.

Apel, J. (1987). *Principals of ocean physics.* Academic Press.

Ardhuin, F., Rawat, A., & Aucan, J. (2014). A numerical model for free infragravity waves: Definition and validation at regional and global scales. *Ocean Modelling*, *77*, 20 - 32.

Aucan, J., & Ardhuin, F. (2013). Infragravity waves in the deep ocean: An upward revision. *Geophysical Research Letters*, *40*(13), 3435-3439.

Audet, P., & Janiszewski, H. (2020, june). *Obstools: Software for processing broadband ocean-bottom seismic data.* Zenodo.

Baillard, C., Crawford, W. C., Ballu, V., Hibert, C., & Mangeney, A. (2013, 11). An Automatic Kurtosis-Based P- and S-Phase Picker Designed for Local Seismic Networks. *Bulletin of the Seismological Society of America*, *104*(1), 394-409.

Beaucé, E., Frank, W., Paul, A., & Campillo, M. (2019). Systematic detection of clustered seismicity beneath the southwestern alps. *Journal of Geophysical Research. Solid Earth*, *124*(11), 11531-11548.

Bell, S. W., Ruan, Y., & Forsyth, D. W. (2015, 08). Shear Velocity Structure of Abyssal Plain Sediments in Cascadia. *Seismological Research Letters*, *86*(5), 1247-1252.

Bensen, G. D., Ritzwoller, M. H., Barmin, M. P., Levshin, A. L., Lin, F., Moschetti, M. P., ... Yang, Y. (2007, 06). Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements. *Geophysical Journal International*, *169*(3), 1239-1260.

Bergen, K. J., Johnson, P. A., de Hoop, M. V., & Beroza, G. C. (2019). Machine learning for data-driven discovery in solid earth geoscience. *Science*, *363*(6433).

Bishop, C. (1994). Mixture density networks [WorkingPaper]. *NCRG/94/004*.

Bishop, C. (1995). *Neural networks for pattern recognition.* Oxford University Press.

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

Breiman, L. (2001). Random forests. *Machine Learning*, 5-32.

Brenguier, F., Shapiro, N. M., Campillo, M., Nercessian, A., & Ferrazzini, V. (2007). 3-d surface wave tomography of the piton de la fournaise volcano using seismic noise correlations. *Geophysical Research Letters*, *34*(2).

Brooks, S., Gelman, A., Jones, G., & Meng, X. (2011). *Handbook of markov chain monte carlo.* Chapman and Hall/CRC.

Canals, M., Lastras, G., Urgeles, R., Casamor, J., Mienert, J., Cattaneo, A., ... Bryn, P. (2004). Slope failure dynamics and impacts from seafloor and shallow sub-seafloor geophysical data: case studies from the costa project. *Marine Geology*, *213*(1), 9 - 72. (COSTA - Continental Slope Stability)

Carbotte, S. M., Canales, J., Carton, H. D., Nedimovic, M. R., Han, S., Marjanovic, M., ... Parker, B. (2012, December). Evolution and hydration of the Juan de Fuca crust and uppermost mantle: a plate-scale seismic investigation from ridge to trench. In *Agu fall meeting abstracts* (Vol. 2012, p. T13H-01).

Carlson, R., & Raskin, G. (1984). Density of the ocean crust. *Nature*, *311*, 555-558.

Cesca, S., & Grigoli, F. (2015). Chapter two - full waveform seismological advances for microseismic monitoring. In R. Dmowska (Ed.), (Vol. 56, p. 169 - 228). Elsevier.

Chassande-Mottin, E. (2002). Testing the normality of the gravitational wave data with a low cost recursive estimate of the kurtosis. *in Proc. 3rd workshop on Physics in Signal and Image Processing 2003, Grenoble, France, 157–160*.

Chen, C., & Breiman, L. (2004). Using random forest to learn imbalanced data. *University of California, Berkeley*.

Choquette, P. W., & Pray, L. C. (1970). Geologic nomenclature and classification of porosity in sedimentary carbonates. *AAPG Bulletin*, *54*(2), 207-250.

Christensen, N. I., & Shaw, G. H. (1970, 08). Elasticity of Mafic Rocks from the Mid-Atlantic Ridge. *Geophysical Journal International*, *20*(3), 271-284.

Clennell, M. B., Hovland, M., Booth, J. S., Henry, P., & Winters, W. J. (1999). Formation of natural gas hydrates in marine sediments: 1. conceptual model of gas hydrate growth conditioned by host sediment properties. *Journal of Geophysical Research: Solid Earth*, *104*(B10), 22985-23003.

Cochrane, G. R., Moore, J. C., MacKay, M. E., & Moore, G. F. (1994). Velocity and inferred porosity model of the oregon accretionary prism from multichannel seismic reflection data: Implications on sediment dewatering and overpressure. *Journal of Geophysical Research: Solid Earth*, *99*(B4), 7033-7043.

Cracknell, M. J., & Reading, A. M. (2013). The upside of uncertainty: Identification of lithology contact zones from airborne geophysics and satellite data using random forests and support vector machines. *GEOPHYSICS*, *78*(3), WB113-WB126.

Crawford, W. C. (2004, 06). The sensitivity of seafloor compliance measurements to sub-basalt sediments. *Geophysical Journal International*, *157*(3), 1130-1145.

Crawford, W. C., & Singh, S. C. (2008). Sediment shear properties from seafloor compliance measurements: Faroes-shetland basin case study. *Geophysical Prospecting*, *56*(3), 313-325.

Crawford, W. C., Webb, S. C., & Hildebrand, J. A. (1991). Seafloor compliance observed by long-period pressure and displacement measurements. *Journal of Geophysical Research: Solid Earth*, *96*(B10), 16151-16160.

Crawford, W. C., Webb, S. C., & Hildebrand, J. A. (1998). Estimating shear velocities in the oceanic crust from compliance measurements by two-dimensional finite difference modeling. *Journal of Geophysical Research: Solid Earth*, *103*(B5), 9895-9916.

Dai, H., & MacBeth, C. (1995). Automatic picking of seismic arrivals in local earthquake

data using an artificial neural network. *Geophysical Journal International*, *120*(3), 758-774.

Dales, P., Audet, P., & Olivier, G. (2017). Seismic interferometry using persistent noise sources for temporal subsurface monitoring. *Geophysical Research Letters*, *44*(21), 10,863-10,870.

Das, T., & Mukherjee, S. (2020, 01). Porosity in carbonates. In (p. 9-18). doi: 10.1007/978-3-030-13442-6_2

Dash, R., & Spence, G. (2011a, 12). P-wave and S-wave velocity structure of northern Cascadia margin gas hydrates. *Geophysical Journal International*, *187*(3), 1363-1377.

Dash, R., & Spence, G. (2011b, 12). P-wave and S-wave velocity structure of northern Cascadia margin gas hydrates. *Geophysical Journal International*, *187*(3), 1363-1377.

Davy, R. G., Collier, J. S., Henstock, T. J., & Consortium, T. V. (2020). Wide-angle seismic imaging of two modes of crustal accretion in mature atlantic ocean crust. *Journal of Geophysical Research: Solid Earth*, *125*(6), e2019JB019100. (e2019JB019100 2019JB019100)

de Wit, R., Käufl, P., Valentine, A., & Trampert, J. (2014). Bayesian inversion of free oscillations for earth's radial (an)elastic structure. *Physics of the Earth and Planetary Interiors*, *237*, 1 - 17.

de Wit, R. W. L., Valentine, A. P., & Trampert, J. (2013, 06). Bayesian inference of Earth's radial seismic structure from body-wave traveltimes using neural networks. *Geophysical Journal International*, *195*(1), 408-422.

Dickens, G. R., & Quinby-Hunt, M. S. (1997). Methane hydrate stability in pore water: A simple theoretical approach for geophysical applications. *Journal of Geophysical Research: Solid Earth*, *102*(B1), 773-783.

Doran, A. K., & Laske, G. (2019). Seismic structure of marine sediments and upper oceanic crust surrounding hawaii. *Journal of Geophysical Research: Solid Earth*, *124*(2), 2038-2056.

Dowla, F. U., Taylor, S. R., & Anderson, R. W. (1990). Seismic discrimination with artificial neural networks: Preliminary results with regional spectral data. *Bulletin of the Seismological Society of America*, *80*(5), 1346-1373.

Dunn, R. A., Martinez, F., & Conder, J. A. (2013). Crustal construction and magma chamber properties along the eastern lau spreading center. *Earth and Planetary Science Letters*, *371-372*, 112-124. doi: https://doi.org/10.1016/j.epsl.2013.04.008

Dysart, P. S., & Pulli, J. J. (1990). Regional seismic event classification at the NORESS array: Seismological measurements and the use of trained neural networks. *Bulletin of*

the Seismological Society of America, *80*(6B), 1910-1933.

Earp, S., Curtis, A., Zhang, X., & Hansteen, F. (2020, 07). Probabilistic Neural Network Tomography across Grane field (North Sea) from Surface Wave Dispersion Data. *Geophysical Journal International*. (ggaa328)

Farouki, R. T. (2012). The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, *29*(6), 379-419.

Farouki, R. T., & Rajan, V. T. (1987). On the numerical condition of polynomials in bernstein form. *Computer Aided Geometric Design*, *4*(3), 191-216.

Fomel, S. (2007). Local seismic attributes. *GEOPHYSICS*, *72*(3), A29-A33.

Frank, W. B., & Shapiro, N. M. (2014). Automatic detection of low-frequency earthquakes (lfes) based on a beamformed network response. *Geophysical Journal International*, *197*(2), 1215-1223.

Frank, W. B., Shapiro, N. M., Husker, A. L., Kostoglodov, V., Romanenko, A., & Campillo, M. (2014). Using systematically characterized low-frequency earthquakes as a fault probe in guerrero, mexico. *Journal of Geophysical Research: Solid Earth*, *119*(10), 7686-7700.

Gandomi, M., Soltanpour, M., Zolfaghari, M. R., & Gandomi, A. H. (2016). Prediction of peak ground acceleration of iran's tectonic regions using a hybrid soft computing technique. *Geoscience Frontiers*, *7*(1), 75-82. (Special Issue: Progress of Machine Learning in Geosciences)

Geiger, L. (1912). Probability method for the determination of earthquake epicenters from the arrival time only. *Bull. St. Louis Univ.8,60-71*.

Gibbons, S. J., & Ringdal, F. (2006, 04). The detection of low magnitude seismic events using array-based waveform correlation. *Geophysical Journal International*, *165*(1), 149-166.

Glorot, X., Bordes, A., & Bengio, Y. (2011, 11–13 Apr). Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, & M. Dudík (Eds.), (Vol. 15, pp. 315–323). Fort Lauderdale, FL, USA: JMLR Workshop and Conference Proceedings.

Gosselin, J. M., Dosso, S. E., Cassidy, J. F., Quijano, J. E., Molnar, S., & Dettmer, J. (2017, 08). A gradient-based model parametrization using Bernstein polynomials in Bayesian inversion of surface wave dispersion. *Geophysical Journal International*, *211*(1), 528-540.

Grigoli, F., Cesca, S., Amoroso, O., Emolo, A., Zollo, A., & Dahm, T. (2013). Automated seismic event location by waveform coherence analysis. *Geophysical Journal International*, *196*(3), 1742-1753.

Gross, M. G., McManus, D. A., & Ling, H.-Y. (1967, 09). Continental shelf sediment, northwestern United States. *Journal of Sedimentary Research*, *37*(3), 790-795. doi: 10.1306/74D7179E-2B21-11D7-8648000102C1865D

Gulick, S., Austin, J., McNeill, L., Bangs, N., Martin, K., Henstock, T., ... Permana, H. (2011). Updip rupture of the 2004 sumatra earthquake extended by thick indurated sediments. *Nature Geoscience*, *4*, 453–456.

Günaydın, K., & Günaydın, A. (2008). Peak ground acceleration prediction by artificial neural networks for northwestern turkey. *Mathematical Problems in Engineering*(919420).

Hale, D. (2006a). An efficient method for computing local cross-correlations of multidimensional signals. *CWP Report, 656.*.

Hale, D. (2006b). Recursive gaussian filters. *CWP Report, 546.*.

Hamilton, E. L. (1971). Elastic properties of marine sediments. *Journal of Geophysical Research (1896-1977)*, *76*(2), 579-604.

Hamilton, E. L. (1978). Sound velocity–density relations in sea-floor sediments and rocks. *The Journal of the Acoustical Society of America*, *63*(2), 366-377.

Hamilton, E. L. (1979). Vp/vs and poisson's ratios in marine sediments and rocks. *The Journal of the Acoustical Society of America*, *66*(4), 1093-1101.

Han, S., Bangs, N., Carbotte, S., Saffer, D., & Gibson, J. (2017). Links between sediment consolidation and cascadia megathrust slip behaviour. *Nature Geoscience*, *10*, 954–959.

Hebb, D. (1949). The organization of behaviour: A neuropyschological theory. In (p. 63). John Wiley & Sons Inc.

Herceg, M., Artemieva, I., & Thybo, H. (2015, 12). Sensitivity analysis of crustal correction for calculation of lithospheric mantle density from gravity data. *Geophysical Journal International*, *204*(2), 687-696.

Hyndman, R. (1979). Poisson's ratio in the oceanic crust — a review. *Tectonophysics*, *59*(1), 321 - 333. (Crustal properties across passive margins)

Hyndman, R., & Davis, E. E. (1992). A mechanism for the formation of methane hydrate and seafloor bottom-simulating reflectors by vertical fluid expulsion. *Journal of Geophysical Research: Solid Earth*, *97*(B5), 7025-7041.

Hyndman, R., Moore, G., & Moran, K. (1993, 04). Velocity, porosity, and pore-fluid loss from the nankai subduction zone accretionary prism. *Proc., scientific results, ODP, Leg 131, Nankai Trough*, *131*.

Idehara, K., Yabe, S., & Ide, S. (2014). Regional and global variations in the temporal clustering of tectonic tremor activity. *Earth, Planets and Space*, *66*, 66.

Janiszewski, H. A., Gaherty, J. B., Abers, G. A., Gao, H., & Eilon, Z. C. (2019a, 01). Amphibious surface-wave phase-velocity measurements of the Cascadia subduction zone. *Geophysical Journal International*, *217*(3), 1929-1948.

Janiszewski, H. A., Gaherty, J. B., Abers, G. A., Gao, H., & Eilon, Z. C. (2019b, 01). Amphibious surface-wave phase-velocity measurements of the Cascadia subduction zone. *Geophysical Journal International*, *217*(3), 1929-1948.

Jiao, P., & Alavi, A. H. (2020). Artificial intelligence in seismology: Advent, performance and future trends. *Geoscience Frontiers*, *11*(3), 739-744.

Kamer, Y. (2015). Magnitude frequency, spatial and temporal analysis of large seismicity catalogs: The californian experience. *ETHzürich*.

Kao, H., & Shan, S.-J. (2004). The Source-Scanning Algorithm: mapping the distribution of seismic sources in time and space. *Geophysical Journal International*, *157*(2), 589-594.

Kastner, M. (2001). *Gas hydrates in convergent margins: Formation, occurrence, geochemistry, and global significance*. American Geophysical Union (AGU).

Kennett, B. L. N., Engdahl, E. R., & Buland, R. (1995). Constraints on seismic velocities in the Earth from traveltimes. *Geophysical Journal International*, *122*(1), 108-124.

Knuth, D. E. (1990). Arthur lee samuel, 1901-1990. *TUGboat*, *11*(4).

Kong, Q., Trugman, D. T., Ross, Z. E., Bianco, M. J., Meade, B. J., & Gerstoft, P. (2018, 11). Machine Learning in Seismology: Turning Data into Insights. *Seismological Research Letters*, *90*(1), 3-14.

Kvenvolden, K. A. (1994). Natural gas hydrate occurrence and issues. *Annals of the New York Academy of Sciences*, *715*(1), 232-246.

Käufl, P., P. Valentine, A., W. de Wit, R., & Trampert, J. (2016, 03). Solving probabilistic inverse problems rapidly with prior samples. *Geophysical Journal International*, *205*(3), 1710-1728.

Langet, N., Maggi, A., Michelini, A., & Brenguier, F. (2014, 01). Continuous Kurtosis-Based Migration for Seismic Event Detection and Location, with Application to Piton de la Fournaise Volcano, La Réunion. *Bulletin of the Seismological Society of America*, *104*(1), 229-246.

Lapedes, A., & Farber, R. (1988). *How neural nets work*.

Laske, G., Collins, J. A., Wolfe, C. J., Solomon, S. C., Detrick, R. S., Orcutt, J. A., ... Hauri, E. H. (2009). Probing the hawaiian hot spot with new broadband ocean bottom instruments. *Eos, Transactions American Geophysical Union*, *90*(41), 362-363.

Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural*

Networks, *6*(6), 861-867.

Li, L., Tan, J., Schwarz, B., Staněk, F., Poiata, N., Shi, P., ... Gajewski, D. (2020). Recent advances and challenges of waveform-based seismic location methods at multiple scales. *Reviews of Geophysics*, *58*(1), e2019RG000667.

Lomax, A., Satriano, C., & Vassallo, M. (2012). Automatic Picker Developments and Optimization: FilterPicker—a Robust, Broadband Picker for Real-Time Seismic Monitoring and Earthquake Early Warning. *Seismological Research Letters*, *83*(3), 531-540.

Longuet-Higgins, M. S. (1950). A theory of the origin of microseisms. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, *243*(857), 1-35.

MacKay, D. J. C. (1992). The evidence framework applied to classification networks. *Neural Computation*, *4*(5), 720-736.

MacKay, M. E. (1995). Structural variation and landward vergence at the toe of the oregon accretionary prism. *Tectonics*, *14*(6), 1309-1320.

Marone, F., & Romanowicz, B. (2007, 07). Non-linear crustal corrections in high-resolution regional waveform seismic tomography. *Geophysical Journal International*, *170*(1), 460-467.

McNamara, D. E., & Buland, R. P. (2004). Ambient Noise Levels in the Continental United States. *Bulletin of the Seismological Society of America*, *94*(4), 1517-1527.

Meier, M., Ross, Z. E., Ramachandran, A., Balakrishna, A., Nair, S., Kundzicz, P., ... Yue, Y. (2019). Reliable real-time seismic signal/noise discrimination with machine learning. *Journal of Geophysical Research: Solid Earth*, *124*(1), 788-800.

Meier, U., Curtis, A., & Trampert, J. (2007). Global crustal thickness from neural network inversion of surface wave data. *Geophysical Journal International*, *169*(2), 706-722.

Ming-Kuei Hu. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, *8*(2).

Minksy, M., & Papert, S. A. (1969). *Perceptrons: An introduction to computational geometry.* MIT Press.

Mitchell, T. M. (1997).
In *Machine learning* (p. 2). McGraw Hill.

Montagner, J.-P., & Jobert, N. (1988). Vectorial tomography—ii. application to the indian ocean. *Geophysical Journal*, *94*(2), 309-344.

Mosher, S., & Audet, P. (2017). Recovery of P Waves from Ambient-Noise Interferometry of Borehole Seismic Data around the San Andreas Fault in Central California. *Bulletin of the Seismological Society of America*, *108*(1), 51-65.

Mosher, S., & Audet, P. (2020). Automatic detection and location of seismic events from time-delay projection mapping and neural network classification. *Journal of Geophysical Research: Solid Earth*.

Mosher, S., Eilon, Z., Janiszewski, H., & Audet, P. (2021). Probabilistic inversion of seafloor compliance for oceanic crustal shear velocity structure using mixture density neural networks. *Geophysical Journal International*, *GJI-S-20-1152*.

Mousavi, S. M., Zhu, W., Sheng, Y., & Beroza, G. C. (2019). Cred: A deep residual network of convolutional and recurrent units for earthquake signal detection. *Sci Rep 9, 10267*.

Métaxian, J.-P., Lesage, P., & Valette, B. (2002). Locating sources of volcanic tremor and emergent events by seismic triangulation: Application to arenal volcano, costa rica. *Journal of Geophysical Research: Solid Earth*, *107*(B10), ECV 10-1-ECV 10-18.

Nadeau, R. M., & Dolenc, D. (2005). Nonvolcanic tremors deep beneath the san andreas fault. *Science*, *307*(5708), 389–389.

Obara, K. (2002). Nonvolcanic deep tremor associated with subduction in southwest japan. *Science*, *296*(5573), 1679–1681.

Perol, T., Gharbi, M., & Denolle, M. (2018). Convolutional neural network for earthquake detection and location. *Science Advances*, *4*(2).

Poiata, N., Satriano, C., Vilotte, J.-P., Bernard, P., & Obara, K. (2016). Multiband array detection and location of seismic sources recorded by dense seismic networks. *Geophysical Journal International*, *205*(3), 1548-1573.

Poiata, N., Vilotte, J.-P., Bernard, P., Satriano, C., & Obara, K. (2018, 02). Imaging different components of a tectonic tremor sequence in southwestern Japan using an automatic statistical detection and location method. *Geophysical Journal International*, *213*(3), 2193-2213.

Polet, J., & Kanamori, H. (2000). Shallow subduction zone earthquakes and their tsunamigenic potential. *Geophysical Journal International*, *142*(3), 684-702.

Prahl, F., Ertel, J., Goni, M., Sparrow, M., & Eversmeyer, B. (1994). Terrestrial organic carbon contributions to sediments on the washington margin. *Geochimica et Cosmochimica Acta*, *58*(14), 3035-3048. doi: https://doi.org/10.1016/0016-7037(94)90177-5

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *CoRR*, *abs/1710.05941*.

Reynen, A., & Audet, P. (2017). Supervised machine learning on a network scale: application to seismic event classification and detection. *Geophysical Journal International*, *210*(3), 1394-1409.

Rojas, O., Otero, B., Alvarado, L., Mus, S., & Tous, R. (2019). Artificial neural networks

as emerging tools for earthquake detection. *Computación y Sistemas*, *23*, 335-350.

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

Ross, Z. E., Trugman, D. T., Hauksson, E., & Shearer, P. M. (2019). Searching for hidden earthquakes in southern california. *Science*, *364*(6442), 767–771.

Ruan, Y., Forsyth, D. W., & Bell, S. W. (2014). Marine sediment shear velocity structure from the ratio of displacement to pressure of rayleigh waves at seafloor. *Journal of Geophysical Research: Solid Earth*, *119*(8), 6357-6371.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533-536.

Rychert, C. A., Harmon, N., & Tharimena, S. (2018). Scattered wave imaging of the oceanic plate in cascadia. *Science Advances*, *4*(2).

Saffer, & Tobin. (2011). Hydrogeology and mechanics of subduction zone forearcs: Fluid flow and pore pressure. *Annual Review of Earth and Planetary Sciences*, *39*(1), 157-186.

Saffer, & Wallace. (2015). The frictional, hydrologic, metamorphic and thermal habitat of shallow slow earthquakes. *Nature Geoscience*, *8*, 594–600.

Saikia, U., Rychert, C., Harmon, N., & Kendall, J. (2020). Sediment structure at the equatorial mid-atlantic ridge constrained by seafloor admittance using data from the pi-lab experiment. *Marine Geophysical Research*, *41*(3).

Sambridge, M., & Mosegaard, K. (2002). Monte carlo methods in geophysical inverse problems. *Reviews of Geophysics*, *40*(3), 3-1-3-29.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal*, *3*(3).

Saragiotis, C. D., Hadjileontiadis, L. J., & Panas, S. M. (2002). Pai-s/k: A robust automatic seismic p phase arrival identification scheme. *IEEE Transactions on Geoscience and Remote Sensing*, *40*(6), 1395-1404.

Sauter, A. W., Dorman, L. M., & Schreiner, A. E. (1986). *A study of sea floor structure using ocean bottom shots and receivers* (T. Akal & J. M. Berkson, Eds.). Boston, MA: Springer US.

Seats, K. J., Lawrence, J. F., & Prieto, G. A. (2012). Improved ambient noise correlation functions using welch's method. *Geophysical Journal International*, *188*(2), 513-523.

Seising, R. (2018). The emergence of fuzzy sets in the decade of the perceptron—lotfi a. zadeh's and frank rosenblatt's research work on pattern classification. *Mathematics*, *6*(7).

Shahraeeni, M. S., Curtis, A., & Chao, G. (2012). Fast probabilistic petrophysical mapping

of reservoirs from 3d seismic data. *GEOPHYSICS*, *77*(3), O1-O19.

Shelly, D. R., Beroza, G. C., Ide, S., & Nakamula, S. (2006). Low-frequency earthquakes in shikoku, japan, and their relationship to episodic tremor and slip. *Nature*, *442*, 188-191.

Shiuly, A., Roy, N., & Sahu, R. B. (2020). Prediction of peak ground acceleration for himalayan region using artificial neural network and genetic algorithm. *Arabian Journal of Geosciences*, *13*.

Solheim, A., Forsberg, C., Yang, S., Kvalstad, T., Longva, O., & Rise, L. (2006). The role of geological setting and depositional history in offshore slope instability. *Offshore Technology Conference, Houston, Texas*.

Sorrells, G. G., & Goforth, T. T. (1973, 10). Low-frequency earth motion generated by slowly propagating partially organized pressure fields. *Bulletin of the Seismological Society of America*, *63*(5), 1583-1601.

Stewart, S. W. (1977). Real-time detection and location of local seismic events in central California. *Bulletin of the Seismological Society of America*, *67*(2), 433-452.

Straume, E. O., Gaina, C., Medvedev, S., Hochmuth, K., Gohl, K., Whittaker, J. M., ... Hopper, J. R. (2019). Globsed: Updated total sediment thickness in the world's oceans. *Geochemistry, Geophysics, Geosystems*, *20*(4), 1756-1772. doi: https://doi.org/10.1029/2018GC008115

Sutton, G. H., Maynard, G. L., & Hussong, D. M. (2013). *The structure and physical properties of the earth's crust*. American Geophysical Union (AGU).

Tang, L., Zhang, M., & Wen, L. (2020). Support vector machine classification of seismic events in the tianshan orogenic belt. *Journal of Geophysical Research: Solid Earth*, *125*(1), e2019JB018132. (e2019JB018132 2019JB018132)

Tary, J. B., Herrera, R. H., Han, J., & van der Baan, M. (2014). Spectral estimation—what is new? what is next? *Reviews of Geophysics*, *52*(4), 723-749.

Toomey, D. R., Bell, S. W., Bromirski, P. D., Carlson, R. L., Chen, X., Collins, J. A., ... Wilcock, W. S. (2014, June). The cascadia initiative: A sea change in seismological studies of subduction zones. *Oceanography*, *27*(2), 138-150.

Valentine, A. P., & Woodhouse, J. H. (2010, 08). Approaches to automated data selection for global seismic tomography. *Geophysical Journal International*, *182*(2), 1001-1012.

Van Der Malsburg, C. (1986). Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. In G. Palm & A. Aertsen (Eds.), *Brain theory* (pp. 245–248). Berlin, Heidelberg: Springer Berlin Heidelberg.

Vanneste, M., Forsberg, C. F., Glimsdal, S., Harbitz, C. B., Issler, D., Kvalstad, T. J., ...

Nadim, F. (2013). *Submarine landslides and their consequences: What do we know, what can we do?* (C. Margottini, P. Canuti, & K. Sassa, Eds.). Berlin, Heidelberg: Springer Berlin Heidelberg.

Vannucchi, P., Spagnuolo, E., Aretusini, S., Di Tori, G., Ujiie, K., Tsutsumi, A., & Nielsen, S. (2017). Past seismic slip-to-the-trench recorded in central america megathrust. *Nature Geoscience*, *10*, 935–940.

Vassallo, M., Satriano, C., & Lomax, A. (2012). Automatic picker developments and optimization: A strategy for improving the performances of automatic phase pickers. *Seismological Research Letters*, 541-554.

Wasserman, P. D., & Scwhartz, T. (1988). Neural networks, ii: What are they and why is everybody so interested in them now? *IEEE Expert: Intelligent Systems and Their Applications*, *3*(1), 10–15.

Webb, S. C., Zhang, X., & Crawford, W. (1991). Infragravity waves in the deep ocean. *Journal of Geophysical Research: Oceans*, *96*(C2), 2723-2736.

Weiss, E. A. (1992). Biographies: Eloge: Arthur lee samuel (1901-90). *IEEE Annals of the History of Computing*, *14*(3), 55-69.

Whitmarsh, R. B., & Miles, P. R. (1991). *In situ measurements of shear-wave velocity in ocean sediments* (J. M. Hovem, M. D. Richardson, & R. D. Stoll, Eds.). Dordrecht: Springer Netherlands.

Willoughby, E. C., & Edwards, R. N. (2000). Shear velocities in cascadia from seafloor compliance measurements. *Geophysical Research Letters*, *27*(7), 1021-1024.

Willoughby, E. C., Latychev, K., Edwards, R. N., Schwalenberg, K., & Hyndman, R. (2008). Seafloor compliance imaging of marine gas hydrate deposits and cold vent structures. *Journal of Geophysical Research: Solid Earth*, *113*(B7).

Withers, M., Aster, R., Young, C., Beiriger, J., Harris, M., Moore, S., & Trujillo, J. (1998). A comparison of select trigger algorithms for automated global seismic phase and event detection. *Bulletin of the Seismological Society of America*, *88*(1), 95-106.

Yamamoto, T., & Torii, T. (1986, 04). Seabed shear modulus profile inversion using surface gravity (water) wave-induced bottom motion. *Geophysical Journal International*, *85*(2), 413-431.

Young, I. T., & van Vliet, L. J. (1995). Recursive implementation of the gaussian filter. *Signal Processing*, *44*(2), 139 - 151.

Yuan, T., Spence, G. D., & Hyndman, R. (1994). Seismic velocities and inferred porosities in the accretionary wedge sediments at the cascadia margin. *Journal of Geophysical Research: Solid Earth*, *99*(B3), 4413-4427.

Zha, Y., & Webb, S. C. (2016). Crustal shear velocity structure in the southern lau basin constrained by seafloor compliance. *Journal of Geophysical Research: Solid Earth*, *121*(5), 3220-3237.

Zhihu Huang, & Jinsong Leng. (2010). Analysis of hu's moment invariants on image scaling and rotation. In *2010 2nd international conference on computer engineering and technology* (Vol. 7, p. V7-476-V7-480).

Zhu, J., Canales, J. P., Han, S., Carbotte, S. M., Arnulf, A., & Nedimović, M. R. (2020). Vp/vs ratio of incoming sediments off cascadia subduction zone from analysis of controlled-source multicomponent obs records. *Journal of Geophysical Research: Solid Earth*, *125*(6), e2019JB019239.

Zhu, W., & Beroza, G. C. (2019, 10). PhaseNet: a deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International*, *216*(1), 261-273.

E

Acronyms

ATaCR Automated Tilt and Compliance Removal

BA Balanced Accuracy

CCF Cross Correlation Function

CF Characteristic Function

CICE Cascadia Initiative Community Experiment

CSZ Cascadia Subduction Zone

FN False Negative

FP False Positive

GMM Gaussian Mixture Model

JdF Juan de Fuca

LCC Local Cross Correlation

LFE Low Frequency Earthquake

LTA Long Term Average

MCMC Markov Chain Monte Carlo

MDN Mixture Density Network

ML Local Magnitude

MLP Multilayer Perceptron

NA North America

OBS Ocean-Bottom Seismometer

PA Pacific

PDF Probability Distribution Function

PPSD Probabilistic Power Spectral Density

PSD Power Spectral Density

ReLU Rectified Linear Unit

SLF Spatial Likelihood Function

STA Short Term Average

TDE Time Delay Estimate

TDOA Time Difference of Arrival

TN True Negative

TP True Positive

TTD Travel Time Difference

USGS United Stated Geological Survey

$\mathbf{V_P}$ P-Wave Velocity

$\mathbf{V_S}$ Shear-Wave Velocity