APPLICATIONS OF DEEP LEARNING IN SEISMOLOGY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF GEOPHYSICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Weiqiang Zhu
August 2021

This dissertation is online at: https://purl.stanford.edu/hf729kf3190

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Greg Beroza, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Eric Dunham

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

William Ellsworth

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Seismic waveforms contain valuable information about earthquakes and earth structure. Dense seismic monitoring networks are deployed across the world and collect massive amounts of observational data; however, the sheer amount of this data poses a challenge for seismic data processing and analysis. Developing effective and efficient algorithms and models for seismic data analysis is thus important for studying earthquake physics, improving earthquake forecasting, and mitigating earthquake hazards. In the work presented in this thesis, I have explored a promising approach to advancing earthquake detection and inversion using deep learning. Deep learning has in recent years achieved super-human performance in solving many challenging problems, such as image recognition, protein folding, and playing Go or Atari games. In contrast to conventional algorithms that rely on expert-designed features and decision rules, deep neural networks can automatically learn characteristic features and statistical criteria from large training data sets accompanied by manual labels. The huge amount of archived seismic data collected in the past few decades provides excellent training resources for deep learning, making it a very promising approach to studying seismic signals and addressing research challenges, such as detecting hidden small earthquakes whose numbers dominate earthquake catalogs. However, at the time I started my PhD research, there was limited work on deep learning applications in seismology. To explore the potential of deep learning in seismology, I focused on two directions: First, I developed modular deep learning algorithms to improve earthquake monitoring including signal denoising, phase picking, phase association, and earthquake detection. The results of my work show that these deep learning algorithms significantly improve earthquake monitoring by detecting up to orders of magnitude more small earthquakes than are detected in standard catalogs, and by doing so reveal a far more detailed picture of earthquake sequences and fault structures. In addition, I utilized cloud computing to scale-up our detection workflow to solve the big data challenge in mining large archived data sets. Second, I studied the connection between deep learning optimization and conventional seismic inversion, such as full-waveform inversion. I developed a new inversion approach to solving seismic inverse problems using automatic differentiation and proposed a new regularization method by parameterizing inversion targets using neural networks. The results show that the rapid development of deep learning frameworks and neural network architectures can improve seismic inversion to constrain physical

parameters of interest from detected seismic waveforms. In all, these applications of deep learning to both earthquake monitoring and inverting underlying parameters demonstrate that deep learning is an effective tool to improve the extraction of useful information from seismic data and that it holds great promise for future developments in seismology.

Acknowledgments

The past five years at Stanford have been a wonderful experience for me. In the spring of 2016, I received the best offer I could ever imagine to be co-advised by Greg Beroza and Eric Dunham at Stanford, which opened my journey to explore research that I am passionate about and to pursue both a PhD major in Geophysics and a PhD minor in Computer Science. I would like to thank many people who have made this journey an invaluable experience for me.

First and foremost, I want to thank my advisor, Greg Beroza, who guided me through my PhD studies to become an advanced researcher. Greg always encourages me to aim high and look ten years ahead. Although I did not accomplish every goal we discussed, I have learned and achieved a lot in the process of pursuing these ambitious goals. When I pitched an idea to Greg, he was always excited and pushed me to think deep into experiments and applications to verify the idea. Greg set a role model for me with his passion for learning new knowledge together with his students. Greg patiently helped me improve my terrible drafts and gave me suggestions on academic writing. Greg has been very supportive of my decisions in exploring both research directions of machine learning and numerical simulation, finding industry internship opportunities, pursuing a PhD Minor degree, and applying for postdoc and faculty jobs. He kept encouraging me when I was not confident about myself. I am grateful to have had an advisor as supportive as Greg.

I want to thank Eric Dunham for spending a lot of time teaching me numerical simulation and helping me design and complete my secondary research project. I was fortunate to have Eric as my co-advisor in the first two years of my PhD. Eric helped me start PhD research smoothly in every aspect: discussing research projects, teaching me numerical simulation, recommending important courses, and holding weekly meetings to solve my problems. Although I decided to pursue machine learning as my thesis direction after the first year, Eric remained very supportive and helped me re-design the project scope to meet the requirements of a secondary research project. Without Eric's help, I could not have finished the project, let alone publish a high-quality paper. Eric's passion for teaching courses at Stanford and designing outreach events for elementary schools has also motivated me to become a good teacher. I am grateful for the opportunity to have Eric as my co-advisor.

I want to thank William Ellsworth for being on my committee throughout the past five years and giving me many valuable suggestions. I am very lucky to have had group meetings with Bill

during my PhD studies. Bill always uses questions to guide us to think deeply and broadly on the discussion topic. Bill is very patient in answering my questions, suggesting potential solutions, and even providing hands-on help to look at data together. I am grateful to Bill for his help in completing my PhD research.

I would like to thank Philip Levis and Kevin Arrigo for agreeing to join my defense committee. Phil helped me pursue a PhD minor degree in Computer Science. Kevin helped chair my defense exam. I enjoyed his class where I learned about radar and remote sensing. I want to thank Mary McDevitt of the Technical Communicaton Program at Stanford for teaching me about academic writing and helping me prepare my thesis.

I would like to thank many collaborators whom I have had the pleasure to work with and learn from: Mostafa Mousavi, Kailai Xu, Kai Sheng Tai, Yixiao Sheng, Yongsoo Park, Miao Zhang, Yen Joe Tan, Ian McBrearty, Chengping Chai, Songqiao Wei, Yang Lei, Lise Retailleau, Alvin Brian Hou, Robert Yang, Avoy Datta, Daniel J. Wu. In particular, Mostafa Mousavi taught me how to design a research project not just for a paper but as a comprehensive plan to address every challenge. Kailai Xu taught me how to implement numerical simulation and design rigorous testing and debugging procedures. Kai Sheng Tai taught how to design complex neural network architects and systematically decompose and build each module. I will miss the time we spent together coding and working on projects.

I would like to thank Marc Stogaitis's team and my mentor Alexei Barski during my internship at Google. Learning how a team collaborates and advances projects forward in industry was a valuable lesson for me.

I want also to thank my friends who have made the life of studying abroad pleasant. I want to thank my roommate Fanming Dong; labmates Yixiao Sheng, Chao Liang, Dongzhuo Li, and Tianze Liu; friends I was fortunate to get to know–Ye Yuan, Shifan Mao, Chao Chen, and Chen Liang; and my high-school classmates who live in the Bay Area. They helped me survive the difficult times when I first came to the US, looked for internships, and struggled in my research. I want to thank my student mentor Cansu Culha and everyone in Greg's and Eric's groups for helping me through five years of PhD studies.

And finally, I would like to thank my parents Jinquan Chen and Qiaozhen Zhu and my wife Mingyang Liu for their unconditional love. They have provided me with the strongest emotional support. Making them proud is the most important force driving me forward.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Large earthquakes can cause severe damage and casualties. Understanding earthquake mechanisms, improving earthquake forecasting, and characterizing earthquake hazards are important but very challenging tasks. Improved earthquake monitoring would provide fundamental information for illuminating earthquake activity and understanding the underlying physics. There are two main approaches to improving earthquake monitoring. One approach is to improve observational hardware. For example, dense networks of instruments, such as seismometers, GPS, and InSAR, can be deployed to improve observational sensitivity to weak earthquake-related signals. The other approach is to improve processing software, particularly developing data mining algorithms using high performance computing to extract detailed information from from a large amount of observational data.

Modern earthquake seismology has greatly benefited from dense seismic monitoring networks, powerful computational resources, and other emerging observational techniques. Massive seismic data volumes have been archived over the last few decades and the growth rate is continuously accelerating with improvements in sensor technology. For example, more than 700 TiB data have been archived at the IRIS (Incorporated Research Institutions for Seismology) data center[1]. New sensing technologies, such as distributed acoustic sensing (DAS) which uses fiber-optic cables to provide sensors every few meters, are generating very dense measurements of ground motion and huge amounts of data (Zhan, 2020). Advanced data processing and analysis techniques thus play an important role in extracting new insights from these large data sets to make breakthroughs in earthquake characterization and forecasting. For example, the detection of aftershocks is a commonly used way to study earthquake processes and fault zone structure. Based on the Gutenberg-Richter (GR) relationship (Gutenberg & Richter, 1944), for which the b-value is typically $\sim$1, if we could improve earthquake monitoring algorithms to detect one or two magnitude smaller earthquakes, we would be able to detect a factor of ten more small earthquakes. This large number of small earthquakes would illuminate the fault zone structures with unprecedented spatial resolution and

[1] https://ds.iris.edu/data/distribution/

may provide insights into how large earthquakes happen.

When I started my PhD study in 2016, several breakthroughs occurred in deep learning. AlphaGo defeated a world champion in the board game Go (Silver et al., 2016); deep convolutional neural networks achieved a lower error rate than humans on ImagNet, a large-scale object recognition benchmark dataset (He et al., 2016; Russakovsky et al., 2015); and deep recurrent neural networks achieved a performance similar to that of humans in language translation (Wu et al., 2016). As illustrated by these cases, deep learning presents a new data-driven approach to transform large data sets into effective models. By training on data sets accompanied by manual labels, deep neural networks can achieve performance similar to that of human analysts, or even super-human performance. This data-driven approach can be particularly effective for previous challenging tasks, such as recognizing a cat in an image, which is easy for humans, but difficult in traditional algorithms that require explicit mathematical formulas of specific challenging targets. Deep neural networks, in contrast, can implicitly learn the knowledge from manual labels to solve the corresponding classification/regression tasks. The weights of deep neural networks, up to millions and billions parameters, are optimized during training to fit manual labels accurately. Once trained, the neural networks can efficiently process the remaining large amounts of unlabeled or unanalyzed data, from which we can extract much more information and grain new insights into related research problems. Deep learning thus has the potential to be a very effective data-driven approach for studying earthquake-related signals. However, in 2016, there were no applications of deep learning to seismological problems, even with the availability of many hand-picked labels from years of manual inspection of seismic waveforms. These manual labels provide an outstanding opportunity to develop deep learning models (e.g., deep neural networks) for seismic signals, because the performance of deep neural networks can improve with the increasing amount of training data. In addition to the effectiveness of deep neural networks, the optimization methods for training deep neural networks is also closely connected to the adjoint-state method used in seismic inversion. The rapidly developing deep learning frameworks, which effectively use computational power of CPU, GPU or TPU, can also benefit seismic inversion to constrain underlying physical parameters.

In the work presented in this thesis, I was motivated by the tremendous opportunity that deep learning could offer and thus explored that potential in seismic applications. It should be noted that although the high precision and efficiency of deep neural networks make them promising tools to process large volumes of seismic data, detect small earthquakes missing from standard catalogs, and reveal a detailed picture of earthquake activities and fault structures, to realize the potential of deep learning for seismological problems, deep learning must be applied carefully. To do this, I carefully studied various issues in building the necessary training data, in designing effective neural network architecture, in data augmentation for improved performance, and in training strategies to ensure good performance and generalization of deep neural networks. Specifically, as the subsequent chapters of this thesis detail, I explored the potential applications of deep learning in seismology

from two directions: earthquake monitoring and inversion problems. First, I solved the denoising problem by applying deep learning to separate signal from noise in noisy seismic waveforms to improve subsequent data analysis (Chapter 2). Second, I solved the phase picking problem by applying deep learning to recognize the two fundamental seismic phases, the P- and S-phases, which are important for detecting and locating earthquakes (Chapter 3). Third, I solved the phase association problem using an unsupervised clustering method to aggregate the picked P- and S-phases across multiple stations in a seismic network to detect earthquakes and filter out false positives (Chapter 5). Next, based on these algorithms, I developed a detection workflow with cloud computing for both real-time earthquake monitoring and mining archived earthquake data sets (Chapter 6). Compared with conventional algorithms, these learning-based models significantly improve earthquake detection performance and have been applied to studying complex earthquake sequences at many regions over the world, such as Guy-Greenbrier AR (Park et al., 2020), Ridgecrest CA (Liu et al., 2020), Central Italy (Tan et al., 2021), and China (Su et al., 2021). Furthermore, I developed a novel end-to-end approach to solve the earthquake detection problem (Chapter 7) that aimed to simultaneously optimize phase detection and association to improve detection performance. Finally, I explored the combination of deep learning and seismic inversion to constrain underlying physical parameters like the earthquake source time function and subsurface wave velocity models. I developed a new approach to solve seismic inversion problems using a deep learning framework (Chapter 8) based on the connection between the reverse-mode automatic differentiation used in deep learning and the adjoint-state methods used in seismic full-waveform inversion. I explored an approach to parameterize physical parameter models using neural networks, which impose spatial regularization and improve seismic inversion stability (Chapter 9). In all, these efforts have explored the strategic intersection of the fields of deep learning and seismology, demonstrating both the effectiveness of deep learning in earthquake monitoring and the close connection between techniques used in deep learning and seismic inversion. These deep learning models have advanced the state-of-art performance of earthquake detection. The successful applications of deep learning to a wide range of seismic problems demonstrates its effectiveness and potential for solving challenging tasks and sparking new insights in future research.

Chapter 2

DeepDenoiser: Seismic Signal Denoising and Decomposition Using Deep Neural Networks

Frequency filtering is widely used in routine processing of seismic data to improve the signal-to-noise ratio (SNR) of recorded signals and by doing so to improve subsequent analyses. In this study we develop a new denoising/decomposition method, DeepDenoiser, based on a deep neural network. This network is able to learn simultaneously a sparse representation of data in the time-frequency domain and a non-linear function that maps this representation into masks that decompose input data into a signal of interest and noise (defined as any non-seismic signal). We show that DeepDenoiser achieves impressive denoising of seismic signals even when the signal and noise share a common frequency band. Because the noise statistics are automatically learned from data and require no assumptions, our method properly handles white noise, a variety of colored noise, and non-earthquake signals. DeepDenoiser can significantly improve the SNR with minimal changes in the waveform shape of interest, even in presence of high noise levels. We demonstrate the effect of our method on improving earthquake detection. There are clear applications of DeepDenoiser to seismic imaging, micro-seismic monitoring, and preprocessing of ambient noise data. We also note that potential applications of our approach are not limited to these applications or even to earthquake data, and that our approach can be adapted to diverse signals and applications in other settings.

2.1 Introduction

Recorded seismic signals are inevitably contaminated by noise and non-seismic signals from various sources including: ocean waves, wind, traffic, instrumental noise, electrical noise, etc. Spectral filtering (usually based on the Fourier transform) is frequently used to suppress noise in routine seismic data processing; however, this approach is not effective when noise and seismic signal occupy the same frequency range. Moreover, selecting optimal parameters for filtering is non-intuitive, typically varies with time, and may strongly alter the waveform shape such that it degrades the analysis that follows.

Due to these limitations, numerous efforts have been made to develop more effective noise suppression in seismic data (Abma & Claerbout, 1995; Bonar & Sacchi, 2012; Chen, 2018; Chen & Fomel, 2015; Chen, Huang, et al., 2016; Chen, Zhang, et al., 2016; Chen, Zhou, et al., 2017; Huang et al., 2018; Huang, Wang, Chen, et al., 2016; Huang, Wang, Yuan, et al., 2016; Huang et al., 2017; Liu et al., 2012; Liu et al., 2015; Naghizadeh, 2012; Oropeza & Sacchi, 2011; Tian et al., 2014; Zhou, Li, et al., 2017; Zhou, Shi, et al., 2017; Zhu et al., 2015). Methods based on time-frequency denoising (Donoho & Johnstone, 1995; Donoho & Johnstone, 1994) form a large class of seismic denoising techniques. In this approach, noisy time series are first transformed into the time-frequency domain using a time-frequency transform, such as a wavelet transform (Cao & Chen, 2005; Gaci, 2014; Liu, Cao, et al., 2016; Mousavi & Langston, 2017; Mousavi, Langston, & Horton, 2016; Mousavi, Langston, Mostafa Mousavi, et al., 2016), Short Time Fourier Transform (STFT) (Mousavi & Langston, 2016a), S-transform (Tselentis et al., 2012), curvelet transform (Hennenfent & Herrmann, 2006; Neelamani et al., 2008; Tang & Ma, 2011), dreamlet transform (Wang et al., 2015), contourlet transform (Shan et al., 2009), shearlet transform (Zhang & van der Baan, 2018), and empirical mode decomposition (Bekara & Van der Baan, 2009; Chen, Xie, et al., 2017; Chen & Ma, 2014; Han & van der Baan, 2015; Liu et al., 2014). The resulting time-frequency coefficients are modified (thresholded) to attenuate the coefficients associated with noise and to find an estimate of the signal coefficients. The modified coefficients are inverse transformed back into the time domain to reconstruct the denoised signal. The basic idea is to promote sparsity by transforming seismic data to other domains where the signal can be represented by a sparse set of features so that signal and noise can be separated more easily.

These methods can suppress the noise even when it occupies the same frequency range as the signal, however, the choice of a suitable thresholding function to map the noisy data into optimally denoised signal can be challenging. Denoising performance of time-frequency methods can be improved in two ways: either by using a more effective sparse representation of the data, or by using a more flexible and powerful mapping function. Machine learning techniques, such as dictionary learning, have been used to improve seismic denoising by learning better sparse representations (Chen, 2017; Chen, Ma, & Fomel, 2016; Liu et al., 2018; Siahsar et al., 2017). The focus of this paper is on improving both sparsity and the mapping function using deep learning.

Deep learning (Goodfellow et al., 2016; LeCun et al., 2015) is a powerful machine learning technique that can learn extremely complex functions through neural networks. Deep learning has been shown to be a powerful tool for learning the characteristics of seismic data (DeVries et al., 2018; Mousavi, Zhu, Sheng, et al., 2019; Perol et al., 2018; Ross, Meier, & Hauksson, 2018; Ross, Meier, Hauksson, & Heaton, 2018; Ross, Yue, et al., 2019; Zheng et al., 2018; Zhu & Beroza, 2018).

In this study we present DeepDenoiser, a novel time-frequency denoising method using deep neural networks. This network is able to simultaneously learn a sparse representation of the input data and a high-dimensional non-linear function that maps this representation into desired masks from the training data set. Given input data, DeepDenoiser produces two individual masks, one for the seismic signal and the other for the noise signal. The masks are further used to extract the corresponding waveforms from the input data. We use earthquake seismograms manipulated with various types of noise and non-earthquake signals recorded by seismic stations to train the network and demonstrate its performance. We apply the method to unseen noisy seismograms to illustrate its generalizability, to compare its performance with other denoising methods, and to document its ability to improve earthquake detection results.

2.2 Method

In the time-frequency domain, we represent recorded data, $Y(t, f)$, as the superposition of seismic signal, $S(t, f)$, and some additive natural/instrumental noise or non-seismic signals, collectively termed noise, $N(t, f)$:

$$Y(t, f) = S(t, f) + N(t, f) \tag{2.1}$$

The objective of denoising is to estimate the underlying seismic signal (i.e., the denoised signal), $\hat{S}(t)$, from its noise contaminated version that minimizes the expected error between the true and estimated signal:

$$error = E||\hat{S}(t) - S(t)||_2^2 \tag{2.2}$$

where $\hat{S}(t) = TFT^{-1}\{M(t, f)Y(t, f)\}$, TFT^{-1} is the inverse time-frequency transform, $Y(t, f)$ is the time-frequency representation of noisy data, and $M(t, f)$ is a function that maps $Y(t, f)$ to a time-frequency representation of the estimated signal. Donoho and Johnstone (1994) showed that this mapping can be carried out through a simple thresholding in a sparse representation where the thresholding value can be estimated from noise level assuming a Gaussian distribution.

Here, we cast the problem as a supervised learning problem where a deep neural network will learn a sparse representation of the data to generate an optimal mapping function based on training samples of signal and noise data distribution. This is analogous to Wiener deconvolution as follows. We define our mapping functions as two individual masks, $M_S(t, f)$ and $M_N(t, f)$ for signal and

noise respectively:

$$M_S(t,f) = \frac{1}{1+\frac{|N(t,f)|}{|S(t,f)|}} \tag{2.3}$$

$$M_N(t,f) = \frac{\frac{|N(t,f)|}{|S(t,f)|}}{1+\frac{|N(t,f)|}{|S(t,f)|}} \tag{2.4}$$

Each mask has the same size as the input time-frequency representation, $Y(t,f)$, and contains values between 0 and 1 that attenuate either noise or signal in time-frequency space.

Inspired by the capability of auto-encoders in learning a sparse representation of data with respect to an optimization objective, we designed our network in the form of a series of fully convolutional layers with descending and then ascending sizes (Fig. 2.1). Following Ronneberger et al. (2015), we use skip connections to improve the convergence of training and prediction performance.

The inputs to the first layer are the imaginary and real parts of the time-frequency coefficients of the data, $Y(t,f)$. In the last layer, masks of signal and noise ($M_S(t,f)$ and $M_N(t,f)$) are provided as labels for training. The input time-frequency coefficients are processed and transformed through a series of 2D convolutional layers with a ReLU (rectified linear unit) activation layer and batch normalization (Ioffe & Szegedy, 2015). The convolution filter size is kept constant (3×3), but the feature space in the first half of the network is gradually reduced using strides of 2×2. These layers act as an effective feature extractor that accelerate learning of a very sparse representation of input data at the bottleneck layer. In the second half of the network, deconvolution (the transpose of the convolution layers) is used to generate a high-dimensional non-linear mapping of this sparse representation into output masks. In the last layer, a softmax normalized exponential function is used to produce masks. Through the training process the network learns both how to construct a sparse representation of data and optimal masks to separate signal from noise by optimizing a loss function (cross-entropy loss function).

Fig. 2.2 shows the data-flow diagram of our method. First, the seismic waveform is transferred into the time-frequency domain. The trained network takes the real and imaginary parts of time-frequency coefficients as the input and produces individual masks for both signal and noise as the outputs. The masks are the targets in optimizing the neural network during training. The estimated time-frequency coefficients of the seismic signal, $\hat{S}(t,f)$, and noise, $\hat{N}(t,f)$, are obtained by applying the associated mask to the imaginary and real parts of the data coefficients, $Y(t,f)$. The final denoised signal and noise are obtained after inverse transforming $\hat{S}(t,f)$ and $\hat{N}(t,f)$ back into the time domain. Instead of defining different features and thresholds manually to enhance signal and attenuate noise, DeepDenoiser automatically learns richer features from semi-real seismic data that allows it to separate signal and noise in the time-frequency domain. Deep learning has the potential to provide a more effective and accurate automatic denoising tool for seismic data preprocessing, which can be applied to challenging tasks such as micro-earthquake detection.

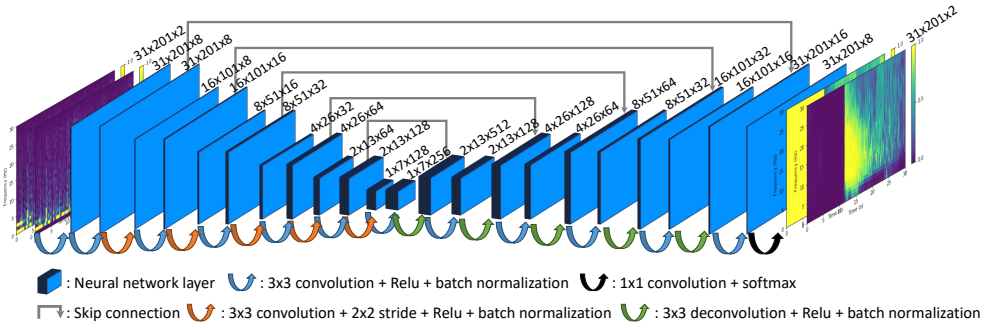


Figure 2.1: Neural network architecture. Inputs are the real and imaginary parts of the time-frequency representation of noisy data. Outputs are two masks for signal and noise extraction. Blue rectangles represent layers inside the neural network. The dimension of each layer is presented above it, and contains "frequency bins × time points × channels". Arrows represent different operations applied to layers. The input data first go through 3×3 convolution layers with 2×2 strides for down-sampling and then go through deconvolution layers (Noh et al., 2015) for up-sampling. Batch normalization and skip connections are used to improve convergence during training. The softmax normalized exponential function is applied in the last layer to predict the masks for signal and noise.

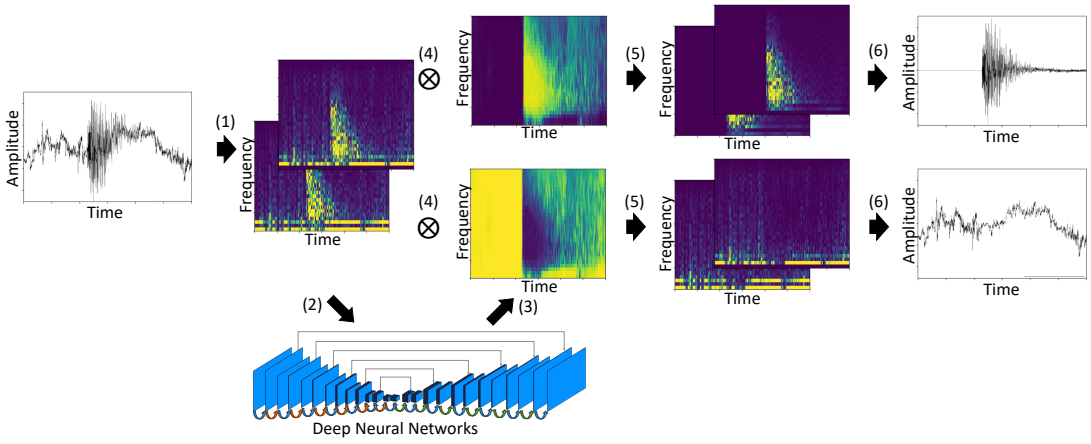


Figure 2.2: Data flow diagram of our denoising method. (1) The noisy data is transformed into the time-frequency domain using Short Time Fourier Transform (STFT). (2) The real and imaginary parts of time-frequency coefficients are fed into our deep neural network. (3) The neural network produces two masks for signal and noise based on input data. (4, 5) The associated masks are applied to the noisy-signal coefficients to estimate the time-frequency coefficients of the seismic signal and noise. (6) The denoised signal and noise in time domain are obtained using inverse STFT.

2.3 Network Training

We use 30-second seismograms recorded by the high broadband channels (HN*) of the North California Seismic Network to train the neural network and test its performance. The dataset consists of 56,345 earthquake waveforms with very high signal-to-noise ratios (SNRs) as the signal samples and 179,233 seismograms associated with various types of non-earthquake waveforms as the noise samples. Both the signal and noise datasets are randomly split into a signal training set with 45104 samples, a signal validation set with 5576 samples, a signal test sets with 5665 samples, a noise training set with 143403 samples, a noise validation set with 17890 samples, and a noise test set with 17940 sample. To construct 'noisy' seismograms for training the neural network, we iterate through the signal training set repeatedly. In each iteration, we first randomly shift every signal sample, then randomly select one noise sample from the noise training set, randomly scale its amplitude, and stack it to the signal sample to generate training data of different SNR levels. This random combination between the signal and noise samples builds up a final training dataset of over millions samples. The Short Time Fourier Transform (STFT) is applied to produce the time-frequency representation of the noisy waveforms. The sampling frequency used in STFT is 100Hz and the length of each segment is 0.3 seconds. We normalize the 2D time-frequency matrices of the constructed noisy waveforms by removing the mean and dividing by the standard deviation. During prediction, the mean and standard deviation are recorded to recover the 2D time-frequency matrices back to the original scale after denoising. The real and imaginary parts are fed to the neural network as two separate channels so that the network is able to learn from both the time and phase information. The prediction targets of the neural network are two masks: one each for signal and noise, constructed with equations (3) and (4). The same procedure is used to generate validation and test sets. The validation set is only used for tuning the hyper-parameters of the network, which helps to identify and prevent over-fitting. The test set is used to document performance. We train the deep neural network using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.001 and a min-batch size of 200.

2.4 Results

2.4.1 Test Set

We use the test set to analyze the final performance and illustrate the denoising results. Fig. 2.3 demonstrates that the network can successfully decompose the noisy inputs with different characteristics into denoised signal and noise. The algorithm can recover denoised signal with high accuracy (Fig. 2.3(b, d)(iv)). As can be seen from the same example, signal leakage is minimal and the waveform shape, frequency content, and amplitude characteristics are well preserved after denoising. These characteristics hold for the extracted noise as well (Fig. 2.3(b, d)(v)).

One advantage of our learning-based denoising method is that the network not only learns the features of seismic signals but also the features of various noise sources. The wide variety of noise sources along with their time-varying signatures makes it very difficult for hand-engineered features to represent each type of noise effectively. However, DeepDenoiser shows the ability to learn a sparse feature representation for different types of noise. Some of them are recognized in our test dataset: The first type of noise is band-limited (Fig. 2.4(a, b)), with relatively strong values within narrow frequency bands; The second type of noise is low-frequency noise (Fig. 2.4(c, d))), which exhibits strong background fluctuations that the signal rides on; The third type of noise is cyclic noise (Fig. 2.5), which is combined with different modes with the frequency band varying with time. The first two types of noise could be effectively attenuated by band-pass or low-pass filtering given an accurate estimate of the noise frequency bands. The third type of noise is challenging for traditional denoising methods because the noise changes with time and its frequency band overlaps with the frequency band of the target signal. For all of these types of noise, DeepDenoiser automatically predicts a mask that adapts to the noise features. The mask not only estimates the noise frequency bands needed for denoising but also reflects the changes of frequency content over time. The denoised signal and the separated noise (Fig. 2.4(b, d)(iv, v) and Fig. 2.5(b, d)(iv, v)) indicates that DeepDenoiser has robust denoising performance on various kinds of noise.

We also test DeepDenoiser on waveforms with pure noise. Fig. 2.6 shows that it accurately distinguishes these waveforms from those containing earthquake signals, i.e. no recovered signal is predicted and the recovered noise is equivalent to the input noise. With traditional denoising methods, the input noise waveform on the other hand could be contaminated. This important feature shows that DeepDenoiser has the ability to preserve noise signals with or without the presence of earthquakes at reasonable computational cost. It could have significant applications in data preprocessing for ambient seismic noise studies where the contamination of noise data with earthquake signals can adversely affect the cross-correlation results (Liu, Ben-Zion, et al., 2016). Current practice for dealing with earthquake contamination is to simply discard the associated windows (Bensen et al., 2007; Sheng et al., 2017).

2.4.2 Generalization

The neural network is trained on seismic data that is synthesized by superposing noise onto high-SNR seismic signals. To test the network's generalizability, we applied it on the 91,000 samples of real noisy seismograms recorded in Northern California. These seismograms are from detected earthquakes in the Northern California Earthquake Catalog, but are contaminated heavily by noise (Fig. 2.7(i)) and therefore have low SNRs (Fig. 2.8).

The results suggest that DeepDenoiser successfully recovers clean seismic waveforms (Fig. 2.7(ii)),

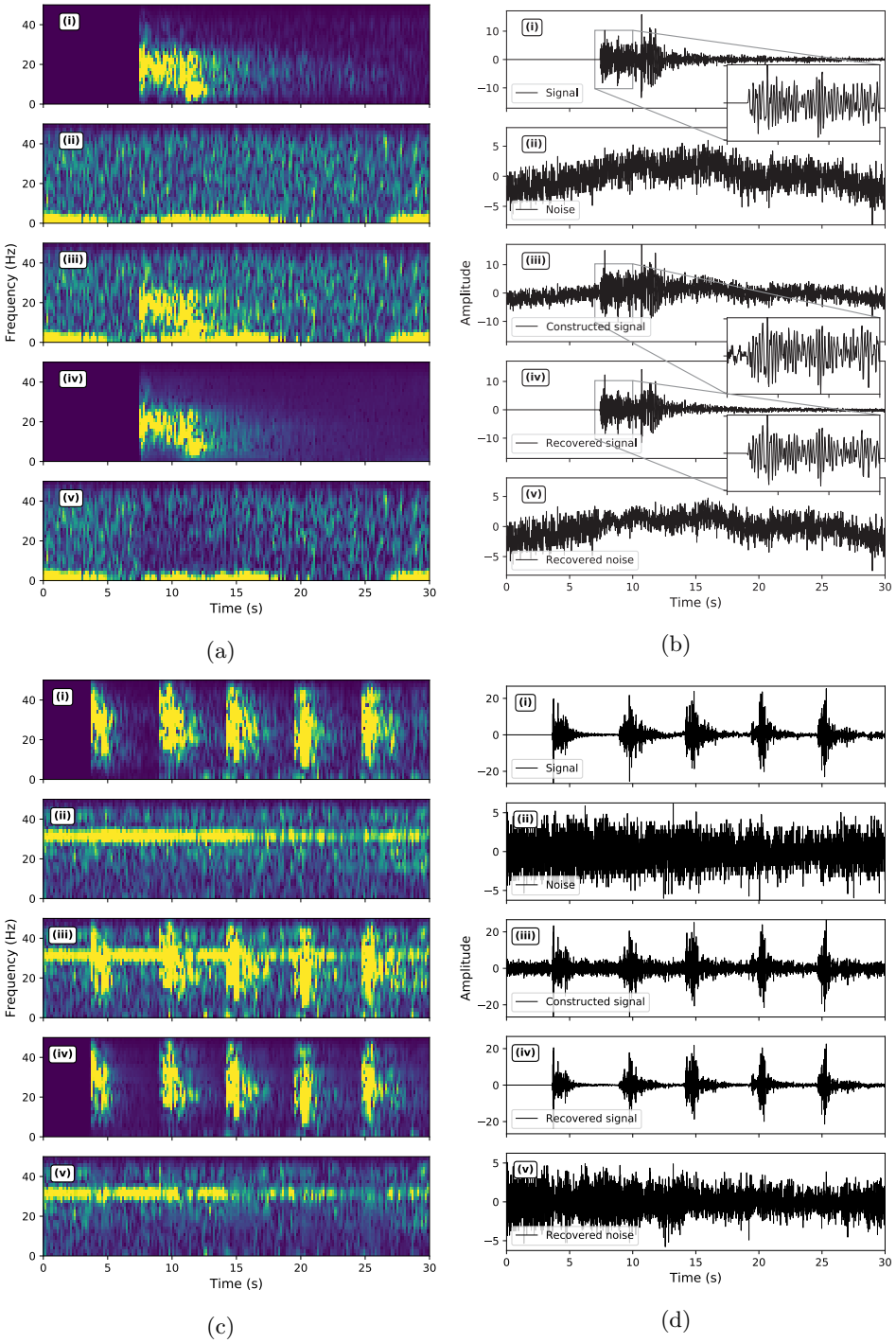


Figure 2.3: Denoising examples: (a, c) time-frequency domain; (b, d) time domain. Panels (i, ii, v) show the "clean" signals, the real noise, and the constructed noisy signals. Panels (iv, v) show the recovered signals and noise by DeepDenoiser. The events in (a, b) are shifted to start from 7.5s for plotting. Five events with increased shifting times are stacked together to construct the multi-event noisy samples in (c, d).

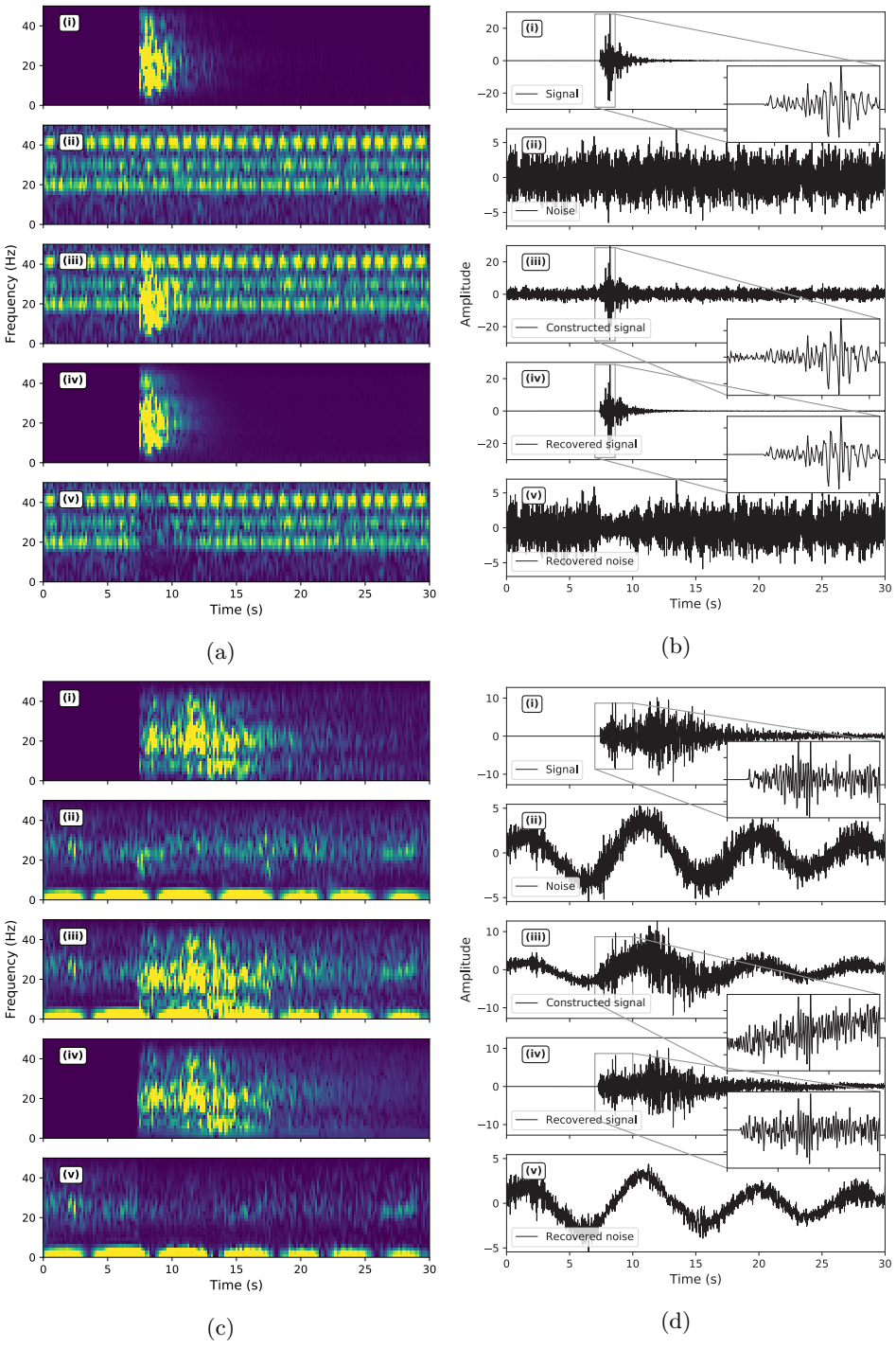
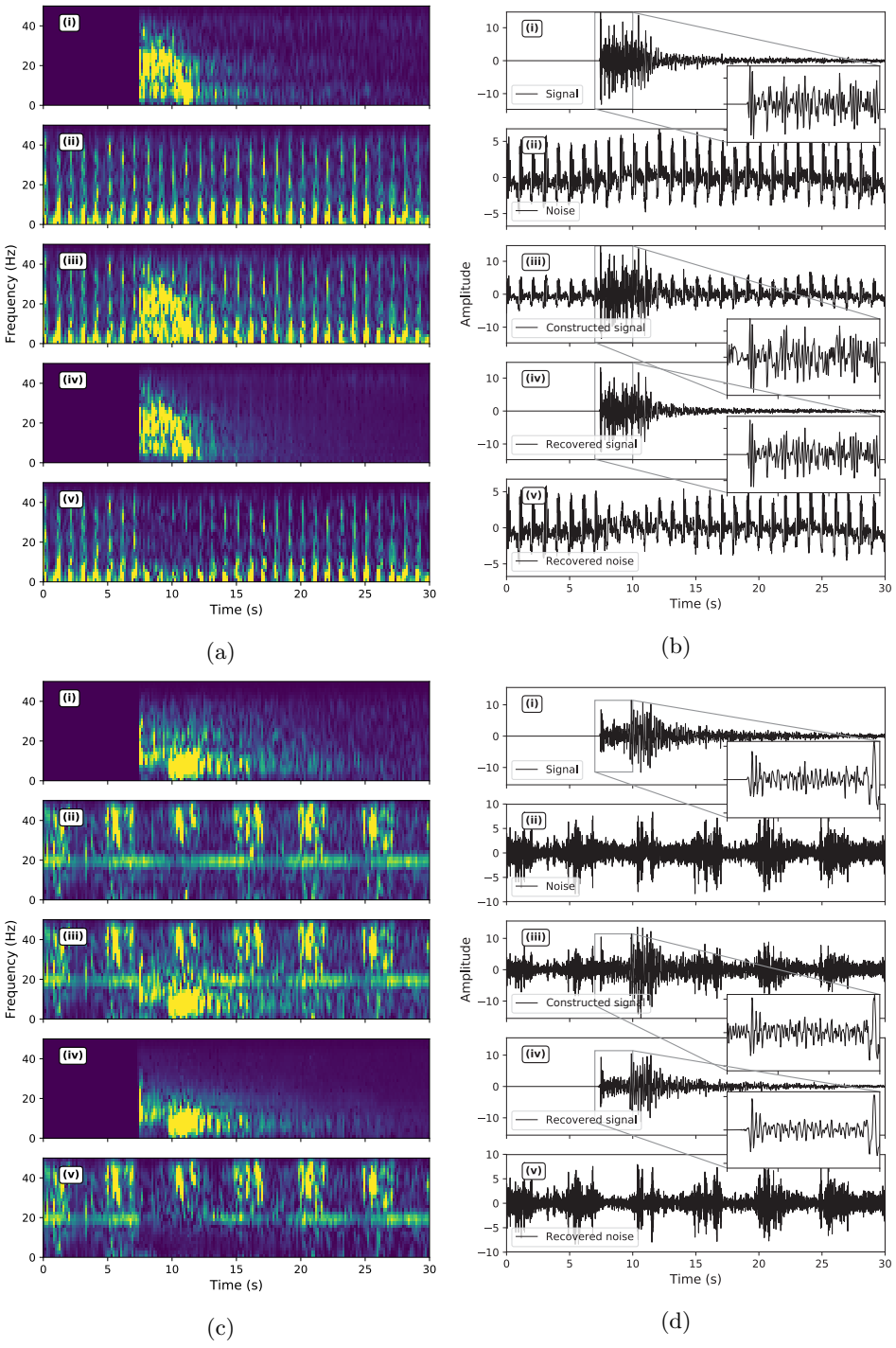


Figure 2.4: Denoising performance in presence of strong colored noise, plotted in a same manner as Fig. 2.3.



(a) (b) (c) (d)

Figure 2.5: Denoising performance in presence of cyclic non-seismic signals, plotted in a same manner as Fig. 2.3.

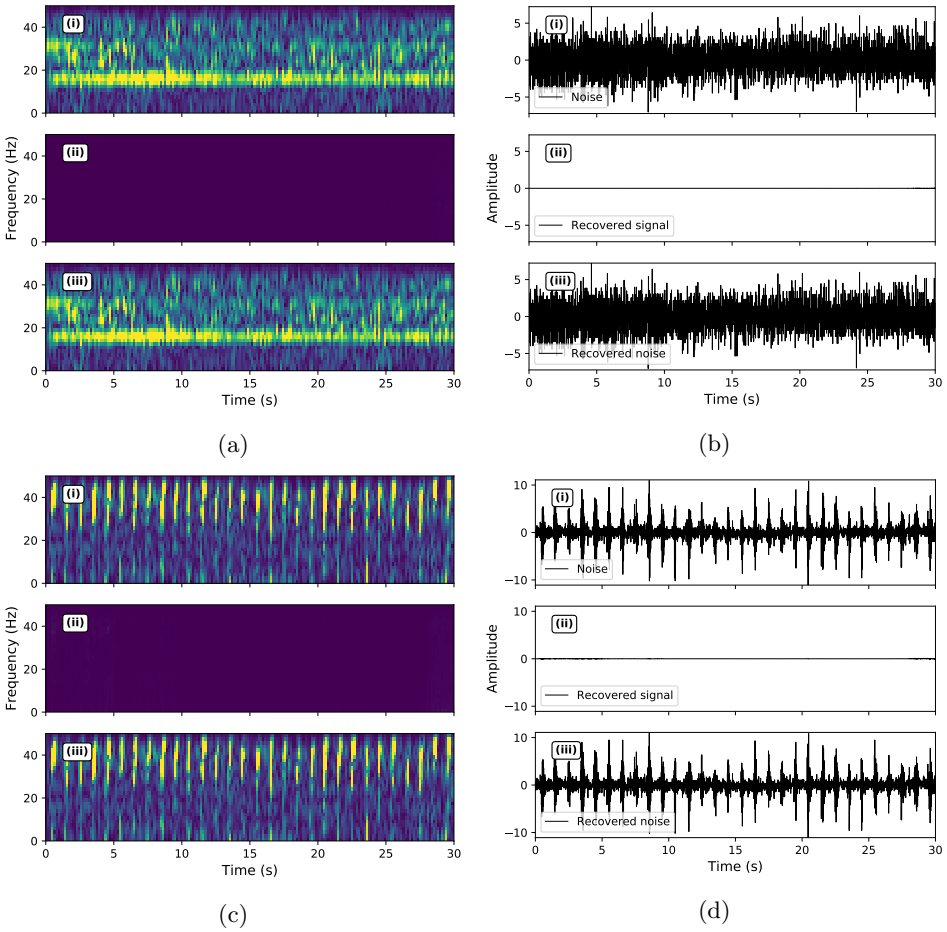


Figure 2.6: Denoising examples of pure noise: (a, c) time-frequency domain; (b, d) time domain. Panels (i) show input samples with pure noise. Panels (ii, iii) show the recovered signals and noise by DeepDenoiser. There is no artifact signals recovered by DeepDenoiser.

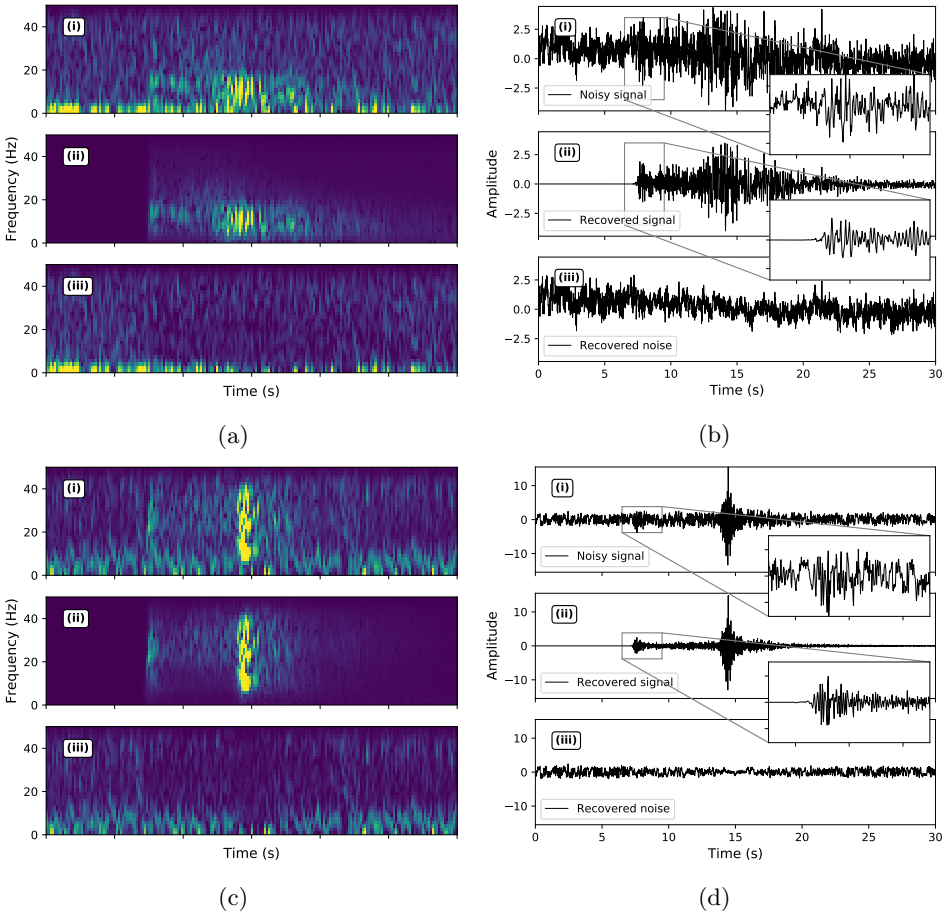


Figure 2.7: Denoising performance on real noisy seismograms: (a, c) time-frequency domain; (b, d) time domain. Panels (i) show real noisy seismograms. Panels (ii) and (iii) shows the denoised signal and remaining noise using DeepDenoiser.

e.g. the first arrivals, and improves the SNR by around 15 dB (Fig. 2.8). The SNR is calculated as:

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{\sigma_{\mathrm{signal}}}{\sigma_{\mathrm{noise}}} \right) \tag{2.5}$$

where σ_{signal} and σ_{noise} are the standard deviations of waveforms before and after the first arrival. Although DeepDenoiser is trained on synthetic data, it generalizes well to real seismograms. This suggests we can directly apply the deep neural network trained in this study to denoising tasks in real life whose performance relies on clean, undistorted seismic signals.

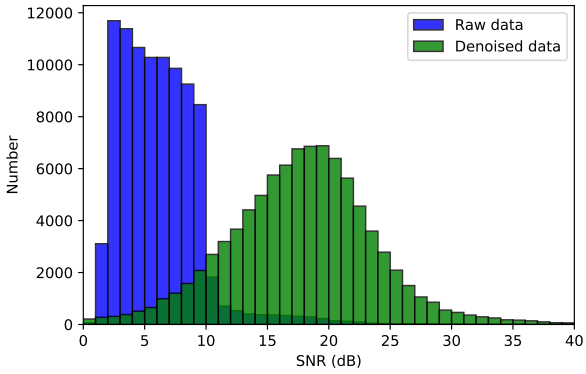


Figure 2.8: Histogram showing SNR improvement of real noisy seismograms at Northern California.

2.4.3 Comparison with Other Methods

To compare the performance of our method with other denoising approaches, we select one independent seismic waveform of "EHZ" channel in obspy (Beyreuther et al., 2010), which is recorded at station RTSH of the BayernNetz (BW) Network in Germany. We also cut a 30-second window prior to the event as the noise sample (Fig. 2.12 in appendix). We scale the noise waveform to vary the noise level and stack it with the seismic waveform to generate a sequence of constructed noisy signals with decreased SNRs.

We used normal filtering and general cross-validation denoising (Mousavi & Langston, 2017) for the comparison. For normal filtering, we design the filter based on the smoothed frequency distribution of the clean signal (Fig. 2.13 in appendix), which is intended to emphasize the frequency band where the true signal resides. We measure the SNR improvements between the denoised and construed noisy signals, and changes of the maximum amplitude, the correlation coefficient and the picked arrival times as the bases of the comparison between the performance of these methods (Fig. 2.9). The arrival time is picked using the same short-term averaging/long-term averaging (STA/LTA) method (Allen, 1978) same as the section after.

Fig. 2.9 indicates that DeepDenoiser achieves a better denoising performance while introducing smaller distortion to the signal waveform. The SNR improvement of DeepDenoiser is more significant and more robust than the GCV method (Fig. 2.9(a)). Fig. 2.9(b) shows that DeepDenoiser recovers the true amplitude of the signal more accurately. The max amplitude of the denoised signal is closer to the true signal when the SNR is larger than 2 dB, while the GCV method largely attenuates both noise and signal to achieve a good denoising performance. The higher correlation coefficients in Fig. 2.9(c) further demonstrate that DeepDenoiser introduces smaller waveform distortion during denoising. In contrast, the GCV method distorts the signal waveform significantly. DeepDenoiser is designed based on a way to separate signal and noise effectively, which enables it to improve SNR and

preserve the signal waveform simultaneously. The denoised waveform also improves the recovery of arrival times from constructed noisy signal. With a fixed activation threshold for STA/LTA method, the picked arrival times show a systematically increasing delay and eventual failure at increasing noise levels. Fig. 2.9(d) shows that on the noisy signal the arrival time has a delay of 0.5 s at SNR of 4 dB and fails to be recovered beyond this level (smaller dB). On the other hand, the signal after running DeepDenoiser yields an accurate arrival time even at SNR of 2dB, which is also superior to the other two denoising methods.

2.4.4 Application to Earthquake Detection

Background noise can significantly affect the performance of common detection algorithms such as STA/LTA (short-term average/long-term average) for detecting small and weak events. Moreover, the presence of non-earthquake signals will increase the false positive rate, and as a result, degrade detection precision. In Fig. 2.10 we present two examples of how DeepDenoiser can improve the STA/LTA characteristic function. The short and long time windows are set to be 0.5 seconds and 5 seconds. Fig. 2.10(a) shows an example when the noise smooths out the sharp jump at the arrival of seismic waves and makes the earthquake undetectable. DeepDenoiser makes this earthquake easy to detect and increases the recall rate by making the STA/LTA characteristic function sharp after denoising. Fig. 2.10(b) shows another example when impulsive non-seismic signals bring sharp peaks to the STA/LTA characteristic function. These peaks will be falsely detected as earthquakes by the STA/LTA method. DeepDenoiser can effectively remove these non-seismic signals and increase the prediction precision. We compare the earthquake detection results on our test dataset of 10,800 samples before and after denoising. With a threshold of 5, we calculate the precision of earthquake detection to be 35.14%, 92.34% and the recall rates to be 17.69%, 93.76% for the noisy signal and denoised signal respectively. The accuracy is defined as the ratio of true positive detections over the total positive detections. The recall rate is defined as the ratio of true positives over the total true number of earthquakes. Both the accuracy and recall rate are improved significantly by DeepDenoiser.

We further investigate if DeepDenoiser would bias the timing information by measuring differential times via cross-correlation, which determines the accuracy of earthquake locations (Schaff et al., 2004). We measure the cross-correlation between the true signals and the denoised signals by DeepDenoiser. The cross-correlation window is chosen to be 2.56 seconds centered on the P wave. We create three test datasets with low mean SNRs of 3.3, 1.4 and 0.5, where DeepDenoiser's performance starts to fail. The results show that 98.9%, 96.0% and 85.8% of the samples have zero time-shifts. The standard deviations of time-shifts are 0.09, 0.14, and 0.29 seconds. Although expressing it using the standard deviation, the resulting distribution is not Gaussian and the values are strongly influenced by outliers with bad waveforms. This demonstrates that DeepDenoiser would not introduce systematic bias to the time information of seismic events.

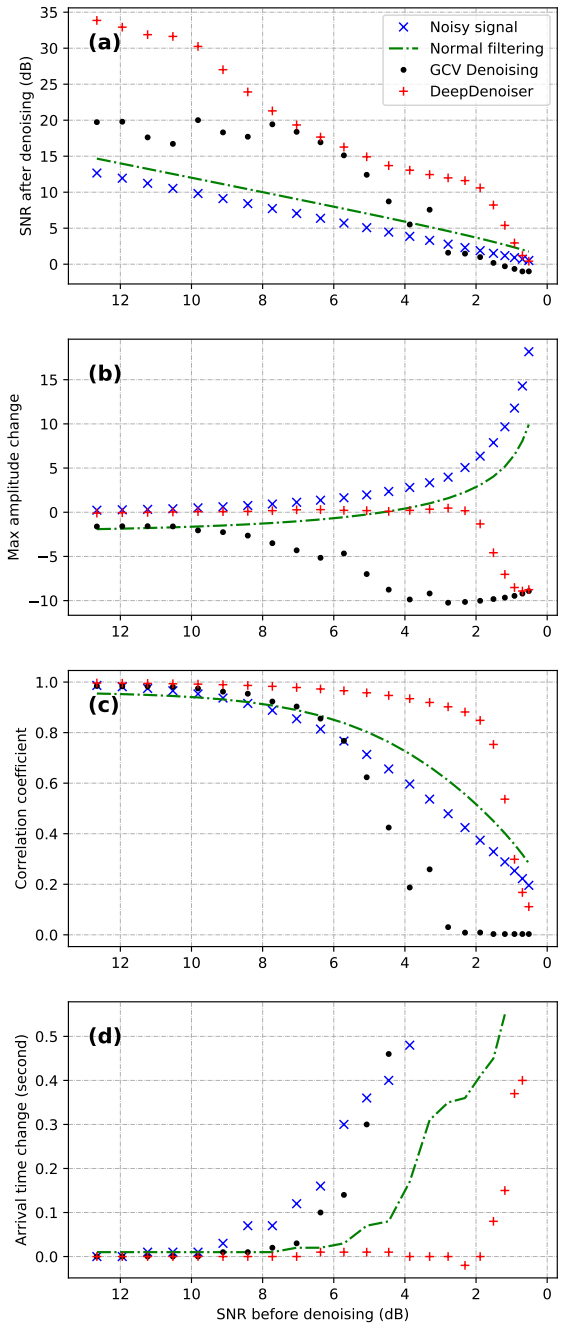
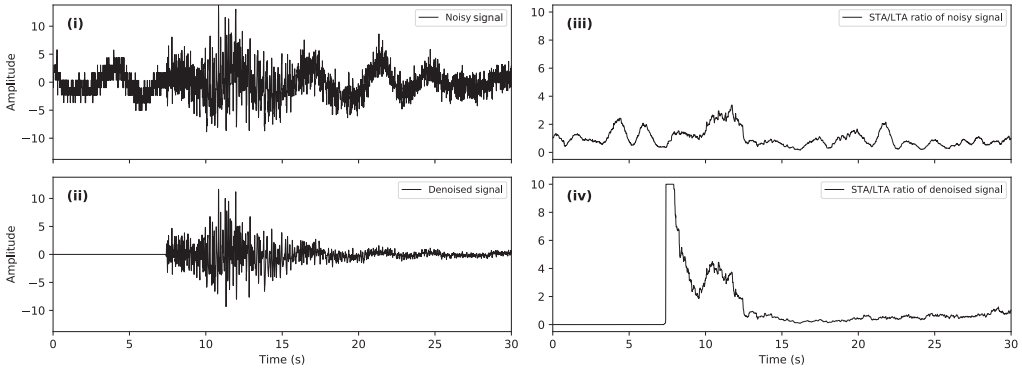
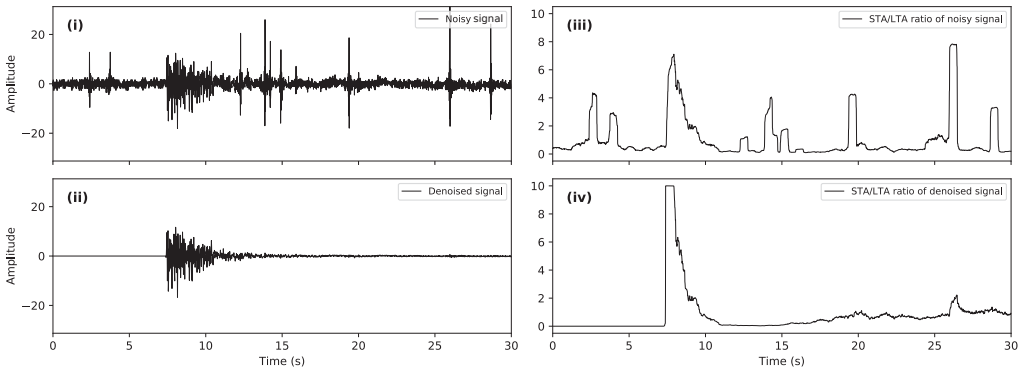


Figure 2.9: Performance comparison between normal filtering, GCV denoising and DeepDenoiser: (a) improvement of SNR; (b) max amplitude changes; (c) correlation coefficient; (d) picked arrival time changes. Values in (b) and (d) are differences compared with the signal in Fig. 2.12(a) in appendix. Values in (c) are calculated by zero-lag cross-correlation between the denoised waveforms and the signal in Fig. 2.12(a) in appendix.

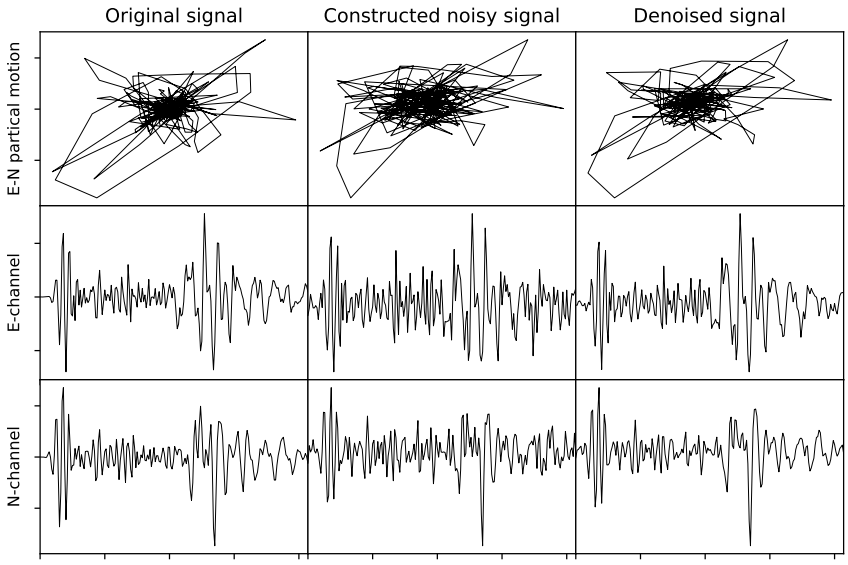


(a)

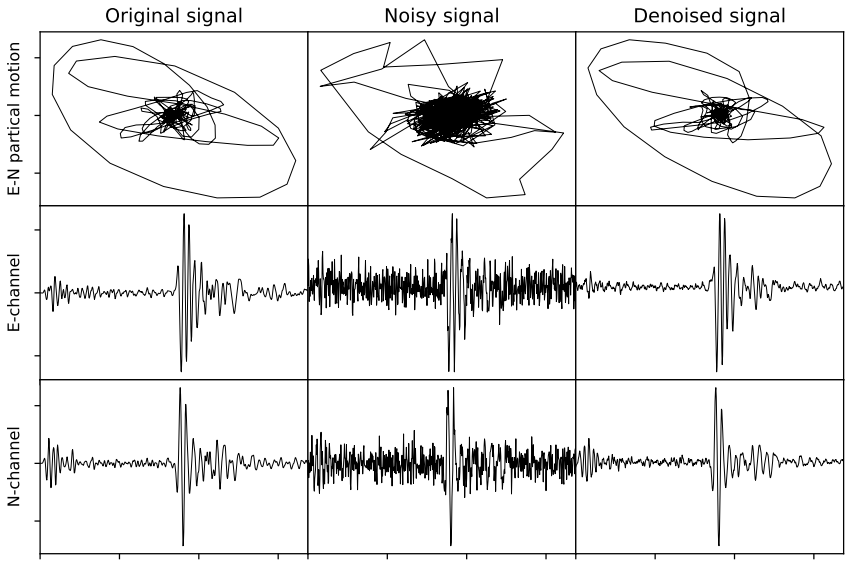


(b)

Figure 2.10: Improvement of STA/LTA characteristic function after denoising: (i) construed noisy signals; (ii) denoised signals; (iii) and (iv) the corresponding STAT/LTA characteristic functions.



(a)



(b)

Figure 2.11: Examples of particle motion between the East-West (E) and North-South (N) horizontal channels. The left columns are the original high SNR signals. The center columns are the constructed noisy signals. The right column are the denoised signals by DeepDenoiser. The particle motions are shown in the first rows. The waveforms of E and N channels are shown in the second and third rows.

2.5 Discussion and Conclusion

We have developed a novel deep-learning-based denoising algorithm for seismic data, DeepDenoiser. This learning-based approach can learn a collection of sparse features with the aim of signal and noise separation from samples of data. These features reflect more accurately the statistical characteristics of a signal of interest and can be used for effective denoising or decomposition of input waveforms. The neural network automatically determines the percentage of signal (mask) present at each data point in the time-frequency space. DeepDenoiser learns the sparse representation of data and to predict two adaptive thresholding masks simultaneously by optimizing the loss function. The masks determined by the network effectively decompose the input data into a signal of interest and noise.

Our results show this algorithm can achieve robust and effective performance in denoising of data even when the signal and noise share a common frequency band. The denoising ability of our method is not limited to random white noise, but performs well for a variety of colored noise and non-earthquake signals as well. Our tests indicate that DeepDenoiser significantly improves the SNR with minimal change to the underlying signal of interest. DeepDenoiser preserves waveform shape more faithfully than other denoising methods, even in presence of high noise levels. Because we apply the same masks to both the real and imaginary parts of the input noisy signal (Fig. 2.2 step (5)), DeepDenoiser does not introduce a phase shift during denoising. Fig. 2.11 shows the particle motions between the two horizontal channels when we apply DeepDenoiser to the three channels of seismograms separately. The results further demonstrate that DeepDenoiser does not introduce large bias to the amplitude, time, and phase information after denoising. Although the network is trained using constructed noisy signals, it generalizes to real noisy datasets outside of training set. We have only demonstrated the capability of our method in improving event detection; however, the potential applications of our approach are widespread. DeepDenoiser can be adapted for various types of seismic and non-seismic signals and different applications. Seismic imaging, micro-seismic monitoring, test-ban treaty monitoring, and preprocessing of ambient noise data are other potential applications of our method.

Although the performance of DeepDenoiser is impressive, it does not result in a perfect separation of signal and noise. Perfect separation of signal and noise in the time-frequency domain requires recovering two complex spectra for signal and noise from the complex spectra of the noisy signal. DeepDenoiser uses the same mask for both the real and imaginary parts of the spectrum, and the mask, which has a value between [0, 1], can not recover signal values larger than the input noisy signal. Predicting the signal and noise values directly without the use of a mask is a potential future direction for improvement. For three-component seismographs or seismic arrays, extending DeepDenoiser to consider all components simultaneously during denoising is another possibility.

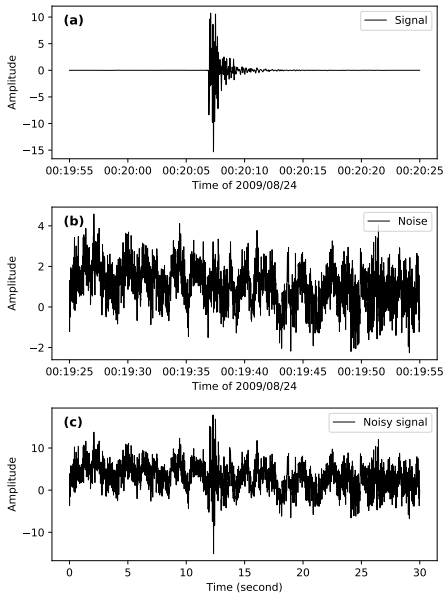


Figure 2.12: The waveforms cut from "EHZ" channel of RTSH station, BayernNetz (BW) Network in Germany: (a) the "clean" signal; (b) the noise waveform obtained from the same station prior to the event origin time; (c) an example of generated noisy waveform with a stacking ratio of 3 for noise in (b) over signal in (a).

Appendix

The waveform of the signal and noise used for section "comparison with other methods" are shown in Fig. 2.12. One example of the constructed noisy signal is shown in Fig. 2.12(c). Frequency distribution of the filter used for normal filtering are shown in Fig. 2.13. This is built based on the frequency distribution of clean signal in Fig. 2.12(a), so that it retains the frequency bandwidth of the clean signal.

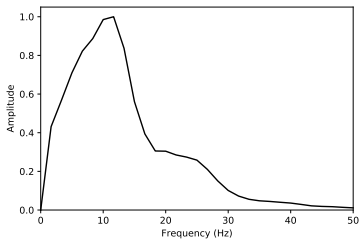


Figure 2.13: Frequency distribution of the filter used for normal filtering in the comparison section

Chapter 3

PhaseNet: A Deep-Neural-Network-Based Seismic Arrival Time Picking Method

As the number of seismic sensors grows, it is becoming increasingly difficult for analysts to pick seismic phases manually and comprehensively, yet such efforts are fundamental to earthquake monitoring. Despite years of improvements in automatic phase picking, it is difficult to match the performance of experienced analysts. A more subtle issue is that different seismic analysts may pick phases differently, which can introduce bias into earthquake locations. We present a deep-neural-network-based arrival-time picking method called "PhaseNet" that picks the arrival times of both P and S waves. Deep neural networks have recently made rapid progress in feature learning, and with sufficient training, have achieved super-human performance in many applications. PhaseNet uses three-component seismic waveforms as input and generates probability distributions of P arrivals, S arrivals, and noise as output. We engineer PhaseNet such that peaks in the probability distributions provide accurate arrival times for both P and S waves. PhaseNet is trained on the prodigious available data set provided by analyst-labeled P and S arrival times from the Northern California Earthquake Data Center. The dataset we use contains more than seven hundred thousand waveform samples extracted from over thirty years of earthquake recordings. We demonstrate that PhaseNet achieves much higher picking accuracy and recall rate than existing methods when applied to the waveforms of known earthquakes, which has the potential to increase the number of S-wave observations dramatically over what is currently available. This will enable both improved locations

and improved shear wave velocity models.

3.1 Introduction

Earthquake detection and location are fundamental to seismology. The quality of earthquake catalogs depends critically on both the number and the accuracy of arrival time measurements. Earthquake arrival time measurement, or phase picking, is often carried out by network analysts who base their phase picks on expert judgment and years of experience. As the rate of seismometer deployment continues to accelerate; however, it is becoming increasingly difficult to keep up with the data flow. This is particularly true for dense networks in areas of particular interest or concern that now may contain over 1000s of sensors. Phase pickers are particularly challenged by S waves, because they are not the first arriving waves, and they emerge from the scattered waves of the P coda. S-wave arrival times are particularly useful because they can be used to reduce the depth-origin trade-off that can afflict earthquake locations based on P waves alone, and because S-wave structure is important for strong ground motion prediction.

Decades of research has been devoted to automatic phase picking, including: methods based on amplitude, standard deviation or energy; statistical methods and shallow neural networks. The short-term average/long-term average (STA/LTA) method (Allen, 1978) is commonly used and tracks the ratio of energy in a short-term window with that in a long-term window. Peaks above a threshold mark impulsive P or S wave arrivals. This method is efficient, often effective, but susceptible to noise and has low accuracy for arrival times, particularly for shear waves. Baer and Kradolfer (1987) improved the STA/LTA method using the envelope as the characteristic function. Sleeman and Van Eck (1999) applied joint autoregressive (AR) modeling of the noise and seismic signal and used the Akaike Information Criterion (AIC) to determine the onset of a seismic signal. Approaches based on higher-order statistics (HOS), including kurtosis and skewness, were developed to identify the transition from Gaussianity to non-Gaussianity, which coincides with the onset of the seismic event, even in the presence of noise (Küperkoch et al., 2010; Saragiotis et al., 2002). Traditional shallow neural networks were tested by Gentili and Michelini (2006) to pick P and S phases, based on four manually defined features: variance, absolute value of skewness, kurtosis and a combination of skewness and kurtosis predicted based on sliding windows. While most phase picking algorithms focus on P waves, Ross and Ben-Zion (2014) utilized polarization analysis to distinguish between P and S waves primarily to improve S-wave arrival time measurements. Despite the substantial efforts outlined above, the accuracy of automated phase picking algorithms lags that of experienced analysts. This is attributable to the fact that earthquake waveforms are highly complex due to multiple effects including source mechanism, stress drop, scattering, site-effects, phase conversions, and interference from a multitude of noise sources. Traditional automated methods use manually defined features that require careful data processing, like band-pass filtering

and setting an activation threshold.

In this study, we present a deep neural network algorithm, PhaseNet, for seismic phase picking. Instead of using manually defined features, deep neural networks learn the features from labeled data, both noise and signal, which proves a powerful advantage for complex seismic waveforms. The network is trained on the substantial catalog of available P and S arrival-times picked by experienced analysts. Unfiltered three-component seismic waveforms are the input to PhaseNet, which is trained to output three probability distributions: P wave, S wave, and noise. The neural network is trained on the target probability distributions of known earthquake waveforms. Peaks in the P wave and S wave probability distributions are designed to correspond to the predicted P and S arrival times. We demonstrate that PhaseNet provides high accuracy and recall rate for both P and S picks, and achieves significant improvement compared with a traditional STA/LTA method. PhaseNet has the potential to provide comprehensive, superior performance for standard earthquake monitoring.

3.2 Data

Seismological archives include tremendous numbers of manually picked P and S wave arrivals, which represent an exceptionally rich training set of labeled data that is ideal for deep learning (Figure 3.1). In this study, we gathered available digital seismic waveform data based on the Northern California Earthquake Data Center Catalog (NCEDC, 2014). We use three-component data that have both P and S arrival times. This leaves us 779,514 recordings in the dataset. We use stratified sampling based on stations to divide this dataset into training, validation and test datasets, with 623,054, 77,866 and 78,592 samples respectively. The training and validation sets are used during training, fine-tuning parameters and model selection. The test set is only used to evaluate the final performance and results of PhaseNet. This dataset has a diversity of waveform characteristics. It includes different types of instruments in Northern California Seismic Network and covers a wide range of signal-to-noise ratio (SNR). The proportion of each instrument in the dataset is shown in Figure 3.2. The distribution of signal-to-noise ratio is shown in Figure 3.3. The SNR is calculated by the ratio of standard deviations of the five seconds following and the five seconds preceding the P arrival. The complexity of this dataset makes it challenging for automatic phase picking, but it provides a more comprehensive performance evaluation.

We apply minimal data preprocessing to the training data. We randomly select a 30-second time window that includes the P and S arrival times as the input of PhaseNet. The position of the arrivals within the window are varied to ensure that the algorithm doesn't just learn the windowing scheme. All data are sampled at 100 Hz, which is the most common sampling rate in the raw dataset, so that the 30-second input waveforms have 3001 data points for each component. We normalize each component waveform by removing its mean and dividing it by the standard deviation (Figure 3.4(a-c)). The manually picked time points in the dataset may not be the true P/S arrivals, but we expect

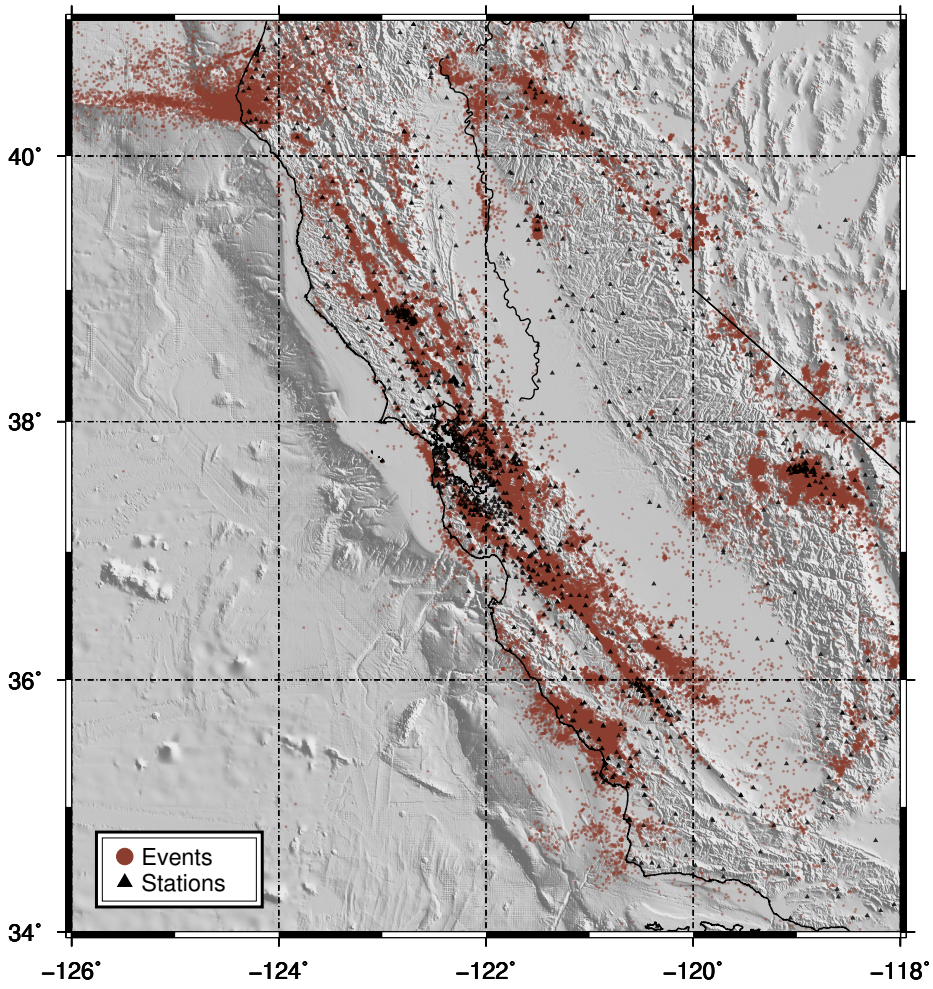


Figure 3.1: The locations of 234,117 earthquakes (red points) and 889 seismic stations (black triangles) in the Northern California Earthquake Catalog.

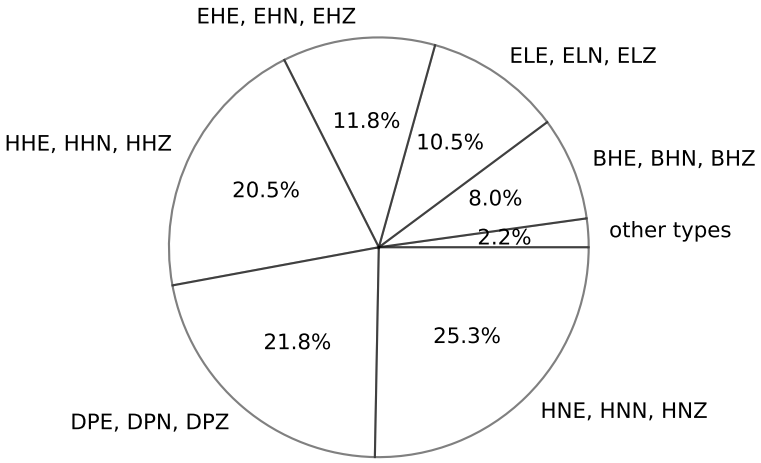


Figure 3.2: The proportion of different instrument types in the dataset. The first letter is the band code: H: high broad band, D: very very short period, E: short period. The second letter is the instrument code: N: accelerometer, P: very short period seismometer, H: high gain seismometer, L: low gain seismometer. The third letter is the orientation code: E: east-west direction, N: north-south direction, Z: vertical direction.

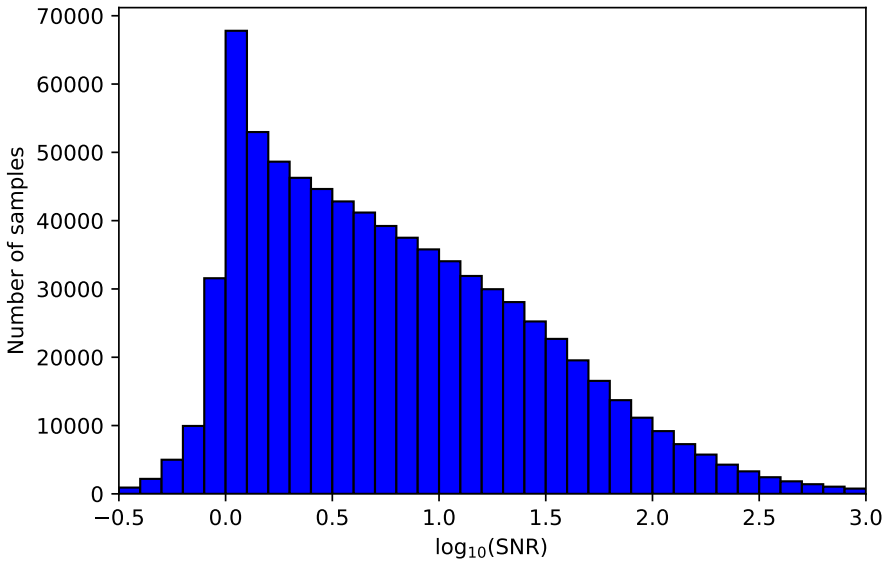


Figure 3.3: Signal-to-noise ratio (SNR) distribution. The SNR is calculated by the ratio of standard deviations of two five-second windows following and preceding the P arrival.

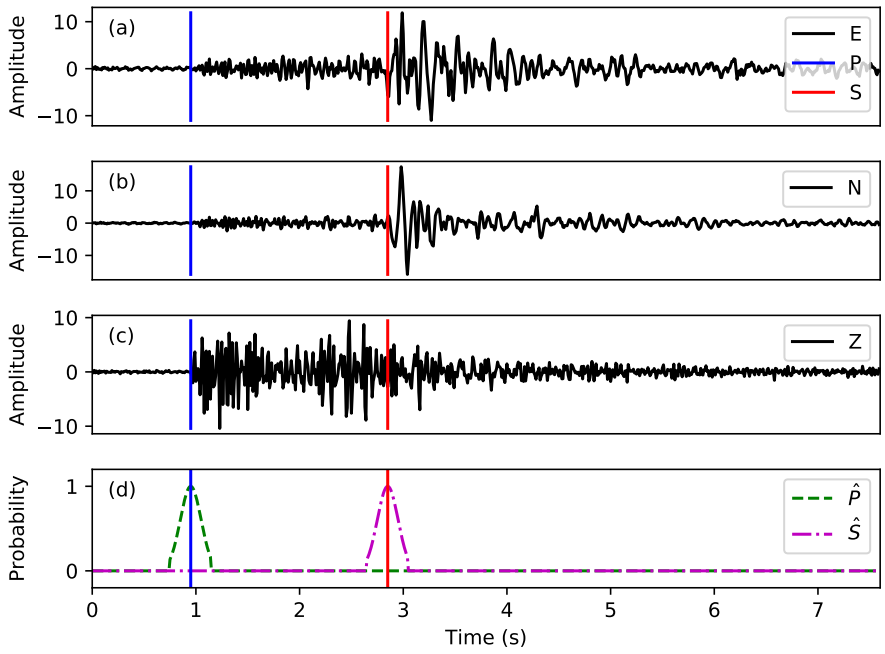


Figure 3.4: A sample from the dataset. (a) - (c) Seismograms of the "ENZ" (East, North, Vertical) components. The blue and red vertical lines are the manually picked P and S arrival times. (d) The converted probability masks for P and S picks. The shape is a truncated Gaussian distribution with a mean (μ) of the arrival time and a standard deviation (σ) of 0.1 seconds.

the ground-truth arrival times will be centered on the manual picks with some uncertainty. For this reason we apply a mask with the shape of a Gaussian distribution around the manual picks. Therefore, the time point picked by analysts has the highest probability, while the nearby data points have reduced probabilities (Figure 3.4(d)). The standard deviation of the Gaussian distribution is set to 0.1 seconds. Representing manual picks probabilistically allows the algorithm to reduce the influence of picking errors in the dataset. Because we have considered the probabilities of the nearby data points, the mask increases the amount of information in P and S picks relative to noise, and helps accelerate convergence. Here the noise includes all data points that are not first arrivals of P or S waves. The probability distribution of noise is calculated by:

$$Prob(\text{noise}) = 1 - Prob(\text{P}) - Prob(\text{S})$$

where $Prob$ is the probability of each class. After the conversion using a Gaussian distribution mask, we can extract accurate arrival times from the peaks of probability distributions predicted by PhaseNet.

3.3 Method

The architecture of PhaseNet (Figure 3.5) is modified from U-Net (Ronneberger et al., 2015) to deal with 1-D time series data. U-net is a deep neural network approach used in biomedical image processing, that seeks to localize properties in an image. The mapping to our problem is to localize the properties of our time series into three classes: P-pick, S-pick, and noise. The inputs are three-component seismograms of known earthquakes. The outputs are probability distributions of P wave, S wave, and noise. In our experiments, the input and output sequences contain 3001 data points for each component (30 seconds long, sampled at 100 Hz). The input seismic data go through four down-sampling stages and four up-sampling stages. Inside each stage, we apply 1D convolutions and rectified linear unit (ReLU) activations. The down-sampling process is designed to extract and shrink the useful information from raw seismic data to a few neurons, so each neuron in the last layer makes up a broadly receptive window. The up-sampling process expands and converts this information into probability distributions of P wave, S wave and noise for each time point. A skip connection at each depth directly concatenates the left output to the right layer without going through the deeper layer. This helps improve convergence during training (Li et al., 2017; Ronneberger et al., 2015). The 1D convolution size is set to 7 data points. The stride step for down-sampling is set to 4 data points, so after each stride the channel length is condensed into one-fourth of its original dimension, while the deconvolution operation (Noh et al., 2015) for up-sampling expands the condensed layers by a factor of four to recover its previous length. We have added padding at the front and the back of each layer during convolutions to make the input and output sequences have the same length. Figure 3.5 shows the size of each layer and the operations of convolution and deconvolution. The

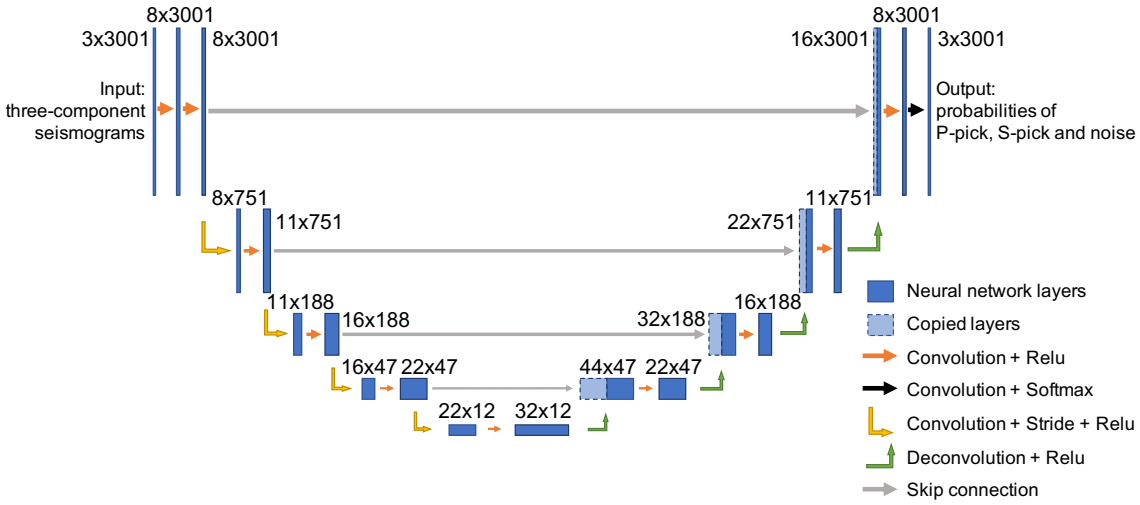


Figure 3.5: The network architecture. The input is the 30-second three-component seismograms sampled at 100 Hz, so the input has a dimension of 3×3001. The output is three probabilities with the same length as input for P-pick, S-pick, and noise. The blue rectangles represent layers inside the neural network. The numbers near them are the dimensions of each layer, which follow a format of "number of channels × length of each channel". The arrows are operations applied between layers, whose meanings are noted in the low right corner. The input seismic data go through four down-sampling stages and four up-sampling stages. The down-sampling is done by 1D convolution and stride. We have set the length of convolution kernel to 7 data points and the stride step to 4 data points. The up-sampling is done by deconvolution, which recovers the input length of the previous stage. A skip connection at each stage directly concatenates the left output to the right layer without going through the deeper layers, which improves convergence during training. The blue rectangles with dashed boundaries are the layers copied directly by the skip connection. The softmax normalized exponential function is used to set probabilities in the last layer.

Table 3.1: Evaluation metrics on the test dataset. Pickers with residuals ($\Delta t < 0.1s$) are counted as true positive picks. The mean ($\mu(\Delta t)$) and standard deviation ($\sigma(\Delta t)$) are calculated on residuals ($\Delta t < 0.5s$) whose distributions are shown in Figure 3.6

Evaluation Indicator	Phase	PhaseNet	AR picker
Precision	P	0.939	0.558
	S	0.853	0.195
Recall	P	0.857	0.558
	S	0.755	0.144
F1 score	P	0.896	0.558
	S	0.801	0.165
$\mu(\Delta t)(ms)$	P	2.068	11.647
	S	3.311	27.496
$\sigma(\Delta t)(ms)$	P	51.530	83.991
	S	82.858	181.027

softmax normalized exponential function is used to set probabilities in the last layer:

$$q_i(x) = \frac{e^{z_i(x)}}{\sum_{k=1}^{3} e^{z_k(x)}}$$

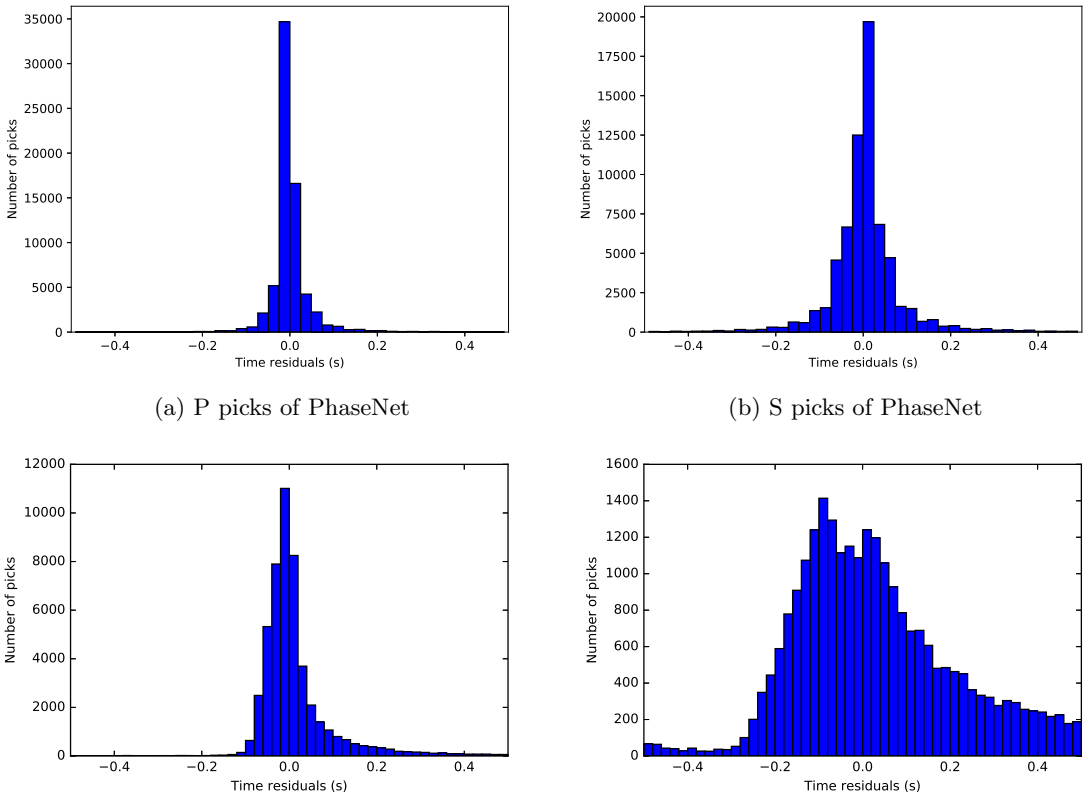
where $i = 1, 2, 3$ represents noise, P and S classes. $z(x)$ are the unscaled values of the last layer. The loss function is defined using cross-entropy between the true probability distribution ($p(x)$) and predicted distribution ($q(x)$):

$$H(p, q) = -\sum_{i=1}^{3} \sum_{x} p_i(x) \log q_i(x),$$

which measures the divergence between the two probability distributions. The P and S arrival times are extracted from the peaks of output probability distributions (Duarte, 2015).

3.4 Experiments

We have chosen the evaluation metrics: precision, recall, F1 score, mean (μ) and standard deviation (σ) of time residuals (Δt) between picks of PhaseNet and analysts to test the performance of PhaseNet (Powers, 2011). Precision, recall and F1 are standard measures of performance defined



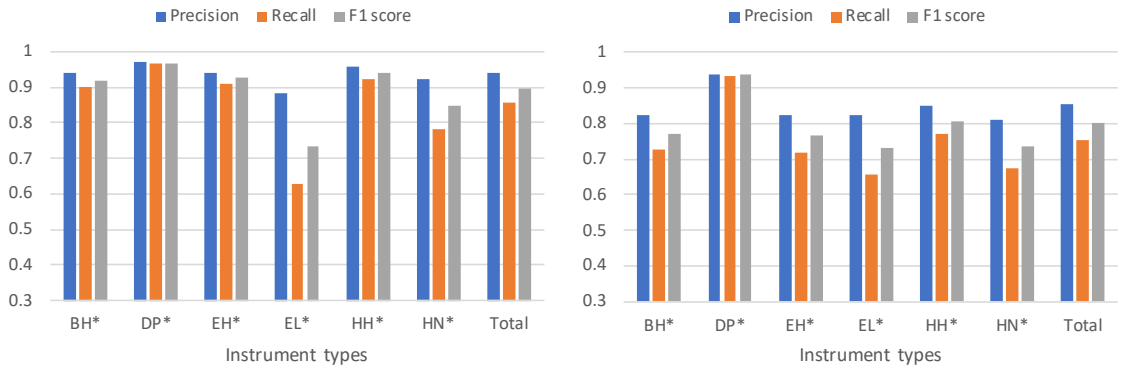
(a) P picks of PhaseNet

(b) S picks of PhaseNet

(c) P picks of AR picker

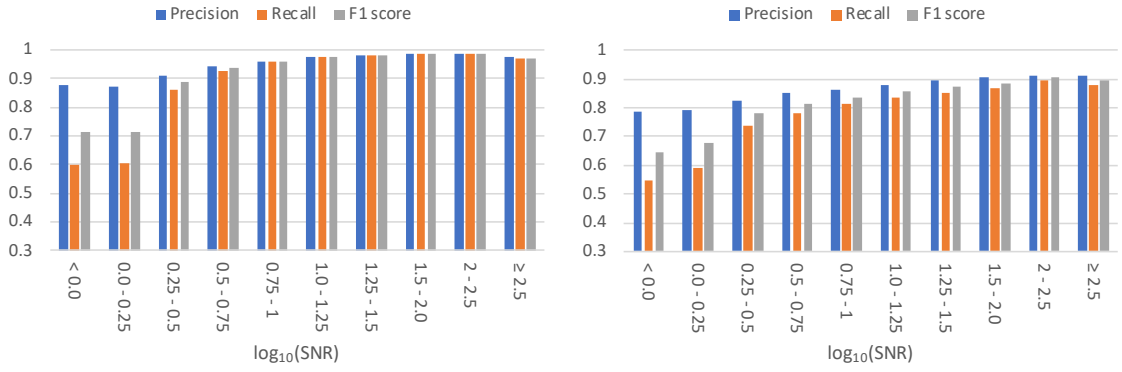
(d) S picks of AR picker

Figure 3.6: The distribution of residuals (Δt) of PhaseNet (upper panels) and AR picker (lower panels) on the test dataset



(a) (b)

Figure 3.7: Performances on different instrument types. (a) P picks. (b) S picks. The meaning of x-axis labels is the same as in Figure 3.2. The "total" dataset is the same test dataset used in Table 3.1.



(a) (b)

Figure 3.8: Performances of different signal-to-noise ratios (SNR). (a) P picks. (b) S picks.

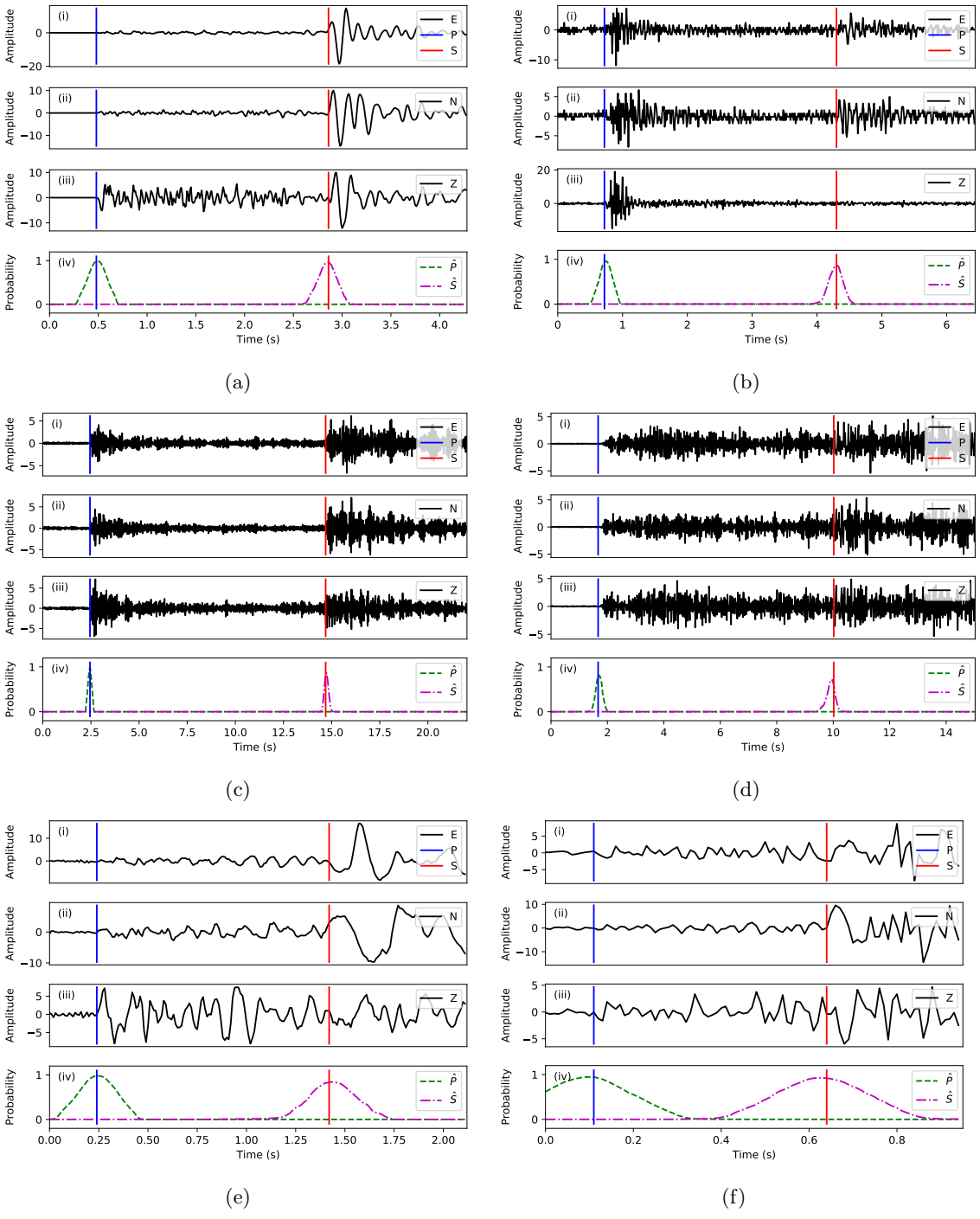


Figure 3.9: Examples of good picks ($\Delta t < 0.1s$) in the test dataset. The upper sub-figures (i - iii) are the "ENZ" components of seismograms. The lower sub-figures (iv) are the predicted probability distributions of P wave ($\hat{P}$) and S wave ($\hat{S}$). The blue and red vertical lines are the P and S arrival times picked by analysts.

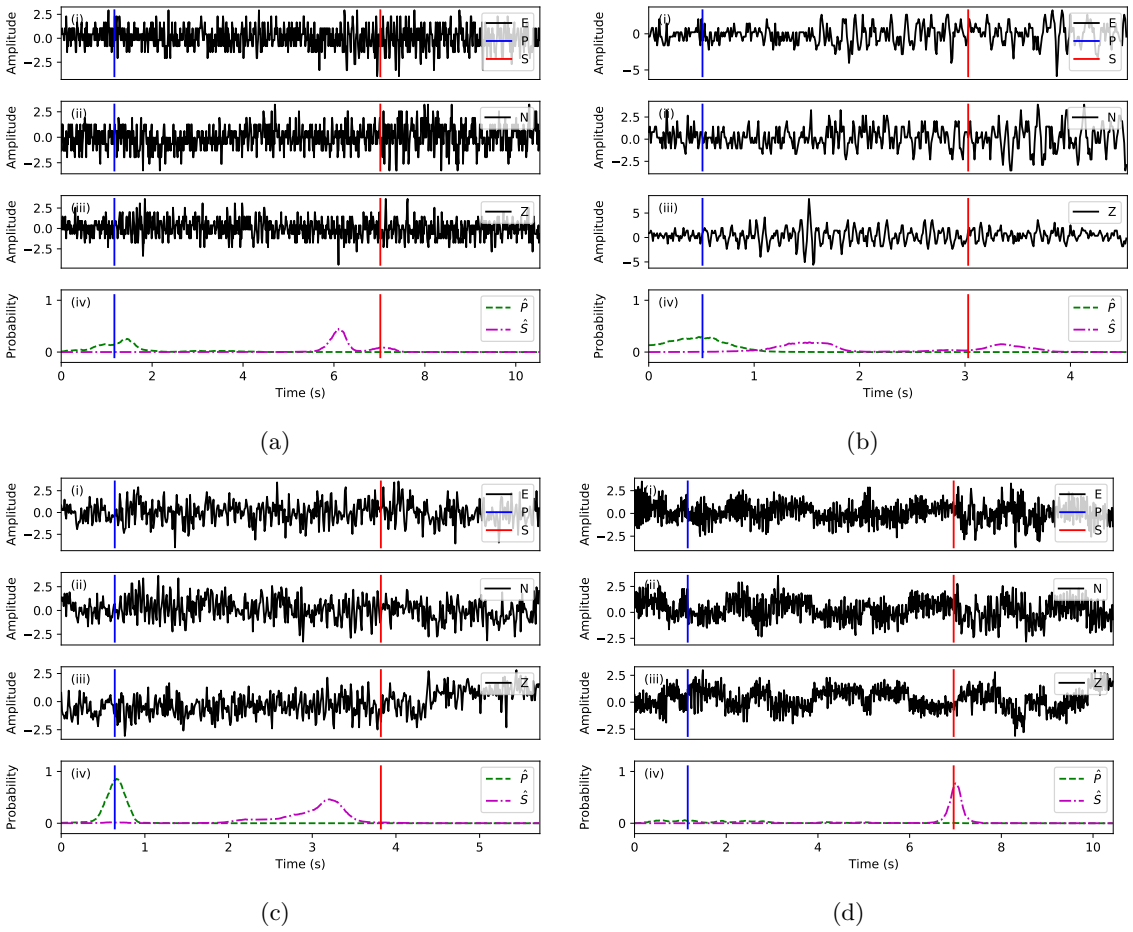


Figure 3.10: Examples of bad picks in the test dataset. (a) and (b) are examples of no P or S picks predicted. (c) is an example of bad S picks. (d) is an example of bad P picks.

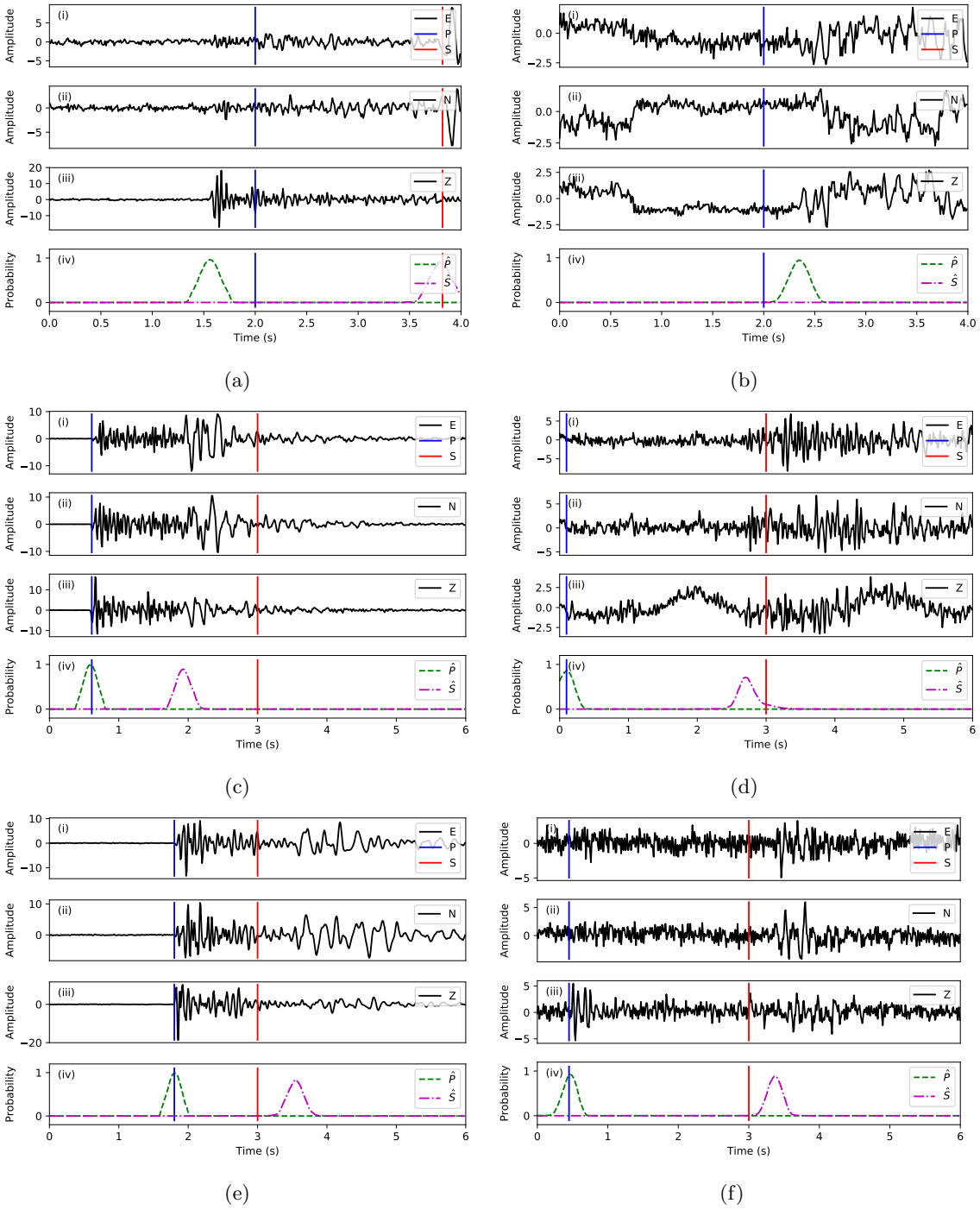


Figure 3.11: Examples where manual picks may be not accurate in the test dataset. (a) and (b) are ambiguous P picks. (c) - (f) are ambiguous S picks.

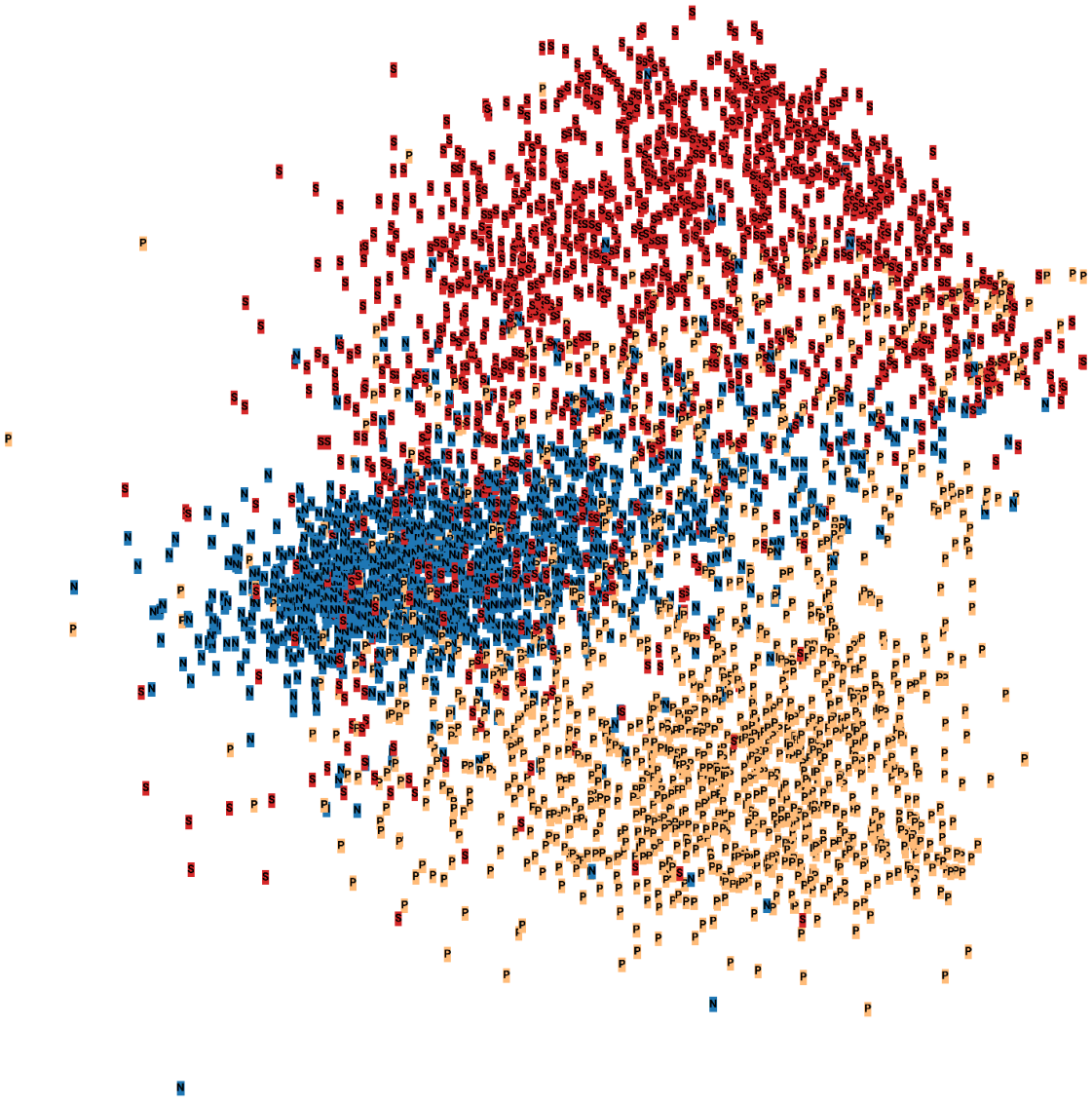


Figure 3.12: PCA visualization of weights in the deepest layer. The red, yellow and blue dots represent input data with P arrivals, S arrivals or only noise.

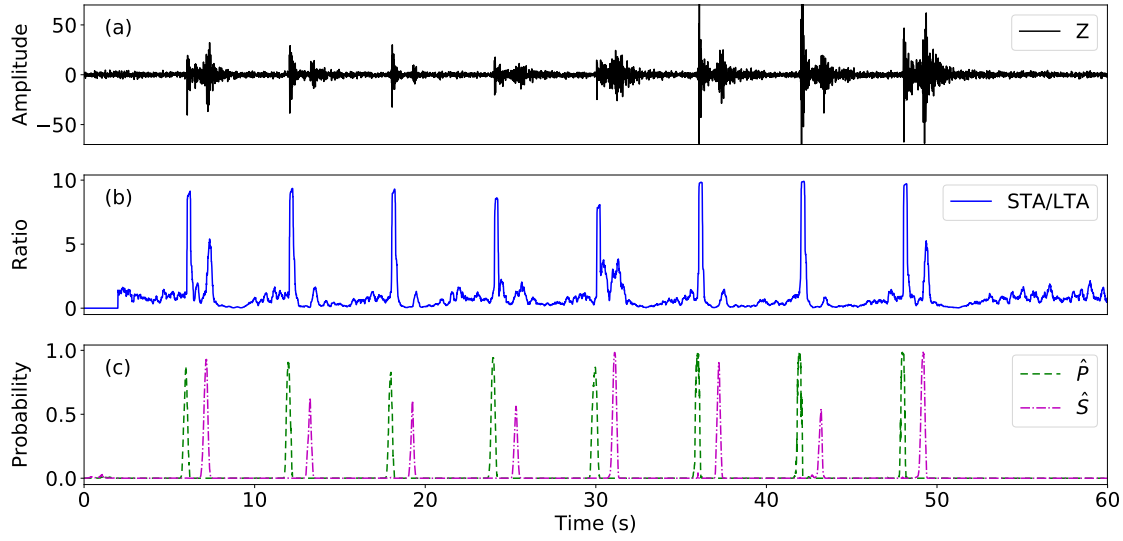


Figure 3.13: Synthetic continuous seismic waveforms. (a) waveform of vertical component. (b) output of basic STA/LTA in Obspy. (c) output of PhaseNet. The continuous data is created by stacking waveforms of eight events. The first-arrival-time interval between adjacent events is six seconds. The STA/LTA method runs on the vertical component. The PhaseNet runs on three components.

as:

$$\text{precision} : P = \frac{T_p}{T_p + F_p} \tag{3.1}$$

$$\text{recall} : R = \frac{T_p}{T_p + F_n} \tag{3.2}$$

$$\text{F1 score} : F1 = 2\frac{P \times R}{P + R} \tag{3.3}$$

where T_p is the number of true positives, F_p is the number of false positives, and F_n is the number of false negatives. Peak probabilities above 0.5 are counted as positive picks. Arrival-time residuals that are less than 0.1s ($\Delta t < 0.1s$) are counted as true positives. Picks with larger residuals are counted as false positives. F1 score is a balanced criterion between precision and recall. For example, if we set a very high threshold for positive picks, only the best picks are reported positive which is only a small portion of total true picks, so that we can get a very high precision but a very low recall. The opposite case of a very low threshold will lead to low precision but high recall. For both cases, F1 score will be low, which is less sensitive to the selection of threshold and could provide more accurate evaluation of algorithms' performance. We compare our results with those obtained by the open-source "AR picker" (Akazawa, 2004) implemented in Obspy (Beyreuther et al., 2010). The results of both PhaseNet and AR picker are shown in Table 3.1. For our data set, our method

achieved significant improvements, particularly for the S waves. Because S waves emerge from the scattered waves of the P coda, picking S arrivals is more challenging for automatic methods.

Figure 3.6 shows the distribution of time residuals between the automated and human-labeled P and S picks. The residual distributions of the P-picks are much narrower than for the S-picks, which is consistent with the fact that P-wave arrivals are expected to be clearer and hence easier to pick. The residual distributions of both P and S picks for PhaseNet are distinctly narrower and do not have obvious biases compared with the results from the AR picker.

Figure 3.7 shows performances of PhaseNet on different instrument types. The same model, which is trained on all instrument types, is used for testing, while the test dataset is divided based on each instrument type. Without changing any parameters or thresholds, the performance of PhaseNet is robust on different instruments. Despite the waveform differences between short period and broad band, high gain and low gain, accelerometer and seismometer, PhaseNet learns the common features needed to detect P and S phases and picks the correct arrival times.

Figure 3.8 shows the change of performances of PhaseNet with different signal-to-noise ratios. The test set is divided into 10 different categories based on the value of $\log_{10}$(SNR). Precision, recall and F1 score are calculated for each category. All the three evaluation metrics increase as the improvement of signal-to-noise ratio. The F1 score exceeds 0.9 for P and 0.8 for S when the value of $\log_{10}$(SNR) is over 0.5. The precision of PhaseNet is high even for low SNR data while the recall rate becomes relatively smaller. This reflects the fact that for noisy data the seismic signals may be overwhelmed by noise and become harder to detect.

It is instructive to look at a handful of representative results. Figure 3.9 shows good examples from the test dataset. For different waveform characteristics and P-S time intervals, PhaseNet successfully picks the P and S arrivals. The peaks of the predicted distributions accurately align with the analyst-labeled P and S picks. Figure 3.10 shows some apparently failed cases. The P and S first arrivals are harder to distinguish and the waveforms are more noisy and complex than those in Figure 3.9. Figure 3.11 shows some interesting cases where the P or S arrival times picked by analysts may be incorrect. The predictions of the neural networks appear more reasonable and consistent. Because there are subjective factors in seismic-phase picking, analysts may use different criteria to pick arrivals. Picks by the same analysts may also differ at different times.

To analyze the representations that PhaseNet has learned, we train another model without the skip connection which forces all the information to go through the deepest layer, and apply PCA (Principal Component Analysis) analysis to the neural weights of the deepest layer (Figure 3.5). The neural network condenses the knowledge from high dimensional raw waveforms into a few parameters in the deepest layer, which means that these low dimensional neural weights should contain the information needed to determine P vs. S arrivals. We feed in seismic data with P arrivals, S arrivals or only noise, and record the corresponding vectors in the deepest layer. The PCA visualization (Figure 3.12) shows that these condensed vectors group to different regions for

P-pick, S-pick, and noise. This demonstrates that the neural network has learned to extract the characteristic features to differentiate between P-pick, S-pick, and noise from the raw data and capture them in the condensed neural weights in the deepest layer.

PhaseNet predicts the probability distributions of P and S picks for every data point in the time series, so it may be applied to continuous data for earthquake detection. We have created continuous seismic data by stacking waveforms of eight different events (Figure 3.13). These events are shifted to make the arrival-time interval between adjacent events equal to six seconds. We have applied both basic STA/LTA in Obspy and our PhaseNet method to this sequence. The short and long sliding windows of the STA/LTA method are set to 0.2s and 2s. PhaseNet does not need a sliding window, but takes the whole 60-second waveforms as input and outputs three probability sequences for P-pick, S-pick, and noise. The output sequences in Figure 3.13 show that PhaseNet produces similar spikes as STA/LTA methods, which are commonly used for earthquake detection; however, PhaseNet can also differentiate between P and S arrivals. This information could also be used to reduce false detections, because events with both P and S picks are more likely to be a true earthquakes compared with the undifferentiated spikes reported by STA/LTA. The PhaseNet model in this study is trained on a dataset of detected earthquakes. It is designed to pick seismic phases accurately. In order to apply PhaseNet for detection on continuous data, a new dataset that includes more non-seismic signals should be used for training to let it learn the features to differentiate the noise spikes which look similar to seismic phases.

3.5 Discussion

We have shown that PhaseNet can detect and pick P and S arrivals effectively within known earthquake waveforms. The F1 score provides a balanced assessment of algorithm performance in both precision and recall. PhaseNet achieves an F1 score of 0.896 for P arrivals and 0.801 for S arrivals, which is substantially better than the AR picker (0.558 for P arrivals and 0.165 for S arrivals). We have chosen a strict threshold for true positive ($\Delta t < 0.1s$) during evaluation. If we were to relax this standard, the F1 score would be even higher. Our method differs from that proposed by Ross and Ben-Zion (2014), because PhaseNet does not explicitly use polarization analysis to separate P from S waves. PhaseNet automatically learns features, which might implicitly include polarization, to distinguish P from S waves. We find that the improvement S-picks is more significant than the improvement to P-picks, which suggests that the features learned from data are more effective than manually defined features.

The STA/LTA method is based on detecting a sudden change in waveform amplitude. But the S phase is always contaminated by the P coda, which degrades the STA/LTA ratio to select an accurate S-pick. PhaseNet has an advantage here in that it can learn features other than amplitude both to detect S waves and to differentiate between P and S waves. Figure 3.14 shows examples of

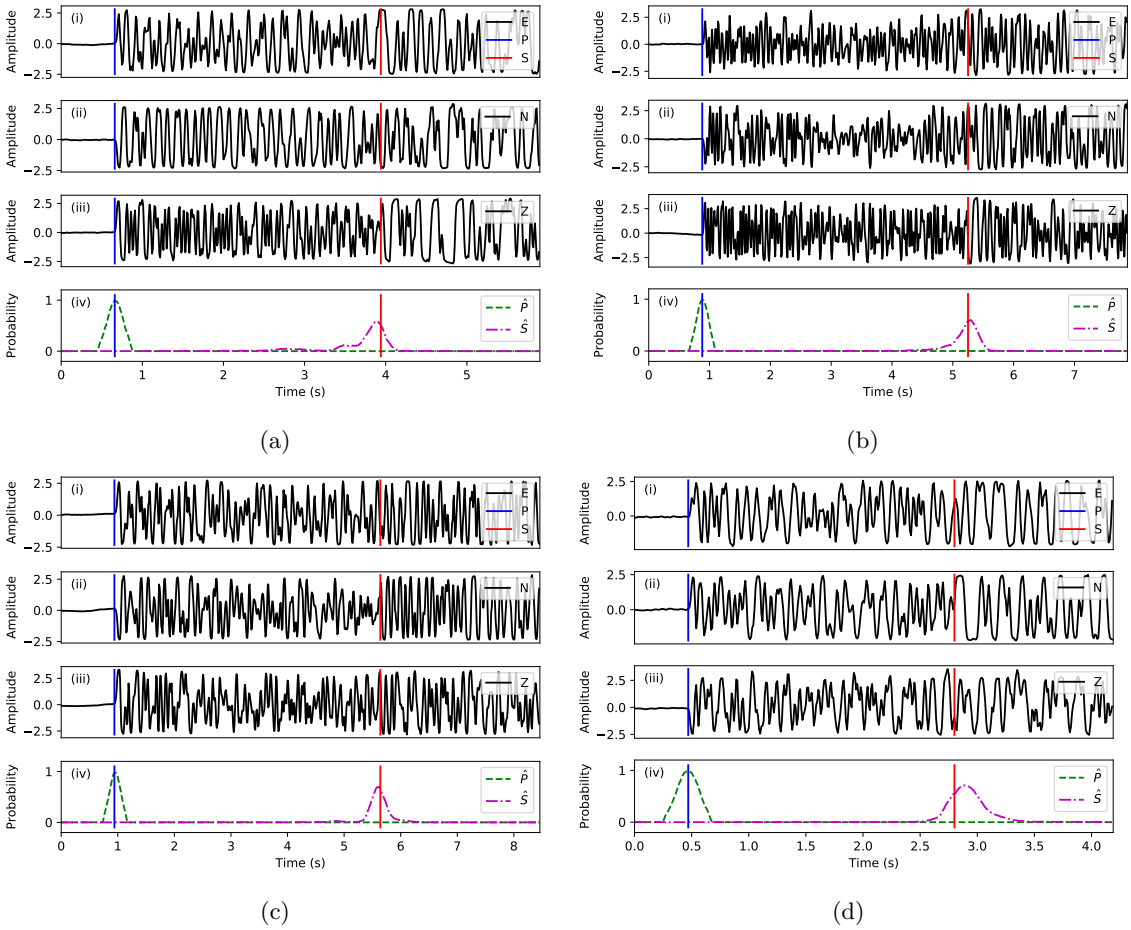


Figure 3.14: Examples of amplitude clipped waveforms.

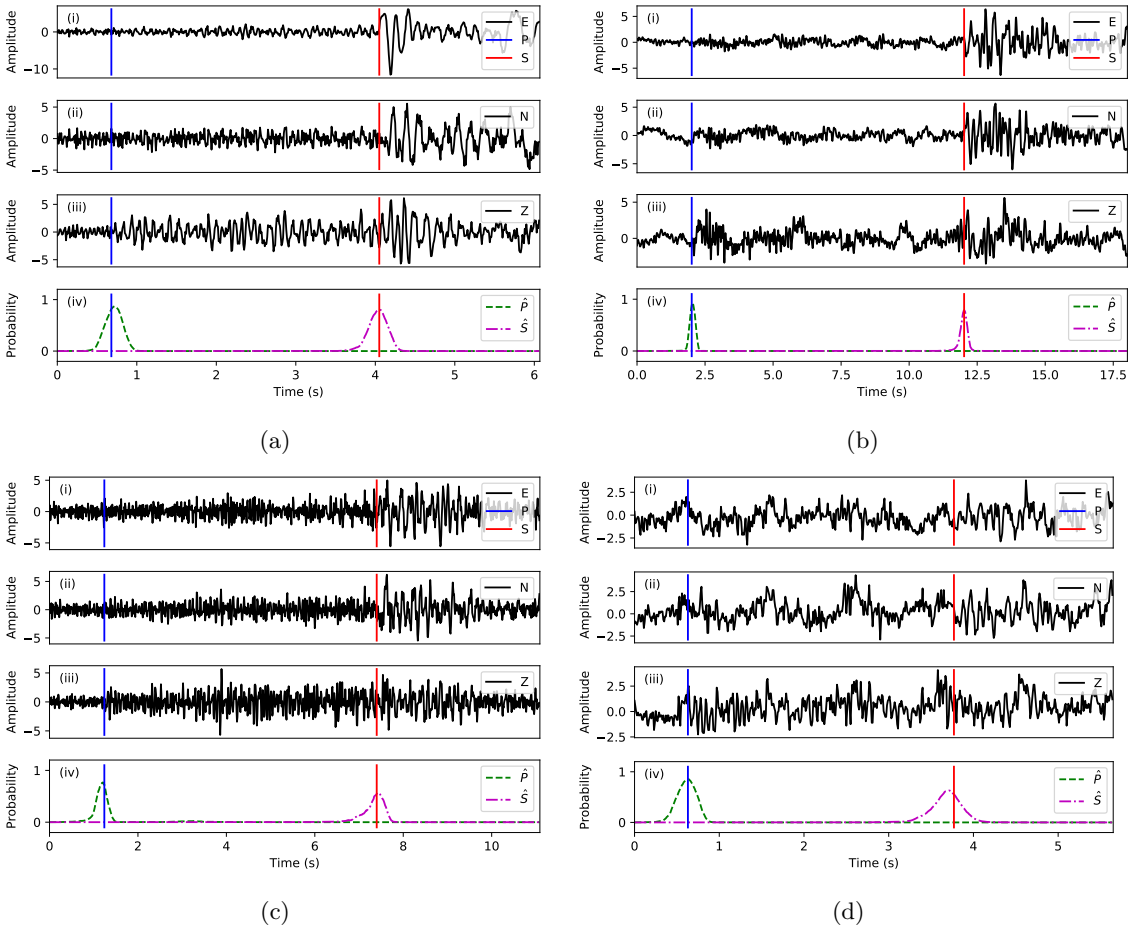


Figure 3.15: Examples of low SNR data

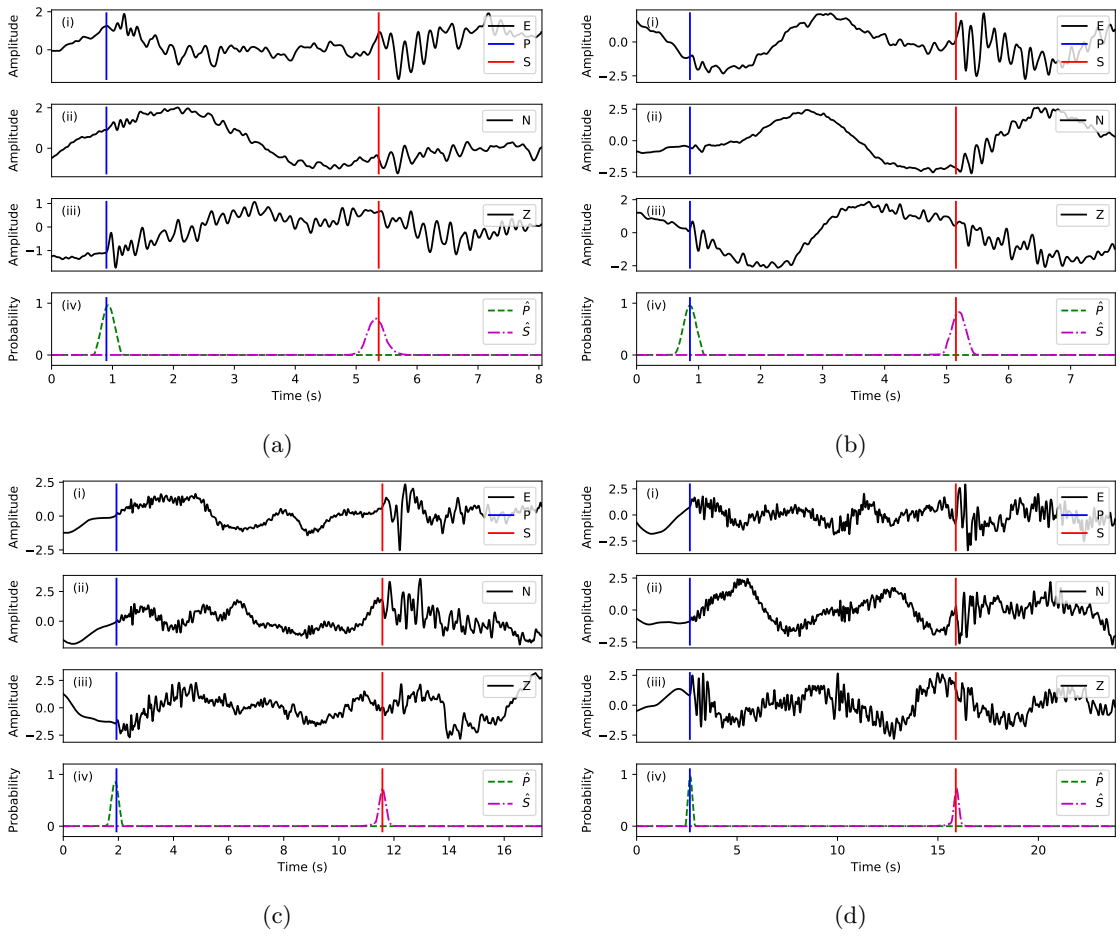


Figure 3.16: Examples with background variation

PhaseNet applied to clipped waveforms. Although the amplitude is strongly clipped, PhaseNet is still able to pick S arrivals successfully.

We have not pre-processed the data with denoising techniques such as band-pass filtering. As a result, our dataset contains a number of low signal-to-noise ratio data. We apply the AR picker after pre-processing the data with a band-pass filter of 0.1Hz - 30Hz. Without filtering, its performance would be substantially degraded. PhaseNet does not require this pre-processing because it not only learns the characteristics of P and S waves, but it also learns what kind of data is noise. Figure 3.15 shows several prediction results on low SNR data, for which it would be difficult for analysts to pick P and S arrivals. Despite these challenges, PhaseNet predicts accurate arrival-times at high probabilities. Figure 3.16 shows examples with strong low frequency background noise. PhaseNet can accurately pick both P and S phases without the need for filtering.

The STA/LTA method is sensitive to the threshold selected to determine P or S arrivals, and there is an inevitable trade-off between too high and too low thresholds. Moreover, it is prone to a delayed arrival-time if the threshold is set too high. Instead of an unbounded STA/LTA ratio, PhaseNet estimates probability distributions between [0, 1]. We have set the threshold of probability to 0.5 for both P and S picks. Tuning this threshold can further improve the performance, but the effect is not significant. Unlike STA/LTA, this threshold will not systematically bias arrival times, because this threshold is only used to decide if it is a positive pick. The accurate arrival time is measured from the peak of the probability distribution and does not depend directly on this threshold.

PhaseNet is not constrained by the number of earthquakes inside one time window. Because it predicts probability sequences of the same length as input waveforms, there could be several peaks or no peaks in P or S probability distributions inside the window. As shown in Figure 3.13, PhaseNet converts the 60-second waveforms into probability distributions with several spikes of P and S arrivals. We can apply PhaseNet to continuous data to generate running probability distributions of P and S arrivals, which can be used as the basis of earthquake detector when paired with an association algorithm. Accurate phase arrival times can be used to get absolute earthquake locations and to develop seismic velocity models. PhaseNet provides an improved method to get accurate S arrivals, which will be useful for developing better S-wave velocity models and improving earthquake locations.

3.6 Conclusion

Deep learning methods are improving rapidly. An important ingredient for improving them is the existence of large labeled data sets. In seismology, we are fortunate to have such large data sets ready at hand in the form of decades of arrival times with accompanying waveforms. We are on the verge of, or perhaps have already arrived at, a threshold where neural networks are “superhuman”

in the sense that they can outperform human analysts. In this study, we have built a training dataset using manually picked P and S arrival-times from the Northern California Seismic Network catalog. We have developed PhaseNet, a deep neural network algorithm that uses three component waveform data to predict the probability distributions of P-pick, S-pick, and noise. We extract arrival times from the peaks of these distributions. Test results show that our method achieves significant improvements compared with existing methods, particularly for S waves. PCA visualization shows that the condensed neural weights contain characteristics that allow the separation of P wave, S wave, and noise. While further testing against existing methods is required, we are not far from making such a capability operational. An increase in accurate P and S arrival-times will help us to continue to extract as much information as possible from rapidly growing waveform data sets for earthquake monitoring, and the ability to extract reliable S arrivals will allow us to improve shear wave velocity models substantially, which will be especially useful for prediction of path effects in strong ground motion prediction. Finally, we note that PhaseNet can also be used for other phases for which manually labeled training datasets are available.

Chapter 4

Seismic Signal Augmentation to Improve Generalization of Deep Neural Networks

Deep learning has emerged as an effective approach for seismic data processing in general, and for earthquake monitoring in particular. The ability of deep learning models to generalize beyond the training and validation data is important for comprehensive earthquake monitoring; this ability, furthermore, depends on the availability of a sufficiently large and complete training dataset. However, this requirement can prove challenging to meet due to significant effort and time for data collection and labeling. Data augmentation provides an efficient and effective approach for increasing the dimension of training samples and improving generalization to unseen samples. In this study, we present augmentation methods appropriate for seismic waveforms and demonstrate their ability to reduce bias and increase performance. These augmentation methods can be applied to a wide range of deep learning applications designed for seismic data.

4.1 Introduction

Deep learning has been successfully applied to a wide range of seismic problems, such as earthquake detection (Dokht et al., 2019; Mousavi, Zhu, Sheng, et al., 2019; Perol et al., 2018; Wu, Lin, Zhou, et al., 2018; Zhou et al., 2019; Zhu, Peng, et al., 2019), clustering (Mousavi, Zhu, Ellsworth, et al., 2019), phase detection and picking (Ross, Meier, Hauksson, & Heaton, 2018; Wang, Xiao, et al., 2019; Zheng et al., 2018; Zhu & Beroza, 2018), polarity determination (Ross, Meier, & Hauksson, 2018), earthquake location (Mousavi & Beroza, 2019; Zhang et al., 2020), magnitude estimation (Mousavi & Beroza, 2020), denoising (Si & Yuan, 2018; Zhu, Mousavi, et al., 2019), phase association

(McBrearty, Delorey, et al., 2019; Ross, Yue, et al., 2019), among others. Based on deep neural network layers, deep learning algorithms exploit the unknown structure in seismic data, extract useful features and representations of seismic waveforms, and learn a map to a target distribution of interest, such as probabilities to separate earthquake from non-earthquake signals. Because the performance of a deep neural network improves with the number of diverse training samples, large seismic datasets like STEAD (Mousavi, Sheng, et al., 2019) have been created for deep-learning-based research. However, large-scale training datasets do not exist for every problem, e.g., there is a limited number of very large earthquakes, which due to the power-law distribution of earthquake magnitudes are (fortunately) rare. Moreover, building a high-quality large training dataset, with sufficient labeling and quality control, requires significant effort and time. One way to circumvent these problems is through data augmentation, which consists of various techniques to generate new training samples based on collected datasets to expand the size and variety of training samples. Data augmentation has proven successful in avoiding overfitting and improving the generalization of deep learning models trained on small training datasets and thus shows potential for applications on seismic datasets.

"Generalization" commonly is used to refer to the process of recognizing that a specific feature belongs to a larger category. In deep learning, "generalization" denotes the ability of a trained neural network to perform well on data that were not used in its training or validation (e.g., a unique seismic source or seismograms from a region not used in training). Among the factors that affect generalization are the network architectures, optimization techniques, and training datasets. The size, accuracy (of labels), and completeness of the training datasets are key elements for developing a well-performing model. A training dataset may lack any or all of these properties; for this situation, data augmentation can provide an effective option to improve a model's performance. In high-dimension data space (Figure 4.1), the limited training samples, including signals and noise, may only span a small subspace and provide weak constraints on the possible decision boundaries learned by the neural network. This can result in poor performance either in the form of low true positives or high false positives. Data augmentation is designed to increase the training sample size and complexity, thus expanding the sampled feature space such that the neural network can learn an improved decision boundary to reduce both false positives and false negatives and to improve generalization on unseen samples.

Data augmentation is commonly used in training deep neural networks to boost performance in classification and recognition problems for computer vision (Krizhevsky et al., 2012; Mikołajczyk & Grochowski, 2018; Perez & Wang, 2017; Simard et al., 2003), audio processing (Cui et al., 2015; Salamon & Bello, 2017), and other areas (Fadaee et al., 2017; Frid-Adar et al., 2018; Um et al., 2017). When implemented correctly, data augmentation reduces the risk of over-fitting by increasing the number and variability of training data. It can be especially effective for applications with scarce labeled data. Most data augmentation methods are designed for images, for which

semantic meaning can be easily preserved. However, some augmentations appropriate for vision, e.g., horizontal flipping, rotating, and shearing, are inappropriate for seismic data because these processes violate the physical properties of the waveform data of interest. Very few studies exist on augmentation techniques for seismic data. Because seismic data are usually collected and organized in a standard way, i.e., using a fixed time window around the P-wave arrival, obvious augmentations are needed to prevent the neural networks from learning and memorizing any artifacts in how the training data are organized. Less obvious augmentations, such as random shifting, recombining events and noise, and channel (and station) dropout, are consistent with the character of seismic signals and can mitigate bias in training data and increase model performance. In this study, we demonstrate these techniques and explain how they help to improve generalization of the model to cases of interest including: low quality data, complex noise, and closely recorded earthquakes in earthquake swarms. Our examples demonstrate that with appropriate augmentation, we can expand the application of deep learning to smaller datasets than would otherwise be possible for purposes of earthquake monitoring.

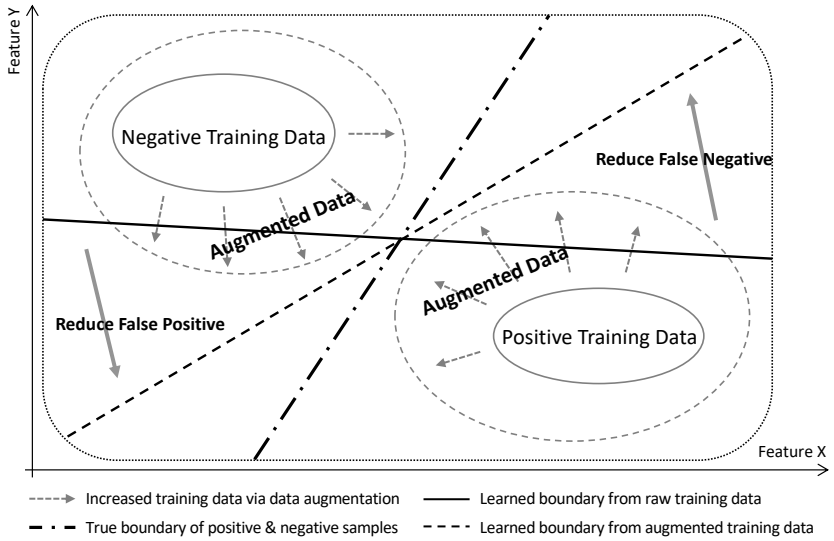


Figure 4.1: Data augmentation expands the data space (small dashed arrows) spanned by collected seismic signals and noise, and results in improved constraints on the decision boundary (the shift from solid to dashed lines). With a broader data space exploited by data augmentation, the trained deep neural networks can generalize better, with reduced false negative and reduced false positive rates, to unseen signals and noise.

4.2 Benchmark Data and Training Procedure

We collected a small, high signal-to-noise ratio (SNR) training dataset with accurate manual labels of 500 earthquake waveforms from the Northern California Earthquake Data Center (NCEDC) (NCEDC, 2014) recorded before 2018 to demonstrate the application of augmentation for training deep learning models on seismic data. In the same way, we created a validation dataset with another 500 high SNR earthquake waveforms before 2018, used to choose the best model during training. We evaluated the augmentation methods with a much larger test dataset of 10,000 earthquake waveforms recorded in 2018. This choice of ratios between training, validation, and test datasets, is purposely designed to evaluate the effect of augmentation on small datasets. For real applications, we would need to choose a larger sample size for the training dataset than the validation and test datasets. The SNR distribution of the data is shown in Figure 4.2. Here the SNR is calculated based on the standard deviations of waveforms before and after the first manually picked P-wave arrival. The distribution of epicentral distances of the test dataset is shown in Figure 4.2d, with source-station paths mainly ranging from 0-120 km. The training and validation sets (not illustrated) have similar distributions of epicentral distances.

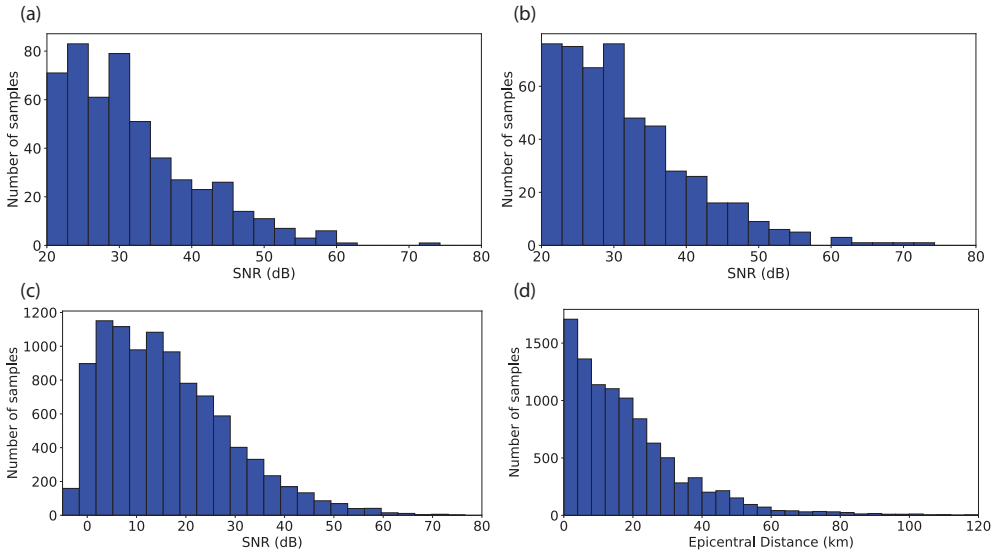


Figure 4.2: Statistics of the benchmark datasets: the SNR distributions of (a) 500 high quality training samples (SNR > 20dB) and (b) 500 high quality validation samples (SNR > 20dB), both sampled from earthquakes that occurred prior to 2018; (c) the SNR distribution of 10,000 test samples from earthquakes that occurred in 2018; (d) the epicentral distance distribution of the test samples. The training and validation datasets have similar epicentral distance distributions, which are not shown here

We used the now well-studied phase picking problem as an exemplar for the effects of data

augmentation for training deep neural networks. We used the same neural network architecture as PhaseNet (Zhu & Beroza, 2018), i.e., a fully convolutional neural network designed for seismic phase picking problems. To avoid the complex effects of hyper-parameter tuning, we removed dropout, learning rate decay, and weight decay (regularization) and keep only batch normalization in the architecture. We used the Adam optimization with a learning rate of 0.01, a batch size of 20, and 100 total epochs. Augmentation increases the size of the training dataset by generating a large number of new training samples. Here, we kept the total number of training samples the same in each epoch (500 samples) and apply data augmentation on the fly, so that augmentation increases the variety of training samples in each epoch. The neural network architecture, training procedure, and optimization method will also affect the generalization for different problems and data formats (He et al., 2019). In this study, we keep the training procedure simple and focus on the effect of data augmentation for seismic data. With suitable modification, our results should apply to related problems like earthquake detection, phase detection, as well as other deep learning applications to seismic data.

4.3 Augmentations

Many data augmentation techniques have been proposed and applied in image and acoustic signal processing. Here we examine those augmentations of particular relevance to processing seismic data. Note that due to the randomness in initialization and optimization of neural networks and data augmentation, the reported performance will have a certain randomness.

4.3.1 Random shift

Seismic training data are usually collected based on the phase arrival information from earthquake catalogs. Therefore, it is tempting to define a cutout window based on the time relative to a particular seismic phase such as the first P-wave arrival. Training neural networks using data with a fixed reference time or with a limited time shift of a few seconds around the reference time point can introduce positional bias into the model. Such a neural network model is prone to memorizing the location of the anchor time point rather than learning more general functions from the feature space.

Here, we train three models on data with a 30-second time window using: no random shift, limited random shift from 10-15s, and full random shift from 0-30s. The random shift is applied on the fly so that each training waveform is shifted differently in each epoch. We examine the P-wave arrival predictions by sliding the test waveform from the left side to the right side of the window and record the predicted activation scores at the true P-arrival locations (Figure 4.3). For the model trained without random shift, regardless of where the true P-wave arrival is, the neural network continues to predict a high probability score at 10s, which is the fixed P-wave arrival time during

Table 4.1: Comparison between two implementations of random shift: (a) precomputing a fixed random shift for training samples; (b) calculating random shifts on the fly so that each training sample has a different shift in each epoch.

Random shift type		Precomputed	On the fly
P-wave	Precision	**0.908**	0.902
	Recall	0.550	**0.588**
	F1-score	0.685	**0.712**
S-wave	Precision	0.738	**0.748**
	Recall	0.529	**0.571**
	F1-score	0.616	**0.647**

training (Figure 4.3a-c). For the model trained with limited shift within 10-15s, the result shows a statistical bias that phases are expected to exist within a limited window. The predicted activations fall off at the edges of the time window (Figure 4.3d). This bias can be neglected in evaluation if the test dataset is shifted within the same window from 10-15s; however, the bias result in deteriorated performance when applied to continuous data where the signal arrival time is not known a priori. A very narrow shift range can make a model ineffective for detecting earthquakes outside the shift window used in training. In contrast, the case with random shift from 0-30s has high activation scores across the whole window. This potential bias from random shift could also occur in other deep learning applications, such as earthquake detection, based on a specific window size.

In addition to mitigating the potential bias across the prediction window, random shift can also increase sample variety and improve detection performance. Table 4.1 compares two implementations of random shift: applying a fixed set of shifting times to the 500 training samples, and applying random shift on the fly so that each sample is shifted differently in each epoch. Random shift on the fly allows a greater variety of time shifting on different training epochs and leads to better performance. In contrast, a fixed set of time shifting for all epochs may only sample limited time points, especially for a small number of training data. In order to effectively apply random shift on the fly within the 30s window in this case, we cut the training sample to a larger window of 90s to avoid the need for zero padding. Zero padding may itself introduce a subtle bias by having the neural network learn the transition from zeros to signals and use this artifact as the basis for its predictions. Investigating the distribution of arrival times in a training dataset prior to and after the augmentation is a good practice for deep-learning-based seismic phase detecting/picking models.

4.3.2 Superimposing events

With the exception of earthquake swarms and aftershock sequences, catalogued earthquakes tend to occur in isolation; thus, training samples contain only one earthquake, commonly referred to as an “event” in the training window. However, training only on single event data can lead neural networks to learn a subtle bias of expecting only one event within the duration of the window and

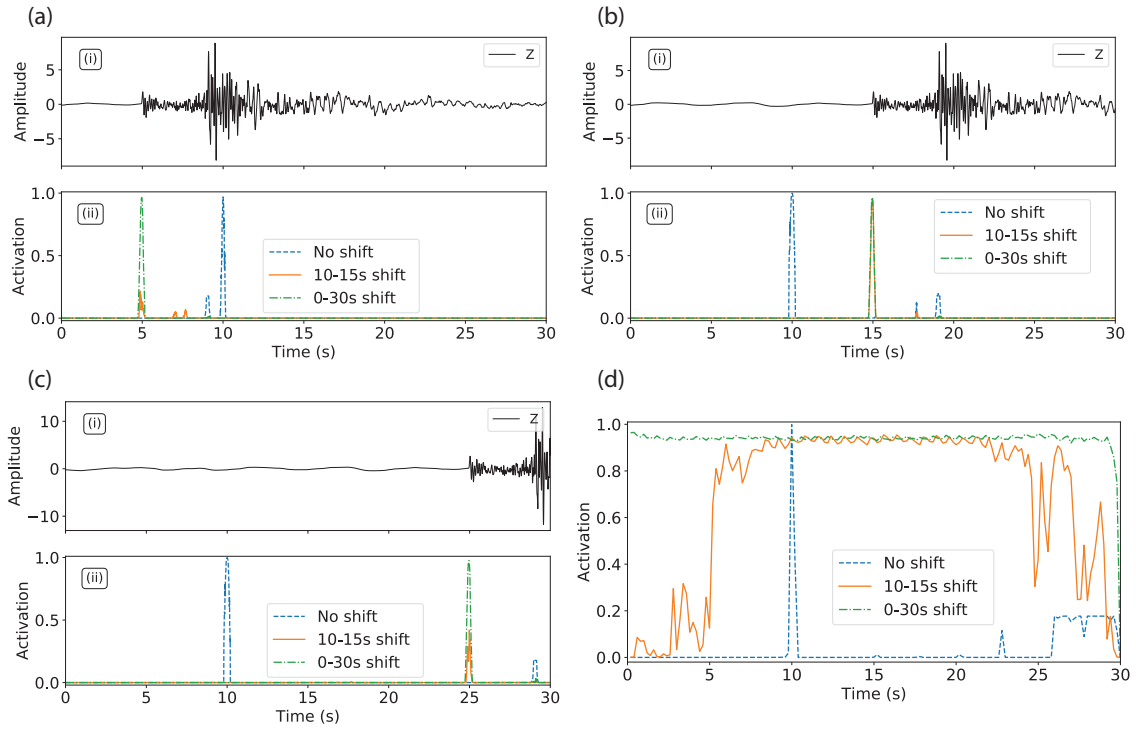


Figure 4.3: Activation scores from three models trained with different random shift augmentations. (a)-(c) show the predicted probability sequences of neural networks trained with no random shift (blue short-dashed line), random shift within 10-15s (orange solid line), and random shift within 0-30s (green alternate short-long dashed line). The test waveform slides from the left to the right edge of the window, and the cases with P-wave arrivals at 5s, 15s, 25s are shown in (a)-(c). (d) shows the P-wave arrival predictions at every time point when sliding waveform across the window. Training without random shift leads the neural network to learn a fixed time regardless of the waveform content. Training using incomplete shift between 10-15s leads to activations decay at the edges of the considered window, causing missed detections when applied to continuous waveforms.

suppressing the detection of smaller events that are also present in the time window. This bias can result in missed events for semantic-segmentation-based methods (Mousavi, Zhu, Sheng, et al., 2019; Zhu & Beroza, 2018), which are designed to detect every event in a time window. We would like a well-trained neural network to perform appropriately on normal earthquakes, but also to generalize to extreme cases, like earthquake swarms and induced earthquakes, when information is dense and earthquakes occur at much more frequent intervals than is usually the case.

An effective augmentation to address the case of multiple events in a short window is to artificially superimpose events in a way that mimics such cases and removes the bias of only one event existing in each window. Superimposition simply means adding two or more time series together, often referred to as "stacking" in seismological parlance. During superimposition, we also apply a random ratio between event amplitudes, which further enhances the neural network's ability to

detect smaller earthquakes that occur close in time to larger ones. Figure 4.4 shows one example with two earthquakes occurring close to one another in time. The neural network model trained without superimposed events only detects the first large event and neglects the second smaller one (Figure 4.4(iv)); however, the model trained with superimposed events predicts the second smaller event with high probability scores for both P and S waves (Figure 4.4(v)).

Although event waveforms in real data may completely overlap each other when a station simultaneously records two earthquakes, we avoid synthesizing these cases. These are event waveforms that usually left unused during the manual processing. Because neural networks' performance is highly dependent on training data, these cases have the potential to increase false positives on common seismic waveforms. Thus, we avoid implicitly introducing fully masked events to the training through augmentation. Our observations suggest that a well-trained model should be capable of generalizing to the events with overlapping waveforms. In special applications to realistically replicate an earthquake swarm, stacking much more overlapping events could be useful. Since most of the datasets provide no information regarding the end of waveform (or end of the earthquake coda), we can roughly estimate the end of the earthquake coda using the time between the P and S arrival times. Another option is to use measurements based on envelope functions similar to those used in coda magnitude estimation.

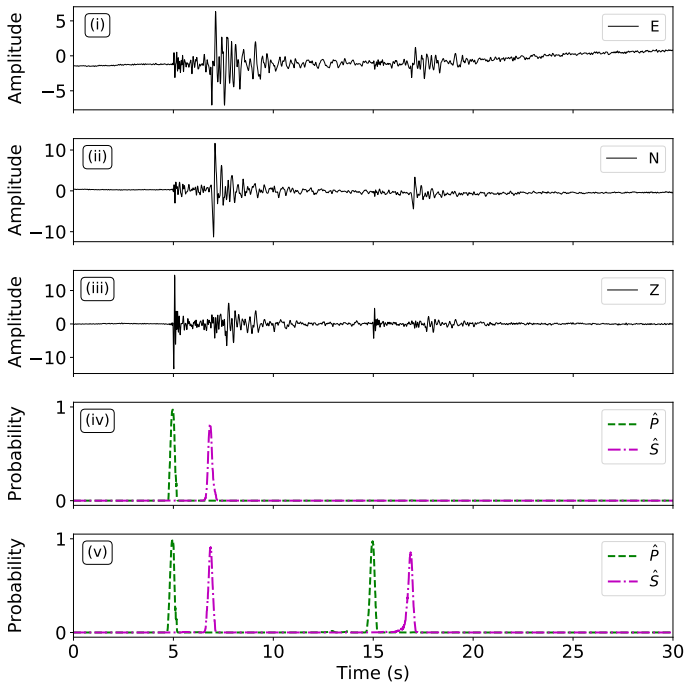


Figure 4.4: Comparison of predicted activations for training without and with superimposed events: (i)-(iii) waveforms of ENZ channels; (iv) training without superimposed events; (v) training with superimposed events.

4.3.3 Superposing noise

Although most manual labels are selected from high-quality seismic data, a robust neural network should also be able to work on low quality or otherwise complex data. Superposing noise is a straightforward way to increase the performance of neural networks applied to low signal-to-noise ratio (SNR) data. A distinct advantage of this augmentation is that the high reliability of labels from high SNR data can be retained when the waveforms are superposed with strong noise. Because the augmented weak signals are de-amplified versions of known high SNR signals, their labels are more accurate than those on low SNR signals. By controlling the ratio between the signal and the superposed noise, we can influence the detection limits of the neural network. In particular, by superposing strong noise, we can push the neural network to detect weak signals hidden inside the background noise; however, it should be noted that the potential for false positives may increase as well. Nevertheless, superposing noise is also an effective way to mitigate overfitting on a small training dataset because noise samples are easily obtained from continuous seismic recordings or from synthetically generated random noise.

Figure 4.5 compares the neural network's performance with and without superposing noise. It is clear that high scores of precision, recall, and F1 score can be obtained on the high SNR test samples (> 20 dB) training with only a small training dataset. However, recall is much lower for the low SNR test samples (≤ 20 dB) when only high-quality samples are used for training. After training with superposing noise as augmentation, the recall and F1 score are significantly improved for the low SNR data; meanwhile, the performance for high SNR data is maintained. Note that the increase of recall is more significant than precision; this is because the neural network trained with augmented noisy data becomes more sensitive to weak signals buried inside noise and recovers more events, but this may also increase the potential of false positives, thus limit the improvement in precision. Here we have fixed the activation score threshold for comparison. In practice, we can tune a sequence of activation thresholds and generate a receiver operating characteristic (ROC) curve to balance between precision and recall and to determine a better activation threshold with both improved precision and recall.

The choice of superposed noise depends on the specific environment, such as applications to borehole data, urban data, ocean bottom seismometer (OBS) data. Using real seismic noise recorded by seismic instruments in situ should result in a more realistic augmentation and better performance; however, care must be taken to avoid including undetected events within the noise windows, which would inadvertently increase the mis-labeling rate in the dataset and deteriorate the performance. There is also a risk of biasing the performance for a specific type of instrument if the noise waveforms are not sampled appropriately. Although the use of Gaussian noise would be safe in terms of avoiding mislabeling, the result would be a less realistic representation of real seismic noise, which is usually strongly coherent and non-stationary. Thus, real seismic noise is preferred for augmentation provided that a reliable set of noise samples that have been checked for the existence of non-cataloged

earthquakes is available.

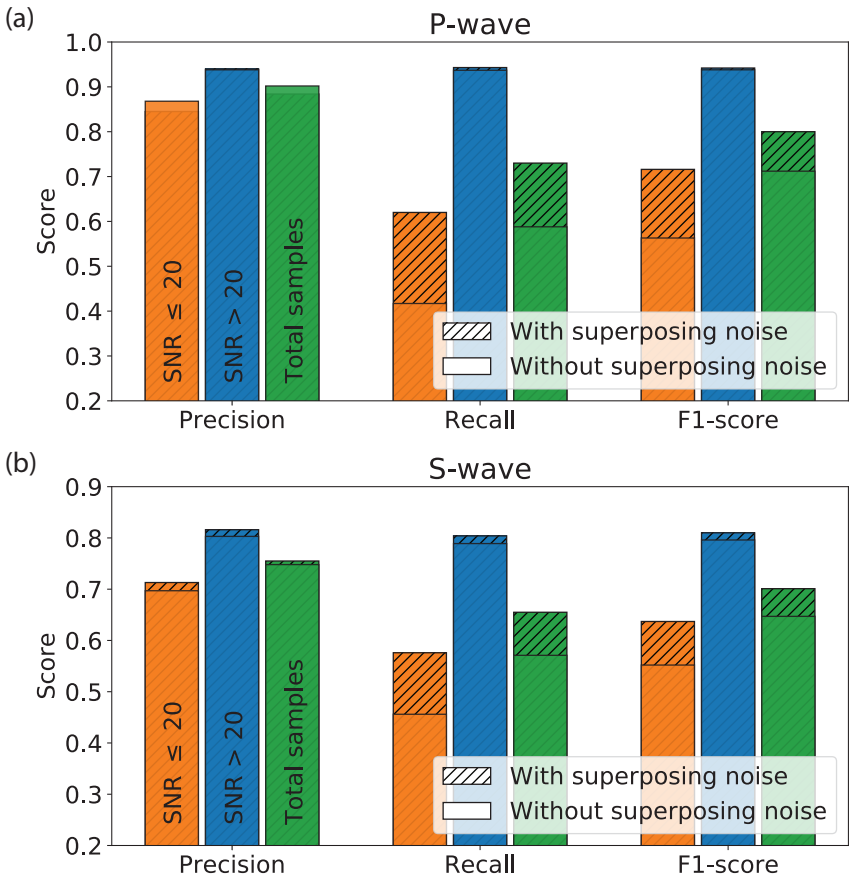


Figure 4.5: Comparison of detection performance with and without noise superposition augmentation: (a) P-wave arrival; (b) S-wave arrival. For high quality test data (SNR $>$ 20dB), both models achieve a high performance; however, the model with augmentation significantly increases the recall rate and F1 score for low quality test data (SNR $\leq$ 20dB).

4.3.4 False positive noise

As shown above, superposing noise on earthquake signals helps improve the performance of neural networks for low SNR data. Adding false positive signals (non-earthquake signals) is another way to deal with complex noise effects, such as shaped pulses from urban vibration. This is particularly effective for training neural networks to recognize negative samples (i.e., reducing the false positive rate). Due to the complex noise and non-stationary sources of noise in continuous seismic data, a small training dataset can only cover a limited range of noise. As a result, the neural network

trained on limited noise samples may result in many false positives on unseen noise with features similar to seismic signals. To tackle this problem, we can add these false positive noise examples or synthesize similar non-earthquake signals into the training dataset to retrain or fine-tune the neural network to learn features to recognize these false positives and correct their predictions.

Figure 4.6 presents a common case in seismic data acquisition when part of the data is missing due to instrumental or telemetry errors. This introduces abrupt changes in continuous data that may result in false positives. These types of false positives are common in traditional methods, like STA/LTA, and they can confuse deep learning methods if this kind of noise is absent from the training dataset. The model trained without appropriate augmentation can produce a false prediction of P and S waves at abruptly waveform change, as shown in Figure 4.6(iv). Adding a small number of similar kinds of noise samples into the training data, however, can effectively suppress false-positive predictions of this type. The same logic applies to other types of common false positive noise, such as impulsive signals from human activities. In practice, identifying different classes of common false positives is challenging due to the need for comprehensive test data and manual examination. To improve the efficiency of identifying false positive samples, active learning, active learning (Cohn et al., 1994; Kirsch et al., 2019; Settles, 2009) could be one option. Because manually recognizing false positives from a large number of unlabeled samples during application is often difficult, active learning aims to design a strategy to rank these unlabeled samples and annotate the most informative samples first, such as samples with the largest uncertainty. These samples are more likely to be recognized as false positives or false negatives, and adding them into training can improve learning efficiency.

4.3.5 Channel dropout

Three-component seismic data are the most common data form in modern earthquake seismology; however, single channel recordings dominate many historical archives and are still used in some deployments. Moreover, with three-component recordings, it is not uncommon for one of the channels to fail due to either instrument malfunctions or errors in telemetry. Augmentation is an effective strategy to improve the performance of models trained using three-component data on single-channel data. A suitable approach is to use a technique similar to dropout (Gal & Ghahramani, 2016; Srivastava et al., 2014). In the input layer, we randomly drop one or two channels from the ENZ input channels. This channel dropout trains the neural network to also predict on data with missing channels. For applications like phase association, where the training is done based on data from multiple stations, we can apply a similar approach to dropping data from part of the stations during training (Zhu, Tai, et al., 2019). This can prevent the network from overfitting on dominant stations and increases the neural network's robustness in the inevitable cases where data from some stations are missing or corrupted.

Table 4.2 compares the performance between training with and without channel dropout. We

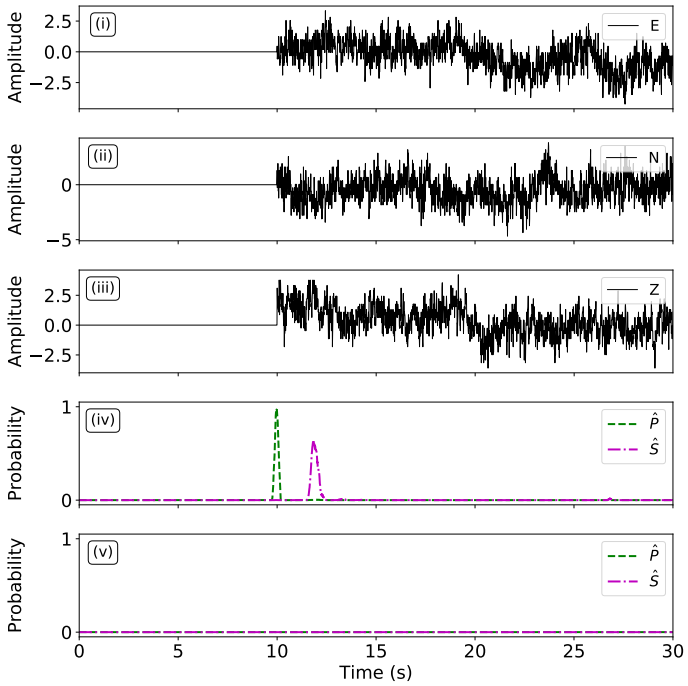


Figure 4.6: Comparison of predicted activations before and after adding false positive noise: (i-iii) waveforms of ENZ channels; (iv) training without false positive noise; (v) training with false positive noise.

apply the trained model on high quality test data (SNR $>$ 20dB) and examine the performance on different components. Both models show performance similar to that of three-component data; however, the model trained with channel dropout works better on single-component data. The performance on single E-, N-, Z-, and EN-component combinations is instructive and reflects the information learned by the neural network to distinguish P and S waves. For picking the P-wave arrival, performance scores using only the Z-component are similar to those using all ENZ-components, reflecting that the Z-component provides most of the information used for picking P-wave arrivals. In contrast, the horizontal EN-components contain the essential information for picking S-wave arrivals. This is in agreement with the polarization of P and S waves. The P waves appear stronger on the vertical component, while the S waves show up more strongly on the horizontal components.

4.3.6 Resampling

In deep learning, effective training using imbalanced datasets can be challenging (He & Garcia, 2009; Kotsiantis et al., 2006). This issue may be especially significant when training a neural network using earthquake signals due to the imbalance of earthquake magnitude distributions. A

Table 4.2: Comparison of detection performances on different channels.

P-wave performance		Z	E	N	EN	ENZ
With channel dropout	Precision	**0.952**	**0.869**	**0.886**	**0.893**	**0.957**
	Recall rate	**0.944**	**0.825**	**0.856**	**0.869**	**0.943**
	F1-score	**0.948**	**0.846**	**0.870**	**0.881**	**0.950**
Without channel dropout	Precision	0.944	0.718	0.712	0.795	0.938
	Recall rate	0.928	0.684	0.705	0.777	0.937
	F1-score	0.936	0.700	0.709	0.786	0.938
S-wave performance		Z	E	N	EN	ENZ
With channel dropout	Precision	0.523	0.748	0.777	**0.824**	**0.827**
	Recall rate	**0.436**	**0.732**	**0.767**	**0.808**	**0.810**
	F1-score	**0.476**	**0.740**	**0.772**	**0.816**	**0.818**
Without channel dropout	Precision	**0.630**	**0.771**	**0.793**	0.806	0.803
	Recall rate	0.019	0.683	0.740	0.787	0.789
	F1-score	0.036	0.724	0.766	0.796	0.796

power law relationship (Gutenberg & Richter, 1944) exists between earthquake magnitude and the number of earthquakes, meaning that the number of large magnitude earthquakes for the training is much more limited compared to the number of small magnitude earthquakes. This imbalance directly impacts applications like magnitude estimation using neural networks (Mousavi & Beroza, 2020). Similar issues can exist for distance, depth, geographic location, tectonic setting, source mechanism, magnitude type, instrument type, and SNR in a specific training set. Station coverage and configuration can also vary significantly among seismic monitoring networks. These imbalances can reduce the generalizability of a model trained on a specific dataset to a broader range of earthquakes. For this reason, it is necessary to investigate data properties during the construction of a training dataset. Based on such preliminary investigations, an appropriate resampling approach can be developed to address possible imbalance problems within a dataset.

Random resampling is a technique to deal with the imbalance issue by oversampling the minority class or undersampling the majority class during training, so that the class distribution does not become biased towards a few specific classes, and better generalization can be achieved by training on a more balanced sample distribution. Resampling can, however, bring about undesirable side-effects. Undersampling the majority classes comes with the cost of losing part of the training data and reducing the training size. Extreme oversampling, by repeating a few minority samples of similar magnitudes or from a same region, can also bias the neural network to simply memorizing these samples, which clearly works against generalization. Moreover, oversampling may have limited applications to large earthquakes. Not only are large events rarer, but they are also are more complex compared with small ones. Large earthquakes usually exhibit complex spatial and temporal rupture patterns involving multiple faults. Thus, oversampling may be insufficient to capture the full variety of large magnitude earthquakes. Combining oversampling with the augmentation methods discussed above could be a more effective way to increase both the ratio and variety of the minority

samples. Another alternative would be synthesizing training samples from existing instances using more advanced approaches, such as SMOTE (Chawla et al., 2002), ADASYN (He et al., 2008), and GAN (Goodfellow et al., 2014).

4.3.7 Augmentation for synthetic data generation

In some applications, augmentation techniques can be used to generate semi-synthetic data for training. Two such examples are the seismic denoising problem (Zhu, Mousavi, et al., 2019) and earthquake detection on scanned analog-seismograms (Wang, Zhu, et al., 2019), for which the ground truth (the training target) is unknown and infeasible to obtain from manual labeling. To tackle this problem, we can use augmentation to synthesize training input and target pairs from the abundant seismic waveforms. For example, Zhu, Mousavi, et al. (2019) generated an accurate denoising mask as the training target for neural networks based on high SNR earthquake signals and a group of noise waveforms. As a result, this augmentation provided a sufficiently large number of training samples by randomly combining signal and noise with a random ratio on the fly during training. In this way, the neural network is trained to learn a challenging inverse process to separate signal and noise in opposite to the forward synthesizing process.

Here we show another example of clipped seismic waveform recovery. Clipped waveforms commonly occur for moderate to large earthquakes recorded on nearby weak motion instruments (Yang & Ben-Zion, 2010; Zhang et al., 2016). Because the true unclipped waveform cannot be observed at the station, we cannot directly get training data from historical waveforms. We can, however, synthesize training data from unclipped waveforms by manually clipping these waveforms. In this way, the input data for the neural network is the synthetically clipped waveforms and the training target the true unclipped waveforms, so that we can easily collect a large number of training data through augmentation. As in denoising, this augmentation has the advantage of being derived from a signal where the (unclipped) ground truth is known and provides an accurate training label. In this case, we use the same network architecture as the other cases but use a mean squared error (MSE) to measure the waveform difference between the recovered and the true unclipped waveforms. Figure 4.7 shows the recovered waveforms using the neural network model trained on the synthetic clipped waveforms.

Applying augmentation to synthesizing training data solves the problem of unknown ground truth for some applications. The idea is similar to generating training data using numerical simulations; however, the augmentation method generates the training data based on real seismic waveforms, which is efficient and results in samples that are ipso facto realistic. The trained model can generalize better from the semi-synthetic data to real seismic recordings. If we think of the data generation process as a forward operation, the neural network essentially learns an inverse modeling from the synthesized training data to the true signal of interest that underlies the synthetic data. On the other hand, for cases where not only the label is missing but the real data is also scarce, numerical

simulations could become a source for training data, such as finite fault modeling of large complex earthquakes. In this case, we could combine the synthetic earthquake waveforms with real noise to generate training data to improve detection on large magnitude earthquakes. However, the model trained with simulation data may have a generalization issue when applied to real seismic data. Model fine-tuning or transfer learning on a few real seismic waveforms would be needed to narrow the generalization gap. Many other algorithms in computer vision can also be used to bridge the domain gap between simulation and real word, such as adversarial discriminative domain adaptation (Tzeng et al., 2017). The importance of large earthquakes provides strong motivation for future research in this direction.

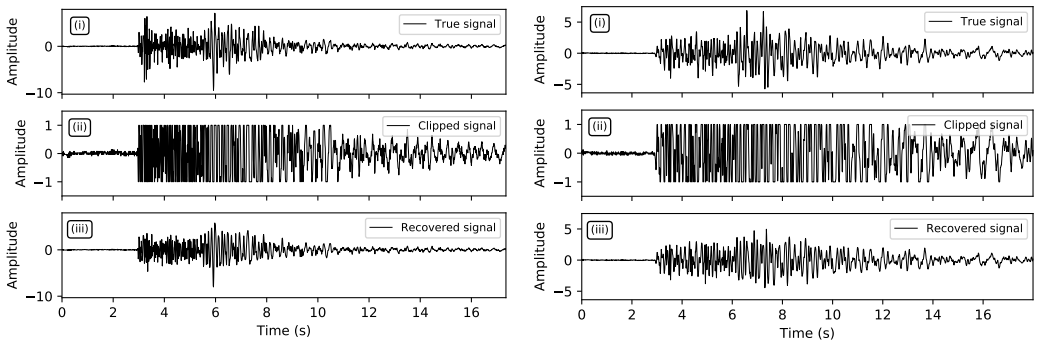


Figure 4.7: Example of clipped waveform recovery: (i) true seismic signals; (ii) manually clipped signals; (iii) recovered signals based on neural networks trained with input-target pairs in (i) and (ii).

4.4 Discussion

In this study, we have introduced and discussed several augmentation techniques that can improve the performance of deep learning methods for seismological applications. Combining these augmentations can expediently increase the possible training samples and improve the model generalization even on small training datasets. In addition to the augmentations discussed above, other augmentation methods used in image and speech processing could also be applicable to seismic data, such as: (1) filtering seismograms in different narrow frequency bands; (2) time or frequency stretching (Salamon & Bello, 2017); (3) masking part of signal by zeros (DeVries & Taylor, 2017b; Zhong et al., 2017); (4) vertically or possibly horizontally flipping signal; (5) rotating the horizontal components to account station orientation issues and create novel source-station paths; (6) scaling between three components using PCA augmentation (Krizhevsky et al., 2012); and (7) feature space augmentation (DeVries & Taylor, 2017a). Some augmentation such as time stretching or vertically flipping can result in potential side effects like changing the phase or polarization information. Hence, some caution is advised in selecting augmentation techniques. Beyond signal-processing-based augmentation, the

Generative Adversarial Network (GAN) approach can be used as a synthetic signal generator to make new training samples (Frid-Adar et al., 2018; Li, Meier, et al., 2018; Shin et al., 2018; Yi et al., 2019). AutoML-based methods, such as AutoAugment, can be used to automatically search for appropriate data augmentations for different problems and datasets (Cubuk et al., 2019). The effectiveness of these methods for seismic data remains as an important area for future research.

The augmentations discussed above were designed during training neural networks, while test time augmentation (He et al., 2016; Krizhevsky et al., 2012) can also help to improve the prediction performance after training. In image classification, several fixed crops and scales are applied to the test image. Similar to ensemble learning, the final prediction score is averaged over these augmentations to improve the overall prediction. The training time and test time augmentations serve different purposes. Training time augmentations, like superposing noise, aim to increase the variety and complexity of training samples, which renders the recognition task much more challenging and pushes the neural networks to learn more accurate decision boundaries between signal and noise (Figure 4.1). The test time augmentations aim to make the recognition task easier by sampling certain transforms that properly represent the data features and make the prediction more robust by aggregating different augmentations. For seismic data, data pre-processing methods, such as filtering, can be used as test time augmentation. Filtering can transform the signal into certain high SNR frequency bands covered by the training dataset, thus improving the prediction accuracy on noisy data. Table 4.3 shows the improved prediction performance by applying 1Hz high-pass filtering on the test dataset. Another potential method for test time augmentation is to compress the large epicentral distance waveform into a shorter time window in order to mitigate the learning bias due to the imbalanced distribution of epicentral distances (91% samples $<$ 40km) in the training dataset. We applied a 50% down-sampling to compress the waveforms of earthquakes with epicentral distance larger than 40km (Figure 4.2d) into half of the original time window. Table 4.4 shows the improved prediction performance after time compression. Note that we used prior information that these observations are from 40 km away before applying the compression, which may not be available in applications.

Table 4.3: Detection performance using a 1Hz high-pass filtering as test time augmentation.

Filtering as test time augmentation		Without compressing	With compressing
P-wave	Precision	0.902	0.902
	Recall rate	0.588	**0.626**
	F1-score	0.712	**0.739**
S-wave	Precision	**0.748**	0.725
	Recall rate	0.571	**0.650**
	F1-score	0.647	**0.685**

In addition to using data augmentation methods to improve neural network performance for

Table 4.4: Detection performance using time compressing as test time augmentation.

Time compressing as test time augmentation		Without compressing (> 40km)	With compressing (>40km)
P-wave	Precision	0.785	**0.889**
	Recall rate	0.374	**0.394**
	F1-score	0.507	**0.545**
S-wave	Precision	0.545	**0.708**
	Recall rate	0.342	**0.424**
	F1-score	0.421	**0.531**

small datasets, generalization of deep learning models is also determined by other factors such as neural network architectures, loss functions, optimization methods, and various training techniques, including batch normalization (Ioffe & Szegedy, 2015), layer normalization (Ba et al., 2016), dropout (Srivastava et al., 2014), early stopping (Prechelt, 1998), learning rate decay, weight decay (weight regularization), label smoothing (Müller et al., 2019), model ensemble (Deng & Platt, 2014; He et al., 2016). Most of these training techniques aim to stabilize the training and prevent overfitting. Note that overfitting and poor generalization do not necessarily go hand-in-hand. A model without overfitting can still suffer from poor generalization to unseen datasets with significantly different characteristics from the training dataset, while an overfitted model can always have generalization problems. With the development of deep learning, a variety of neural network architectures have been designed, modified and tested on seismic signals, and this trend will continue. The data augmentation techniques discussed here can generally apply to these deep learning models to improve their performance and generalization.

For seismic data, choosing the data processing domains, such as, the time domain, time-frequency domain, or wavelet domain, is also important for model performance depending on specific applications. Neural networks can flexibly process multi-dimensional data, like time sequence, image, and videos, which allows presenting seismic signals in either time or frequency domains when designing deep learning models for earthquake detection, denoising, and other problems. For convolutional neural networks, the convolutional kernels have a certain degree of similarity to sine/cosine or wavelet kernels used in Fourier or wavelet transform. The convolutional kernels after training show a variety of frequency- and orientation- features in image recognition (Krizhevsky et al., 2012). Although a neural network can learn frequency information directly from time domain waveforms, transforming seismic data into time-frequency domain explicitly presents the change of frequency distribution with time, making it easier to capture the frequency information during training. But training and testing in the time-frequency also comes with an expensive computational cost and a loss of high time resolution.

Another approach to addressing the issue of insufficient labeled training data is transfer learning and domain adaptation methods, which are developed to apapt the features and knowledge learned

from a large training dataset to improve the training and generalization on a new dataset or task of interest, which often has much less training data (Bengio, 2012; LeCun et al., 2015; Pan & Yang, 2009). For example, pre-trained models on ImageNet dataset (Deng et al., 2009; Oquab et al., 2014) has been used for a wide range of problems, such as object detection (Girshick et al., 2014; Ren et al., 2015), image segmentation (He et al., 2017; Long et al., 2015), medical image recognition (Tajbakhsh et al., 2016) and remote sensing (Marmanis et al., 2015). Commonly, transfer learning means transferring low-level features and representations trained on large datasets from the designed task to a different task on a new dataset, while domain adaptation refers to the case of the same task on two different datasets. Because of the similarity among earthquake signals, pre-trained deep neural networks on large datasets like STEAD (Mousavi, Sheng, et al., 2019) extract common low-level features for seismic waves, which could be used for applications without enough training data by transfer learning or domain adaptation. Unsupervised pretraining, such as auto-encoders (Vincent et al., 2008), can also be used to extract good data representation for transfer learning. In contrast to pre-training, self-training is another way to use the extensive unlabeled data (Zoph et al., 2020). Self-training first trains a model on the labeled dataset, then generates pseudo labels on a much large unlabeled dataset. The new pseudo-labeled data is combined with the labeled dataset to train a new model. This same process can be iterated a few times to advance model performance.

4.5 Conclusions

Data augmentation is an efficient way to improve the performance and generalization of deep neural networks for common cases where labeled data is scarce. We have presented and analyzed data augmentation methods that are well-suited for seismic data. Although we used only a small training dataset, our results show data augmentation can mitigate the bias in training data and improve the performance on a dataset with different statistics. Because data augmentation is independent of the particular neural network model and the computational cost is negligible compared with training, augmentation methods can widely benefit the training of deep learning models on seismic data.

Chapter 5

GaMMA: Earthquake Phase Association using a Bayesian Gaussian Mixture Model

Earthquake phase association algorithms, which aggregate detected phases from a network of seismometers into individual earthquakes that generate them, play an important role in earthquake monitoring. Dense seismic networks and improved phase picking methods can produce massive earthquake phase datasets, particularly for earthquake swarms and aftershocks, and many of these picks can be incorrect, making phase association a challenging problem. Here we present a new association method, the Gaussian Mixture Model Association (GaMMA), that combines the Gaussian mixture model for phase measurements, with earthquake location, origin time, and magnitude estimation. We treat earthquake phase association as an unsupervised clustering problem in a probabilistic framework, where each earthquake corresponds to a cluster of P and S phases with hyperbolic moveout of arrival times and a decay of amplitude with distance. We use a Gaussian distribution to model the collection of phase picks for an event, the mean of which is given by the predicted arrival time and amplitude. Through this approach we can calculate the probability of the picks based on the time and amplitude residuals. We carry out the pick assignment to each earthquake and determine the value of the underlying latent variables of earthquake location, origin time, and magnitude, under the maximum likelihood criterion using the Expectation-Maximization (EM) algorithm. The GaMMA method does not require the typical association steps of other algorithms such as grid-search or training. The probability framework can flexibly consider multiple phase types and arrival times, phase picking quality scores, velocity/acceleration amplitudes, and the residual covariance from association. The results show that GaMMA effectively associates phases from a temporally and spatially dense earthquake sequence while producing estimates of earthquake

location and magnitude.

5.1 Introduction

Earthquake catalogs are the fundamental representation of seismicity. Extensive research has been devoted to generating comprehensive catalogs with up to orders of magnitude more small earthquakes than are present in standard earthquake catalogs. These newly cataloged earthquakes have the potential to reveal relationships among earthquakes and illuminate active structures that would otherwise remain hidden (Hauksson et al., 2012; Park et al., 2020; Ross, Trugman, Hauksson, et al., 2019; Tan et al., 2021; Waldhauser & Schaff, 2008; Yoon et al., 2015). A standard earthquake monitoring workflow from seismic waveforms to earthquake catalogs includes several tasks, including phase picking (Allen, 1978), phase association (Yeck et al., 2019), earthquake location (Klein, 2002), and magnitude estimation (Richter, 1935). The phase picking step detects seismic phases such as P-wave and S-wave phases at each seismic station. The phase association step aggregates and groups these phases from multiple stations of a seismic network into individual earthquakes. Earthquake location and magnitude are then estimated from the associated phase information, i.e., arrival time and amplitude. The resulting catalog can be further enhanced through template matching (Gibbons & Ringdal, 2006; Peng & Zhao, 2009; Shelly et al., 2007) or subspace projection (Barrett & Beroza, 2014; Harris & Dodge, 2011) that use the detected earthquakes as templates to re-scan the waveforms and detect small earthquakes with similar waveforms.

The phase picking step has been significantly improved by deep-learning-based pickers that learn from manual picks labeled by analysts to detect millions of phase picks from raw seismic waveforms (Mousavi et al., 2020; Ross, Meier, Hauksson, & Heaton, 2018; Zhu & Beroza, 2019). The rapidly growing volume of automatic picks and the ongoing growth of seismic networks makes developing effective phase association methods critical. Phase association has not yet received the attention that has been devoted to other earthquake monitoring tasks. Association methods based on back-projection are most commonly used in earthquake monitoring systems, such as GLASS3 (Yeck et al., 2019), Earthworm (Friberg et al., 2010), and SeisComP3 (Weber et al., 2007). These approaches usually deploy a grid-search and back-project phase picks based on the expected moveout with distance. An earthquake and its location is declared based on the number of phase picks inside a spatial grid that are consistent with a candidate location. Although back-projection-based association is robust and effective, its performance is limited for dense earthquake sequences when earthquakes occur so closely in time and space that interpreting the maxima resulting from back-projection becomes problematic. Studies have continued to focus on improving the grid-search and back-projection approach for different scenarios (Draelos et al., 2015; Gibbons et al., 2016; Zhang et al., 2019). Meanwhile, several new approaches have been proposed to solve the earthquake phase association using: graph theory (McBrearty, Gomberg, et al., 2019), the RANSAC algorithm

(Woollam et al., 2020), and deep learning (McBrearty, Delorey, et al., 2019; Ross, Yue, et al., 2019). When combined with the rapid development of phase picking methods, better association methods have the potential to improve significantly the overall performance of earthquake monitoring pipelines.

We propose an association method based on a Bayesian Gaussian mixture model, which is an unsupervised machine learning method for clustering (Bishop, 2006) that has been widely used in different research fields, such as image processing (Permuter et al., 2006), speech recognition (Reynolds & Rose, 1995), and earthquake studies (Ross, Trugman, Azizzadenesheli, et al., 2020; Seydoux et al., 2020). Earthquake phase association can be treated as an unsupervised clustering problem, with groups of phase picks arising from a discrete set of earthquakes. We combine the Gaussian mixture model with earthquake location, origin time, and magnitude estimation, so that the GaMMA method can cluster phase picks based on the physical constraints of arrival time moveout and amplitude decay with distance. Phase amplitude information is usually neglected because it can be difficult to account for in conventional association methods; however, GaMMA is designed such that it can use phase arrival time, phase-type identification, and amplitude information for association while simultaneously estimating the underlying event location and magnitude. Moreover, GaMMA does not require extra association steps of grid-search, back-projection, or training. These features make GaMMA a promising approach to address the challenge posed by large numbers of automatic picks in earthquake monitoring workflows.

5.2 Method

The objective of earthquake phase association is to group seismic phases produced by an earthquake from a large collection of phases detected on a seismic network, so that subsequent earthquake characterization tasks can be performed on each group of seismic phases. Phase arrival times from the same earthquake follow a hyperbolic moveout that is determined by the hypocentral distance and the wavespeed structure of the Earth. This moveout allows association algorithms to separate phases from different earthquakes. In this work, we extend the association problem by using both phase arrival time and phase amplitude information. The phase amplitude scales with earthquake magnitude and decays with the hypocentral distance. Thus, the amplitude provides additional information to improve phase association. We formulate the association problem as follows: Given N seismic phases $(x_i, y_i, z_i, t_i, a_i)$, i.e., arrival time t_i and amplitude recorded a_i at the i-th seismic station located at (x_i, y_i, z_i), we seek to group these phases into K earthquakes and estimate the underlying source parameters $(x_k, y_k, z_k, t_k, m_k)$, i.e., location (x_k, y_k, z_k), origin time t_k, and magnitude m_k of the k-th earthquake. We solve this association problem using the Gaussian mixture model (Permuter et al., 2006), which is an unsupervised clustering method that groups N data points into K clusters by maximizing the probability that these N data can be explained by a mixture of K

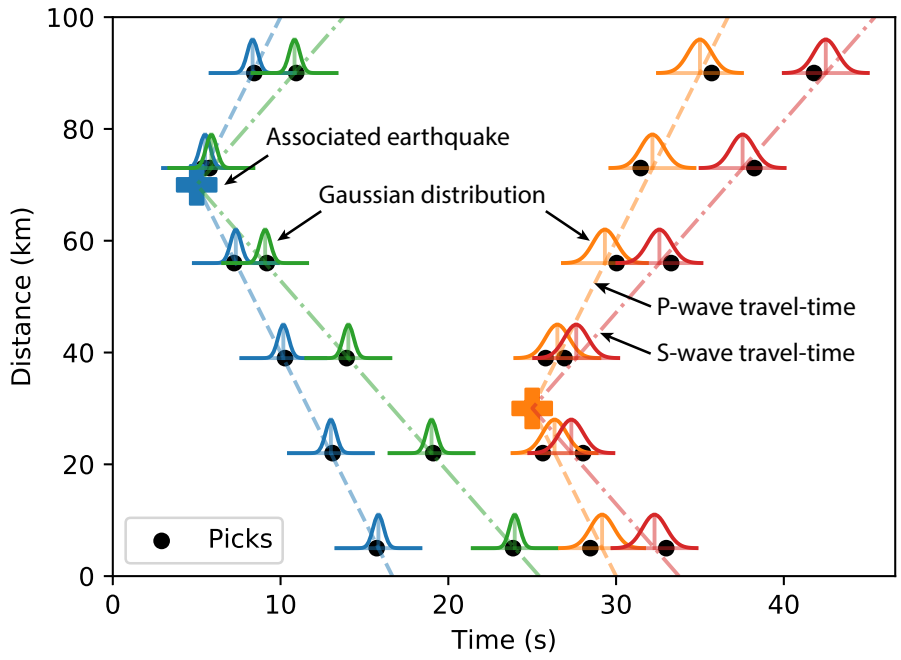


Figure 5.1: Gaussian Mixture Model for Association. The GaMMA method models Gaussian distributions based on the theoretical phase travel-time and amplitude and uses the Expectation-Maximization (EM) algorithm to update iteratively: phase assignment, earthquake source parameters, and the mean and standard deviation of Gaussian distribution. This iteration converges to the correct phase assignment and earthquake source parameters to solve the association problem. Note that we use both phase arrival time and amplitude information to model the Gaussian distributions.

Gaussian distributions. We incorporate the physical constraints on phase arrival time and amplitude into a Gaussian mixture model to make it suitable for our association problem. Figure 5.1 illustrates how we model the Gaussian distributions to calculate the probability of the sequence of phases generated by the two causative earthquakes. The mathematical details of GaMMA are explained in the following sections.

5.2.1 Bayesian Gaussian Mixture model

We cast the association problem in a probabilistic framework where we use a Gaussian mixture distribution to model the probability of each phase pick:

$$p(\mathbf{x}_i) = w_i \sum_{k=1}^{K} \phi_k \mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Lambda}_k^{-1}) \tag{5.1}$$

$$\mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Lambda}_k^{-1}) = \frac{1}{(2\pi)^{n/2}} |\mathbf{\Lambda}_k|^{1/2} \exp\left(-\frac{1}{2}(\mathbf{x_i} - \mu_k)^T \mathbf{\Lambda}_k (\mathbf{x_i} - \mu_k)\right) \tag{5.2}$$

$$\sum_{k=1}^{K} \phi_k = 1 \tag{5.3}$$

where $\mathbf{x}_i$ represents a phase pick including arrival time, phase type, and amplitude (t_i, a_i) values at the i-the station. ϕ_k is the mixture component coefficient of the k-th earthquake. $\mathcal{N}$ represents a Gaussian distribution, and μ_k is the mean of Gaussian distribution. μ_k represents the theoretical phase arrival time and amplitude $(\hat{t}_{ik}, \hat{a}_{ik})$ determined by the k-th earthquake. $\mathbf{\Lambda}_k$ is the precision (inverse covariance) matrix of the Gaussian distribution. w_i is the phase picking quality score between [0, 1]. n is the number of feature dimensions, which is 1 if only time information is used or 2 if both time and amplitude information are used. Based on the Gaussian mixture distribution, we can calculate the probability of a set of recorded phases $(x_1, x_2, ..., x_N)$. We assume these observations $(\mathbf{X})$ are independent and identically distributed (i.i.d.), then the log likelihood function is given by:

$$\log\left(p(\mathbf{X}|\phi, \mu, \mathbf{\Lambda})\right) = \sum_{i=1}^{N} \log\left(w_i \sum_{k=1}^{K} \phi_k \mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Lambda}_k^{-1})\right) \tag{5.4}$$

We can find the assignment from N phase picks to K earthquakes, which is the goal of association, and the corresponding earthquake source parameters by maximizing the log likelihood.

A limitation of the Gaussian mixture model formulation is that we need to assume the number of underlying earthquakes K. To address this unknown, we implement the Bayesian Gaussian mixture model (Bishop, 2006), which uses variational inference to calculate approximate posterior distributions for the parameters of a Gaussian mixture distribution. Three conjugate priors are introduced. In particular, we use a Dirichlet prior for the mixture component coefficient, $p(\phi) = \mathcal{D}(\alpha_0)$ (Ferguson, 1973), which controls the concentration of mixture components; a Gaussian prior for the mean conditioned on the precision, $p(\mu_k|\Lambda_k) = \mathcal{N}(m_0, \beta_0\Lambda_k)$; and a Wishart prior for the precision, $p(\Lambda_k) = \mathbf{W}(\mathbf{W}_0, \nu_0)$ (Wishart, 1928), which controls the estimation of covariance. The Bayesian model penalizes parameters that are away from the priors, which balances data fitting and model complexity. The mixture components (i.e., earthquakes) that do not contribute to the explaining the data (i.e., picks) will have approximately zero mixture coefficients so that we can choose a large number of components in the mixture model without over-fitting.

5.2.2 Expectation-Maximization (EM) algorithm

We use the the Expectation-Maximization (EM) algorithm to solve the maximum likelihood estimation of $p(\mathbf{x})$. To consider the physical constraints on phase arrival time and amplitude for association, we incorporate the estimate of earthquake location, origin time, and magnitude into the EM algorithm. We then iteratively update the assignments from picks to earthquakes in the E-step and optimize the earthquake parameters in the M-step:

E-step:

$$\gamma_{ik} = \frac{\phi_k \mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Sigma}_k)}{\sum_{k=1}^{K} \phi_k \mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Sigma}_k)} \tag{5.5}$$

where γ_{ik} is the probability that phase pick $\mathbf{x}_i$, i.e., arrival time t_i and wave amplitude a_i, is generated by the k-th earthquake.

M-step:

1. Effective number of picks assigned to the k-th earthquake:

$$N_k = \sum_{i=1}^{N} \gamma_{ik} \tag{5.6}$$

$$\phi_k = \frac{N_k}{N} \tag{5.7}$$

2. Earthquake location, origin time, and magnitude of the k-th earthquake:

$$\underset{(x_k, y_k, z_k, t_k)}{\text{minimize}} \; l(x_k, y_k, z_k, t_k) = \sum_{i=1}^{N} \gamma_{ik} \mathcal{L}\left(t_i, \hat{t}_{ik}(x_k, y_k, z_k, t_k)\right) \tag{5.8}$$

$$m_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} \mathcal{F}'_a(a_i, d_{ik}) \tag{5.9}$$

3. Theoretical travel time, amplitude, and statistics of residuals:

$$\mu_k = \begin{bmatrix} \hat{t}_{ik} \\ \hat{a}_{ik} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_t(x_k, y_k, z_k, t_k) \\ \mathcal{F}_a(m_k, d_{ik}) \end{bmatrix} \tag{5.10}$$

$$\mathbf{\Lambda}_k^{-1} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T \tag{5.11}$$

where $\mathcal{L}$ is a loss function of the residuals between the picked phase arrival time t_i and the theoretical arrival time $\hat{t}_{ik}$ from the k-th earthquake. Minimization of the loss function l gives an estimate of the earthquake location and origin time (x_k, y_k, z_k, t_k). m_k is the magnitude of the k-th earthquake.

$\mathcal{F}_t$ represents the function used to calculate theoretical phase arrival time $\hat{t}_{ik}$. $\mathcal{F}_a$ represents the function to calculate theoretical phase amplitude $\hat{a}_{ik}$ based on earthquake magnitude m_k, and $\mathcal{F}'_a$ represents the function to estimate earthquake magnitude using phase amplitude. d_{ik} is the distance from the k-th earthquake to i-th seismic station. Here we decouple the optimization of earthquake magnitude (Equation (5.9)) from the optimization of earthquake location and time (Equation (5.8)). We use arrival times to constrain earthquake location and use phase amplitudes to constrain earthquake magnitude. Note that although we decouple the two optimizations, the precision matrix (Equation (5.11)) considers the correlation between arrival time and amplitude residuals. Both arrival time and amplitude information are used for the association process of clustering picks among earthquakes (Equation (5.5)).

For the Bayesian Gaussian mixture model, we add another stage in the M-step to update the posterior parameters:

$$\alpha_k = \alpha_0 + N_k \tag{5.12}$$

$$\beta_k = \beta_0 + N_k \tag{5.13}$$

$$\mathbf{m}_k = \frac{1}{\beta_k}\left(\beta_0 \mathbf{m}_0 + N_k \mu_k\right) \tag{5.14}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{\Lambda}_k^{-1} + \frac{\beta_0 N_k}{\beta_0 + N_k}\left(\mu_k - \mathbf{m}_0\right)\left(\mu_k - \mathbf{m}_0\right)^{\mathrm{T}} \tag{5.15}$$

$$\nu_k = \nu_0 + N_k \tag{5.16}$$

The E-step is modified as:

$$\gamma_{ik} \propto \widetilde{\pi}_k \widetilde{\Lambda}_k^{1/2} \exp\left\{-\frac{D}{2\beta_k} - \frac{\nu_k}{2}\left(\mathbf{x}_n - \mathbf{m}_k\right)^{\mathrm{T}} \mathbf{W}_k \left(\mathbf{x}_n - \mathbf{m}_k\right)\right\} \tag{5.17}$$

$$\ln \widetilde{\Lambda}_k = \sum_{i=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + D \ln 2 + \ln\left|\mathbf{W}_k\right| \tag{5.18}$$

$$\ln \widetilde{\pi}_k = \psi\left(\alpha_k\right) - \psi(\widehat{\alpha}) \tag{5.19}$$

where $\widehat{\alpha} = \sum_k \alpha_k$ and ψ is the digamma function (Abramowitz & Stegun, 1964). The explanation of these updating rules is detailed in Bishop (2006)'s textbook.

5.2.3 Earthquake location and magnitude estimation

The iteration of the EM algorithm both updates the clustering of picks based on earthquakes and optimizes the corresponding earthquake source parameters. We focus on efficient association rather than accuracy of earthquake source parameters, which can be realized once phases are properly associated, so we choose two basic approaches to estimate approximate earthquake locations and magnitudes. We optimize Equation (5.8) with a Huber loss function (Huber, 1992) as the target to

reduce the effect of outliers:

$$\mathcal{L}_\delta(t-\hat{t}) = \begin{cases} \frac{1}{2}(t-\hat{t})^2 & \text{for } |t-\hat{t}| \le \delta \\ \delta\left(|t-\hat{t}| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases} \tag{5.20}$$

where the hyper-parameter δ is set to 1 second. For this proof-of-concept study, we use a uniform velocity model to calculate the theoretical phase travel-time:

$$\widehat{t}_{ik}(x_k, y_k, z_k, t_k) = \mathcal{F}_t(x_k, y_k, z_k, t_k) = \frac{d_{ik}}{v} + t_k \tag{5.21}$$

We then solve the minimization of Equation (5.8) using the BFGS algorithm (Fletcher, 2013). Advanced earthquake location algorithms and complex velocity models can also be applied to solving Equation (5.8) but at a higher computational cost.

To estimate earthquake magnitude in Equation (5.9), we use a linear relationship between phase amplitude and earthquake magnitude:

$$\widehat{m}_{ik} = \mathcal{F}'_a(a_i, d_{ik}) = c_0 + c_1 \log a_i + c_2 \log d_{ik} \tag{5.22}$$

Based on the measured phase amplitude type, e.g., displacement, peak ground velocity, or peak ground acceleration, we can choose from among the Richter empirical magnitude relationship (Richter, 1935), the Richter simulation-based prediction (Al-Ismail et al., 2020), or a simplified ground motion prediction equations (Picozzi et al., 2018) for Equation (5.22).

5.3 Results

We demonstrate the performance of GaMMA first on a synthetic example and then on data from the 2019 Ridgecrest, California earthquake sequence.

5.3.1 Synthetic test

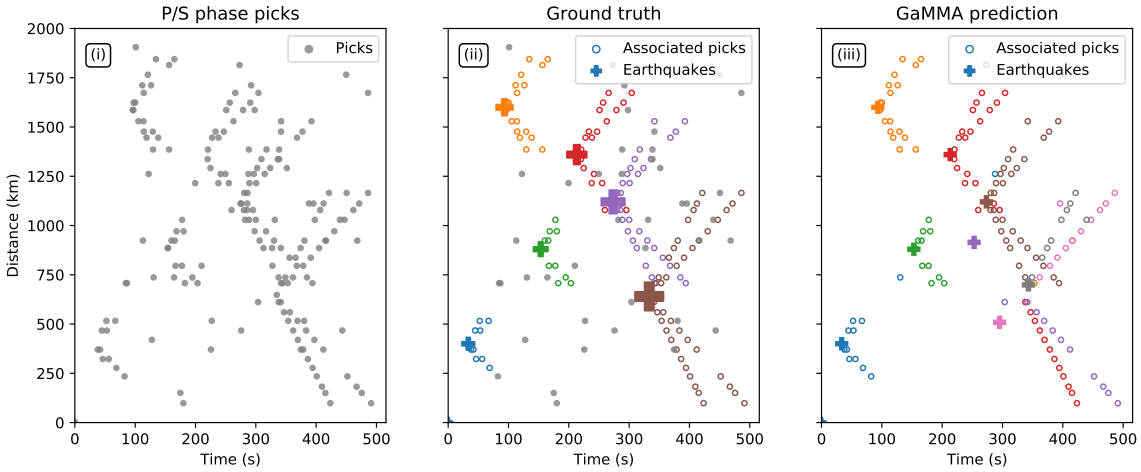
We first created a synthetic experiment to demonstrate the association results of GaMMA. We generated a sequence of phases including both P- and S- phases from six earthquake events. To model the errors that exist in real data, we added a 0.5s random error in the phase arrival times and scaled the phase amplitude (peak ground velocity (PGV)) by a random factor between 0.3-3. We further added 30% false positive picks at random times. In total, 178 P- and S-phase picks from 40 stations were used for association (left panels of Figure 5.2). We used a simplified ground motion prediction equation of PGV from Picozzi et al. (2018)'s work: $\log PGV = -2.175 - 1.68 \log R + 0.93M$, where R is hypocentral distance and M is earthquake magnitude. The ground truth result is shown in the middle panels of Figure 5.2. The symbol size represents the relative size of phase amplitude

and earthquake magnitude. We conducted two association tests using in the first test only the arrival time information (Figure 5.2(a)) and in the second test both the arrival time and amplitude information (Figure 5.2(b)). The same parameters and initialization were used for both cases. For each we initialized the earthquake locations at the center of research area and uniformly distributed the earthquake origin times. The association results are shown in the right panels of Figure 5.2. Five of the six true earthquakes are successfully associated in both tests. However, the sixth event in the lower right corner was not associated using only the arrival time information (Figure 5.2(a)(iii)). This results in multiple incorrect event associations marked by colors pink and purple. This occurs because the P-phases of this event have arrival times that overlap with another distant event's moveout (the event marked in red) and could be mistakenly associated with this distant event. This mis-association is resolved and the sixth event is recovered when we use both the arrival time and amplitude information (Figure 5.2(b)(iii)). The extra information provided by amplitude adds the necessary extra constraint on distance, in addition to time, that allows the sixth event to be associated correctly. The test with amplitude information also correctly estimates the earthquake magnitudes. This synthetic experiment demonstrates that GaMMA benefits from using both the time and amplitude information in the association process.

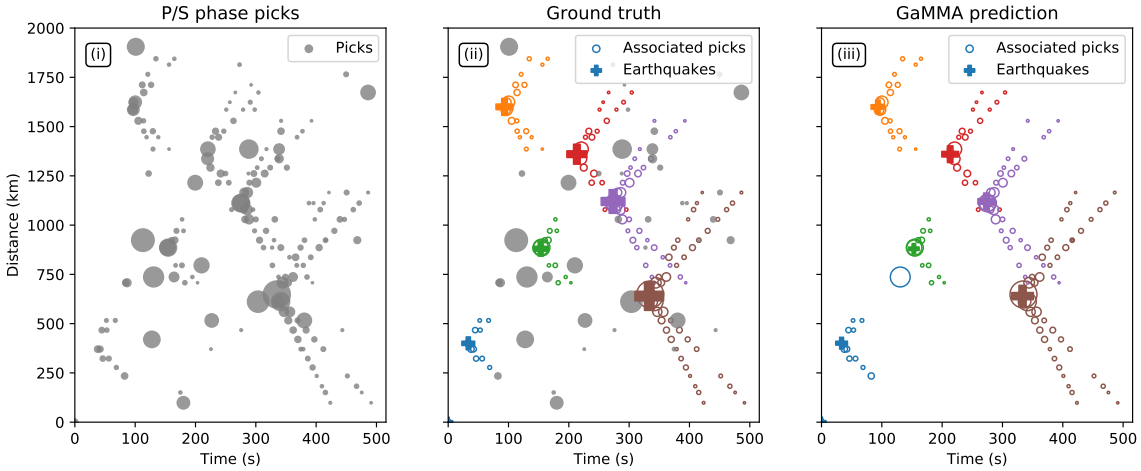
5.3.2 Test on the 2019 Ridgecrest earthquake sequence

We next applied the GaMMA method to part of the 2019 Ridgecrest earthquake sequence to evaluate its performance on a real earthquake sequence. We focused on the initial six days of the sequence when a large number of earthquakes occurred and migrated from the southwest-striking fault to the northwest-striking fault. We applied the the PhaseNet model (Zhu & Beroza, 2019) to extract picks from waveforms of "HH," "BH," "EH," and "HN" channels of 23 stations within 1 degree of the location (-117.504W, 35.705N). We measured the PGV value over a 8s window after the phase arrival time. We then associated the detected 651,994 P-picks and 686,291 S-picks using the GaMMA method. We used a uniform velocity model with v_p = 6 km/s and $v_s = v_p/1.75$ for earthquake location estimation and a simple ground motion prediction equation as above for earthquake magnitude estimation. Because we used such a simple moveout behavior, we do not expect the earthquake locations to be accurate, but they are close enough for successful association, given the relatively short source-receiver distances involved.

Figure 5.3 shows the statistics of the 34,791 associated earthquakes from 598,218 P-picks and 633,010 S-picks, leaving 53,776 P-picks and 53,281 S-picks unassociated. We also plotted the 9,873 earthquakes in the SCSN catalog (SCEDC, 2013) for comparison. Both the associated earthquake locations and magnitudes agree with the SCSN catalog (Figure 5.3(b) and (c)). Figure 5.4 shows an association example with a dense sequence of picks occurring during a 5-minute period. The GaMMA method successfully associated 23 earthquake events from these picks. Figure 5.5 shows the residual distributions of the associated earthquake location and magnitude compared with the SCSN catalog.



(a)



(b)

Figure 5.2: Synthetic example: (a) association using only time; (b) association using both time and amplitude. The left panels plot the P- and S-phase picks. The middle panels are plots of the ground truth of association result. The unassociated false positives are plotted in grey. The circle size represents phase amplitude and the cross size represents earthquake magnitude. The right panels show the association results of the GaMMA method. Note that some phases in the lower right corner of panel (a)(iii) are mis-associated with another distant earthquake marked in red, because these phases can fit the moveout of both events. Amplitude information provides an extra constraint in distance that resolves this ambiguity as shown in panel (b)(iii).

Table 5.1: Statistics of residual distributions in Figure 5.5

Error	Δx (km)	Δy (km)	Δz (km)	Δt (s)	Δm
Mean	-0.85	0.45	10.75	-0.35	-0.064
STD	2.61	2.11	3.80	0.80	0.235
MAE	2.13	1.53	10.75	0.62	0.154

Table 5.2: Comparison with other catalogs

Catalog		SCSN (9,873)	Ross et al. (2019) (29,384)	Liu et al. (2020) (15,421)	Shelly (2020) (16,778)
GaMMA (34,791)	Recall	0.973	0.737	0.987	0.955
	Precision	0.336	0.765	0.576	0.552
	F1-score	0.499	0.751	0.727	0.699

The statistics of mean, standard deviation (STD), and median absolute error (MAE) can be found in Table 5.1. The covariance matrix in Figure 5.5(d) shows that most of the associated earthquakes have small residuals of phase arrival time and amplitude, indicating that the phase picks match well with the theoretical values determined by the causative earthquakes found by association. These location and magnitude estimates can be further improved through the application of established earthquake location and magnitude algorithms once the picks have been associated.

We compared the catalog generated by GaMMA with three state-of-art catalogs (Liu et al., 2020; Ross, Idini, et al., 2019; Shelly, 2020). Table 5.2 shows the earthquake numbers in these catalogs during the same period. In each of these cases we assumed these catalogs as ground truth and analyzed whether the earthquakes they contain are also detected in GaMMA's catalog within a 5s window. Based on the recall rate, more than 95% of earthquakes in the catalogs of SCSN, Liu et al. (2020), and Shelly (2020) are successfully associated by GaMMA. The low precision and F1-score are due to the large number of new earthquakes associated by GaMMA. To verity whether these new earthquakes are reasonable, we compared the magnitude distributions of the four catalogs (Figure 5.6). Most of these new earthquakes associated by GaMMA have a small magnitude and follow the Gutenberg–Richter magnitude-frequency relationship (Gutenberg, 1956), which suggests that they may be legitimate detections of real earthquakes.

5.4 Discussion

The Gaussian mixture model (GMM) is an effective and widely used unsupervised learning method for clustering. We have combined the GMM with earthquake location and magnitude estimation to develop a novel association method (GaMMA). We treat earthquake phase association as a clustering problem that aims to cluster phases based on the causative earthquakes. There are several advantageous features of this unsupervised learning approach. First, GaMMA does not require grid-search or training commonly used in other association methods. Second, GaMMA can

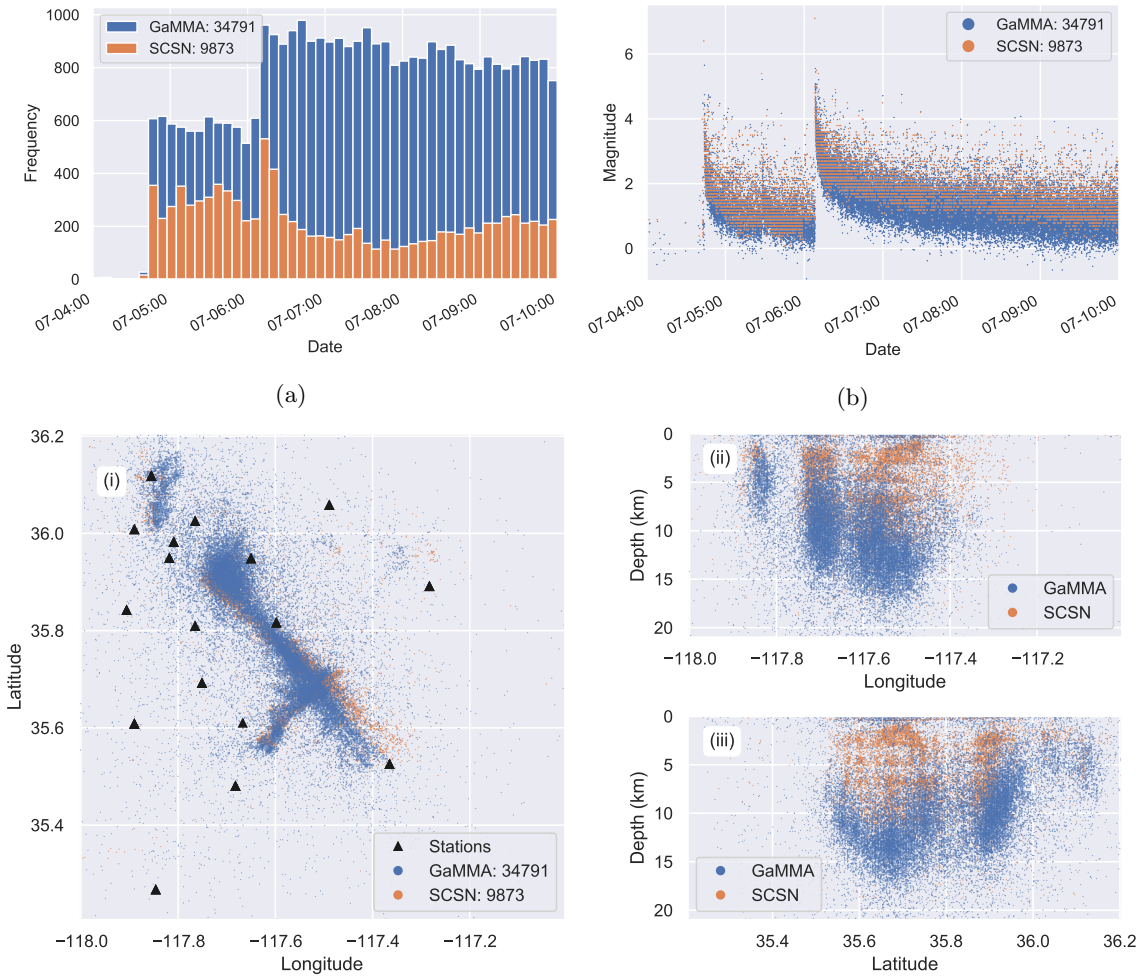


Figure 5.3: Association results of the Ridgecrest dataset: (a) associated earthquake frequency, (b) associated earthquake magnitude, (c) associated earthquake location. Note that because we use a uniform velocity model during association, we do not expect the earthquake locations to be accurate, but they are close enough for effective association.

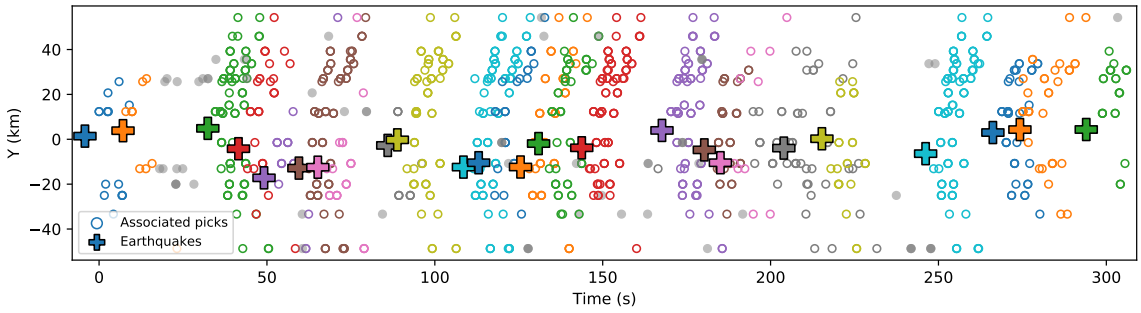


Figure 5.4: An example of association results from a dense sequence of phase picks starting at time 2019-07-04T19:10:52 (UTC). GaMMA associates 23 events during this period, while only 13 events are present in the SCSN catalog.

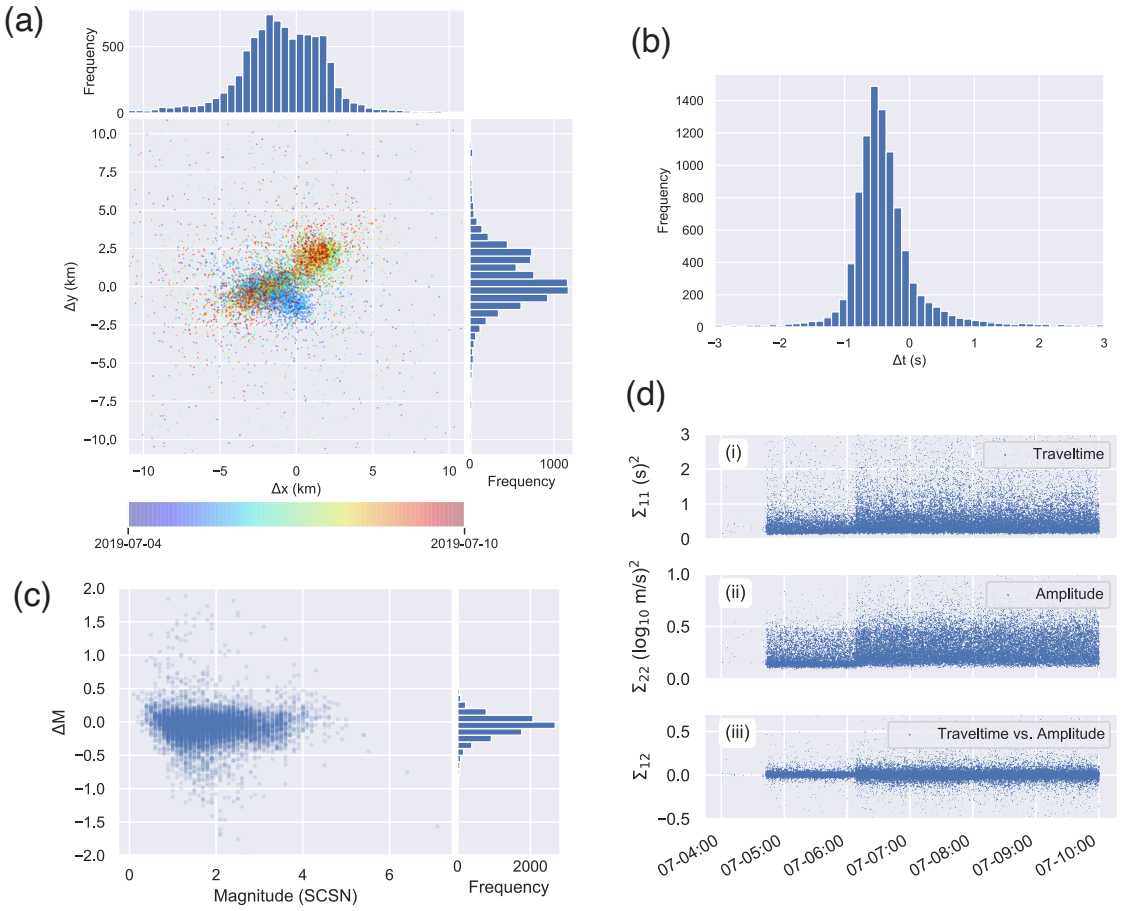


Figure 5.5: Residuals of (a) associated earthquake location, (b) origin time, and (c) magnitude compared with the SCSN catalog. (d) The components of the covariance matrix of travel-time and amplitude residuals estimated by GaMMA.

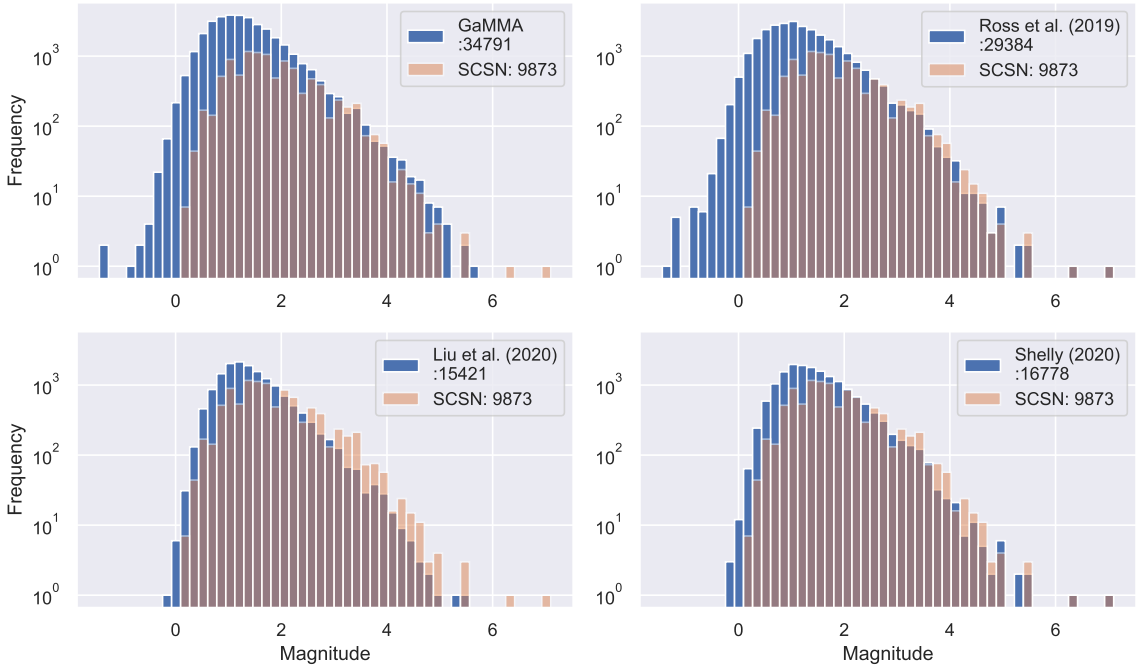


Figure 5.6: Comparison of magnitude distribution.

flexibly consider phase arrival time, amplitude, P/S type, and pick quality. While it is difficult for conventional association methods to consider phase amplitude information, GaMMA can easily include phase amplitude information both to improve association and to estimate earthquake magnitude. Finally, GaMMA optimizes the association result in a probabilistic framework and estimates the covariance of time and amplitude residuals. These advantages suggest that GaMMA a promising approach for improved earthquake phase association, which is in turn an important component of improved earthquake monitoring.

We note that there are several limitations of GaMMA that need to be considered. First, the time complexity of the Gaussian mixture model scales with $O(K \cdot N)$, where K is the number of clusters (earthquakes) and N is the number of samples (phase picks), so the computational cost could become prohibitive for a long earthquake sequence. In practice, however, it is straightforward and effective to segment a long sequence into short windows to improve the association speed by processing data in parallel because phases that are separated by a certain time interval (depending on the source-receiver distance) cannot come from a single earthquake. In this work, we used the DBSCAN algorithm (Schubert et al., 2017) to divide picks into sub-windows for association. The example in Figure 5.4 shows one sub-window from the seven hour sequence. Second, the clustering results of the Gaussian mixture model are affected by the initialization state. In this work, we used a simple strategy to initialize the earthquake locations uniformly in time and space. The

number of initialized earthquakes is proportional to the ratio between the number of phase picks divided and the number of stations. This simple strategy worked well in the experiments described above. Improving initialization strategies has the potential to increase the association performance further. Third, GaMMA ensures that one pick is only assigned to one earthquake, while conventional back-projection methods may attribute one pick to several earthquakes; however, GaMMA does not consider station-based constraints, such as that only one pair of P- or S-picks from one station is assigned to each earthquake. McBrearty, Gomberg, et al. (2019) accounts for this constraint using a constrained ILP (integer linear programming) solution in his association method, and there is a strong correspondence between that technique and our technique. One potential solution that would allow introduction of the station-based constraint is to add a normalizing scheme similar to Equation (5.5) over stations. Finally, GaMMA assumes that the residuals of pick time or amplitude follow a Gaussian distribution. This assumption is not accurate for false positive picks. Adding a mixture component of background uniform distribution to account false positive picks might be a helpful extension to be considered in future research (Melchior & Goulding, 2018).

5.5 Conclusions

We have developed a new association method based on a Bayesian Gaussian mixture model, GaMMA, which solves the phase association problem as an unsupervised clustering problem. To consider the physical constraints of phase arrival time and amplitude with earthquake location and magnitude, we incorporate optimization of earthquake location and magnitude into the Expectation-Maximization (EM) algorithm commonly used for solving the Gaussian mixture model. GaMMA, thus, can use both arrival time and amplitude information to cluster picks from the same earthquake and simultaneously estimates both the earthquake location and magnitude from each cluster of picks. The experiment results on both synthetic tests and the 2019 Ridgecrest earthquake sequence demonstrate the effectiveness of the GaMMA method in associating a dense sequence of P- and S-phase picks. GaMMA provides an unsupervised learning approach to solve the challenging phase association problem resulting from the increasingly wide applications of deep-learning-based phase pickers. The improved performance of GaMMA can associate more earthquakes from massive automatic phase picks, thus enriching earthquake catalogs and improving earthquake monitoring.

Chapter 6

QuakeFlow: A Scalable Machine-learning-based Earthquake Monitoring Workflow with Cloud Computing

Earthquake monitoring workflows are developed to detect earthquake signals and determine earthquake characteristics from continuous seismic waveforms as recorded by a seismic network. An effective earthquake monitoring workflow is important for providing hazard information on large earthquakes, but has a particularly significant impact on studying earthquake activity as expressed by the massive number of small earthquakes. Recent developments in deep learning methods have been used to significantly improve several tasks in earthquake monitoring workflows, such as earthquake detection, phase picking, and phase association. These improvements allow for the detection of up to orders of magnitude more small earthquakes than are present in standard earthquake catalogs, and to do so with high accuracy and speed. This improved performance is particularly useful for mining archived seismic datasets at scale to detect earthquakes missed in routine catalogs and to reveal previously hidden aspects of earthquake occurrence. In this work, we have developed a machine-learning-based earthquake monitoring workflow, QuakeFlow, that automatically processes seismic data and detects earthquakes with cloud computing. We have added a deep learning model, PhaseNet, for picking P/S phases and a machine learning model, GaMMA, for phase association with approximate earthquake location and magnitude. Each component in QuakeFlow is containerized, allowing straightforward updates to the pipeline with new deep learning/machine learning models, as well as the ability to add new components, such as earthquake relocation algorithms. We built QuakeFlow in Kubernetes

both to make it scalable to large datasets using auto-scaling and to make it easy to deploy on most cloud platforms. Cloud computing enables large-scale parallel processing of deep learning models for seismic data mining tasks. We have applied QuakeFlow to processing three years' worth of archived datasets from Puerto Rico within a few hours. In addition to batch processing, we added Kafka and Spark Streaming services for stream processing to deliver real-time earthquake monitoring results. We also applied Quakeflow to monitoring frequent volcanic earthquakes on the Big Island of Hawaii. Our results show that QuakeFlow can be applied to both improving routine seismic network monitoring and to mining massive archived seismic data for studying complex earthquake sequences.

6.1 Introduction

Seismic networks are deployed around the world to monitor earthquake activities and provide rich data resources for earthquake research. Seismic waveforms are continuously collected and processed by earthquake monitoring workflows to detect, locate, and characterize earthquake events. The resulting earthquake catalogs reveal spatial-temporal evolution of earthquake activities and illuminate fault-zone structure. A comprehensive earthquake catalog thus provides crucial information that allows more effective study of complex earthquake sequences and the ability to quantify earthquake hazard. However, the increasing number of dense seismic networks and the massive amount of archived seismic waveform data pose a challenge for data mining of seismic datasets to detect more earthquakes, particularly in regions that may not be well-studied. The rapid progress of deep learning algorithms and cloud computing techniques provide a promising pathway to address the big data challenge in earthquake monitoring and to generate comprehensive earthquake catalogs with high accuracy and speed.

Deep learning algorithms have drastically improved earthquake monitoring performance particularly in earthquake detection and phase picking (Mousavi et al., 2020; Perol et al., 2018; Ross, Meier, Hauksson, & Heaton, 2018; Zhu & Beroza, 2019). Earthquake monitoring workflows based on deep-learning-based phase pickers have been applied to studying, among others, dense earthquake sequences (Liu et al., 2020; Ross, Cochran, et al., 2020; Tan et al., 2021), induced seismicity (Park et al., 2020; Wang et al., 2020; Zhou et al., 2021), and hydraulic fracturing monitoring (Chai et al., 2020). These studies have demonstrated that deep learning algorithms can detect up to orders of magnitude more small earthquakes than conventional algorithms, providing a more complete accounting of seismicity and new insights into earthquake behavior. These applications moreover show that deep-learning-based earthquake monitoring workflows are now feasible. The adoption of deep learning algorithms into routine earthquake monitoring systems are thus under active development (Saurel et al., 2021; Walter et al., 2021; Yeck et al., 2021).

In addition to deep learning's ability to address challenges posed by effective earthquake detection, cloud computing has the potential to address the big data challenge posed by processing large

seismic datasets. Massive amounts of seismic data have been accumulated in the past few decades. For example, by 2021 the archived data volume at the Incorporated Research Institutions for Seismology (IRIS) has reached about 700 TiB (http://ds.iris.edu/data/distribution/). Much of this data has not been well studied due to the lack of effective earthquake detection workflows; however, without parallel computing, re-processing these archived datasets with deep learning algorithms would require a prohibitive amount of time and effort. Cloud computing can help significantly decrease the computational time needed by using thousands of computational nodes provided by cloud computing platforms, such as AWS (Amazon Web Services), GCP (Google Cloud Platform), and Microsoft Azure. The rapidly developing cloud-native software, such as Kubernetes (Bernstein, 2014), Kubeflow, Kafka (Kreps et al., 2011), and Spark (Zaharia et al., 2010), as well as cloud-based datasets, such as the SCEDC earthquake data AWS public dataset (Yu et al., 2019), make it efficient to build and run earthquake monitoring workflow on the cloud.

To explore this significant potential, in this study we combined both techniques: machine learning algorithms for earthquake detection and cloud computing for parallel processing, to build an earthquake monitoring workflow, QuakeFlow, which can be applied to either mining massive archived datasets or processing real-time waveforms. We added the deep neural network model, PhaseNet (Zhu & Beroza, 2019), to pick P- and S-phases and the Gaussian mixture model, GaMMA (Zhu, McBrearty, et al., 2021), to associate picks and estimate approximate earthquake locations and magnitudes. We also added Kafka and Spark Streaming services to support real-time earthquake monitoring. We deployed the QuakeFlow system in Kubernetes, making it platform-independent and scalable to thousands of cloud computing nodes for large-scale seismic data mining. QuakeFlow is a proof-of-concept workflow to combine the state-of-the-art machine learning models and cloud computing techniques to advance earthquake monitoring.

6.2 QuakeFlow System

An earthquake monitoring workflow consists of a sequence of tasks including phase detection/picking, association, location, and characterization, to detect earthquake signals and constrain source parameters. In this proof-of-concept study, we focused on two of these tasks: phase picking and phase association, which have been significantly improved by new machine learning and deep learning models. These models, however, have not been used by traditional earthquake monitoring systems at earthquake data centers. We built a QuakeFlow system to explore the potential of machine learning algorithms and cloud computing infrastructure for earthquake monitoring. Figure 6.1 shows an overview of QuakeFlow system. Two main modules used by QuakeFlow are a deep learning model, PhaseNet, for picking P- and S-phases and a Gaussian mixture model, GaMMA, for associating phases and estimating approximate earthquake location and magnitude. Both models are containerized using

Docker[1], exposed as RESTful services using FastAPI[2], and deployed in Kubernetes[3]. Autoscaling is enabled for parallel processing on the cloud. These models can be flexibly replaced by other machine learning models, such as GPD (Ross, Meier, Hauksson, & Heaton, 2018) or EQTransformer (Mousavi et al., 2020), once containerized. We also built a training and inference pipeline based on the KubeFlow[4] framework for training and updating machine learning models on the cloud. In addition to the default batch processing for mining achieved datasets, we added the Kafka[5] and Spark Streaming[6] services to support stream processing for real-time earthquake monitoring. We received real-time seismic waveforms from seismic networks using the SeedLink API[7] and continuously pushed these waveforms to the Kafka messaging service. The process works as follows: the data is first sent to an ETL (extract, transform, load) pipeline using the Spark Streaming service, which applies a sequence of MapReduce transformations, such as windowing, grouping, filtering, and aggregation. The pre-processed data is then sent to PhaseNet and GaMMA APIs. Finally, the detected earthquakes with approximate location and magnitude are saved to an earthquake catalog and broadcast through Kafka to a web app to display real-time waveform and earthquake information.

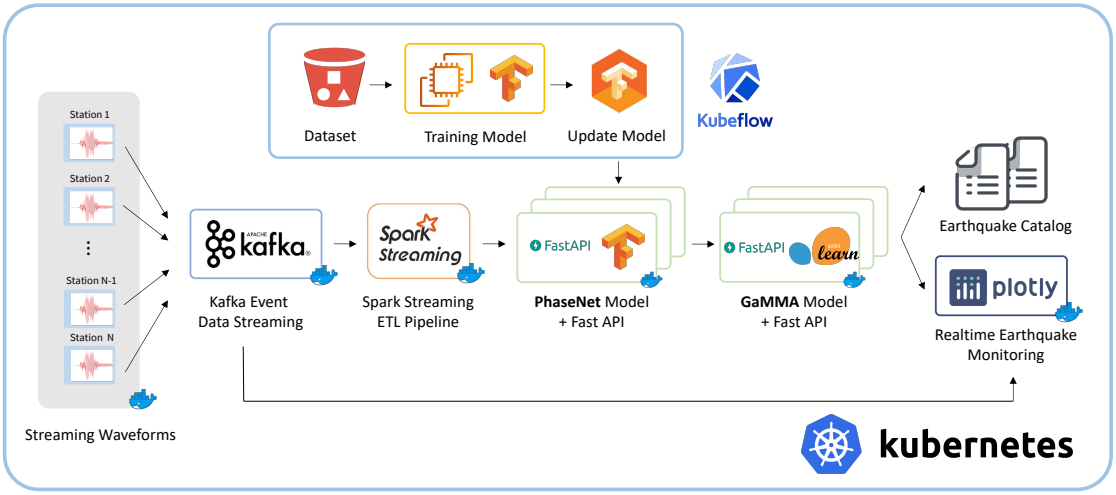


Figure 6.1: Diagram of QuakeFlow

[1] https://www.docker.com/
[2] https://fastapi.tiangolo.com/
[3] https://kubernetes.io/
[4] https://www.kubeflow.org/
[5] https://kafka.apache.org/
[6] https://spark.apache.org/
[7] http://ds.iris.edu/ds/nodes/dmc/services/seedlink/

6.2.1 Machine Learning Models

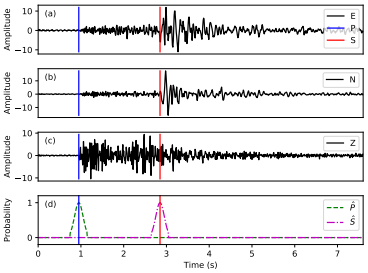
Phase picking and phase association are two important tasks in earthquake monitoring (Figure 6.2). Phase picking detects and picks arrival times of seismic phases, i.e., P-phase and S-phase (Figure 6.2(a)) at each seismic station; phase association aggregates these phase picks from multiple stations into causative earthquakes (Figure 6.2(b)). In QuakeFlow, the conventional phase picking and phase association algorithms are replaced by a deep neural network model, PhaseNet (Figure 6.2(c)), and a Gaussian mixture model, GaMMA (Figure 6.2(d)). The two models together can extract time and amplitude information of P and S phases, detect earthquakes with approximate earthquake locations and magnitudes.

PhaseNet We used the pre-trained PhaseNet model (Zhu & Beroza, 2019) to pick the arrival times of P and S phases from continuous seismic waveforms (Figure 6.2(c)). PhaseNet is a convolutional neural network (CNN) model that can effectively predict two Gaussian-shaped characteristic functions of P and S phases, from which we can extract accurate arrival times. By training on more than 700k samples labeled by human analysts, PhaseNet achieved a much better picking performance than conventional algorithms, detecting an order of magnitude more S-picks with high precision and low bias.

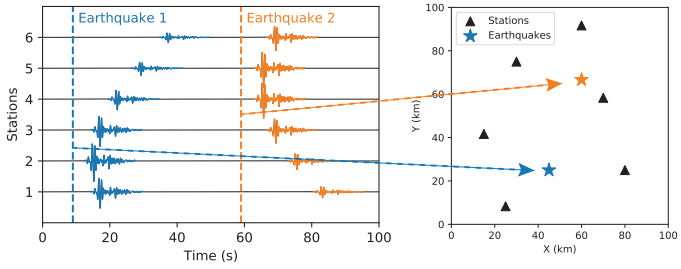
GaMMA We use the unsupervised GaMMA algorithm (Zhu, Ian, et al., 2021), which stands for **Ga**ussian **M**ixture **M**odel **A**ssociation, to associate the picked phases from multiple stations in a seismic network. GaMMA treats earthquake phase association as an unsupervised clustering problem in a probabilistic framework, where phases are clustered following the approximately hyperbolic moveout of phase travel times (Figure 6.2(d)). The probabilistic framework can flexibly consider multiple phase information, such as arrival time, amplitude, phase type, and picking quality score, to associate phases from a dense earthquake sequence effectively.

6.2.2 Data Streaming with Kafka and Spark

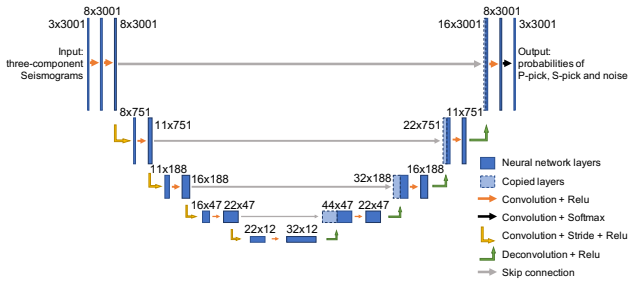
Apache Kafka is a distributed messaging system that is fault-tolerant, highly scalable, and that works well in streaming applications (Kreps et al., 2011). In QuakeFlow, Kafka acts as the central hub for real-time streams of waveform data and model prediction results. Our Kafka settings are as follows: there are 3 pre-defined Kafka topics, which are `waveform_raw`, `phasenet_picks` and `gamma_events`. The monitoring stations continuously send fragments of seismic waveforms to topic `waveform_raw`. We then use Spark Streaming, which is a scalable fault-tolerant streaming processing system that supports both batch and streaming workloads (Zaharia et al., 2013), as an ETL pipeline to apply data transformations and pre-processing to the streaming data. Spark Streaming supports window operations to aggregate streaming data over a sliding window. We group the streaming data in a specified window size (e.g., 30 seconds) using a sequence of MapReduce operations to



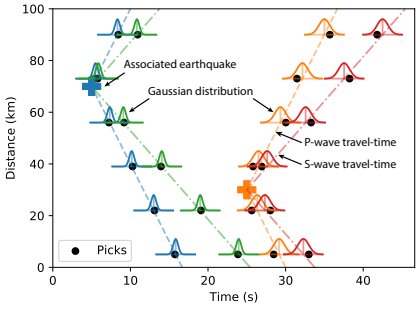
(a) Task 1: Picking P and S phases from waveforms of each seismic station.



(b) Task 2: Associating P and S phases picked across multiple stations and determining earthquake locations and magnitudes



(c) Model 1: PhaseNet model for picking P and S phases.



(d) Model 2: GaMMA Model for associating P and S phases

Figure 6.2: The upper panels (a, b) explain the two tasks (phase picking and phase association) in earthquake monitoring workflows. The lower panels (c, d) show the PhaseNet model used for phase picking (task 1) and the GaMMA model used for phase association (task 2).

prepare a structured data format for subsequent processing of PhaseNet and GaMMA. The outputs of these machine learning models are broadcast to the `phasenet_picks` and `gamma_events` topics, respectively. Last, the earthquake detection results can be saved and visualized by subscribing to the corresponding Kafka topics.

6.2.3 Auto-scaling with Kubernetes

We deployed QuakeFlow in the Kubernetes system, making it platform-independent and applicable to both on-premise servers and any cloud-platforms with Kubernetes services. Kubernetes automatically orchestrates different components of QuakeFlow to make it run on the cloud efficiently. We used both the horizontal pod auto-scaling provided by Kubernetes, as well as the node auto-provision provided by cloud-platforms, such as Google Cloud Platform (GCP), to automatically match computational resources with computational loads. We carried out a simple pressure test on GCP using a maximum of 8 computational nodes of machine type "n2-standard-2"[8] (2 vCPU and 8GB of memory) to evaluate the speedup using auto-scaling when processing a large data volume. Figure 6.3(a) shows that the computational time with auto-scaling is significantly reduced compared to the computational time without auto-scaling. Meanwhile, we can see that the data throughput, i.e., the number of data points processed per second, linearly increases with the data volume when auto-scaling is enabled (Figure 6.3(b)). We can thus apply auto-scaling to the embarrassingly parallel workload of large-scale seismic data mining.

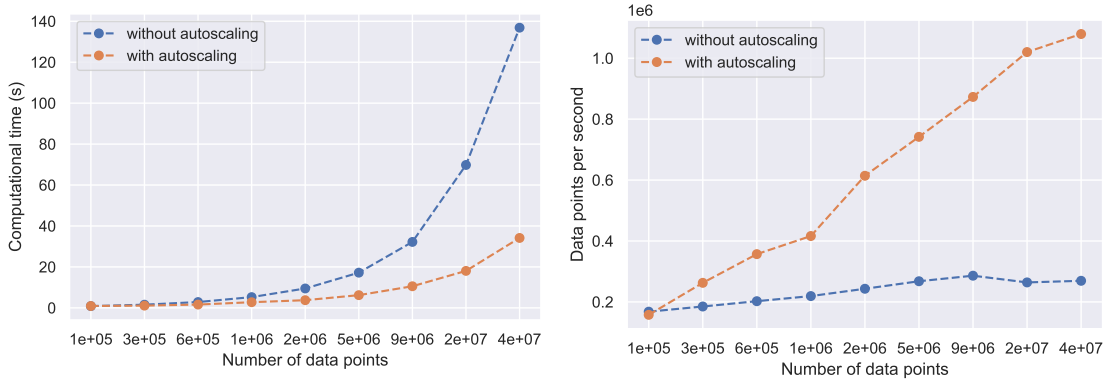


Figure 6.3: Speedup by auto-scaling: (a) computational time; (b) data throughput (number of points processed per second).

[8] https://cloud.google.com/compute/docs/machine-types

6.3 Applications

We applied QuakeFlow to a data mining task of earthquake detection using three years' achieved dataset at Puerto Rico and a simple earthquake monitoring test of one month's streaming data at Hawaii.

6.3.1 Earthquake Detection in Puerto Rico

An earthquake sequence in Puerto Rico started on December 28, 2019 and continues as of August 2021. The largest earthquake (M 6.4) to date occurred on January 7, 2020, causing many injuries and building damage. Several studies of this sequence were presented in the session "The 7 January 2020 South of Indios (M6.4) Earthquake in Puerto Rico, Response and Lessons" at SSA 2021 Annual Meeting (SSA, 2021). For example, Vanacore et al. (2021) relocated the earthquakes detected by the Puerto Rico Seismic Network (PRSN) using hypoDD (Waldhauser & Ellsworth, 2000). Yoon (2021) applied EQTransformer (Mousavi et al., 2020) to detect earthquakes and build a high-resolution catalog. In this experiment, we did not dive into the details of this earthquake sequence, but rather focused on testing QuakeFlow's earthquake detection performance and processing speed using cloud computing. We applied QuakeFlow to processing three years of archived data from 2018-05-01 to 2021-05-01 using stations within a region 65°W - 68°W and 17°N - 19°N (Figure 6.4(a)). We ran QuakeFlow on GCP with auto-scaling capability using a maximum of 60 computational nodes of machine type "n2-standard-2" (2 vCPU and 8GB of memory). Downloading waveform data from IRIS data center[9] using ObsPy (Beyreuther et al., 2010) took averaged around 3.5 hours depending on internet conditions and data center server loads. Picking P and S phase arrivals using PhaseNet took around 3 hours. Associating phases using GaMMA took around 0.5 hours. This data mining task costed around $40 based on a price of $0.07/hour per computational node[10]. The earthquake detection results are shown in Figure 6.4(b)-(e). Compared with the standard catalog retrieved from the IRIS data center, QuakeFlow has detected more than one order of magnitude small earthquakes, particularly during active aftershocks after 2020-01-01. The comprehensive earthquake catalog can provide important information to characterize active aftershock activity, reveal fault structure, and help improve aftershock forecasting. The approximate earthquake locations can be refined using earthquake location algorithms (e.g., hypoinverse (Klein, 2002)) and relocated using the double difference algorithm (e.g., hypoDD (Waldhauser, 2001)). These algorithms and other earthquake characterization components will be added to QuakeFlow in future research.

[9] http://ds.iris.edu/ds/nodes/dmc/data/
[10] https://cloud.google.com/compute/all-pricing

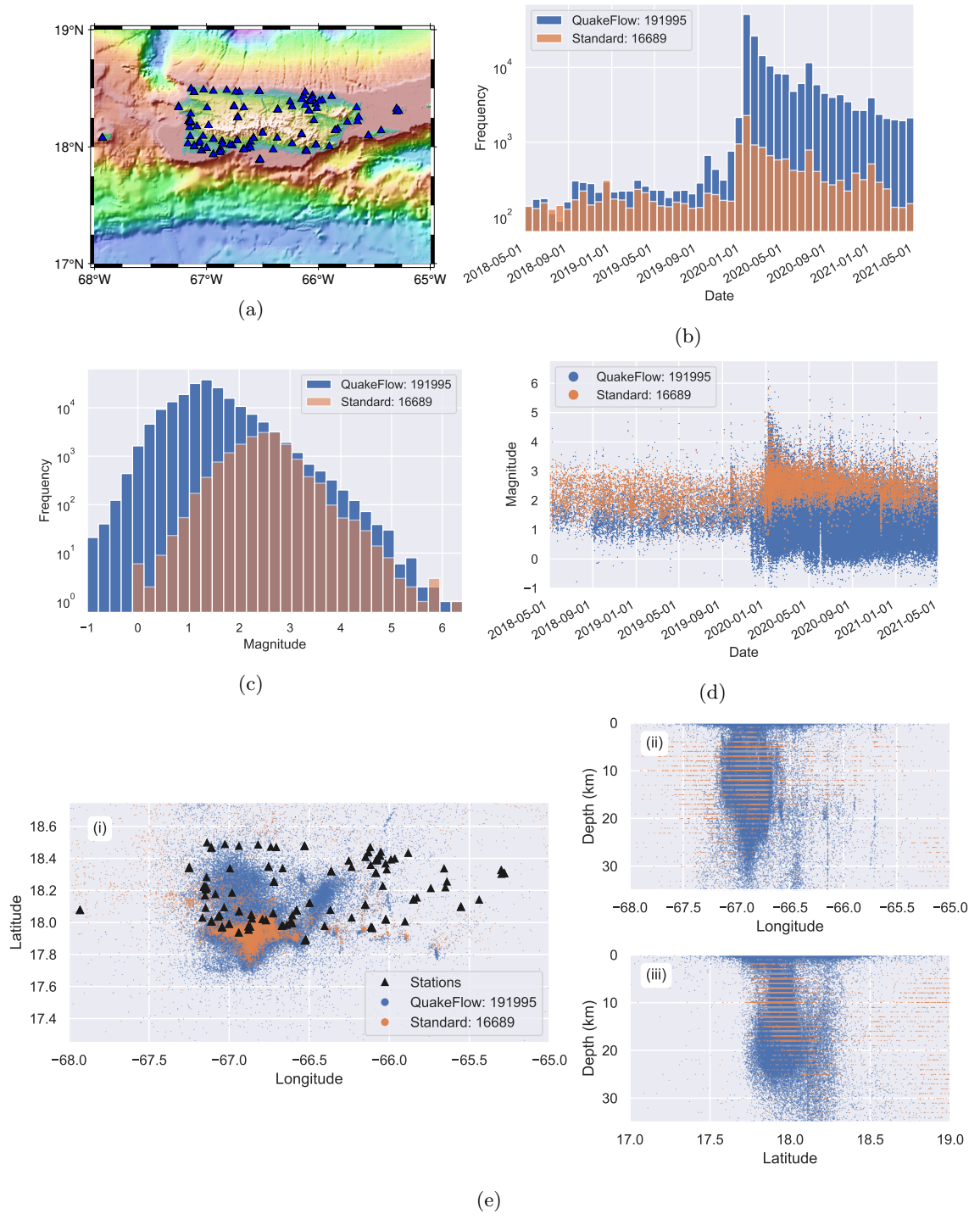


Figure 6.4: Earthquake detection results in Puerto Rico: (a) research region; (b) earthquake number; (c) earthquake magnitude frequency; (d) earthquake magnitude; (e) earthquake location. The blue color represents QuakeFlow results, and the orange color represents the standard catalog retrieved from IRIS data center.

6.3.2 Earthquake Monitoring in Hawaii

Earthquake monitoring systems, such as Earthworm (Friberg et al., 2010) and SeisComP (Weber et al., 2007), are widely used by seismic networks to monitor earthquake events in real-time. QuakeFlow is not designed to be an earthquake monitoring system but as a proof-of-concert study to evaluate improved earthquake detection performance by incorporating machine learning models into earthquake monitoring workflows. Based on the Kafka and Spark Streaming services and Kubernetes platform, however, QuakeFlow can also process streaming seismic waveforms and scale up to hundreds of seismic stations in a seismic network. We thus applied Quakeflow to earthquake monitoring on the Big Island of Hawaii (Figure 6.5(a)). We received real-time seismic data of around 60 stations from IRIS data center using the SeedLink API[11]. The results of one month's earthquake detection (Figure 6.5(b)-(e)) show that QuakeFlow can effectively monitor active volcanic seismicity, and detect many more small earthquakes. Strict quality control and additional processing, such as earthquake location and magnitude, can be added to QuakeFlow to further filter out false positives and improve earthquake source estimation. The results of this experiment show that cloud computing can provide a flexible and efficient way to implement machine learning models for earthquake monitoring workflows. Adding machine learning models into current earthquake monitoring systems (e.g., PhaseWorm (Saurel et al., 2021)) thus offer another promising direction for improving earthquake monitoring capabilities without upgrading current monitoring instrumentation and infrastructure.

6.4 Discussion and Conclusions

Machine learning - especially deep learning - methods have developed rapidly in the last few years. Applications to several earthquake sequences (e.g., 2016-2017 the Central Apennines, Italy sequence (Tan et al., 2021)) have demonstrated that machine learning models trained on large labeled datasets can significantly outperform conventional methods, resulting in an earthquake catalog with unprecedented spatial-temporal resolution. Applying these machine learning models to revisit huge archived seismic data sets can be a rewarding, but computationally challenging, task. Cloud computing addresses this computational challenge using almost unlimited computing nodes to efficiently parallelize seismic data mining workloads. As detailed here, we developed the QuakeFlow project to combine the impressive earthquake detection performance of machine learning algorithms and the powerful parallel processing capability of cloud computing to improve earthquake monitoring workflows. We built QuakeFlow based on the container-orchestration system, Kubernetes, and the Kubeflow projects to run machine learning models in parallel on the cloud. We also add modern data streaming services like Kafka and Spark Streaming to process real-time seismic waveforms. QuakeFlow is made of containerized components, facilitating updates to

[11] http://ds.iris.edu/ds/nodes/dmc/services/seedlink/

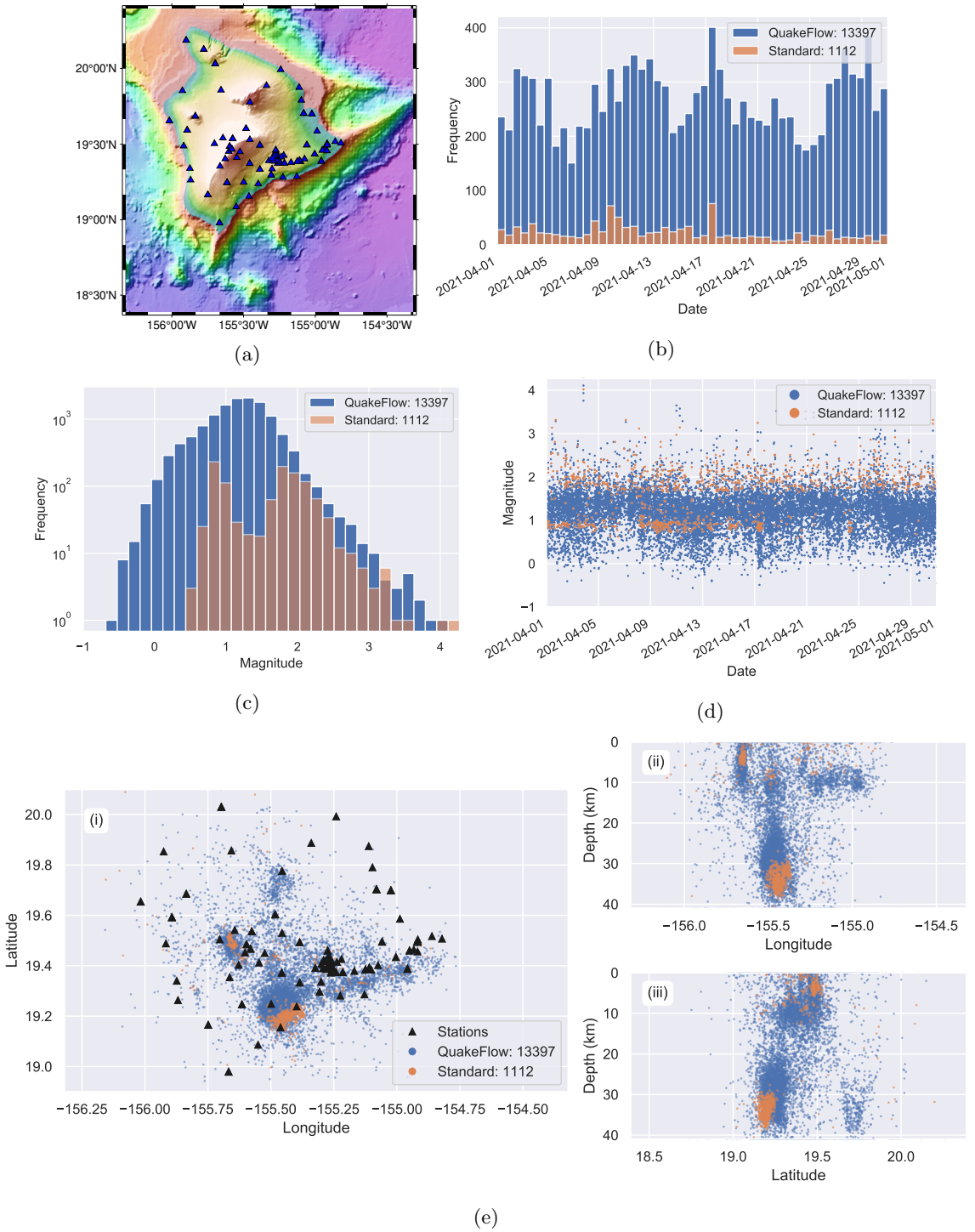


Figure 6.5: Earthquake detection results on the Big Island of Hawaii: (a) research region; (b) earthquake number; (c) earthquake magnitude frequency; (d) earthquake magnitude; (e) earthquake location. The blue color represents QuakeFlow results, and the orange color represents the standard catalog retrieved from IRIS data center.

deep learning/machine learning models current with the state-of-the-art. Quakeflow also facilitates comparison of the performance of competing algorithms by allowing swappable modules while holding data and other algorithms fixed. Additional components, such as, data pre-processing (e.g., denoising (Zhu, Mousavi, et al., 2019)), conventional earthquake location (e.g., hypoinverse and hypoDD), and earthquake characterization, can be added in future work to enhance the processing workflow and improve earthquake catalog quality.

QuakeFlow explores a new approach to the earthquake monitoring workflow based on machine learning and cloud computing. In its current implementation, QuakeFlow contains a deep neural network model for phase picking and a Gaussian mixture model for phase association to detect many more small earthquakes than conventional methods. QuakeFlow runs on most cloud platforms with Kubernetes services to efficiently process huge amounts of seismic datasets and a large number of seismic stations in parallel with auto-scaling. QuakeFlow can thus be applied to many seismic networks and datasets to improve earthquake detection and reveal details of earthquake occurrence.

Chapter 7

EQNet: An End-to-End Earthquake Detection Method for Joint Phase Picking and Association using Deep Learning

Earthquake monitoring by seismic networks typically involves a workflow consisting of phase detection/picking, association, and location tasks. Over the past few years machine learning, and especially deep learning, have been used to improve the performance of individual tasks. In this study, we explore a different approach that builds an end-to-end model that simultaneously optimizes these tasks to improve overall earthquake detection performance. To do this we introduce a novel end-to-end neural network architecture designed for multi-station processing of seismic waveforms recorded over a seismic network. The end-to-end architecture consists of three sub-networks: (1) a backbone network that extracts features from raw waveforms, (2) a phase picking sub-network that picks P- and S-wave arrivals based on these features, and (3) an event detection sub-network that aggregates the features from multiple stations and detects earthquakes. We add a shift-and-stack module to these three sub-networks, similar to back-projection, that introduces kinematic constraints on arrival times and allows the model to generalize to variable station geometry in seismic networks and to different velocity models. Unlike the case where models for each stage are optimized separately, we train the three networks together, simultaneously solving phase picking and association to improve the overall earthquake detection performance. We evaluate our proposed method on the STanford EArthquake Dataset (STEAD) and on the 2019 Ridgecrest, CA earthquake sequence. The results demonstrate our end-to-end approach can effectively pick P- and S-wave arrivals and detect earthquakes with a

performance that rivals that of other state-of-the-art approaches.

7.1 Introduction

Earthquakes are routinely monitored by local and global seismic networks, which consist of tens to thousands of seismographs that continuously record ground motion. The recorded seismic waveforms are processed by earthquake monitoring agencies to detect and catalog earthquakes. Most earthquake monitoring workflows include serial processing through a sequence of stages including seismic phase detection/picking, association, and event location. A phase detection/picking algorithm first identifies P- and S-phases independently at each station. An association algorithm then aggregates these phases from several stations by determining whether the phase arrival times are consistent with travel-times from a common earthquake hypocenter. The associated phases are then used to determine earthquake location, magnitude, and other properties related to the earthquake source (Figure 7.1). To improve earthquake detection performance, especially for frequent small earthquakes, previous research has focused on optimizing each individual task, for example by increasing phase detection/picking sensitivity and improving phase association robustness.

Phase detection/picking methods have been developed to detect P- and S-phases from a continuous seismic waveform. Traditional algorithms are designed with characteristic features, such as changes of amplitude, frequency, and other statistical properties of the time series, to detect the presence of a seismic signal within background noise and/or to localize the accurate arrival time (Allen, 1978; Bai & Kennett, 2000; Baillard et al., 2014; Lomax et al., 2012; Mousavi, Langston, & Horton, 2016; Mousavi & Langston, 2016b; Ross & Ben-Zion, 2014; Saragiotis et al., 2002). Deep learning has emerged as an effective method to learn feature representations of seismic signals automatically by training on a large number of historical data accompanied by manual labels (Mousavi et al., 2020; Mousavi, Zhu, Sheng, et al., 2019; Perol et al., 2018; Ross, Meier, & Hauksson, 2018; Ross, Meier, Hauksson, & Heaton, 2018; Zhou et al., 2019; Zhu, Peng, et al., 2019; Zhu & Beroza, 2019). Applications have demonstrated that deep-learning-based phase detectors and pickers significantly increase the detection of small earthquakes with high efficiency (Park et al., 2020; Ross, Cochran, et al., 2020; Tan et al., 2021; Wang et al., 2020). Most deep-learning-based phase detectors and pickers use only single-station information and ignore the contextual information provided by other stations. Considering multiple stations thus becomes a potential direction to improve phase detection performance.

Once phase detections are thresholded at each station, they are associated across multiple stations in a seismic network. Association methods aggregate these single-station detections into a set of seismic events and filter out false-positives in the process. Most association methods are based on the idea of back-projection, which aggregates phase detections that fit a theoretical travel-time moveout of a hypothetical event source and a wavespeed model (Dietz, 2002; Johnson et al., 1997;

Patton et al., 2016; Yeck et al., 2019; Zhang et al., 2019). An event detection is declared when a number of phases are associated to a common source event. Deep-learning-based methods have been proposed to learn phase arrival-time moveout patterns (Ross, Yue, et al., 2019) or waveform similarity (Dickey et al., 2020; McBrearty, Delorey, et al., 2019) to improve association. Treating phase detection and association separately limits the overall event detection performance. First, accurate phase detection/picking at a single station is a difficult task for the low signal-to-noise ratio (SNR) arrivals of small events whose numbers dominate earthquake catalogs. Second, association using only picked phase times does not exploit potentially informative waveform features across stations. Weak phases that fall below a detection threshold are filtered out in the first step. The information they carry is lost to subsequent processing and unable to contribute to the association step for small events.

An alternative approach to this multi-stage earthquake monitoring workflow of single-station phase detection and multi-station phase association is to develop array-based event detection methods, which can improve the sensitivity for events that are too weak to be detected reliably by a single station. Methods like template matching (Gibbons & Ringdal, 2006; Shelly et al., 2007; Zhang & Wen, 2015) and shift-and-stack (i.e., back-projection or beamforming) (Kao & Shan, 2004; Kiser & Ishii, 2013; Li, Peng, et al., 2018) exploit the coherent waveform signals of multiple stations to enhance detection sensitivity; however, these array-based methods have several disadvantages. Template matching requires *a priori* information in the form of a catalog of detected events as templates and if done comprehensively, can have a high computational cost (Ross, Trugman, Hauksson, et al., 2019). Back-projection relies on high similarity between waveforms. Sequences of tiny earthquakes usually have complex waveforms contaminated by noise, which when coupled with subsurface heterogeneity, complex wave propagation, and wave-speed model uncertainty, lead to a smeared back-projection image from which it can be challenging to extract events. Several deep-learning-based methods have treated an array of seismic recordings as an image and applied 2D convolution neural networks to pick phases or detect events (Shen & Shen, 2021; Yang et al., 2021; Zhang et al., 2021; Zheng et al., 2020). This approach is effective for a fixed geoemtry of seismic recordings, such as a shot gather in exploration seismology, but it unsuitable for the important problem of long-term seismic monitoring for which station geometry will vary with time. This common situation would lead to poor model generalization when applied to a seismic network geometry that differs substantially from that of the training network.

We present a novel approach designed to combine the advantages of effective representation learning by deep neural networks and robustness of array-based event detection. This method, which we call EQNet, combines feature extraction, phase picking, and event detection stages in an end-to-end neural network architecture in order to incorporate knowledge of the downstream association task into the detection step and help the network to learn features that are effective for multiple tasks. The architecture first extracts features from seismic waveforms recorded at each seismic station

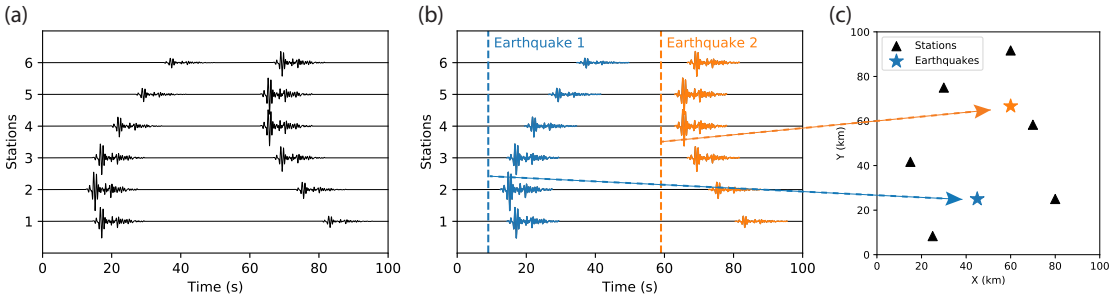


Figure 7.1: Schematic of earthquake event detection over a seismic network: (a) continuously recorded seismic data; (b) multi-stage tasks including: detecting single-station signals, associating signals from a common earthquake, and (c) estimating earthquake source parameters, such as origin time and location.

using a deep backbone network. These feature vectors are then processed by two sub-networks for picking P- and S-phases and for aggregating features from multiple stations and detecting earthquake events. This multi-task approach was also used by Mousavi et al. (2020) for combining phase picking and event detection tasks at a single station. A characteristic of our approach is that we *jointly* optimize the neural network parameters of feature extraction, phase picking, and event detection to maximize the earthquake detection performance using the entire seismic network. This avoids the need for hand-designed characteristic functions and association rules used in conventional multi-stage approaches and preserves information in the input waveforms through all stages. We benchmarked the phase picking performance on the STanford EArthquake Dataset (STEAD) (Mousavi, Sheng, et al., 2019) and evaluated the earthquake detection performance against four state-of-the-art catalogs previously developed for the 2019 Ridgecrest, CA earthquake sequence. Our end-to-end approach provides a promising new direction for further improving earthquake monitoring using deep learning.

7.2 Method

7.2.1 End-to-end architecture

To perform the phase picking, association, and event detection tasks in an end-to-end fashion, we design a novel architecture for EQNet (Figure 7.2) that consists of four components: (1) a backbone feature extraction network, (2) a phase picking network, (3) a shift-and-stack module, and (4) an event detection network. EQNet processes an array of seismograms from multiple stations as input, and produces both picked P/S-phases and detected earthquake events with a rough estimation of earthquake origin time and location as outputs. The backbone feature extraction network maps raw seismic waveforms into a feature space with a condensed time dimension. The feature extraction network is modified from the 18-layer residual network (ResNet-18) (He et al., 2016) that uses

1D convolution to process three-component seismic waveforms and automatically extract condensed features. The extracted features are then processed by two sub-networks. The phase picking network extracts P- and S-phase picks from these features, and the event detection network detects earthquake events from shifted features produced by the shift-and-stack module. The shift-and-stack module performs a transformation in the feature domain, similar to the back-projection process, by sampling candidate hypocenters, shifts the features based on estimated theoretical travel-times, and generates a collection of aligned features. The module introduces prior knowledge of physical constraints on arrival-time moveout, epicentral distance, and the wavespeed model. The event detection network then classifies whether an earthquake exists at a specific candidate location and time based on the aligned views of features. The feature extractor network, phase picker and event detector sub-networks serve functions similar to the backbone network and multiple head networks in object detection methods, such as YOLO (Bochkovskiy et al., 2020). The parameters of these three networks are jointly optimized during training using a summation of three binary cross-entropy losses as a multi-task optimization target:

$$\mathcal{L} = \lambda_P \mathcal{L}_P + \lambda_S \mathcal{L}_S + \lambda_{EQ} \mathcal{L}_{EQ} \tag{7.1}$$

$$\mathcal{L}_P = \sum_T y_P \log \hat{y}_P + (1 - y_P) \log(1 - \hat{y}_P) \tag{7.2}$$

$$\mathcal{L}_S = \sum_T y_S \log \hat{y}_S + (1 - y_S) \log(1 - \hat{y}_S) \tag{7.3}$$

$$\mathcal{L}_{EQ} = \sum_T y_{EQ} \log \hat{y}_{EQ} + (1 - y_{EQ}) \log(1 - \hat{y}_{EQ}) \tag{7.4}$$

where $\mathcal{L}_P$ and $\mathcal{L}_S$ are the losses of the phase picking networks for P- and S-phase respectively, and $\mathcal{L}_{EQ}$ is the loss of the event detection network. T is the number of time points, y is the ground truth label, and $\hat{y}$ is the network prediction. $(\lambda_P, \lambda_S, \lambda_{EQ})$ are weights for each loss function. For this proof-of-concept study, we do not tune for the optimal weighting, but set all weights to 1. The parameters of these three neural networks are shown in Figure 7.3. We use two phase picking sub-networks for picking P- and S-phases, and each network takes half of the extracted features as input. These two sets of features are shifted using P- and S-wave velocities respectively and processed by the event detection sub-network. We train this end-to-end neural network model using the AdamW optimizer (Loshchilov & Hutter, 2017), a weight decay rate of 1×10^{-4}, and a cosine learning rate decay strategy (Loshchilov & Hutter, 2016) with an initial learning rate of 3×10^{-4}.

7.2.2 Dataset and training

We collected a training set from the Northern California Earthquake Data Center (NCEDC, 2014) and selected events with manual P- and S-phase picks from more than 3 stations. The total training dataset contains 99,465 events and 563,790 P- and S-picks. Figure 7.4(a) shows one example recorded

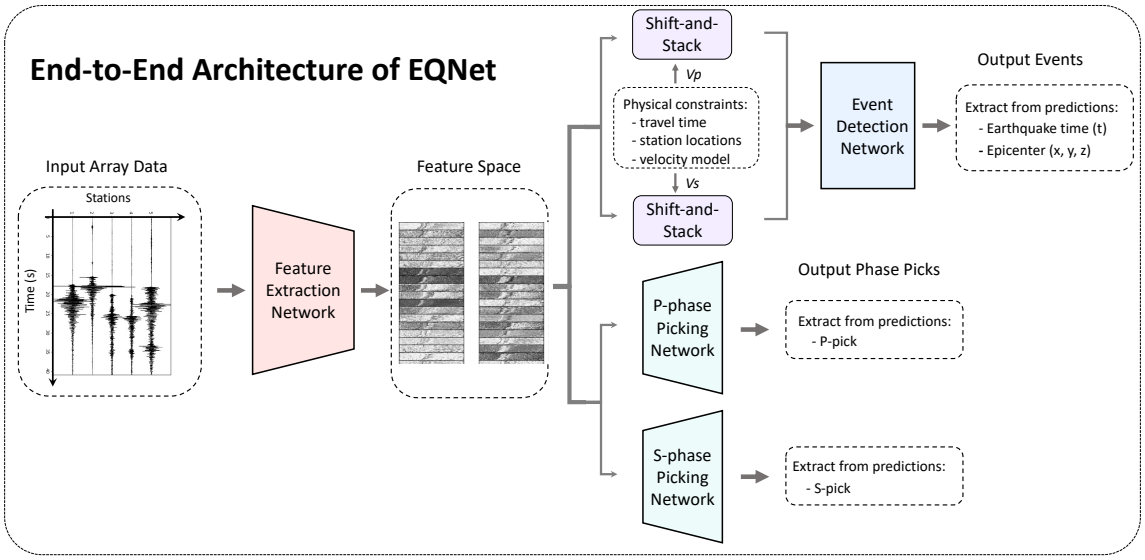


Figure 7.2: Architecture of the end-to-end earthquake detection model (EQNet). The input data are seismic waveforms recorded by multiple stations in a seismic network. The outputs are two activation sequences of P- and S-picks and an activation map of earthquake events with approximate earthquake time and location. EQNet consists of four sub-modules: feature extraction, phase picking, shift-and-stack, and event detection. The feature extraction network transforms raw seismic waveforms into feature representations using a 1D ResNet-18 model. The phase picking network then predicts two activation sequences for P- and S-phase arrivals based on the features. The shift-and-stack module is designed to sample candidate earthquake locations, calculate travel-times at each station location, and shift the features accordingly, allowing generalization to different station locations and seismic wavespeed models. The event detection network predicts an activation map for approximate earthquake times and locations based on the shifted features. These three networks are optimized simultaneously during training to improve earthquake detection performance.

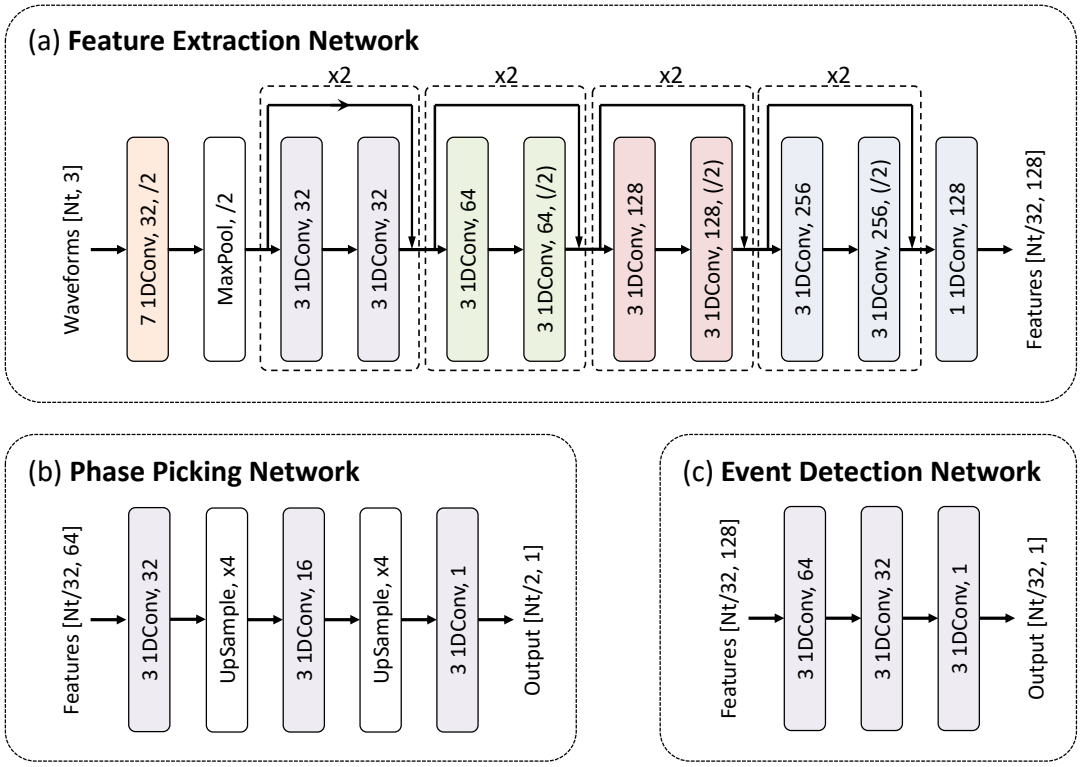


Figure 7.3: Neural network parameters: (a) a backbone neural network for feature extraction; (b) a sub-network for phase picking; (c) a sub-network for event detection. Numbers in the square brackets are the shapes of input and output data. For example, the input waveform has a length of Nt samples from 3 components. The numbers inside each neural network layer are the convolution kernel size, the number of channels, and the stride step (inside brackets). Batch normalization layers and ReLU activation functions are used after each 1D convolutional layer but are not plotted here.

at 14 stations. We use the same truncated Gaussian-shaped target function used for PhaseNet (Zhu & Beroza, 2019) as training labels for P/S-phase arrival times and earthquake origin time. The Gaussian-shape target function allows some uncertainty in manual labels and balances the distribution between a small number of positive sample points of manual labels and the remaining majority negative sample points in a seismic waveform to improve training accuracy and speed. We set two different widths of the Gaussian-shaped target functions: 1s for phase arrival time and 2s for earthquake origin time to account for different uncertainty levels. Figure 7.4(d) and (e) show the corresponding prediction scores after training of the phase picking network and the event detection network respectively. We use a negative sampling approach (Goldberg & Levy, 2014) to collect positive samples from the correct earthquake location and negative samples from other random locations. This negative sampling process helps balance the sparse space of true earthquake locations relative to the much larger space of candidate locations spanned by the entire monitoring area to speed up the training. During inference, we take the continuous seismic waveforms (Nt) as input data. The shift-and-stack module uniformly but coarsely samples the whole space at $Nx \times Ny$ grid points with a horizontal interval of ~4 km. Figure 7.4(f) shows the corresponding activation map of prediction scores. From the spatial-temporal predictions above a threshold, we first extract earthquake times from peaks along the time axis (Figure 7.4(e)) and then determine the earthquake location using the geometric median of the top 20 activated grids (Figure 7.4(f)). We visualize the automatically extracted features in Figure 7.4(b) and (c). Each mini-panel shows one output channel of the feature extraction network in Figure 7.3(a). The horizontal axis presents time, and the vertical axis represents stations. We plot 32 channels out of the total 64 channels for P- and S-phases. We observe significant differences among these features, such as localized activations at P- and S-phase arrivals and broad activations during earthquake signals. These features contain characteristic information that is useful to differentiate earthquake signals from background noise and to pick phase arrivals.

7.3 Results

We evaluate the phase picking performance of EQNet on the previously published and publicly available STanford EArthquake Dataset (STEAD) and evaluate the earthquake detection performance on the 2019 Ridgecrest, CA earthquake sequence.

7.3.1 Benchmark with the STEAD dataset

STEAD is a global dataset of more than one million seismic waveforms (Mousavi, Sheng, et al., 2019) with both P- and S-arrival labels that has been used to compare state-of-the-art automatic phase pickers (Mousavi et al., 2020). We selected the same test set (~120,000 waveforms) used in Mousavi et al. (2020) to evaluate the phase picking performance of EQNet. The waveforms were

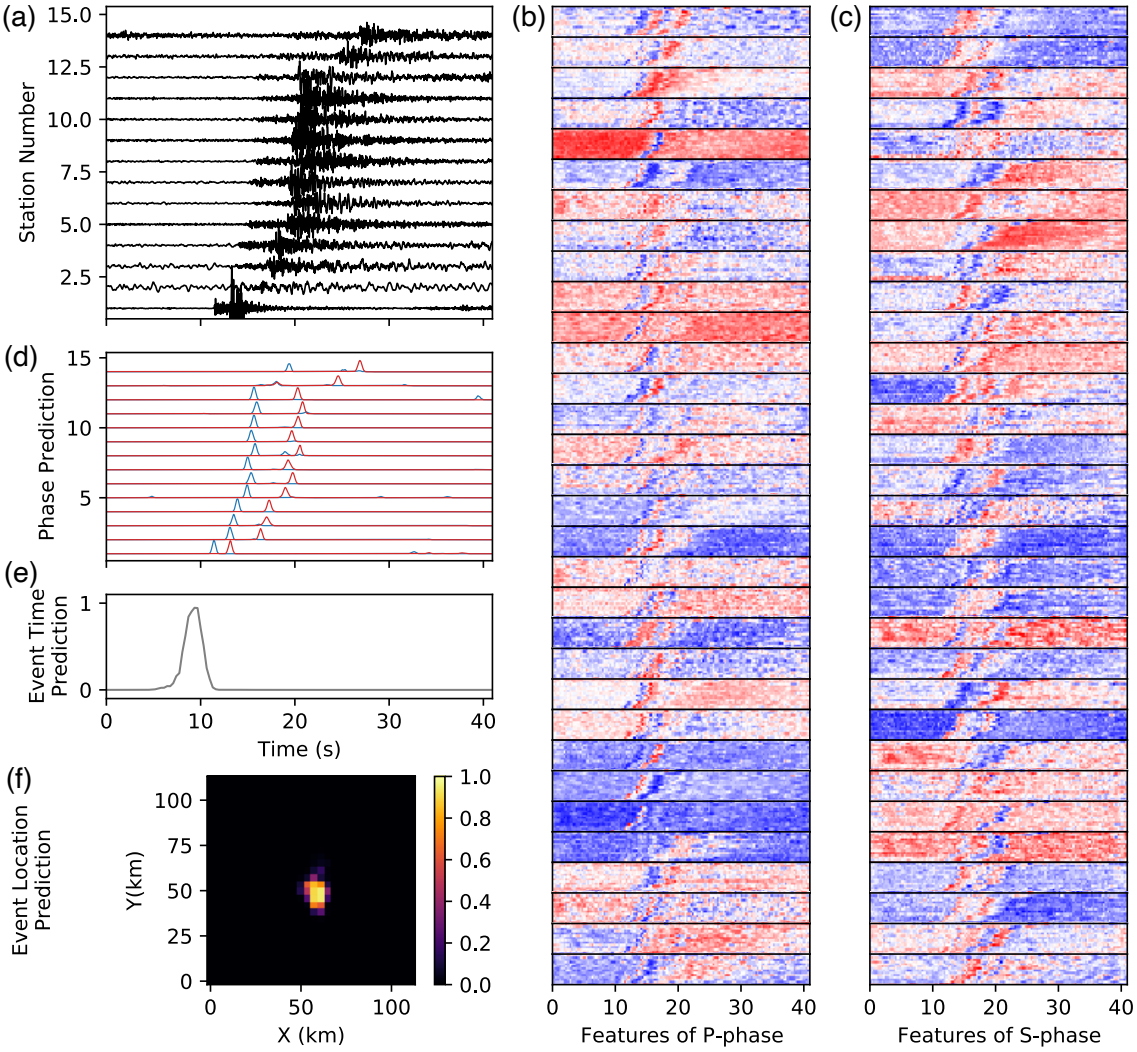
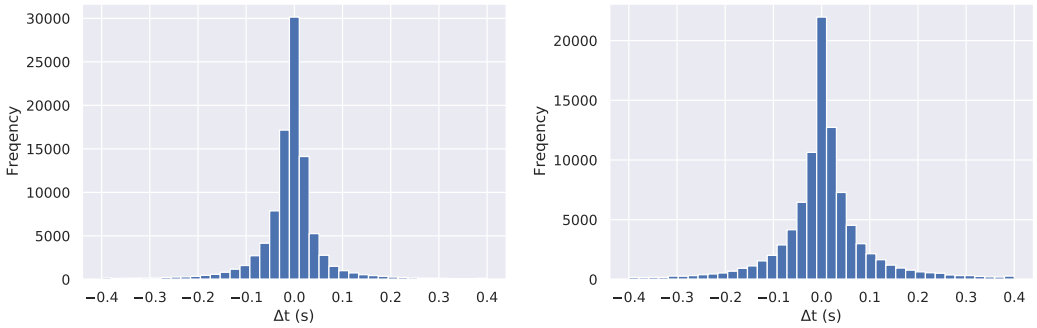


Figure 7.4: Prediction example: (a) seismic waveforms; (b, c) features of P-phase and S-phase extracted by the backbone feature extraction neural network. Each sub-panel shows one output channel of the backbone network. (d) P- and S-phase activation scores predicted by the phase picking neural network. (e, f) Earthquake activation scores predicted by the event detection neural network. Specifically, (e) shows the max score along the time axis from which we extract the event time, and (f) shows the max score along spatial axes from which we extract the event location.



(a) (b)

Figure 7.5: Phase picking performance: (a) time residuals of P-picks; (b) time residuals of S-picks.

Table 7.1: Statistics of picking time residuals.

		Precision	Recall	F1	Mean (s)	Std (s)	MAE (s)
EQNet	P-phase	0.96	0.95	0.95	-0.01	0.08	0.04
	S-phase	0.92	0.92	0.92	0.01	0.11	0.07
PhaseNet	P-phase	0.96	0.96	0.96	-0.02	0.08	0.07
	S-phase	0.96	0.93	0.94	-0.02	0.11	0.09
EQTransformer	P-phase	0.99	0.99	0.99	0.00	0.03	0.01
	S-phase	0.99	0.96	0.98	0.00	0.11	0.01

processed by the feature extraction network and then the phase picking networks to pick P- and S-phase arrivals. We extracted the peaks above a threshold of 0.5 in the predicted activation sequences (Figure 7.4(d)) to detect phases and determine arrival times. The distribution of residuals between the predicted arrival times and the manual labels are shown in Figure 7.5, and the corresponding statistics are listed in Table 7.1. The predicted picks that are within 0.5 seconds from the manual labels are counted as true positives. The rest are counted as false positives. Although EQNet uses a simple CNN model, the picking performance approaches the state-of-the-art models PhaseNet (Zhu & Beroza, 2019) and EQTransformer (Mousavi et al., 2020). Moveover, although EQNet is trained using data from Northern California only, it generalizes well to this global dataset.

7.3.2 Benchmark on the 2019 Ridgecrest earthquake

We chose the Ridgecrest earthquake sequence to evaluate EQNet's event detection performance using multi-station waveforms. Ridgecrest is useful for this comparison because in addition to the routine catalog developed by the Southern California Seismic Network (SCSN), Shelly (2020) and Ross, Idini, et al. (2019) built two different catalogs using the template matching method, which takes earthquake events in the SCSN catalog as templates and scans the continuous seismic data to detect more small earthquakes. Liu et al. (2020) built another catalog using the PhaseNet picker and the REAL (Zhang et al., 2019) association algorithm. These state-of-the-art catalogs provide

good benchmarks for evaluating the performance of our end-to-end method.

We selected five days from July 4th to 8th for the benchmark comparison and downloaded the continuous seismic waveforms from seismic stations within 1 degree of the location (-117.504W, 35.705N). Earthquakes were frequent and closely spaced in time during these five days, which makes the detection task challenging. During this period, the SCSN catalog reports 8,044 events, Shelly (2020) reports 13,775 events, Ross, Idini, et al. (2019) report 23,705 events, and Liu et al. (2020) report 12,357 events. We report EQNet's detection number at two activation thresholds: 11,739 events at a threshold of 0.5 and 24,708 events at a threshold of 0.1. Because the ground-truth event number behind these continuous waveforms is unknown, and because each catalog contains false positive and false negative detections due to different algorithms, hyperparameters, and thresholds, it is a challenging task to compare detection performance accurately across catalogs. We used a cross-validation test among these catalogs by assuming one catalog as ground truth and calculating the relative performance metrics of the other catalogs. We consider earthquake events whose origin time fall within 3s of the benchmark catalog time as true positives. Table 7.2 shows the precision, recall, and F1-score of the cross-validation test. The SCSN catalog (8,044 events) has the highest precision, meaning that most of events in the SCSN catalog also exist in the other catalogs and are more likely to be true earthquakes. The EQNet0.1 catalog (24,708 events) has the highest recall rate meaning that the events in the other catalogs can also be detected by EQNet. The precision and recall values are greatly affected by number of detected earthquakes. The F1-score (the harmonic mean of precision and recall) provides a balanced evaluation criteria of the two measures. Liu et al. (2020)'s catalog (12,357 events) and EQNet0.5 catalog (11,739 events) achieves the highest F1-scores. Note that this cross-validation result mainly reflects the consistency among catalogs instead of determining which catalog is the most accurate. From the point of view of this paper, the result demonstrates that our end-to-end method achieves a performance similar to that of the other state-of-the-art approaches. Through choosing a range of thresholds between 0.1 - 1, we plot the precision and recall curves of EQNet measured by assuming the other four catalogs as ground truth (Figure 7.6(a)). The curves show good consistency between EQNet's catalog and the catalogs of Liu et al. (2020), Shelly (2020), and SCSN. The lower similarity to Ross, Idini, et al. (2019) is potentially due to either false positive events in that catalog or to false negative events in the EQNet catalog. We further analyze the accuracy of the earthquake origin time and locations of the EQNet0.5 catalog compared with the SCSN catalog (Figure 7.6(b) - (d)). The error distributions of earthquake time and location show that most detected earthquakes have a time error within 1 s and an epicenter error within 6 km. This is not surprising because the event detection network of EQNet only estimates very approximate earthquake locations for detection. We note that these locations are sufficiently accurate to carry out phase association, and to be used as initial location input to conventional location (e.g. hypoinverse (Klein, 2002)) or relocation (e.g. hypoDD (Waldhauser & Ellsworth, 2000)) algorithms.

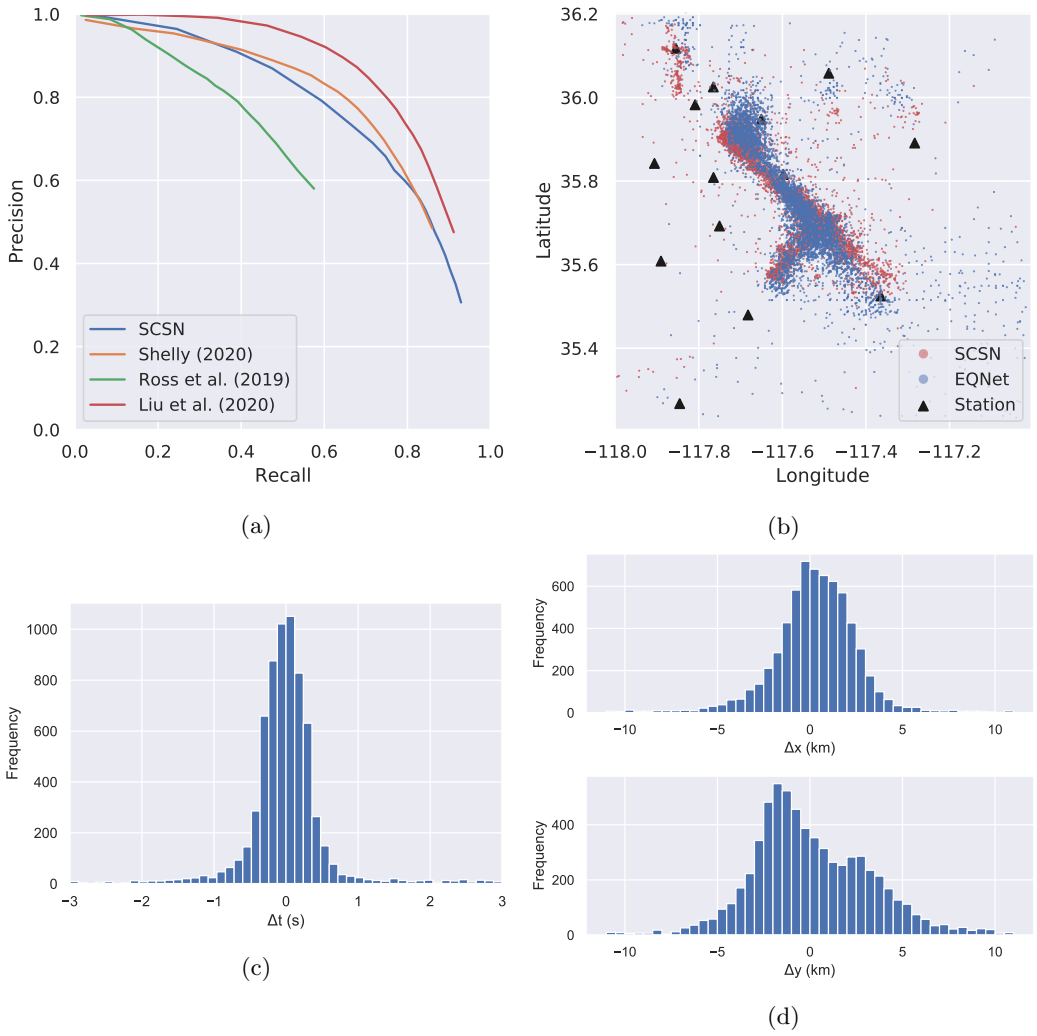


Figure 7.6: Event detection performance: (a) precision-recall curves calculated by assuming the four state-of-the-art catalogs as ground truth; (b) estimated earthquake locations; (c) error distribution of earthquake time; (d) error distribution of earthquake location compared with that of the SCSN catalog.

Table 7.2: Cross-validation among four state-of-the-art catalogs and EQNet's catalog

Testing catalog	Coefficient	Benchmark catalog					
		SCSN	Shelly (2020)	Ross et al. (2019)	Liu et al. (2020)	EQNet0.5	EQNet0.1
SCSN (8,044)	Precision	1.0	**0.844**	**0.896**	**0.833**	**0.830**	**0.929**
	Recall	1.0	0.494	0.299	0.525	0.554	0.306
	F1 score	1.0	0.623	0.448	0.644	0.665	0.460
Shelly (2020) (13,775)	Precision	0.494	1.0	0.791	0.722	0.680	0.859
	Recall	0.844	1.0	0.448	0.774	0.773	0.485
	F1 score	0.623	1.0	0.572	0.747	0.724	0.620
Ross et al. (2019) (23,705)	Precision	0.299	0.448	1.0	0.421	0.392	0.575
	Recall	0.896	0.791	1.0	0.801	0.790	**0.580**
	F1 score	0.448	0.572	1.0	0.552	0.524	0.577
Liu et al. (2020) (12,357)	Precision	0.525	0.774	0.801	1.0	0.796	0.911
	Recall	0.833	0.722	0.421	1.0	0.753	0.475
	F1 score	0.644	**0.747**	0.552	1.0	**0.774**	**0.625**
EQNet0.5 (threshold=0.5) (11,739)	Precision	**0.554**	0.773	0.790	0.796	1.0	1.0
	Recall	0.830	0.680	0.392	0.753	1.0	0.475
	F1 score	**0.665**	0.724	0.524	**0.774**	1.0	0.644
EQNet0.1 (threshold=0.1) (24,708)	Precision	0.306	0.485	0.580	0.475	0.475	1.0
	Recall	**0.929**	**0.859**	**0.575**	**0.911**	1.0	1.0
	F1 score	0.460	0.620	**0.577**	0.625	0.644	1.0

7.4 Discussion

Multi-stage earthquake detection workflows consisting of a sequence of processing tasks, such as, phase detection/picking, association, location, and characterization, have proven to be an effective approach for earthquake monitoring; however, each step of this multi-stage approach involves information loss, such as weak signals undetected because they fall below a threshold at the phase picking step. An end-to-end approach without intermediate steps avoids this information loss by using seismic waveforms to improve overall earthquake detection performance. We have developed an end-to-end neural network architecture for earthquake detection to solve the phase picking and association tasks simultaneously. The end-to-end architecture consists of three neural networks for feature extraction, phase picking and event detection. These networks are jointly optimized during training to maximize their performance. The end-to-end training approach eliminates separate hyper-parameter tuning for each step in conventional earthquake detection workflows and avoids information loss in each step. EQNet combines the advantages of effective feature extraction of deep neural networks with the high detection robustness of array-based methods by processing waveforms from stations across a seismic network. The good performance of both the phase picker and the event detector networks demonstrates that the extracted features (Figure 7.4(b) and (c)) contain characteristic information of seismic waveforms. The event detection network combines these features to recognize coherent patterns across multiple stations and thus improve earthquake detection, particularly for the important case of weak signals.

In contrast to models that are trained on a fixed set of stations and neglect station location information (Yang et al., 2021; Zhang et al., 2020), we design a shift-and-stack module that serves a similar function as the back-projection method commonly employed in earthquake detection and imaging (Ben-Zion et al., 2015; Inbal et al., 2015; Kao & Shan, 2004). This module incorporates physical constraints of travel times determined by station locations and wavespeed models, making

EQNet generalizable to other research regions using different station geometries and wavespeed models. In this work, for example, we demonstrated that the model trained with data from Northern California generalizes well on the Ridgecrest earthquake sequence in Southern California. The sampling module enables generalizability at some computational cost because the event detection network needs to process a collection of shifted features sampled at different spatial locations. To reduce the computational effort, we engineer a deep backbone network, which processes the input waveforms only once to extract features, and a shallow event detection network, which efficiently processes a group of shifted features in parallel. Conventional back-projection methods for local earthquake detection and location have strict requirements of coherent waveforms and polarity, accurate velocity model, and can come at high computational cost from dense grid sampling (Beskardes et al., 2018). Filtered waveforms and characteristic functions, such as the complex envelope (Gharti et al., 2010), STA/LTA (Fischer et al., 2020), and kurtosis (Langet et al., 2014), are used to improve stacking results and to reduce computational cost. The features extracted by EQNet provide a rich representation of characteristic functions that are informed by data and are optimized for the relevant learning tasks (Figure 7.4(b) and (c)), thus EQNet achieves good performance without either strict requirements or preprocessing. In the experiments above, we applied EQNet directly to the raw waveforms, using an assumed uniform velocity model of 6 km/s for P waves and 3.4 km/s for S waves. We applied grid-search to latitude and longitude components with a spatial interval of 4 km. Beyond this proof-of-concept study, it will be possible to use more realistic velocity models and/or improved travel-time predictions to improve the output. The computational time required to process 24 hours of continuous data from 42 three-component seismometers is around 7 minutes on a Tesla V100-SXM2-32GB-LS GPU, and because the problem is embarrassingly parallel, the processing can be easily distributed over multiple GPUs.

Earthquake monitoring bears some similarity to object detection in computer vision, which aims to locate and classify objects in an image (Girshick, 2015; Girshick et al., 2014; Liu, Anguelov, et al., 2016; Redmon et al., 2016; Ren et al., 2015). CNN-based object detection methods, such as YOLO (Redmon et al., 2016), scan a set of coarse grids to classify object categories and predict bounding boxes. Similarly, EQNet detects earthquakes by scanning a spatial-temporal space. Because most earthquakes, and particularly small earthquakes, can be approximated as a point source, we do not need to predict bounding boxes of different shapes as is the case in object detection. Our end-to-end approach opens a new pathway for the earthquake detection problem at the seismic network level. It resembles the way analysts monitor earthquakes by examining a collection of waveforms recorded across multiple stations. New algorithms and architectures in the rapidly evolving field of object detection will likely improve this end-to-end approach for earthquake monitoring in the near future.

7.5 Conclusions

Deep learning is an effective approach for earthquake detection; however, currently a dominant approach in deep learning models for seismic monitoring is to process seismic waveforms one station at a time, which can not realize the full potential of network-based monitoring. To do that, deep learning needs to be combined with effective association methods to aggregate information from multiple stations in a seismic network to detect earthquakes and reduce false positives. We have developed an end-to-end model, EQNet, that combines detection and association for array-based earthquake detection. EQNet consists of three neural networks for feature extraction, phase picking, and event detection, and jointly optimizes these tasks to improve overall detection performance. EQNet processes multiple waveforms from a seismic network to improve detection sensitivity for small earthquakes. Application to the STEAD dataset demonstrates that EQNet can pick P- and S-phases as effectively as other state-of-the-art deep learning models. The cross-validation result on the 2019 Ridgecrest earthquake sequence demonstrates that EQNet achieves good earthquake detection performance and generates earthquake catalogs consistent with four state-of-the-art approaches. EQNet represents a new strategy for developing array-based deep learning models to improve earthquake detection.

Chapter 8

ADSeismic: A General Approach to Seismic Inversion with Automatic Differentiation

Imaging Earth structure or seismic sources from seismic data involves minimizing a target misfit function, and is commonly solved through gradient-based optimization. The adjoint-state method has been developed to compute the gradient efficiently; however, its implementation can be time-consuming and difficult. We develop a general seismic inversion framework to calculate gradients using reverse-mode automatic differentiation. The central idea is that adjoint-state methods and reverse-mode automatic differentiation are mathematically equivalent. The mapping between numerical PDE simulation and deep learning allows us to build a seismic inverse modeling library, ADSeismic, based on deep learning frameworks, which supports high performance reverse-mode automatic differentiation on CPUs and GPUs. We demonstrate the performance of ADSeismic on inverse problems related to velocity model estimation, rupture imaging, earthquake location, and source time function retrieval. ADSeismic has the potential to solve a wide variety of inverse modeling applications within a unified framework.

8.1 Introduction

Inverse modeling is used in seismology to recover physical parameters such as earthquake location, magnitude, and Earth's interior structure. Such inverse problems are usually solved by minimizing a misfit function that measures the discrepancy between predictions and observations. Derivative-based optimization requires calculation of the gradient of the misfit function with respect to the physical parameters. The adjoint-state method is a commonly used technique for computing the

gradient efficiently (Plessix, 2006). This method solves an adjoint linear system, which involves solutions of the forward problem usually implemented in partial differential equations (PDEs). The drawback of the adjoint-state method is that the derivation and implementation can be very challenging. For PDE-constrained optimization, the adjoint equation to calculate the partial derivatives of specific physical equation and its PDE discretization with respect to the model parameter needs to be derived on a case-by-case basis for different systems and physical parameters (Bradley, 2013). Although many frameworks exist for specific inverse modeling applications (Cockett et al., 2015; Rücker et al., 2017), to our knowledge general frameworks that can estimate physical parameters without case-by-case gradient derivation and implementation are lacking.

Automatic differentiation (AD) (Baydin et al., 2017; Paszke et al., 2017), where the gradients are computed automatically based on the computational graph of the forward simulation, provides an alternative approach. In AD, a computational graph of the forward simulation keeps track of arithmetic operational dependencies, stores intermediate results, and computes the gradient using the chain rule. AD has been the dominant approach for training deep neural networks, which is known as "backpropagation" in the deep learning community (Rumelhart et al., 1986). Both deep neural networks and PDE simulations can be viewed as a series of linear or nonlinear operators (Hughes et al., 2019). Moreover, reverse-mode automatic differentiation has been shown to be mathematically equivalent to the adjoint-state method (LeCun et al., 1988; Li, Xu, Harris, et al., 2019; Ren et al., 2020). This correspondence allows us to develop a flexible and general seismic inversion framework, ADSeismic, based on current deep learning frameworks such as TensorFlow (Abadi et al., 2016). We note that AD has already been applied to velocity estimation in exploration seismology (Cao & Liao, 2015; Richardson, 2018b; Sambridge et al., 2007; Vlasenko et al., 2016). Compared to existing open-source seismic inversion software, such as IWAVE (Symes, 2015), JUDI4Flux.jl (Witte et al., 2020), and Devito (Louboutin et al., 2019), ADSeismic has some distinct features for seismic inversion: it allows for flexibly experimenting with different inversion targets, leverages specialized hardware designed for deep learning, and executes numerical simulations on heterogeneous computing platforms. ADseismic provides a high performance environment with easily accessible gradients on CPUs and GPUs. It also supports parallel computing based on MPI (Message Passing Interface) (Clarke et al., 1994) for solving inversion tasks on large-scale super computers.

We demonstrate several applications, including velocity estimation, fault rupture imaging, earthquake location, and source time function retrieval. AD yields the same inversion results as adjoint-state methods. The advantage is that while we need to derive and implement a specific gradient for each parameter in different cases with the adjoint-state method, these inversion problems can be solved similarly by specifying a different parameter as the inversion target with ADSeismic because the gradient for each parameter is automatically calculated by AD. Moreover, both the forward simulation and inversion can be accelerated by GPUs and large-scale CPU clusters. Since deep learning hardware and frameworks are continuously improving, ADSeismic provides seismic

inverse modeling with increasingly powerful automatic differentiation techniques for a wide range of applications.

8.2 Method

Automatic Differentiation

Automatic differentiation (AD) is a general and efficient method to compute gradients based on the chain rule. By tracing the forward-pass computation, the gradient at the final step propagates back to each operator and parameter in a computational graph. AD is mainly used for training neural network models that consist of a sequence of linear transforms and non-linear activation functions. AD calculates the gradients of every variable by propagating the gradients back from the loss function to the trainable parameters. These gradients are then used in a gradient-based optimizer, such as the gradient descent (GD) method, to update the parameters and minimize the differences between the model predictions and the ground-truth labels. Numerical simulations based on PDEs are similar to neural network models in that they are both sequences of linear/non-linear transformations (Fig. 8.1). For example, the Finite-Difference Time-Domain (FDTD) method (Yee, 1966), applies a finite difference operator to consecutive time steps to solve time-dependent PDEs (Hughes et al., 2019). In seismic problems, we specify parameters, such as wave velocity, source location, or source time functions, in forward simulations to generate predicted seismic signals. In ADSeismic, the gradients of the observational differences over these parameters can be computed automatically and thus used in a gradient-based optimizer in the same way as when training neural networks.

8.2.1 Relationship to the Adjoint Method

The adjoint-state method is an efficient technique for computing the gradient of the misfit function with respect to the physical parameters of interest. For example, the adjoint-state method is commonly used to compute the gradient in full-waveform inversion (Plessix, 2006). To clarify the connection between the adjoint-state method and reverse-mode automatic differentiation, we provide a derivation based on Lagrange multipliers.

Consider the explicit discretization of the wave equation, which can be written as

$$\begin{aligned} U_1 &= A(\theta)U_0 + F_0 \\ U_2 &= A(\theta)U_1 + F_1 \\ &\vdots \\ U_{n-1} &= A(\theta)U_{n-2} + F_{n-2} \\ U_n &= A(\theta)U_{n-1} + F_{n-1} \end{aligned} \tag{8.1}$$

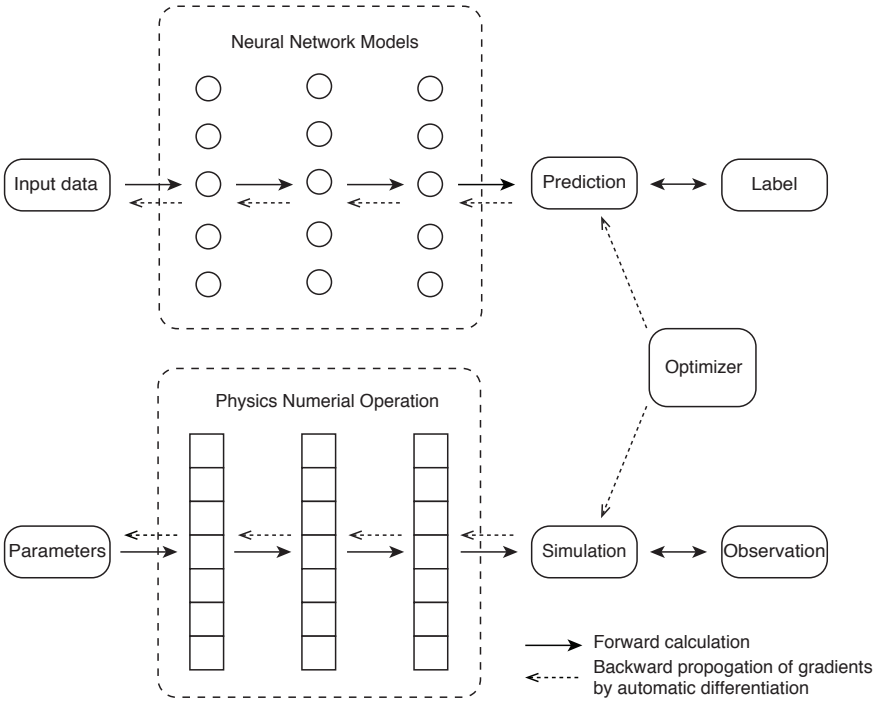


Figure 8.1: The similarity between neural networks and PDE-based physical simulation

where U_k is the seismic wavefield at the k-th time step, F_k is the source term at the k-th time step, $A(\theta)$ is the associated coefficient matrix, and θ is the physical parameter of interest, e.g., the wave velocity. $A(\theta)$ indicates that the entries in the matrix depend on θ. To simplify the notation, we let the misfit function be

$$J(\theta) = \frac{1}{2}\sum_{k=1}^{n} \|U_k(\theta) - U_k^{\text{obs}}\|^2$$

where U_k^{obs} is the observation at the k-th step. The corresponding Lagrangian functional is

$$L(\theta, U_1, \ldots, U_n) = \frac{1}{2}\sum_{k=1}^{n} ||U_k - U_k^{\text{obs}}||^2 + \sum_{k=1}^{n} {\lambda_k}^T \left(A(\theta)U_{k-1} + F_{k-1} - U_k\right) \tag{8.2}$$

where λ_k is the adjoint variable. The Karush–Kuhn–Tucker (KKT) condition (Luenberger & Ye, 1984) for Eq. (8.2) reads

$$\begin{aligned}
\frac{\partial L}{\partial U_n} &= U_n - U_n^{\text{obs}} - \lambda_n = 0 \\
\frac{\partial L}{\partial U_{n-1}} &= U_{n-1} - U_{n-1}^{\text{obs}} - \lambda_{n-1} + A(\theta)^T\lambda_n = 0 \\
&\vdots \\
\frac{\partial L}{\partial U_2} &= U_2 - U_2^{\text{obs}} - \lambda_2 + A(\theta)^T\lambda_3 = 0 \\
\frac{\partial L}{\partial U_1} &= U_1 - U_1^{\text{obs}} - \lambda_1 + A(\theta)^T\lambda_2 = 0
\end{aligned} \tag{8.3}$$

Rearranging Eq. (8.3) we obtain

$$\begin{aligned}
\lambda_n &= U_n - U_n^{\text{obs}} \\
\lambda_{n-1} &= A(\theta)^T\lambda_n + U_{n-1} - U_{n-1}^{\text{obs}} \\
&\vdots \\
\lambda_2 &= A(\theta)^T\lambda_3 + U_2 - U_2^{\text{obs}} \\
\lambda_1 &= A(\theta)^T\lambda_2 + U_1 - U_1^{\text{obs}}
\end{aligned} \tag{8.4}$$

Note that we can compute all the adjoint variables $\lambda_k, k = 1, 2, \ldots, n$ sequentially from $k = n$ to $k = 1$. In this process, we need to perform matrix multiplication with the coefficient matrix $A(\theta)^T$, which is why we call λ_k adjoint variables.

Finally, the gradients of L with respect to θ can be extracted using the computed $\lambda_k, k = 1, 2, \ldots, n$

$$\boxed{\frac{\partial L}{\partial \theta} = \sum_{k=1}^{n} {\lambda_k}^T \frac{\partial A(\theta)}{\partial \theta} U_{k-1}} \tag{8.5}$$

In the following text, we describe how reverse-mode AD is used for computing the gradient $\frac{\partial J}{\partial \theta}$ and show that AD calculates the adjoint variables and gradients in the same way as the adjoint-state method (Eq. (8.4) and (8.5)). A straightforward way to view AD is to consider a specific operator in the computational graph from $k-1$ to k step:

$$\begin{aligned} \text{Forward Computation:} \quad & U_k(U_{k-1},\theta) = A(\theta)U_{k-1} + F_{k-1} \\ \text{Backward Gradient:} \quad & \frac{\partial U_k(U_{k-1},\theta)}{\partial U_{k-1}} = A(\theta)^T \\ & \frac{\partial U_k(U_{k-1},\theta)}{\partial \theta} = \frac{\partial A(\theta)}{\partial \theta} U_{k-1} \end{aligned} \tag{8.6}$$

We assume that the gradient of J with respect to U_k has already been calculated at the k-th time step. We then back-propagate the gradients to the previous time step (Fig. 8.2). For convenience we define

$$\begin{aligned} \mu_k &:= \left(\frac{\partial J(\theta, U_1, \ldots, U_k)}{\partial U_k} \right)^T \quad k = 1, 2, \ldots, n-1 \\ \mu_n &= \left(U_n - U_n^{\text{obs}} \right)^T \end{aligned} \tag{8.7}$$

Here, $J(\theta, U_1, \ldots, U_k)$ can be recursively defined as

$$J(\theta, U_1, \ldots, U_n) := \frac{1}{2} \sum_{k=1}^{n} \| U_k - U_k^{\text{obs}} \|^2 \tag{8.8}$$

$$J(\theta, U_1, \ldots, U_k) := J(\theta, U_1, \ldots, U_k, A(\theta)U_k + F_k) \qquad k = 1, 2, \ldots, n-1 \tag{8.9}$$

where we define $J(\theta, U_1, \ldots, U_k)$ by substituting U_{k+1} in $J(\theta, U_1, \ldots, U_k, U_{k+1})$ with $A(\theta)U_k + F_k$.

We now focus on one specific step shown in bold in Fig. 8.2. In AD, we need to compute the gradients $\frac{\partial J}{\partial U_{k-1}}$ and $\frac{\partial J}{\partial \theta}$ given the so-called "top" gradients $\frac{\partial J}{\partial U_k}$ (noted by the symbol "b" in Fig. 8.2). The gradient backpropagation rule for $\frac{\partial J}{\partial U_{k-1}}$ reads

$$\begin{aligned} \mu_{k-1}^T = \frac{\partial J(\theta, U_1, \ldots, U_{k-1})}{\partial U_{k-1}} &= \underbrace{\overbrace{\frac{\partial J(\theta, U_1, \ldots, U_k)}{\partial U_k}}^{(b)} \frac{\partial U_k(U_{k-1}, \theta)}{\partial U_{k-1}}}_{(a)} + \overbrace{\frac{\partial J(\theta, U_1, \ldots, U_n)}{\partial U_{k-1}}}^{(c)} \\ &= A(\theta)^T \mu_k + \left(U_{k-1} - U_{k-1}^{\text{obs}} \right) \quad k = 2, \ldots, n \end{aligned} \tag{8.10}$$

Note J on the left hand side and on the right hand side have different arguments (Fig. 8.2).

The gradient back-propagation rule for $\frac{\partial J(\theta,U_1,U_2,\ldots,U_{k-1})}{\partial \theta}$ reads

$$g_{k-1} := \overbrace{\frac{\partial J(\theta,U_1,\ldots,U_{k-1})}{\partial \theta}}^{(d)} = \overbrace{\frac{\partial J(\theta,U_1,\ldots,U_k)}{\partial U_k}}^{(b)} \frac{\partial U_k(U_{k-1},\theta)}{\partial \theta} = \mu_k^T \frac{\partial A(\theta)}{\partial \theta} U_{k-1} \tag{8.11}$$

The gradient $\frac{\partial J(\theta)}{\partial \theta}$ is computed by accumulating g_k from all steps

$$\boxed{\frac{\partial J(\theta)}{\partial \theta} = \sum_{k=1}^{n} g_k = \sum_{k=1}^{n} \mu_k^T \frac{\partial A(\theta)}{\partial \theta} U_{k-1}} \tag{8.12}$$

We now demonstrate the equivalence of the gradients (Eq. (8.5)) computed using AD and the gradients (Eq. (8.12)) computed using the adjoint-state method.

Proposition 1. *Assume that* $\{\mu_k\}_{k=1}^n$ *satisfies Eq.* (8.7) *and Eq.* (8.10)*, and* $\{\lambda_k\}_{k=1}^n$ *satisfies Eq.* (8.3)*, then*

$$\lambda_k = \mu_k \quad k = 1, 2, \ldots, n \tag{8.13}$$

And therefore,

$$\sum_{k=1}^{n} \mu_k^T \frac{\partial A(\theta)}{\partial \theta} U_{k-1} = \sum_{k=1}^{n} \lambda_k^T \frac{\partial A(\theta)}{\partial \theta} U_{k-1} \tag{8.14}$$

Proof. Note $\lambda_n = \mu_n = U_n - U_n^{\text{obs}}$ and the recursive relations Eq. (8.4) and Eq. (8.10) are the same. Thus, we have $\lambda_k = \mu_k, \quad k = 1, 2, \ldots, n$. Therefore, Eq. (8.14) holds. □

Proposition 1 implies that reverse-mode automatic differentiation is mathematically equivalent to the adjoint-state method, and the intermediate gradient $\mu_k = \frac{\partial J}{\partial U_k}$ is exactly the adjoint variable λ_k. In the following text, we describe our general approach for seismic inversion based on the connection between the automatic differentiation and the adjoint state method.

8.2.2 Implementation

In this section we describe how automatic differentiation assists computing the gradient of the misfit function with respect to the physical parameters in ADSeismic. We use a staggered grid finite difference method for discretizing both the acoustic wave equation and the elastic wave equation with perfectly matched layer (PML) (Grote & Sim, 2010; Komatitsch & Martin, 2007; Roden & Gedney, 2000). The governing equation for the acoustic wave equation is

$$\frac{\partial^2 u}{\partial t^2} = \nabla \cdot (c^2 \nabla u) + f \tag{8.15}$$

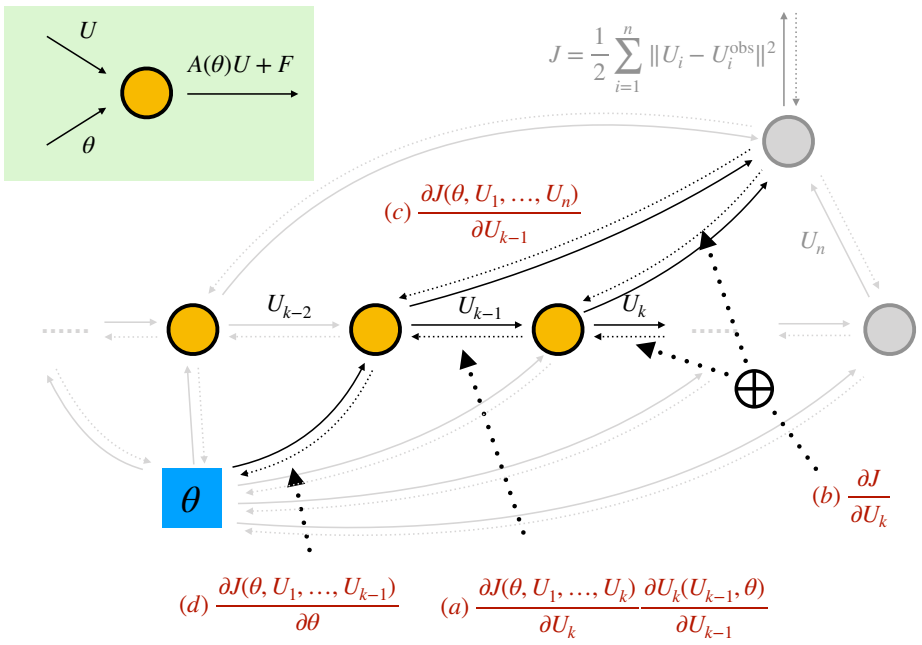


Figure 8.2: Computational graph and gradient back-propagation by automatic differentiation

where u is displacement, f is the source term, and c is the spatially varying acoustic velocity. The inversion parameters of interest are c or f. The governing equation for the elastic wave equation is

$$\begin{aligned} \rho \frac{\partial v_i}{\partial t} &= \sigma_{ij,j} + \rho f_i \\ \frac{\partial \sigma_{ij}}{\partial t} &= \lambda v_{k,k} + \mu(v_{i,j} + v_{j,i}) \end{aligned} \tag{8.16}$$

where v is velocity, σ is the stress tensor, ρ is density, and λ and μ are the Lamé's constants. The inversion parameters in the elastic wave equation case are λ, μ, ρ or f.

The finite difference discretization leads to a system of linear equations Eq. (8.1) for both Eq. (8.15) and Eq. (8.16). For the adjoint-state method, we also need to derive and implement Eq. (8.3) to compute the gradient Eq. (8.5). This step is unnecessary in ADSeismic since the gradient is extracted automatically from the computational graph. We emphasize that only the forward simulation code is required for building a computational graph and the gradient automatically computed by AD is the same as that for the adjoint-state method.

We use the Julia package, ADCME[1], for our implementation since it provides an interface to TensorFlow for automatic differentiation and intuitive Julia syntax for expressing mathematical formulae in numerical simulation. ADCME also provides built-in optimization solvers such as L-BFGS-B (Zhu et al., 1997) for minimizing the misfit function. ADCME allows us to easily extend

[1] https://github.com/kailaix/ADCME.jl

ADSeismic to other equations or models in seismic applications.

8.3 Applications

In this section, we first benchmark the performance of ADSeismic on CPU, GPU, and computer clusters for acoustic and elastic wave equations. We then present three applications of ADSeismic to seismic problems including: velocity model estimation, earthquake location, source time function estimation, and earthquake rupture imaging. The applications are built with the same forward simulation code (acoustic or elastic wave equations) with only minor changes to specify the inversion parameters to be recovered.

8.3.1 Performance Benchmarking

Because the backend of ADSeismic is TensorFlow, the same forward simulation code runs on both the CPU and GPU. We benchmark the performance of ADSeismic on the Intel(R) Xeon(R) CPU E5-2698 and the Tesla V100-SXM2 GPU. For comparison, we also report the speed of a dedicated FORTRAN package SEISMIC_CPML[2] for simulating seismic waves. The comparisons between the CPU and GPU times of ADSesmic and the CPU time of the FORTRAN package for the acoustic and elastic equations are shown in Fig. 8.3a and b. The computation times are averaged over three repeated runs. Because FORTRAN is well optimized for scientific computing, such as vectorization for array-based computing (Loveman, 1993), the FORTRAN package runs 3-5 times faster than ADSeismic on CPUs. With GPU acceleration, ADSeismic achieves similar performance to the FORTRAN code.

ADSeismic also inherits the multi-GPU support from TensorFlow such that we can split the source onto different GPUs so that the forward simulation and the associated gradient are computed using AD in parallel across the GPUs. Then, the gradients are assembled on the CPU and fed to the L-BFGS optimizer to update the inversion parameters. Finally, the updated inversion parameters are distributed to all GPUs for the next integration (Fig. 8.3e).

To extend ADSeismic's applications on large-scale supercomputers, we add MPI into ADSeismic to support distributed and parallel computing. For seismic wave simulation, we can divide the computational domain into several sub-domains and assign the computation of each sub-domain to one of the CPU processors. MPI is used to exchange the updated wavefield values between adjacent sub-domain at each time step. In this way we can both speed up the computation and utilize the massive memory space of supercomputers for large-scale wavefield simulation and parameter inversion. Fig. 8.3c and d show the speedup with the increasing number of CPU processors (Intel(R) Xeon(R) CPU E5-2670).

[2] https://github.com/geodynamics/seismic_cpml

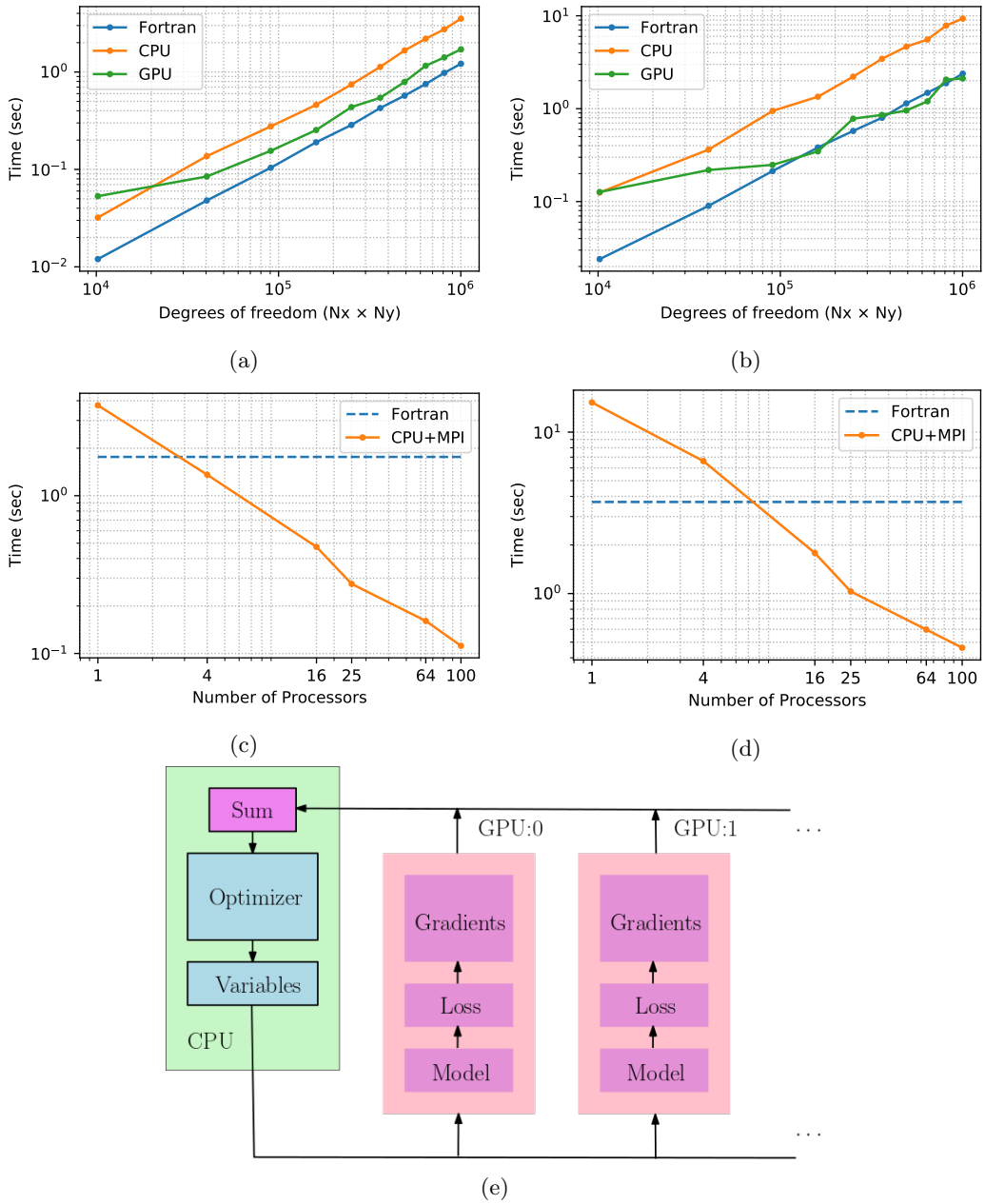


Figure 8.3: High-performance computing of ADSeismic: We benchmark the computation times of ADSeismic on CPU and GPU and against the FORTRAN package SEISMIC_CPML on CPU for the acoustic wave equation (a) and the elastic wave equation (b). The computation time of ADSeismic can be further reduced with more CPU processors for the acoustic wave equation (c) and the elastic wave equation (d), because ADSeismic supports MPI for distributed and parallel computing. ADSeismic also inherits multi-GPU computing from Tensorflow (e).

8.3.2 Full-waveform Inversion

Classic full-waveform inversion (FWI) is based on the adjoint-state method (Fichtner et al., 2006; Plessix, 2006; Tarantola, 1984; Virieux & Operto, 2009). As shown above, AD is mathematically equivalent to the adjoint-state method so that we can apply AD directly to full-waveform inversion without manual derivation of the adjoint-state equations. We demonstrate our method using two cases: the well-known and geometrically complex Marmousi benchmark model (Martin et al., 2002; Versteeg, 1994) (Fig. 8.4) and a layered model of the Earth's crust with embedded anomalies of elliptical shape (Fig. 8.5). We place eight active sources on the surface with a spacing of 850 m for the Marmousi benchmark and seven plane waves with incident angles from -45^o to 45^o from the bottom to mimic incoming teleseismic waves for the layered model. We use a Ricker wavelet as the source time function for both cases. Similar to common FWI applications, we choose L-BFGS optimization and a $L2-$norm loss function for all inversions. We note that ADSeismic supports other optimization techniques such as the stochastic gradient descent (SGD) method (Bottou, 2010; Richardson, 2018b; Witte et al., 2018) although the application and comparison of these optimizers is beyond the scope of this paper. The inversion results in Fig. 8.4c and Fig. 8.5c show good recovery of the complex velocity structure and anomalies, demonstrating that AD accurately estimates the velocity models.

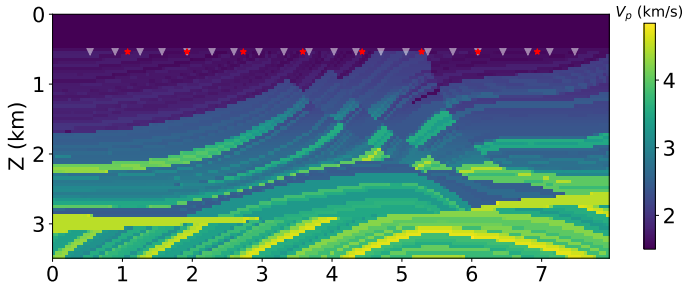
8.3.3 Earthquake Location and Source Time Function Retrieval

Determining earthquake location is a routine but essential earthquake monitoring task for which commonly used methods include 1) linearized inversion for absolute earthquake location (Kissling et al., 1995; Kissling et al., 1994; Klein, 2002; Lienert et al., 1986) and relative earthquake location (Rubinstein & Beroza, 2007; Schaff et al., 2004; Waldhauser & Ellsworth, 2000); 2) non-linear inversion methods (Lomax et al., 2009; Lomax et al., 2000; Thurber, 1985); and 3) migration-based or time-reversal methods (Nakata & Beroza, 2016; Nakata et al., 2016). The migration-based method produces a focused wavefield that is the same as the gradient in the first iteration of the adjoint-state method (Fichtner, 2010); however, this method does not explicitly give the source location but requires post-processing to extract potential earthquake locations from the focused wavefield.

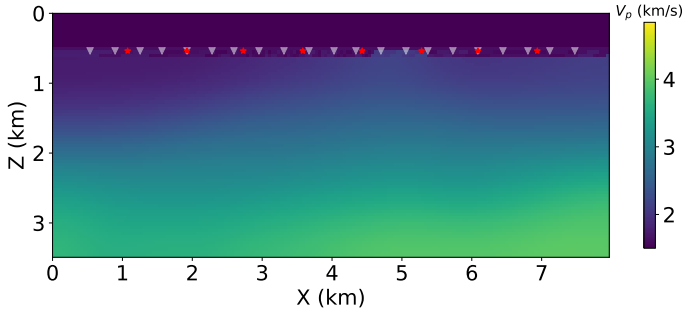
We use a new non-linear earthquake location method based on full waveforms. The inversion target, the source term $f(x,t)$ in equation (8.15), is a delta function in space, whose gradient at zero is not well defined, making the direct application of the adjoint-state method difficult. With AD, we can flexibly re-parameterize the optimization target $f(x,t)$ with a continuous Gaussian form

$$f(x,t) = \frac{g(t)}{2\pi\sigma^2} \exp\left(-\frac{||x-x_0||^2}{2\sigma^2}\right) \tag{8.17}$$

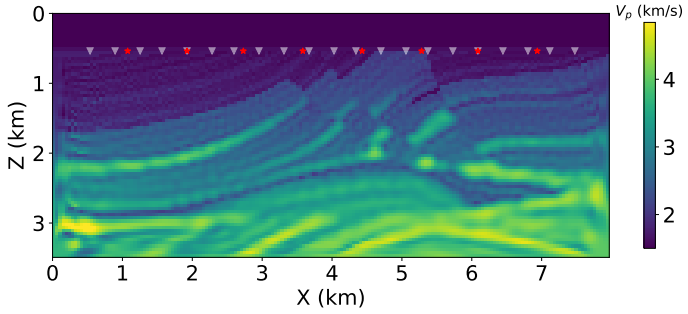
where $g(t)$ is the source time function, x_0 is the earthquake location, and σ is the standard deviation of the Gaussian function, which in our test is set to half of the grid size. In this test,



(a)

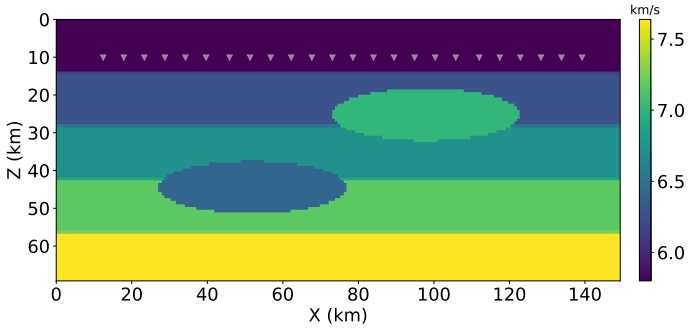


(b)

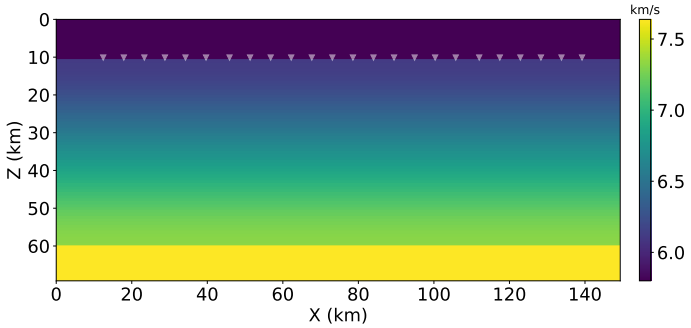


(c)

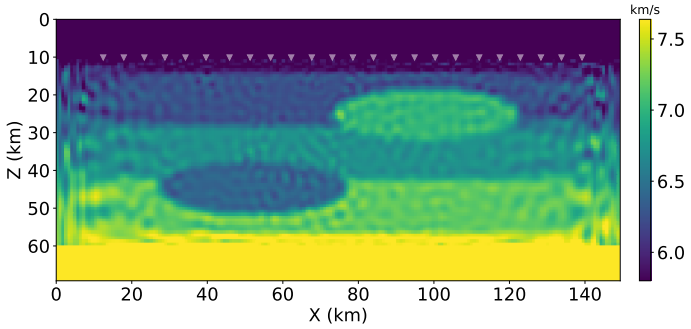
Figure 8.4: The Marmousi benchmark model: (a) the true P-wave velocity model; (b) the initial velocity model; (c) the inverted velocity model. The white triangles at the top represent the receiver locations, while the red stars represent the source locations.



(a)



(b)



(c)

Figure 8.5: The layered model with inclusions: (a) the true P-wave velocity model; (b) the initial velocity model; (c) the estimated velocity model. Here we use seven plane waves propagating from the bottom to the surface with incidence angles ranging from -45^o to 45^o and an interval of 15^o.

we simultaneously estimate the earthquake location x_0 and the source time function $g(t)$ by fitting the recorded waveforms. Fig. 8.6 shows the evolution of the earthquake location and source time function during optimization from an initial state of a random selected earthquake location and a flat source time function. The inversion results agree well with the true earthquake location and source time function.

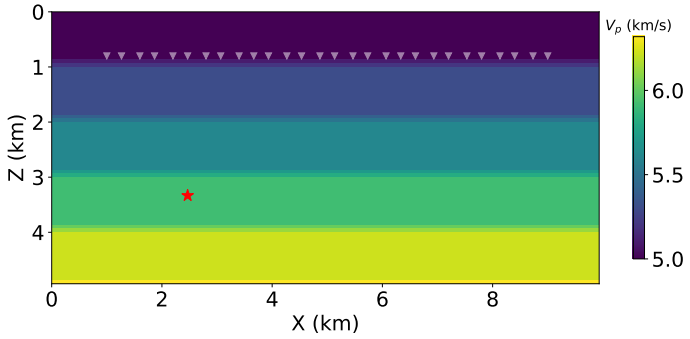
8.3.4 Earthquake Rupture Imaging

The rupture process of large earthquakes has resolvable spatial and temporal extent. Imaging this rupture process from observed seismic data contributes to the understanding the complexity behind the evolution of earthquakes. The linearized kinematic inversion method using elastodynamic Green's functions (Beroza, 1991; Beroza & Spudich, 1988; Hartzell & Heaton, 1983; Kikuchi & Kanamori, 1982; Suzuki et al., 2011; Wald et al., 1990; Zhang et al., 2009) and direct imaging methods, such as back-projection (Ishii et al., 2005; Krüger & Ohrnberger, 2005; Lay et al., 2010; Meng et al., 2012; Simons et al., 2011; Walker et al., 2005; Xu et al., 2009), are the two most commonly used methods for imaging the earthquake rupture process. The adjoint-state method has also been tested for rupture process inversion (Kremers et al., 2011; Somala et al., 2018).

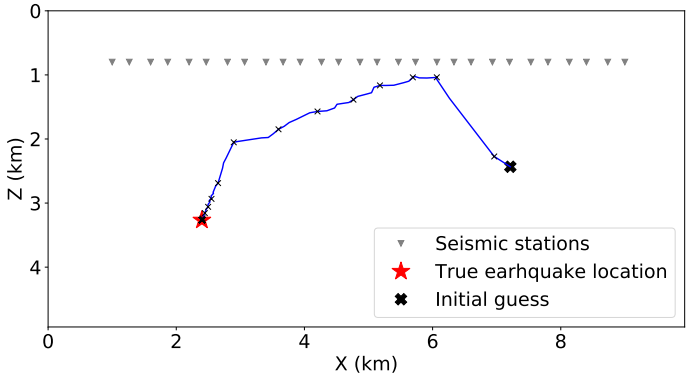
We consider a simplified 2D earthquake rupture case to show the potential applications of ADSeismic for imaging the earthquake rupture process. We mimic a simple rupture process with a group of sources activated from the left to right with different rise times and amplitudes (Fig. 8.7a and b). We consider two inversion targets: the entire rupture history, or the rupture time and amplitude. To estimate the rupture history, we choose the unknown parameter as the source time function $f(t)$. To estimate the rupture time and amplitude, we choose the parameters of rupture time t_0 and amplitude A_0 by assuming that the shape of the source time function is known as a Gaussian function:

$$f(t) = A_0 \exp\left(-\frac{(t-t_0)^2}{2\sigma^2}\right) \tag{8.18}$$

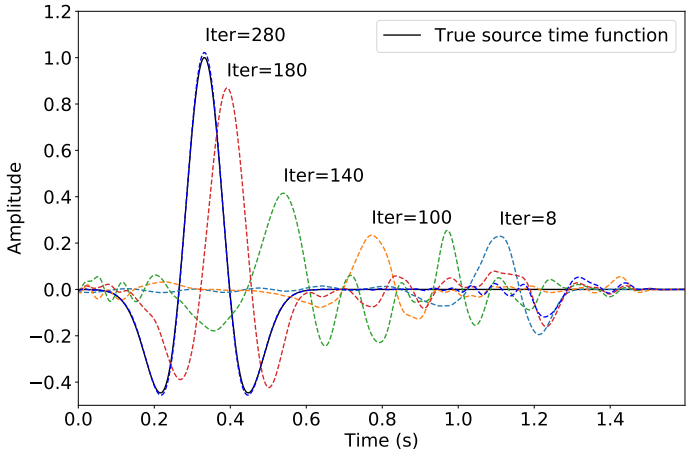
To estimate the entire rupture history, the initial state is set to be zero slip for all locations. To estimate the rupture time and amplitude, the initial state is set to be the same Gaussian function with a constant rupture time and amplitude (Fig. 8.8b). Imaging the entire rupture history requires many more parameters (the total number of time steps) than estimating only the rupture time and amplitude (two parameters A_0 and t_0), with the result that the former problem is less constrained for the same number of receivers. The final inversion results are shown in Fig. 8.7c and 8.8c. Note that we have not incorporated a dynamic rupture model to simulate the rupture propagation in this test. Although deriving the adjoint equation is challenging for the coupled system between dynamic earthquake rupture and seismic waves, AD provides a solution to back-propagate the gradients from the wave equation into the dynamic rupture equation to enable the inversion of fault properties based on seismic wave recordings. The potential applications of AD for complex coupled systems in



(a)



(b)



(c)

Figure 8.6: Inversion of earthquake location and source time function: (a) the velocity model and true source location; (b) the evolution of earthquake location represented by the black ×; (c) the evolution of the source time function from a flat initial state.

geophysics is an obvious direction for future research.

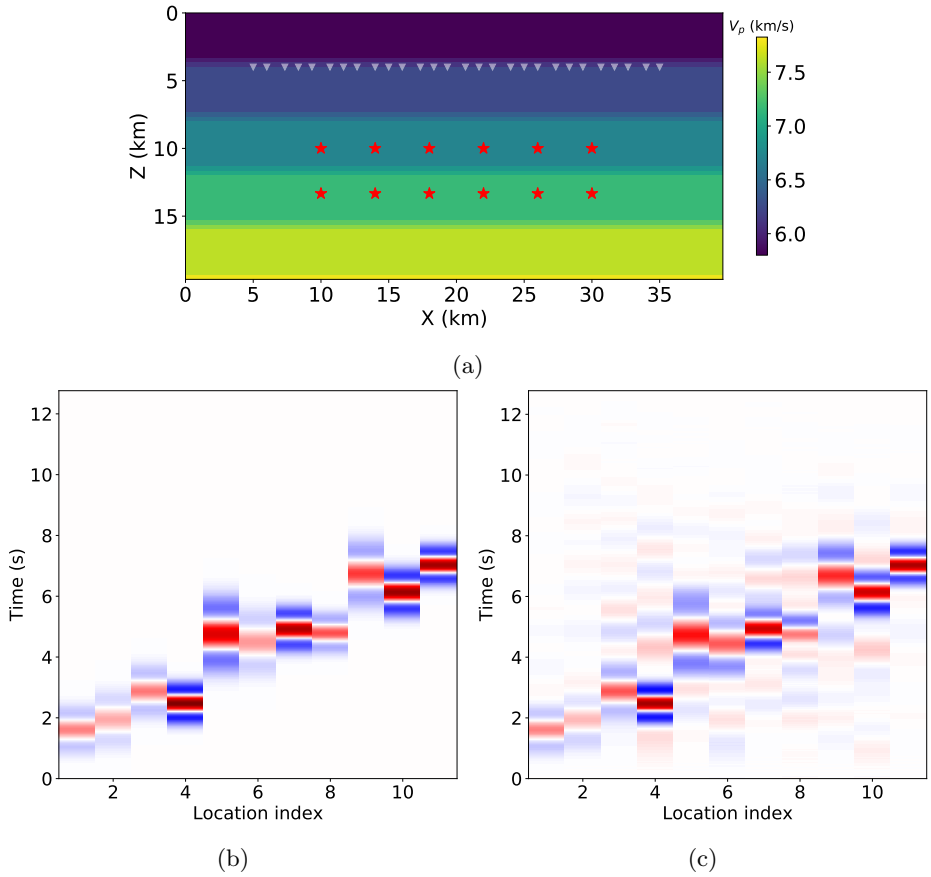
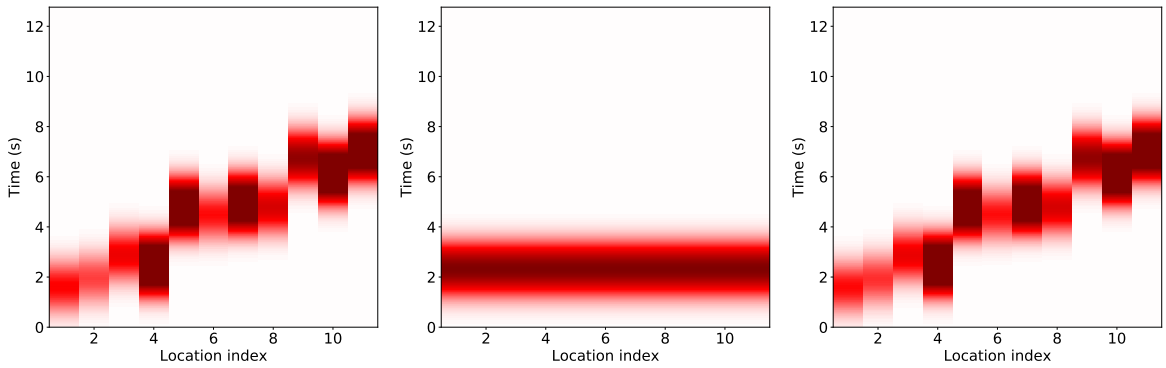


Figure 8.7: Inversion of the whole rupture history: (a) the velocity model, receivers (white triangles), and simplified rupture locations (red stars); (b) true slip waveforms; (c) inverted slip waveforms.

8.4 Limitations

Despite the many strengths of ADSeismic, we note three major limitations:

First, as with any inverse problem, inversion based on AD may suffer from ill-conditioning, among these are the often-encountered problem in seismology of cycle-skipping (Hu et al., 2018; Virieux & Operto, 2009), which produces a local minimum when the predicted signal is shifted more than half a wavelength from the observation due to a poor initial model or lack of low frequency information. Neither AD nor adjoint-state methods can solve the ill-conditioning issue, which is intrinsic to the optimization problem. Nevertheless, there are many techniques for improving the conditioning of the optimization problem (Biondi & Almomin, 2014; Ma & Hale, 2013; Wu et al., 2014; Yang et al., 2018) and these can be applied in our AD framework.



(a) (b) (c)

Figure 8.8: Inversion of slip time and amplitude: (a) true earthquake slip with the shape of a Gaussian; (b) initial inversion state with a same slip time and amplitude; (b) estimated earthquake slip.

Second, reverse-mode AD has demanding memory requirements. This is a noteworthy constraint when running large simulations on GPUs because GPUs usually have lower available memory than CPUs. Techniques such as check-pointing (Chen, Xu, et al., 2016; Griewank & Walther, 2000; Symes, 2007) have been used to reduce memory requirements, but at the cost of more computation. In ADSeismic, we add MPI support to use the large amount of memory of supercomputers for large-scale seismic problems.

Third, the numerical schemes we consider in this work are all explicit. In some applications, implicit schemes are desirable for reasons such as stability, accuracy, and non-linearity (Chu & Stoffa, 2012; Liu & Sen, 2009; Richardson, 2018b). For implicit schemes it is challenging to apply reverse-mode AD techniques since most AD frameworks only provide explicit differentiable operators. Li, Xu, Harris, et al. (2019) introduce the intelligent automatic differentiation method that implements AD for implicit numerical schemes. This approach could be used for extending ADSeismic to implicit schemes.

8.5 Conclusions

In this study we have explained the connection between the automatic differentiation technique in deep learning and adjoint-state methods in seismic inversion. Based on that correspondence we design a general seismic inversion framework, ADSeismic, based on the AD functionality from deep learning software. ADSeismic shows promising results on a series of seismic inversion problems with demonstrated acceleration on GPUs and large-scale CPU clusters. Since deep learning techniques and frameworks are continuously improving, ADSeismic allows for flexibly experimenting with new models, leverages specialized hardware optimized for deep learning, and executes numerical

simulations on heterogeneous computing platforms. This should facilitate general seismic inversion in a high performance computing environment. Furthermore, it opens a pathway for innovation in inverse modeling in geophysics by leveraging AD functionalities in a deep learning framework.

Chapter 9

NNFWI: Integrating Deep Neural Networks with Full-waveform Inversion: Reparametrization, Regularization, and Uncertainty Quantification

Full-waveform inversion (FWI) is an accurate imaging approach for modeling velocity structure by minimizing the misfit between recorded and predicted seismic waveforms. However, the strong non-linearity of FWI resulting from fitting oscillatory waveforms can trap the optimization in local minima. We propose a neural-network-based full waveform inversion method (NNFWI) that integrates deep neural networks with FWI by representing the velocity model with a generative neural network. Neural networks can naturally introduce spatial correlations as regularization to the generated velocity model, which suppresses noise in the gradients and mitigates local minima. The velocity model generated by neural networks is input to the same partial differential equation (PDE) solvers used in conventional FWI. The gradients of both the neural networks and PDEs are calculated using automatic differentiation, which back-propagates gradients through the acoustic PDEs and neural network layers to update the weights of the generative neural network. Experiments on 1D velocity models, the Marmousi model, and the 2004 BP model demonstrate that NNFWI can mitigate local minima, especially for imaging high-contrast features like salt bodies, and significantly improves the inversion in the presence of noise. Adding dropout layers to the neural network model also allows analyzing the uncertainty of the inversion results through Monte Carlo dropout. NNFWI

opens a new pathway to combine deep learning and FWI for exploiting both the characteristics of deep neural networks and the high accuracy of PDE solvers. Because NNFWI does not require extra training data and optimization loops, it provides an attractive and straightforward alternative to conventional FWI.

9.1 Introduction

Full-waveform inversion (FWI) is a high resolution inversion method in exploration seismology (Tarantola, 1984, 2005; Virieux & Operto, 2009), commonly used for estimating subsurface velocity structure based on seismic waves recorded at the surface. FWI falls within the class of PDE-constrained optimization problems. It solves the wave equation, in the acoustic or elastic approximation, to predict seismic waves based on the velocity model. Through the PDE solver, FWI determines the optimal velocity model by minimizing the misfit between predicted and observed seismic waveforms. The gradient of the misfit function can be efficiently calculated by the adjoint method (Plessix, 2006). Although FWI can achieve high accuracy when the full seismic waveform is matched, the non-linearity of the objective function poses a challenge to the optimization process. The inversion result of FWI suffers from local minima due to cycle skipping in the wave oscillations used to form the objective function, i.e., a $\mathcal{L}_2$-norm loss. This is particularly challenging when either a good initial model is lacking or low frequency content of the waveform data is missing. Noise in real seismic recording can also contaminate the inversion results. Because artifacts originating from local minima interfere with imaging results and can lead to misinterpretation of geological structures, a great deal of research has focused on improving the stability of FWI. One solution is to recover or predict the missing low frequency content, such as through envelope inversion (Bozdağ et al., 2011; Wu et al., 2014), sparse blind deconvolution (Zhang et al., 2017), and phase-tracking methods (Li & Demanet, 2016). Another solution is to add regularization and preconditioning, such as Laplacian smoothing (Burstedde & Ghattas, 2009), l_2-norm penalty (Hu et al., 2009), l_1-norm penalty (total variation) (Esser et al., 2018; Guitton, 2012; Kalita et al., 2019), and prior information as constraints (Asnaashari et al., 2013). Many other solutions have also been proposed and tested such as: multiscale inversion (Bunks et al., 1995), wave-equation traveltime inversion (Luo & Schuster, 1991), tomographic full-waveform inversion (Biondi & Almomin, 2014), model extension (Barnier et al., 2018), model reduction (Barnier et al., 2019), model reparameterization (Guitton et al., 2012), and dictionary learning (Li & Harris, 2018; Zhu et al., 2017),. In addition to the strong non-linearity of FWI, uncertainty analysis of FWI results is challenging due both to the high dimensionality of the model space and to the demanding computational cost of solving the wave equation (Gebraad et al., 2020).

In recent studies, the success of deep learning in computer vision, natural language processing, and many other fields has drawn attention to its potential application in FWI (Adler et al., 2021).

One research direction is to build a direct inverse mapping from observations to subsurface structure by training neural networks on paired data of seismic waveforms and velocity models (Kazei et al., 2021; Li, Liu, et al., 2019; Wu, Lin, & Zhou, 2018; Yang & Ma, 2019). This approach does not rely on solving the wave equation but instead treats FWI as a data-driven machine learning problem similar to that in image recognition. The accuracy and generalization of this approach, however, cannot be guaranteed without the PDE constraint in FWI. Another research direction is to apply deep learning as an effective signal processing tool to improve the optimization process of conventional FWI. For example, several studies applied neural networks to extrapolate the missing low frequencies and help mitigate the cycle skipping problem (Hu et al., 2021; Ovcharenko et al., 2019; Sun & Demanet, 2020). In addition to these data-driven approaches, another promising direction is to combine neural networks and PDEs to formulate FWI as a physics-constrained machine learning problem. Richardson (2018a) and Mosser et al. (2020) trained a generative adversarial network (GAN) to build an a prior model of subsurface geological structures and optimized a lower-dimensional latent variable to fit observed data. Wu and McMechan (2019) and Wu and McMechan (2020) proposed a CNN-domain FWI, which reparameterizes the velocity model or the gradient field by a convolutional neural network (CNN) and minimizes the loss by updating the neural network weights. He and Wang (2021) further analyzed the adaptive regularization effect from the convolutional neural network. However, these works relied on pre-training convolutional neural networks on initial velocity models and attributed the regularization effect to the prior information coming from fitting these initial velocity models. In contrast, Ulyanov et al. (2018)'s work on deep image prior demonstrated that the CNN architecture without pre-training can be used as a prior with excellent results in inverse problems of computer vision, such as denoising, super-resolution, and inpainting. Their results showed that although a high-capacity neural network can fit both structured objects and unstructured noise, the parametrization of the neural network offers high impedance to noise during optimization and learns much more quickly towards natural-looking images. Another limitation of the CNN-domain FWI approach is the complex inversion workflow combining two optimization processes of neural network training and full-waveform inversion. Richardson (2018b) and Zhu et al. (2020) have implemented FWI using deep learning frameworks so that reverse-mode automatic differentiation and various effective optimization tools in deep learning frameworks can be used for FWI. The similarity between neural network training and FWI demonstrated by these works makes it possible to greatly simplify the previous inversion workflows within one unified framework for both deep learning and FWI.

In this study, we propose a method, NNFWI, to integrate deep neural networks with full-waveform inversion. Similar to Wu and McMechan (2019)'s idea, we use deep neural networks to generate a physical velocity model, which is then fed to a PDE solver to simulate seismic waveforms. The training process of NNFWI is similar to that of conventional FWI, but with gradients calculated by automatic differentiation instead of by the adjoint-state method. Thus we can easily optimize the two systems of neural networks and PDEs together. Unlike conventional FWI, which directly

estimates the velocity model, NNFWI reparametrizes the velocity model with a generative neural network model and optimizes the neural network's weights. In contrast to previous work that learns prior information from pre-training, NNFWI does not require pre-training neural networks on the initial velocity model. As demonstrated by Ulyanov et al. (2018)'s work on the deep image prior, the inductive bias captured by deep convolutional networks is an effective image prior and can be used as regularization for tasks such as denoising and super-resolution. Due to the regularization effect of neural networks, NNFWI mitigates the effects of local minima and is robust with respect to noise in the data. Furthermore, NNFWI can model uncertainty by adding dropout layers in the neural network. Dropout not only prevents over-fitting during training deep neural networks, but can also approximate Bayesian inference to capture model uncertainty without much extra computational cost (Gal & Ghahramani, 2016). NNFWI has the potential to provide uncertainty quantification for FWI, which otherwise remains a challenging problem. NNFWI exploits the unique advantages of deep neural networks and the high accuracy and generalization of PDEs to improve inversion and uncertainty analysis in FWI.

In the following, we first describe the two components of NNFWI including a generative neural network to parametrize the inversion target (i.e., velocity model) and a acoustic PDE to predict accurate waveforms same as that used conventional FWI. We then compare the performance of NNFWI with conventional FWI and FWI with TV regularization on three benchmark models: 1D velocity profiles, the Marmousi model, and the 2004 BP model. Last, we present an uncertainty estimation method using the Monte Carlo dropout technique and analyzed the computational cost of NNFWI. Additional comparison results are provided in the appendix.

9.2 Method

FWI aims to minimize the discrepancy between the observed seismic data $u(x,t)$ and the numerical prediction $\hat{u}$, which is the solution to a wave equation (denoted as $F(\hat{u},m)=0$), such as the acoustic wave equation:

$$\frac{\partial^2 \hat{u}}{\partial t^2} = \nabla \cdot (c^2 \nabla \hat{u}) + f \tag{9.1}$$

where f is the source term. The inversion parameter here is the acoustic wave speed $m=c$, thus the PDE-constrained optimization problem for estimating an unknown field m is given by

$$\begin{aligned} \min_{m} \quad & D(\hat{u}(x,t,m),u) = \sum_{i=1}^{I}\sum_{j=1}^{J}\int_0^T ||\hat{u}(x_i,x_j,t,m) - u(x_i,x_j,t)||_2^2 dt \\ \text{s.t.} \quad & F(\hat{u},m)=0 \end{aligned}$$

where D is the misfit function, i.e., $\mathcal{L}_2$-norm loss, $\{x_i\}_{i=1}^I$ designates the discrete locations of active sources; and $\{x_j\}_{j=1}^J$ designates the discrete locations where receivers are available. The standard

FWI derives the gradient $\frac{\partial D}{\partial m}$ using the adjoint method and updates m using a gradient-based optimization method, such as L-BFGS (Figure 9.1a). There are two challenges when this method is used:

1. The inverse problem is usually ill-conditioned. One approach is to use a misfit function with a regularization term, such as Tikhonov and total variation regularizations.

2. Quantifying uncertainties of m is very challenging because the forward simulation, i.e., computing $\hat{u}$ by solving $F(\hat{u}, m) = 0$, is very expensive. Conventional methods, such as Markov Chain Monte Carlo (MCMC), usually require a large number of forward simulations, and therefore are computationally demanding.

To address these two issues, we propose Neural-Network-based Full Waveform Inversion (NNFWI). In NNFWI, we use a generative neural network to parametrize the velocity model

$$m = \mathcal{N}(z, w) \tag{9.2}$$

where $\mathcal{N}$ is a generative neural network, z is the latent variable, and w includes the weights of the neural network. Equation (9.2) introduces regularization to the velocity field by representing m with a neural network. In this work we show the regularization effect of two types of neural networks consisting of fully connected layers or convolutional layers. For neural networks of fully connected layers, the latent variable z is set to be the coordinate x, so that $\mathcal{N}(x, w)$ imposes a spatial correlation on m, i.e., $\mathcal{N}(x_1, w)$ and $\mathcal{N}(x_2, w)$ are similar if the spatial coordinates x_1 and x_2 are close. For neural networks of convolutional layers, the local convolutional kernels applied across the entire domain imposes self-similarity and spatial correlation. Ulyanov et al. (2018) demonstrated that this deep image prior of convolutional neural network architectures achieved competitive results with handcrafted self-similarity-based and dictionary-based priors, such as the total variation norm, in a variety of image processing problems, such as denoising, super-resolution, and inpainting. Thus, reparametrizing the velocity m with convolutional neural networks naturally introduces regularization to FWI. We set the latent variable z as a fixed random vector and build a generative neural network to generate a velocity model, which applies a series of convolutional layers and upsampling operations to convert the latent vector into a 2D matrix (Table 9.1). The optimization problem of NNFWI can be written as:

$$\min_{w} D(\hat{u}(x, t, \mathcal{N}(z, w)), u(x_i, t)) = \sum_{i=1}^{I} \sum_{j=1}^{J} \int_0^T ||\hat{u}(x_i, x_j, t, \mathcal{N}(z, w)) - u(x_i, x_j, t)||_2^2 dt \tag{9.3}$$

$$\text{s.t. } F(\hat{u}, \mathcal{N}(z, w)) = 0 \tag{9.4}$$

A diagram of the NNFWI approach is shown in Figure 9.1b. The data setting of NNFWI, i.e., the initial velocity model and the waveform data is same as for conventional FWI, which makes NNFWI

applicable to all conventional FWI problems. In NNFWI, we directly combine the output from the generative neural networks with the initial velocity model as the input for the PDE, so the generative neural networks are trained to predict the updates over the initial model. Because adjoint-state methods and reverse-mode automatic differentiation are mathematically equivalent, we can calculate the gradients of both the generative neural network and the PDEs by automatic differentiation, which simplifies the optimization of both neural networks and PDEs (Xu & Darve, 2020; Zhu et al., 2020). NNFWI is implemented based on ADCME[1] and ADSeismic[2], which simulate both acoustic and elastic waveform equations using the finite-difference time-domain (FDTD) method and automatically calculate gradients using the automatic differentiation method based on the deep learning framework, Tensorflow (Abadi et al., 2016). We use the Adam algorithm (Kingma & Ba, 2014) for the optimization of NNFWI. For comparison, we also conduct conventional FWI using the L-BFGS algorithm (Zhu et al., 1997).

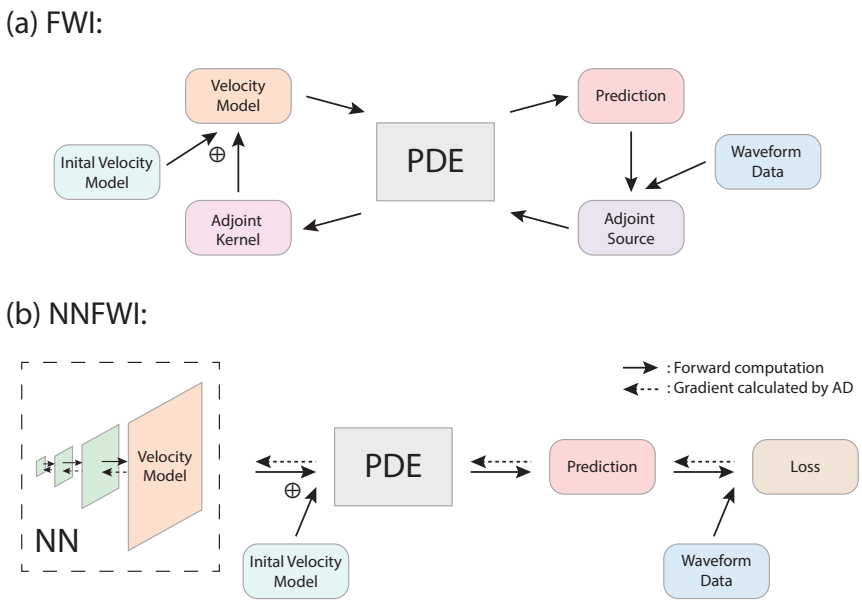


Figure 9.1: (a) Workflow of conventional FWI based on the adjoint-state method. (b) Diagram of NNFWI, which combines neural networks and PDEs of seismic waves. The input velocity to the PDEs is a direct summation of the initial velocity model and the velocity model generated by the neural network. In this way, NNFWI follows the same data format as conventional FWI and can be directly applied to conventional FWI applications. The gradients of both the neural network and PDE are calculated using automatic differentiation in NNFWI using the ADSeismic package (Zhu et al., 2020).

[1] https://github.com/kailaix/ADCME.jl
[2] https://github.com/kailaix/ADSeismic.jl

Table 9.1: The architectures of the generative neural networks in NNFWI. The fully connected neural network (FC) model is used for the 1D FWI case and the convolutional neural network (CNN) model is used for the Marmousi model and the 2004 BP models. We apply scaling factors to the output of neural networks depending on the physical parameters and units. Note that the parameters and layers in these architectures can be modified for different applications.

Model	NN layer	Architecture
FC model	Input	Spatial coordinate x
	Layer 1	Fully-connected layer (channels = 30) + Tanh
	Layer 2	Fully-connected layer (30) + Tanh
	Layer 3	Fully-connected layer (30) + Tanh
	Layer 4	Fully-connected layer (1) + Tanh
CNN model	Input	Random latent vector (8)
	Layer 1	Fully-connected layer (8) + Tanh + Reshape
	Layer 2	2×2 Upsampling + 4×4 Convolutional layer (128) + Leaky Relu (0.1) + Dropout
	Layer 3	2×2 Upsampling + 4×4 Convolutional layer (64) + Leaky Relu (0.1) + Dropout
	Layer 4	2×2 Upsampling + 4×4 Convolutional layer (32) + Leaky Relu (0.1) + Dropout
	Layer 5	2×2 Upsampling + 4×4 Convolutional layer (16) + Leaky Relu (0.1) + Dropout
	Layer 6	4×4 Convolutional layer (1) + Tanh

9.3 Results

In this section, we evaluate the performance of NNFWI by comparing the inversion results between conventional FWI and NNFWI on three cases.

9.3.1 1D model

To demonstrate the regularization effect of neural networks, we design two simple 1D velocity models: one with a linear velocity profile (Figure 9.2) and another with a step-change profile (Figure 9.3). We implement a 1D acoustic wave equation to carry out forward simulation and inversion for these examples. We inject a Ricker wavelet at $x = 0$ km and record the received waveforms (e.g. Figure 9.2b) at both sides of the domain ($x = 0$ km and $x = 1.0$ km). The whole wavefield is also plotted in the last columns of Figures 9.2 and 9.3, but only the waveforms at the two receivers are used for FWI as shown in the middle columns. Due to the limited receivers, the velocity model can not be well constrained by conventional FWI (Figures 9.2 and 9.3 (a)), even though the two received waveforms can be matched (Figures 9.2 and 9.3 (b)). Adding regularization is an effective approach to constrain the inversion results. We add Tikhonov regularization to the loss function, which produces a much smoother velocity profile (Figures 9.2 and 9.3 (d)). Note that the weight of regularization is ad-hoc. The inversion results could be improved by searching for a better weight or by adopting a complex weighting strategy, such as imposing strong initial regularization and relaxing it with further iterations. Instead of adding regularization to the loss function, we use a fully connected neural network (the FC model in Table 9.1) to reparametrize the velocity model. The neural network learns a continuous function $\mathcal{N}(x, w)$ to represent either the linear or step-change profiles $m(x)$, which implicitly imposes a spatial correlation as regularization. The results

demonstrate that the neural network first learns a smooth profile due to the regularization effect and then gradually adapts to complex structures such as the step-change.

9.3.2 Marmousi model

The Marmousi model (Versteeg, 1994) is a benchmark velocity model commonly used for evaluating FWI algorithms. The true velocity model and the 1D initial model used in this case are shown in Figure 9.4. We generated synthetic seismic waveform data using 8 active sources and a sequence of receivers on the surface marked by red stars and a white line respectively in Figure 9.4b. The source spacing is 1.2 km and the receiver spacing is 52 m. The source time function is a Ricker wavelet with a peak frequency of 2.5 Hz. We conducted three experiments by adding, respectively: no random noise, random noise with $\sigma = 0.5\sigma_0$, and random noise with $\sigma = \sigma_0$ to the synthetic data, where σ is the standard deviation of Gaussian white noise and σ_0 is the standard deviation of the synthetic seismic recordings of the entire data set. The same initial model and synthetic data were used for all experiments for both conventional FWI and NNFWI. The number of discretized grid-points of the velocity model in conventional FWI is 13,736, while the number of parameters of the generative neural network in NNFWI is 192,832. The larger number of parameters of NNFWI shows that we can over-parametrize the velocity model with the weights of a generative neural network. However, because of the deep image prior of convolutional neural networks, the inversion results of NNFWI are still robust. The free parameters of NNFWI can be adjusted by changing the architectures and hyper-parameters of the generative neural network.

Figure 9.5 shows the inversion results at different noise levels. We can observe the results of conventional FWI significantly deteriorate with the increase of noise resulting in strong spurious fluctuations in the model. In contrast, the results of NNFWI show much less deterioration. The extracted velocity profiles along depth in Figure 9.6 show that NNFWI can estimate accurate velocity profiles with both clean and noisy data. We further analyzed the inversion quality through three metrics: MSE (mean square error), PSNR (peak signal-to-noise ratio), and SSIM (structural similarity index measure) (Hore & Ziou, 2010) in Table 9.2. The metric scores confirm the improved inversion results of NNFWI on noisy data compared with conventional FWI. This comparison demonstrates the regularization effect from the generative neural network with a deep image prior (Ulyanov et al., 2018). The inversion results of NNFWI become more smooth and robust compared with those of conventional FWI in the presence of noise. Figure 9.7a and b show the change of loss functions of the two methods. Because of the regularization effect of NNFWI, its loss (labeled as "NNFWI + Adam") is higher than that of conventional FWI (labeled as "FWI + BFGS") (Figure 9.7a) and the resolutions is also a bit lower than conventional FWI (Figure 9.5a and b) for data with no random noise. However, for data with $0.5\sigma_0$ random noise, both methods converge to a similar loss (Figure 9.7b), but NNFWI gives a much better recovery of the true model (Figure 9.9c and d). The different intermediate inversion results of conventional FWI and NNFWI at 30%, 10%

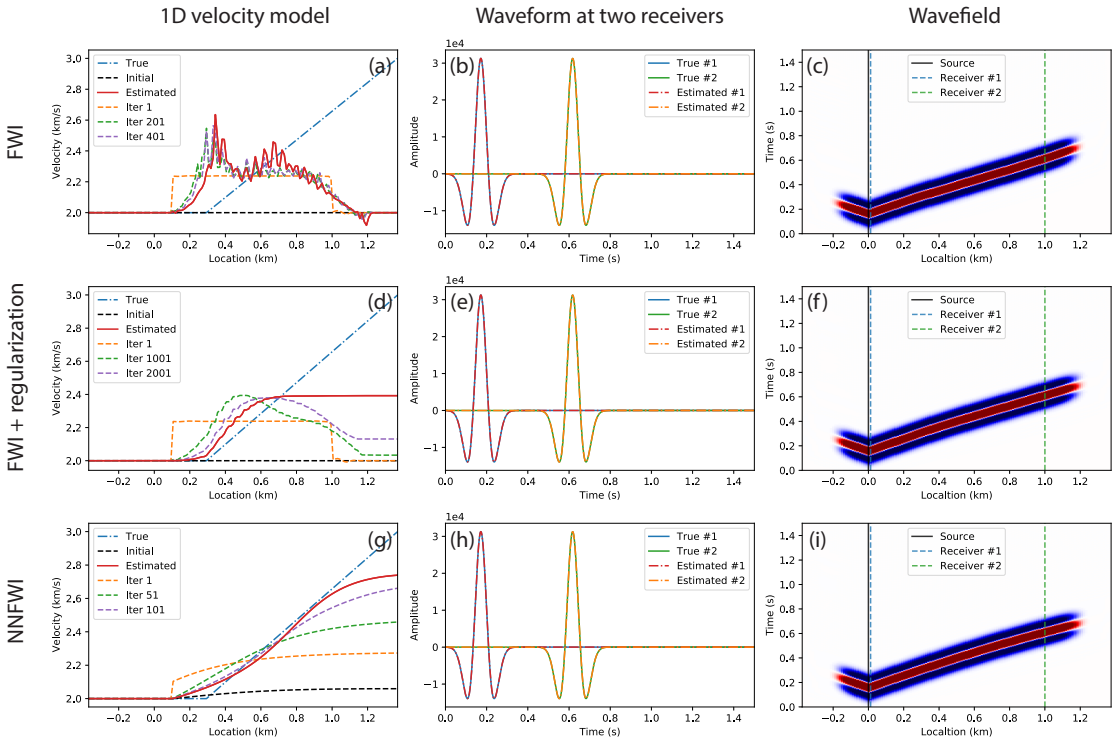


Figure 9.2: Inversion results of an 1D linear velocity profile. The upper panel (a-c) shows results of conventional FWI. The middle panel (d-f) shows results of FWI with Tikhonov regularization. The lower panel (g-i) shows results of NNFWI. The true and estimated velocity profiles are plotted in the left panel (a, d, g). We run 10,000 interactions for the final estimation. The estimated profiles at three selected iterations are also plotted to show the convergence. Note that we only estimate the velocity at $x \geq 0.1$ km to avoid extreme gradient values at the source. Because we only placed two receivers at $x = 0$ and $x = 1.0$, the updates of the first few iterations of FWI mainly occurred in the area between the two receivers, causing a box-like perturbation in the first iteration of FWI in panel (a). NNFWI, however, imposes a spatial regularization to update the whole region, as shown in panel (g). The fitted waveforms are plotted in the middle panel (b, e, h). The whole wavefields are plotted in the right panel (c, f, i).

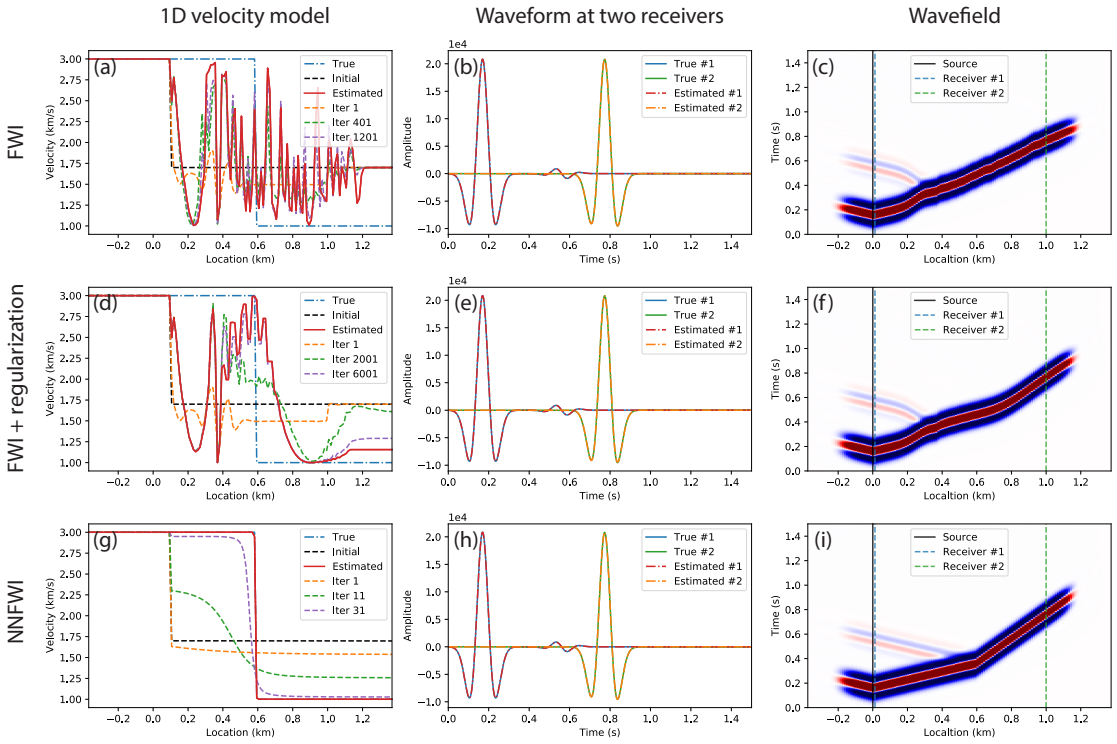
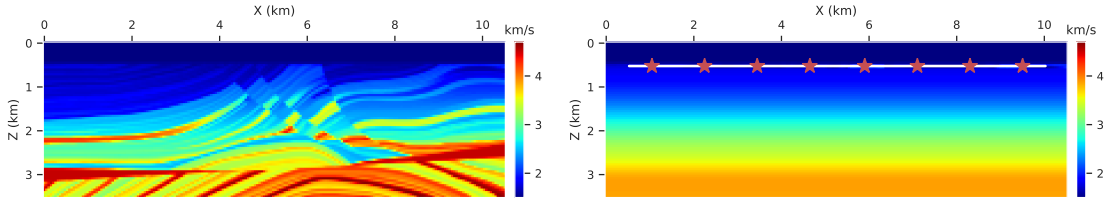


Figure 9.3: Inversion results of an 1D step-change velocity profile. The panels are plotted in the same way as Figure 9.2. Note that the initial step-change at $x = 0.1$ km occurs because we only estimate the velocity at $x \geq 0.1$ km and fix the true velocity value at $x < 0.1$ km to avoid source effects. The waveforms at two receivers are well fit for all three methods, but we observe significant differences in the wavefields (c, f, i) between the two receivers due to the incorrect velocity models.

and 3% of the initial loss can be found in Figure 9.13.

The default architecture of the generative neural network is explained in Table 9.1. In this work we focus on analyzing the effects from re-parametrization instead of specific architectures, so we used a basic architecture with fully-connected layers, convolutional layers, and activation functions of tanh and leaky rectified linear unit (ReLU) (Maas et al., 2013). Tuning the network hyper-parameters and using more advanced architectures such as ResNet (He et al., 2016) may further improve the inversion results. We use 4 upsampling layers of linear interpolation to scale up the latent vector to the size of the velocity model in the experiments above. Meanwhile, we also analyzed architectures using 2 and 3 upsampling layers. The shallow architectures have faster convergence rates (Figure 9.7(c)) and slightly weaker regularization effect (Table 9.3). In addition to inversions using Adam for NNFWI and L-BFGS for conventional FWI, we have also tested inversions using L-BFGS for NNFWI and Adam for conventional FWI. The loss functions are also plotted in Figure 9.7a and the metrics scores are listed in Table 9.5. Both L-BFGS and Adam methods can work well for conventional FWI and NNFWI (Figure 9.14). L-BFGS converges faster than Adam for conventional FWI. This is because Adam is a stochastic gradient descent method with a fixed learning rate strategy, while L-BFGS is a quasi-Newton method with an adaptive learning rate determined by line search. However, Adam proves to be more effective for NNFWI due to the optimization in a high dimensional parameter space of deep neural networks. Thus, we choose the L-BFGS method for conventional FWI and the Adam method for NNFWI as the default setting in this study.



(a) (b)

Figure 9.4: The Marmousi velocity model: (a) the ground-truth model for generating synthetic data; (b) the 1D smoothed initial model for inversion. The source locations (red stars) and the receiver depth (white line) are plotted in (b).

9.3.3 2004 BP model

The 2004 BP benchmark model (Billette & Brandsberg-Dahl, 2005) contains complex salt bodies, which are difficult imaging targets for FWI because of cycle-skipping and amplitude discrepancies caused by sharp contrasts and large-scale salt bodies (Zhang et al., 2018). We extract the left part of the original 2004 BP benchmark model (Figure 9.8), which is based on a geological cross section through the Western Gulf of Mexico. We use the default Ricker wavelet with a peak frequency of 1.2 Hz for this case. The source spacing is 2.8 km and the receiver spacing is 133 m. Conventional

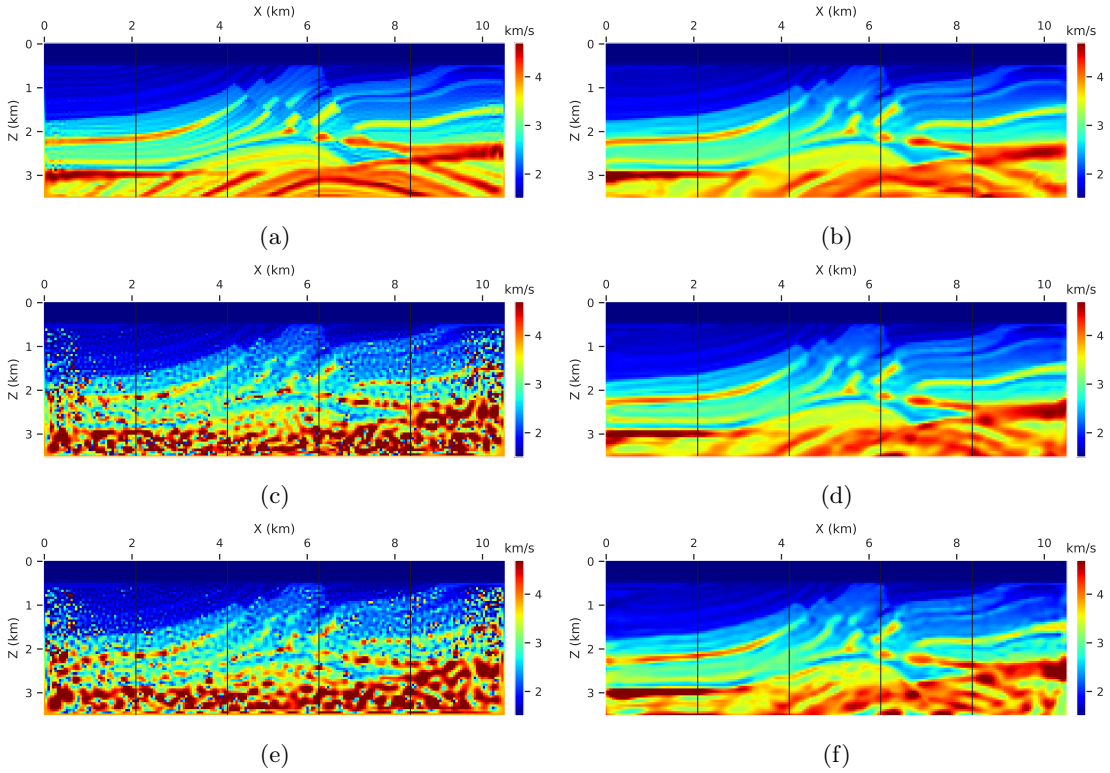
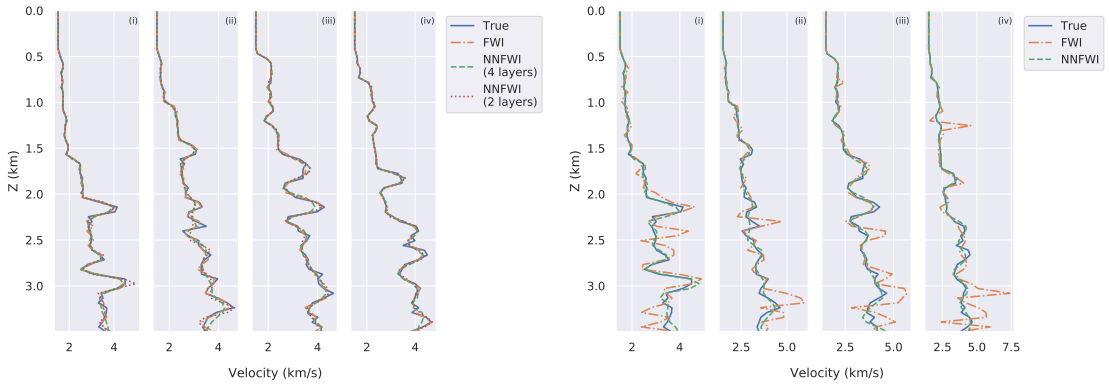


Figure 9.5: Inversion results of conventional FWI and NNFWI on the Marmousi benchmark model. The left panel: conventional FWI results based on data with (a) no random noise, (c) random noise with $\sigma = 0.5\sigma_0$, and (e) random noise with $\sigma = \sigma_0$. The right panel: NNFWI results based on data with (b) no random noise, (d) random noise with $\sigma = 0.5\sigma_0$, and (f) random noise with $\sigma = \sigma_0$. Here σ_0 is the standard deviation of the synthetic seismic data. The velocity profiles along the four black vertical lines are shown in Figure 9.6.

FWI recovers the shallow central portion of the salt body but fails to resolve the U-shaped structure in the left region and the low velocity layers below the salt body Figure 9.9a). Regularization, such as total variation, is needed to skip local minima and estimate the entire shape of the salt body (Esser et al., 2018). However, a proper regularisation weight needs to be determined by trial and error. We tested three regularization weights (γ) of total variation and plotted the best inversion result with ($\gamma = 10^{-3}$) in Figure 9.9a. The inversion results with a too strong regularization effect ($\gamma = 10^{-2}$) and a too weak regularization effect ($\gamma = 10^{-4}$) can be found in Figure 9.16. In contrast, NNFWI can correctly image the salt body without regularization, especially the left U-shaped target (Figure 9.9c). The loss function in Figure 9.7d shows that the loss of NNFWI is not trapped by local minima and continues to decrease to a lower value than conventional FWI and FWI with total variation (labeled as "FWI-TV"). The metrics in Table 9.4 show that NNFWI achieves the best inversion quality. Same as the Marmousi case, we also analyzed the performances with 2 and



(a) (b)

Figure 9.6: Estimated velocity profiles at four locations of the Marmousi benchmark model: (a) no random noise; (b) random noise ($\sigma = 0.5\sigma_0$). The locations of these four profiles are marked by black vertical lines in Figure 9.5.

3 upsampling layers (Table 9.4 and Figure 9.15) and using both Adam and L-BFGS optimization (Table 9.5 and Figure 9.14). We observe the same results that Adam has a better performance for NNFWI and shallower architectures have a slightly weaker regularization effect. To gain insight into the optimization trajectories, we plot the intermediate inversion results when the loss function is reduced to 30%, 10%, and 3% of its initial value in Figure 9.10. The intermediate inversion results of FWI with total variation can be found in Figure 9.17. The result of conventional FWI shows that most of the early updates focus on top shallow layers. NNFWI, in contrast, updates over a much broader depth range and recovers a smooth and approximate shape of the salt body. This difference demonstrates the spatial regularization from convolutional neural networks, which applies convolutional filters across the whole domain, so that the updates of convolutional kernels affect the entire generated velocity model.

9.3.4 Uncertainty Quantification and Computational Analysis

To analyze the uncertainty in the inversion results, which is a challenging task for conventional FWI, we conducted another experiment by adding dropout layers to the generative neural network (Table 9.1). Dropout was initially proposed to prevent overfitting in neural networks by randomly setting a proportion of neurons and their connections to zero (Srivastava et al., 2014). Gal and Ghahramani (2016) demonstrated that dropout in neural networks can also be interpreted in a Bayesian framework to estimate model uncertainty. The dropout operation with a dropout ratio q is the same as applying a binary sampling vector that follows a Bernoulli distribution with a probability $p = 1-q$ to the weight metrics in a neural network. They proved that a neural network with dropout applied before every weight layer mathematically approximates variational inference for a Gaussian

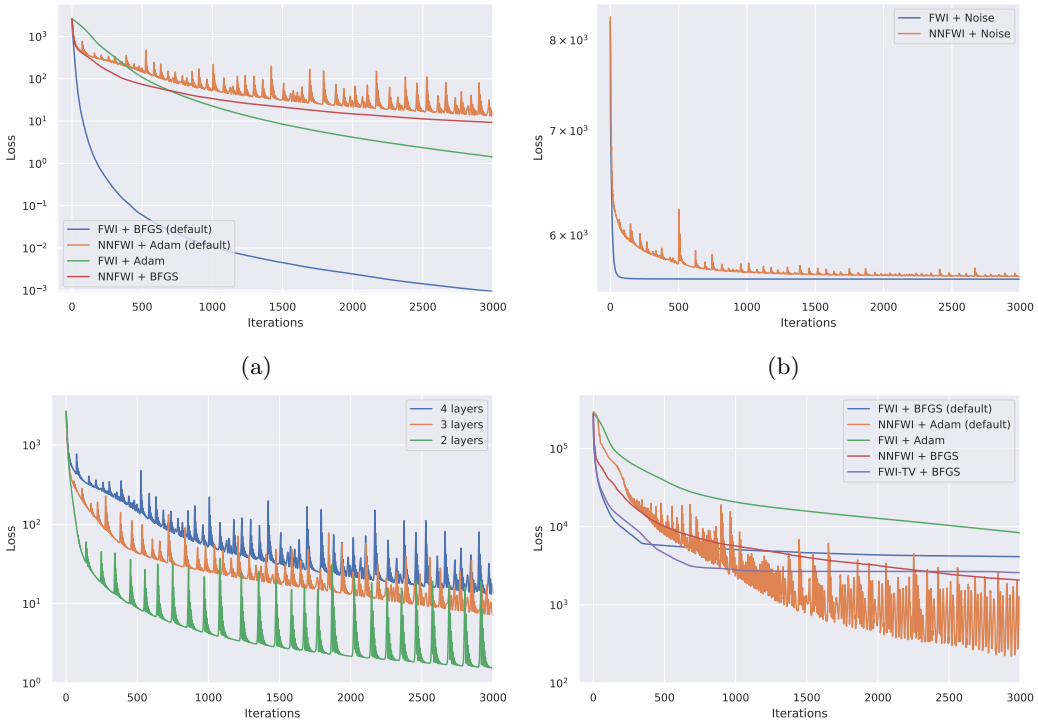


Figure 9.7: Loss functions: (a) inversions with no random noise of the Marmousi model; (b) inversions with random noise ($\sigma = 0.5\sigma_0$) of the Marmousi model; (c) inversions of NNFWI with different number of convolutional layers; (d) inversion with no random noise of the 2004 BP model (explained in Section 9.3.3). (c) is also based on the Marmousi model and the same loss functions based on the 2004 BP model is shown in Figure 9.15. We have tested both L-BFGS and Adam optimization methods for conventional FWI and NNFWI. Adam proves to be more effective for NNFWI because of the optimization of the generative neural networks. We use the L-BFGS method for conventional FWI and the Adam method for NNFWI as the default settings in this study.

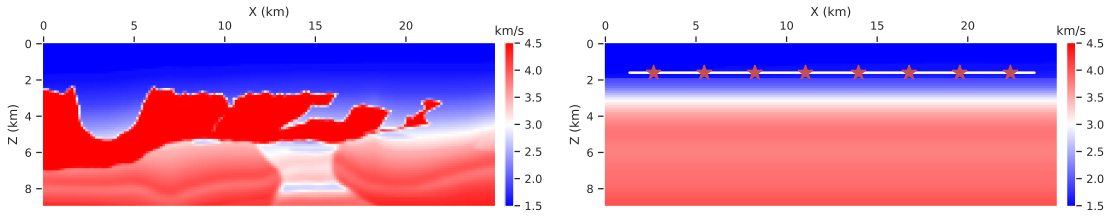


Figure 9.8: The 2004 BP benchmark model: (a) the ground-truth model for generating synthetic data; (b) the 1D smoothed initial model for inversion. The source locations (red stars) and the receiver depth (while line) are plotted in (b).

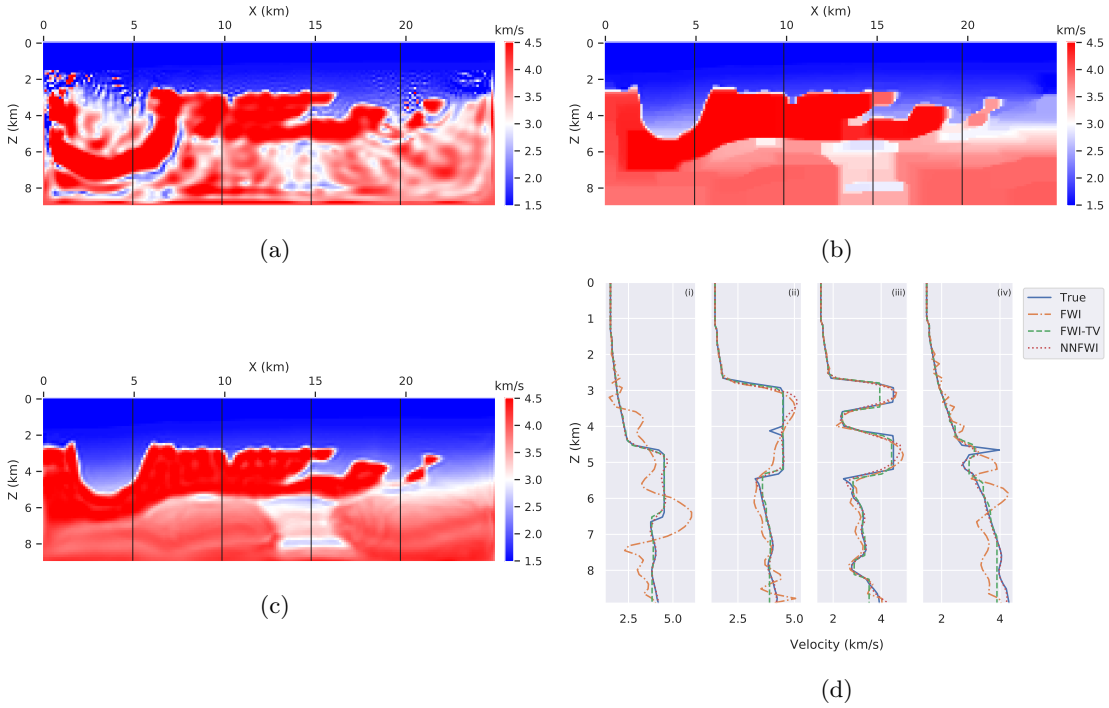


Figure 9.9: Inversion results of the 2004 BP benchmark model: (a) conventional FWI; (b) FWI with TV (total variation) regularization ($\gamma = 10^{-3}$); (c) NNFWI; (d) velocity profiles at four locations that are marked by the black vertical lines in (a, b, c). The other two inversion results with two different TV regularization coefficients of $\gamma = 10^{-2}$ and $\gamma = 10^{-4}$ are shown in Figure 9.16.

process such that model uncertainty can be efficiently estimated by performing stochastic forward passes through the neural network and collecting statistics of the results. We ran 100 Monte Carlo samplings based on the optimized neural network model to calculate the standard deviation of the sampled velocity models. Because the sampling process only requires inference using the generative neural network, uncertainty estimation is orders of magnitude faster than other sampling-based methods that require solving the PDEs.

Figure 9.11a and Figure 9.12a show the estimated velocity models of the Marmousi benchmark model and the 2004 BP model respectively, which are similar to the inversion results in Figure 9.5b and Figure 9.9c. Figure 9.11b and Figure 9.12b show the absolute error between the estimated models and the true models; and Figure 9.11c and Figure 9.12c show the estimated uncertainty. The estimated velocity profiles and estimated uncertainty along depth are shown in Figure 9.11d and Figure 9.12d. The estimated uncertainty by dropout in these two experiments does not exactly reproduce the inversion error but follows a similar trend with the error map. This is not unexpected, since the uncertainty is a statistical estimate, while the model error results from a single computation. To analyze if the estimated uncertainty is consistent using different numbers of network layers and

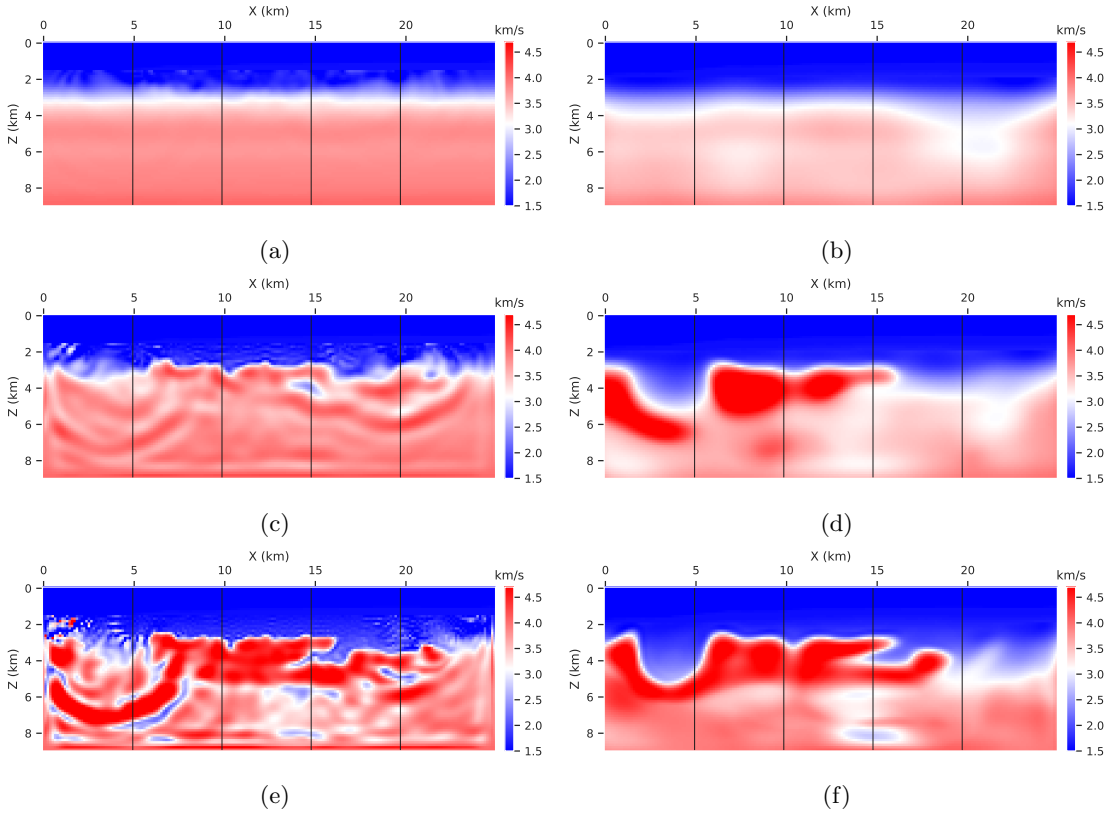


Figure 9.10: Intermediate inversion results of the 2004 BP model: The left panels (a, c, e) show results of conventional FWI; The right panels (b, d, f) show results of NNFWI. We plot three estimated velocity maps at 30%, 10%, and 3% of the initial loss to show the different optimization processes of conventional FWI and NNFWI. Because of the different convergence speeds (Figure 9.7), the iteration numbers are 10, 50, and 240 for FWI in the left panels and 90, 230, 570 for NNFWI in the right panels. The intermediate inversion results of FWI with TV regularization are plotted in Figure 9.17. The intermediate inversion results of the Marmousi model are plotted in Figure 9.13.

dropout ratios, we added one example using 3 upsampling and dropout layers instead of the default 4 layers and another example with a dropout ratio of 20% instead of the default 10%. The results of the Marmousi model can be found in Figure 9.18 and Figure 9.19; and the results of the 2004 BP model can be found in Figure 9.20 and Figure 9.21. The overall estimated uncertainty maps are consistent with each other among these experiments. Meanwhile we can also observe that the estimated standard deviation is slightly smaller for the experiment with a smaller number of upsampling and dropout layers and slightly larger for the experiment with a higher dropout ratio. Due to the lack of uncertainty benchmarks in FWI, further research is needed to verify the accuracy of uncertainty estimation by dropout used in NNFWI.

Finally, we address the computational demands of NNFWI. Based on the compute times per

integration on both CPUs and GPUs in Table 9.6, NNFWI introduces negligible additional computation compared with conventional FWI. This is because generating the velocity model using neural networks is much less expensive than solving the PDEs. Moreover, the optimization of the neural network and PDEs are done simultaneously with automatic differentiation without additional optimization loops as used in dictionary learning (Zhu et al., 2017). The computational efficiency of neural networks allows the exploration of deeper and more complex neural network architectures in future research.

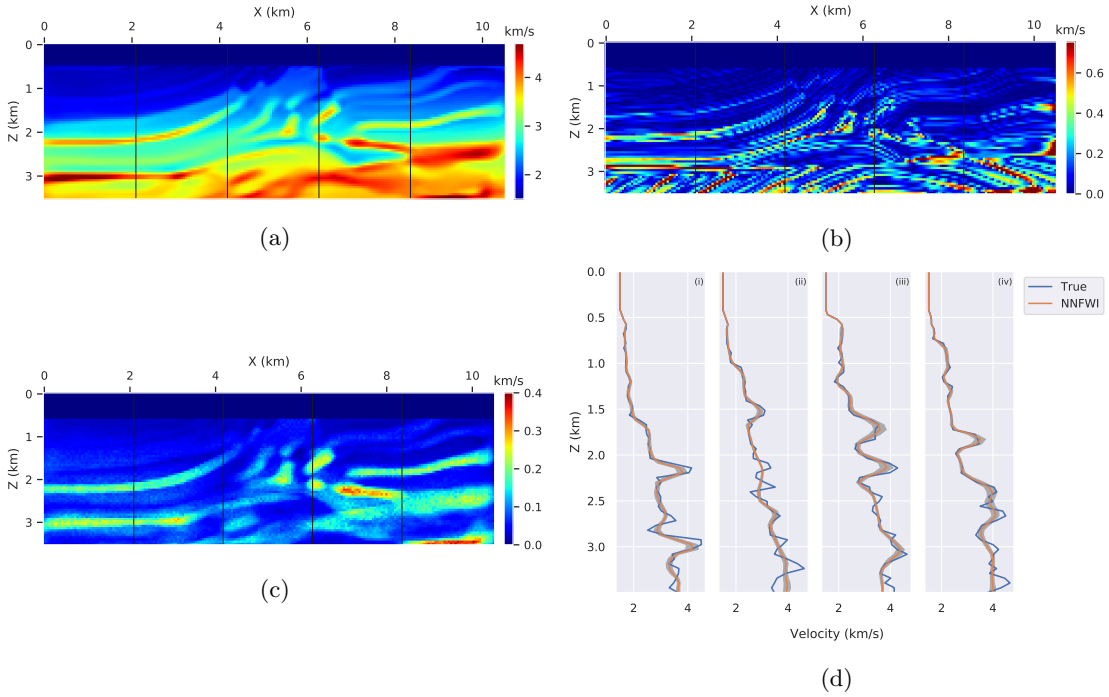


Figure 9.11: Uncertainty quantification of NNFWI by adding dropout layers during training: (a) inversion result of the Marmousi model; (b) inversion error map; (c) estimated standard deviation calculated through Monte Carlo sampling with a dropout rate of 0.1; (d) velocity profiles at four locations (marked by black lines in (a, b, c)). The standard deviation ranges are plotted in gray. The results of uncertainty quantification using three convolutional and dropout layers and using a dropout rate of 0.2 are shown in Figure 9.18 and Figure 9.19 respectively.

9.4 Discussion

NNFWI combines deep neural networks and PDEs for FWI. Compared with data-driven inversion methods relying on only neural networks for direct inverse prediction, NNFWI can use both the properties of neural networks and the physical information represented by PDEs such that its accuracy is similar to that of conventional FWI. Moreover, direct data-driven inverse prediction

Table 9.2: Comparison of FWI and NNFWI under different noise levels based on the Marmousi model.

Experiment	MSE	SSIM	PSNR
FWI (no noise)	114	0.993	49.5
FWI (0.5σ noise)	568	0.851	42.6
FWI (1.0σ noise)	610	0.837	42.4
NNFWI (no noise)	152	0.987	48.2
NNFWI (0.5σ noise)	171	0.983	47.8
NNFWI (1.0σ noise)	233	0.969	46.4

Table 9.3: Comparison of FWI and NNFWI with different numbers of convolutional layers based on the Marmousi model.

Experiment	MSE	SSIM	PSNR
FWI	114	0.993	49.5
NNFWI (4 layers)	152	0.987	48.2
NNFWI (3 layers)	146	0.988	48.4
NNFWI (2 layers)	145	0.988	48.5

Table 9.4: Comparison among FWI, FWI with total variation regularization, and NNFWI with different numbers of convolutional layers based on the 2004 BP model

Experiment	MSE	SSIM	PSNR
FWI	623	0.866	41.7
FWI-TV ($\gamma = 10^{-4}$)	568	0.887	42.1
FWI-TV ($\gamma = 10^{-3}$)	181	0.988	47.1
FWI-TV ($\gamma = 10^{-2}$)	494	0.901	42.7
NNFWI (4 layers)	134	0.993	48.3
NNFWI (3 layers)	209	0.984	46.4
NNFWI (2 layers)	258	0.976	45.4

Table 9.5: Comparison of FWI and NNFWI using different optimization algorithms.

Model	Experiment	MSE	SSIM	PSNR
Marmousi Model	FWI + BFGS	114	0.993	49.5
	FWI + Adam	204	0.976	47.0
	NNFWI + Adam	152	0.987	48.2
	NNFWI + BFGS	196	0.978	47.2
2004 BP Model	FWI + BFGS	623	0.866	41.7
	FWI + Adam	734	0.827	41.0
	NNFWI + BFGS	134.	0.993.	48.3
	NNFWI + Adam	225	0.981	46.1

Table 9.6: The compute time per iteration. The CPU time is averaged over 100 iterations running on a 40-core CPU server with Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz. The GPU time is averaged over 100 iterations running on a 8 GPU server with Nvidia Tesla V100-SXM2-32GB-LS.

Computing time	CPU (s)	GPU (s)
Conventional FWI	6.25	1.91
NNFWI	6.33	1.92

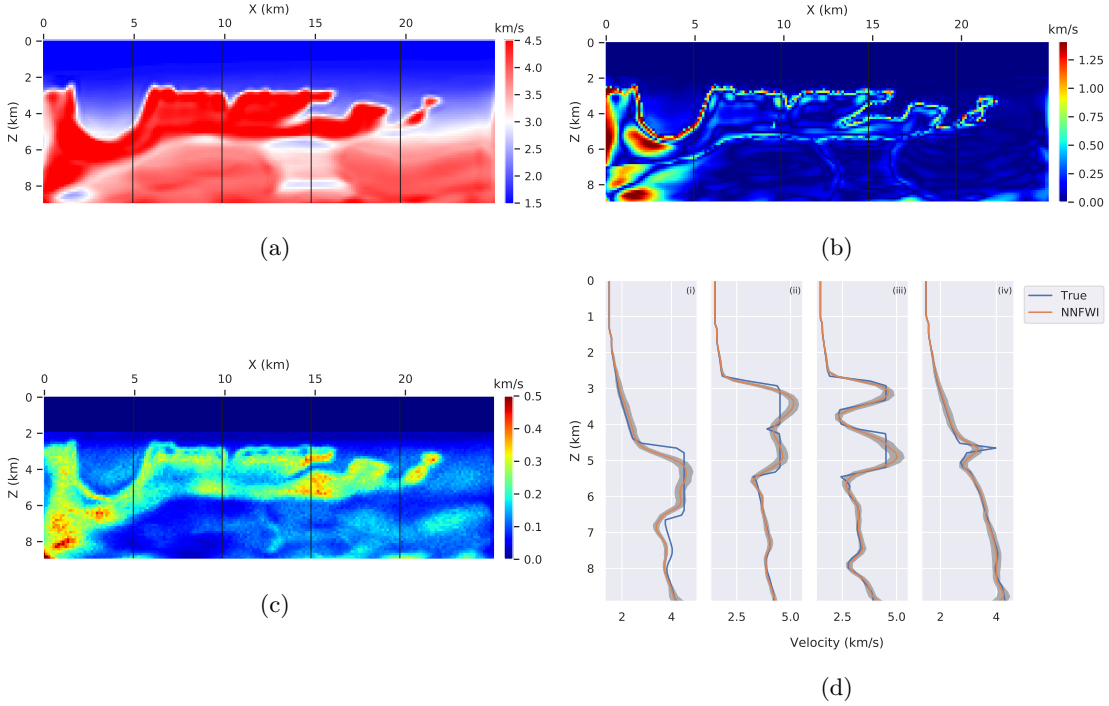


Figure 9.12: Uncertainty quantification of NNFWI by adding dropout layers during training: (a) inversion result of the Marmousi model; (b) inversion error map; (c) estimated standard deviation through Monte Carlo sampling with a dropout rate of 0.1; (d) velocity profiles at four locations (marked by black lines in (a, b, c)). The results of uncertainty quantification using three convolutional and dropout layers and using a dropout rate of 0.2 are shown in Figure 9.20 and Figure 9.21 respectively.

relies on training on a large number of data pairs of velocity models and corresponding seismic data, which is not always available, so it may be susceptible to poor generalization when data does not follow a similar distribution to the training data. NNFWI does not need extra datasets for training the neural network. It applies to the same data settings as conventional FWI. Additionally, NNFWI adds little extra computational cost compared with conventional FWI because the computational cost of deep neural networks is much less than PDEs. The built-in GPU acceleration of deep learning frameworks further speeds up the simulation and optimization processes of NNFWI (Zhu et al., 2020). NNFWI can also be combined with conventional regularization methods in FWI to improve inversion results. For example, we can add the total variation of the generated velocity model $\mathcal{N}(z, w)$ or add the $\mathcal{L}_2$-norm of the weights w of neural networks to the loss function in Equation (9.4) as regularization.

In conventional FWI, adding additional degrees of freedom by extending velocity into non-physical dimensions shows promise in overcoming local minima and improving the inversion results (Barnier et al., 2018; Biondi & Almomin, 2014; Symes, 2008). The generative neural network in

NNFWI can easily over-parametrize physical velocity models. Although theories of deep learning are still developing, over-parametrization is believed to be a key factor in the effective optimization and generalization of deep learning methods (Allen-Zhu et al., 2019; Arora et al., 2018; Cao & Gu, 2019). In this study, we have shown that parameterizing the velocity model by a generative neural network can be used for both regularization and uncertainty quantification for FWI. The regularization effect of NNFWI is similar to dictionary learning in FWI. Dictionary-learning-based FWI adds an extra training loop, which is not needed in NNFWI, to learn sparse representations of complex features in the velocity model using many small training patches collected from previous iterations. In NNFWI, the generative neural network serves as a rich feature bank to represent the complex velocity model. Compared with the complex and computationally expensive dictionary learning step, the update of the generative neural network in NNFWI directly uses the gradients calculated by automatic differentiation. The gradients directly back-propagate from the loss through the PDEs and neural network layers to the weights of the generative neural network. Moreover, no extra training loops of the neural networks are needed, making the training workflow of NNFWI as simple as conventional FWI.

In addition to the architecture of the generative neural network used in this study, other neural network architectures can also be used as the generator in NNFWI. A variety of components of NNFWI can be tuned and added, such as the number of neural network layers, the choice of activation function, batch normalization layer, recurrent neural network layer, attention layer, and a group of optimization algorithms in deep learning. We can also replace the acoustic wave equation used in this work with the elastic wave equation as the PDE solver in NNFWI for elastic FWI applications. In other words, NNFWI is not limited to the specific configuration used in this study. Our work provides a general framework for incorporating deep neural networks with PDEs for FWI applications. NNFWI also enables more advanced deep learning modeling in FWI. For example, we can use Bayesian neural networks (Kendall & Gal, 2017), which model the epistemic uncertainty in the model, as the generator in NNFWI. We can replace the input latent variable z, which is a randomized vector in this work, with the waveform observation y. In this way, NNFWI becomes similar to an encoder-decoder model (Goodfellow et al., 2016) with the neural network as the encoder and the PDE solver as the decoder. The neural network after training learns a mapping from observations to the velocity model. These extensions are natural directions for future research.

9.5 Conclusions

We have introduced NNFWI, an approach to integrate deep generative neural networks to the PDE-constrained optimization of FWI. NNFWI represents the velocity model of interest using a generative neural network and optimizes the weights of the neural network instead. The gradients of both the neural network and PDEs are calculated using automatic differentiation, which simplifies

the optimization of NNFWI without the need of pretraining or extra optimization loops. Our results demonstrate that NNFWI achieves similar accuracy as conventional FWI. More importantly, the deep image prior of generative neural networks automatically filter out noise and introduce a regularization effect, which mitigates cycle-skipping similar as TV regularization and significantly improves the inversion results for noisy seismic data. Additionally, NNFWI provides uncertainty quantification using the dropout technique, which does not require additional computation during training, making NNFWI a much more efficient way to estimate uncertainty in the high dimensional model space of velocity estimation. NNFWI can be directly applied to the same datasets as conventional FWI to improve inversion performance. Extensions to the generative neural work in NNFWI and validating its uncertainty quantification as an absolute error proxy are promising directions for future work.

Appendix

The appendix includes additional figures to compare inversion results between NNFWI and conventional FWI and to analyze uncertainty quantification of NNFWI. The figures are referenced and explained in the main text.

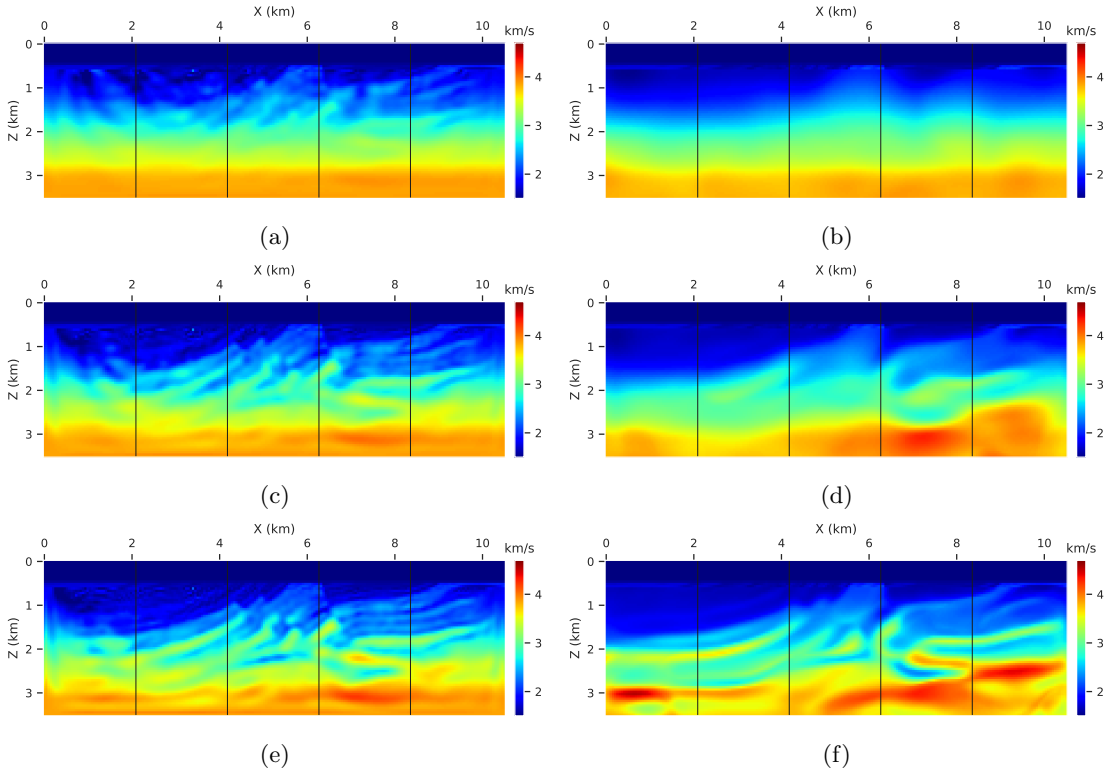


Figure 9.13: Intermediate inversion results of the Marmousi model: the left panels (a, c, e) show results of conventional FWI; and the right panels (b, d, e) show results of NNFWI. We plot three inverted velocity maps at 0.3, 0.1, and 0.03 of the initial loss to show the different optimization processes of conventional FWI and NNFWI. The iteration numbers are 10, 20, and 30 for FWI in the left panels and 10, 270, 890 for NNFWI in the right panels.

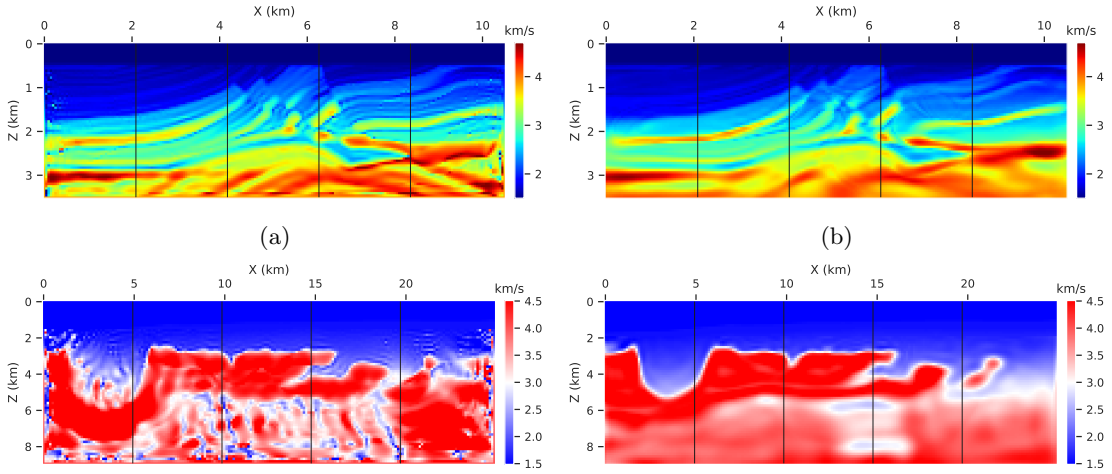


Figure 9.14: Inversion results of (a) conventional FWI with Adam based on the Marmousi model, (b) NNFWI with L-BFGS based on the Marmousi model, (c) conventional FWI with Adam based on the 2004 BP model, and (d) NNFWI with based on the 2004 BP model.

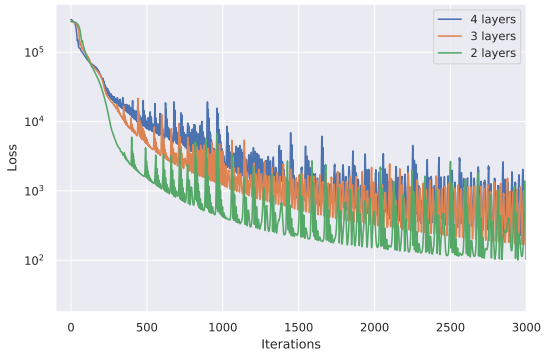


Figure 9.15: Loss functions of NNFWI with different convolutional layers based on the 2004 BP model.

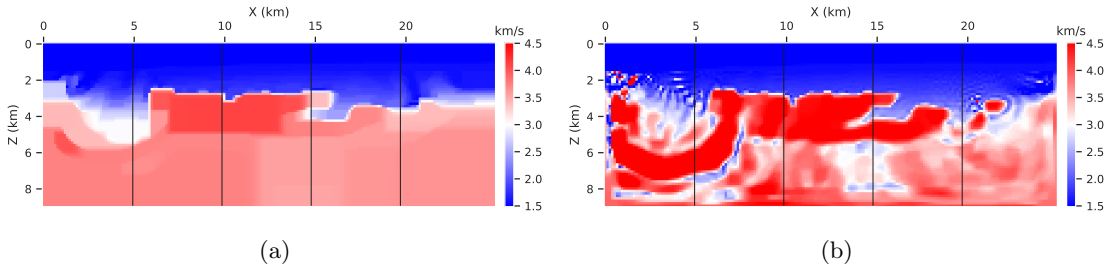


Figure 9.16: Inversion results of conventional FWI with total variation regularization: (a) $\gamma = 0.01$; and (b) $\gamma = 0.0001$. The inversion result with $\gamma = 0.001$ is shown in Figure 9.8(b).

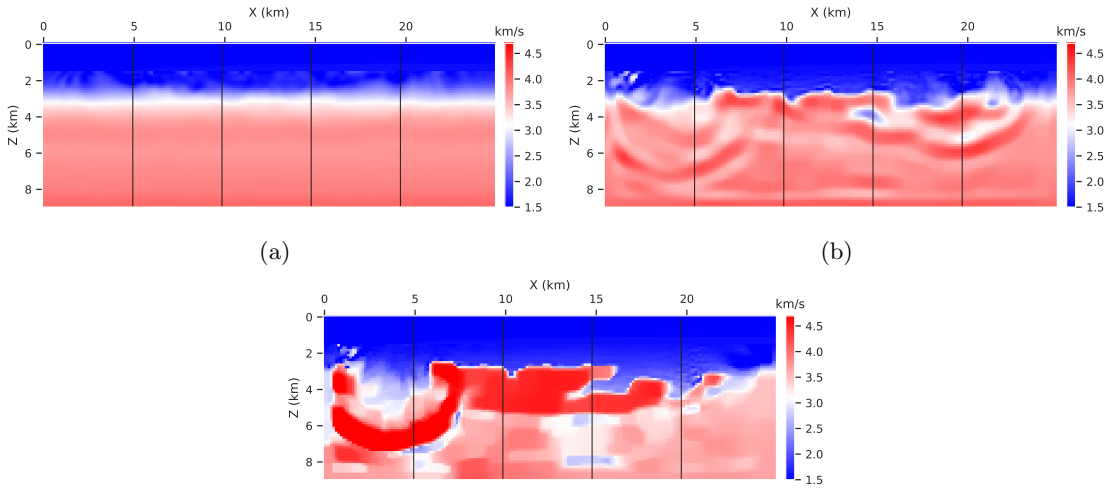


Figure 9.17: Intermediate inversion results of conventional FWI with total variation regularization. We plot three inverted velocity maps at 0.3, 0.1, and 0.03 of the initial loss in (a), (b), and (c) respectively. The iteration numbers are 10, 60, and 330. The results of conventional FWI and NNFWI for comparison are plotted in Figure 9.10.

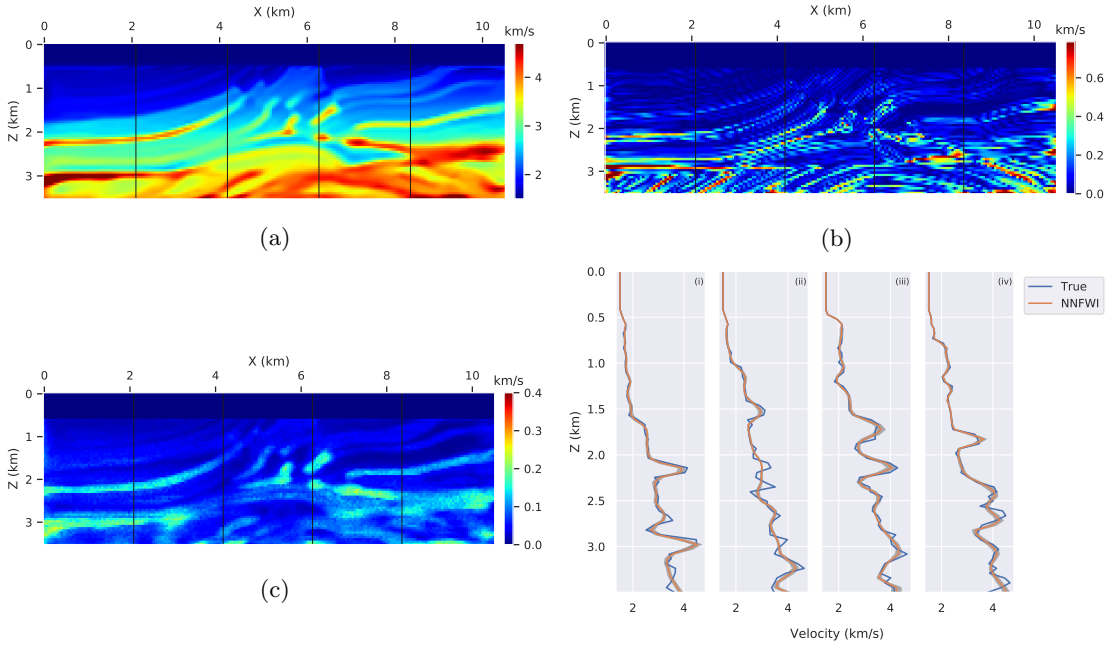


Figure 9.18: Uncertainty quantification of NNFWI with 3 upsampling and dropout layers based on the Marmousi model: (a) inversion result; (b) inversion error map; (c) estimated standard deviation through Monte Carlo samplings with a dropout rate of 0.1; (d) velocity profiles with standard deviation ranges plotted in gray. Their locations are marked by black vertical lines in (a, b, c).

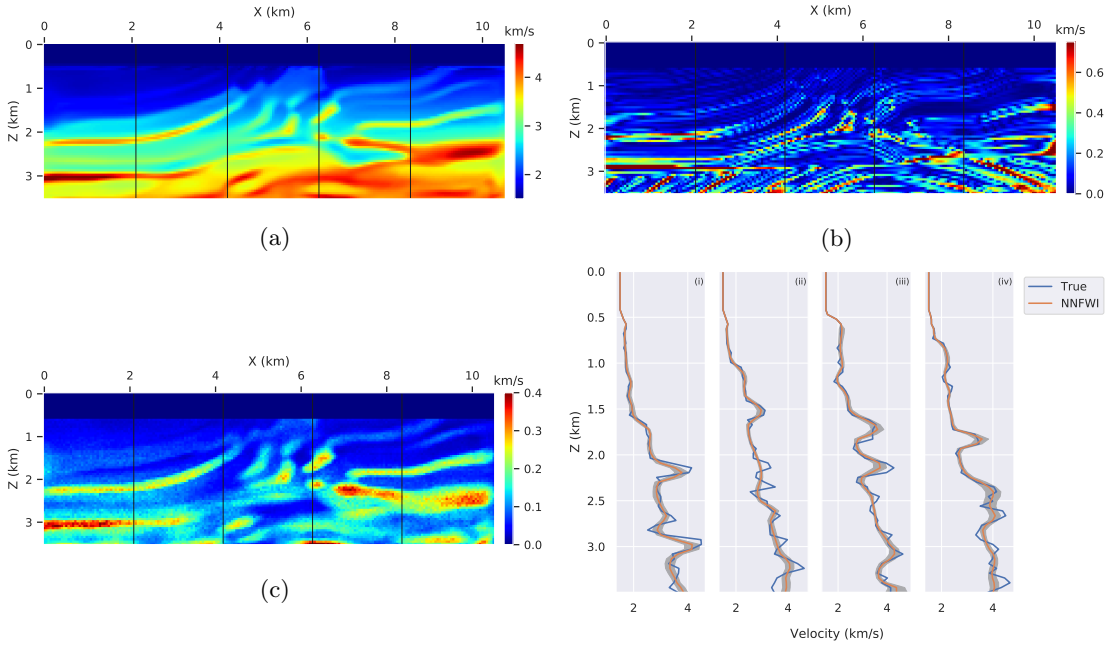


Figure 9.19: Uncertainty quantification of NNFWI with a dropout rate of 0.2 based on the Marmousi model: (a) inversion result; (b) inversion error map; (c) estimated standard deviation through Monte Carlo samplings with a dropout rate of 0.2; (d) velocity profiles with standard deviation ranges plotted in gray. Their locations are marked by black vertical lines in (a, b, c).

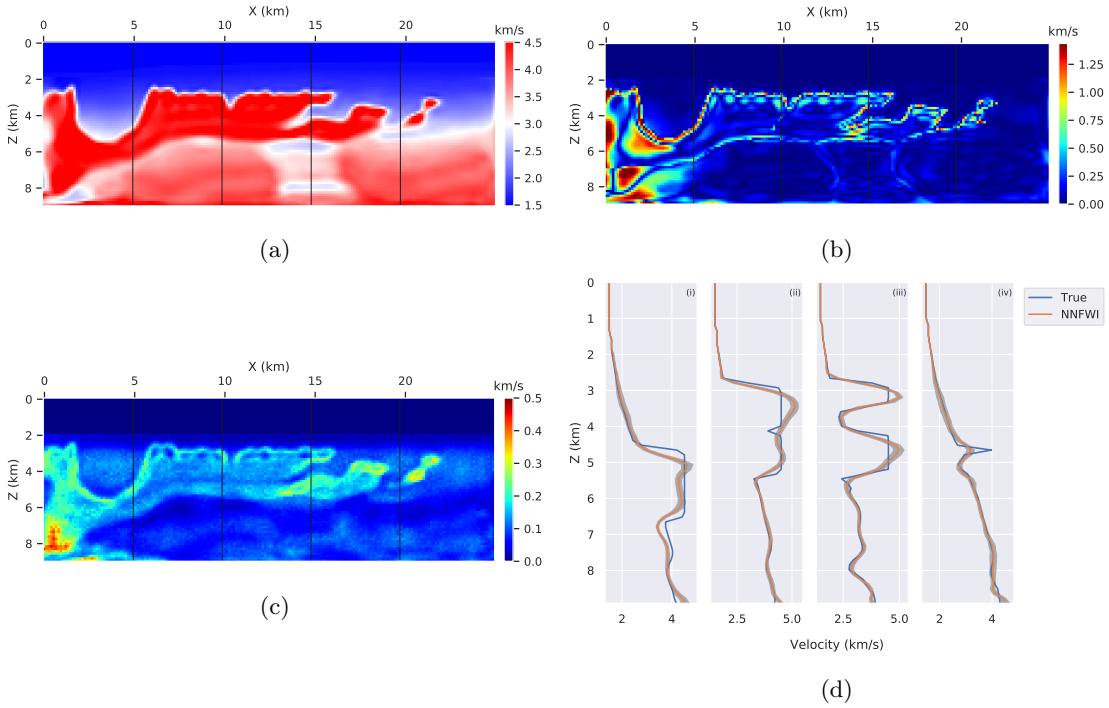


Figure 9.20: Uncertainty quantification of NNFWI with 3 upsampling and dropout layers based on the 2004 BP model: (a) inversion result; (b) inversion error map; (c) estimated standard deviation through Monte Carlo samplings with a dropout rate of 0.1; (d) velocity profiles with standard deviation ranges plotted in gray. Their locations are marked by black vertical lines in (a, b, c).

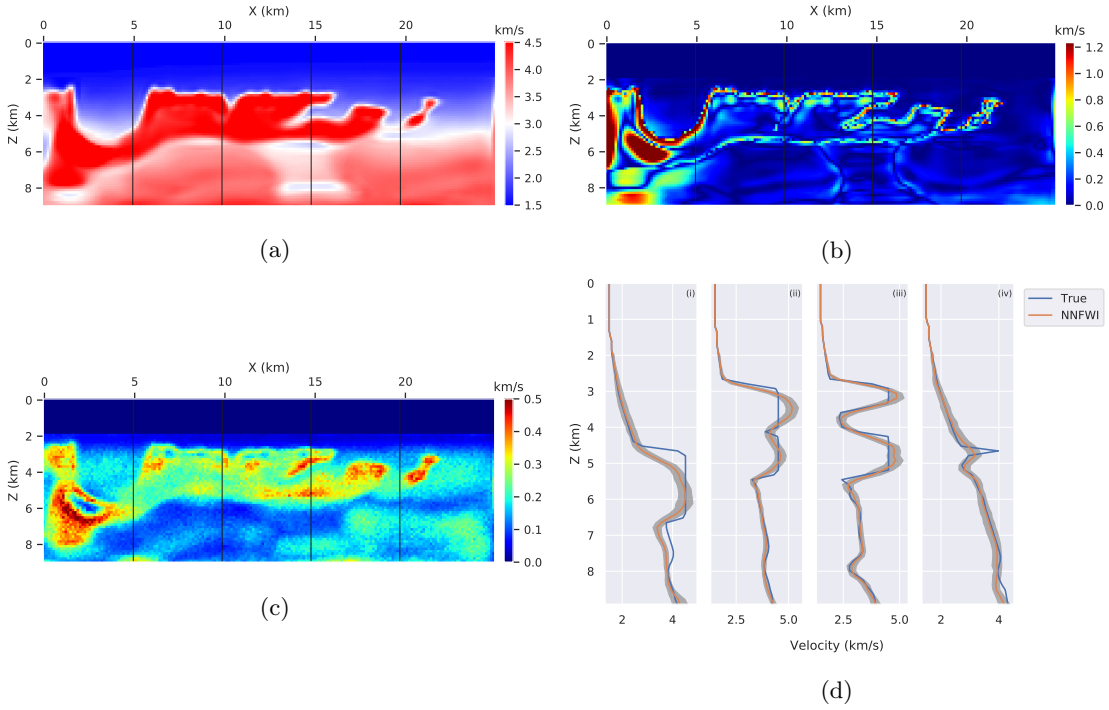


Figure 9.21: Uncertainty quantification of NNFWI with a dropout rate of 0.2 based on the 2004 BP model: (a) inversion result; (b) inversion error map; (c) estimated standard deviation through Monte Carlo samplings with a dropout rate of 0.2; (d) velocity profiles with standard deviation ranges plotted in gray. Their locations are marked by black vertical lines in (a, b, c).

Chapter 10

Conclusions

Applications of deep learning to seismological problems have been growing rapidly in the past five years. The large seismic data sets collected and archived in the past few decades, especially the ones labeled by analysts, provide important training materials for deep learning algorithms (e.g., deep neural networks). With sufficient training, deep learning can automatically learn to extract characteristic features from data and build classification/regression rules for specific research targets of interest. As discussed in thesis, we have applied deep learning to the serial tasks of earthquake monitoring including: signal denoising (Chapter 2), phase picking (Chapter 3), and phase association (Chapter 5), to build a new deep-learning-based earthquake monitoring workflow (Chapter 6) to generate comprehensive earthquake catalogs that reveal detailed information on earthquake sequences and fault zone structures. These deep learning models have significantly improved earthquake monitoring performance by detecting up to orders of magnitude more small earthquakes than are present in standard catalogs. Additionally, we have developed an end-to-end approach to detect earthquakes from seismic waveforms across a network. This approach avoids the information loss at each stage of earthquake monitoring workflows (Chapter 7). Tasks of phase picking, phase association, and event detection are jointly optimized in an end-to-end neural network architecture. We have also analyzed data augmentation techniques that are applicable to seismic waveforms (Chapter 4), using PhaseNet as example, to improve the generalization of these deep learning models. These efforts make the trained deep learning models applicable to a wide range of research problems. The applications have demonstrated deep learning models can recognize seismic signals to a degree similar to human analysts and by doing so can efficiently process large seismic data sets to detect hidden small earthquakes and lead to new insights.

In addition to recognizing seismic signals, we have applied deep learning to seismic inversion to constrain underlying physical parameters. We have applied automatic differentiation to PDE-constrained inversions, such as full-waveform inversion (Chapter 8), so that the optimization of seismic inversion problems can be solved in a way similar to that of training neural networks, where

the gradients of parameters of interest are automatically calculated by deep learning frameworks without manual derivation and implementation. We have in addition explored applying neural networks as a parameterization and regularization method (Chapter 9). The inductive bias of neural network architectures (e.g., convolution neural networks) can impose a prior to the inversion target (e.g., wave velocity model) and improve seismic inversion similar to that of conventional regularization. These applications have shown the close connections between seismic inversion and neural network optimization; thus, the rapid development of deep learning frameworks and neural network architectures will continue to benefit seismic inversion.

Deep learning potentially opens the path to a new paradigm of data-driven research. A large number of data sets have been collected and manually labeled (other than P- and S-phases) in previous research. However, these data sets have been archived and left idle. Deep learning now provides an approach to transforming these data sets into useful models that can be applied to searching for similar signals on a large scale. Two specific features of deep learning makes this data-driven approach promising for seismological problems. First, deep neural networks can achieve performance similar to that of human analysts by training on large data sets labeled by analysts. Although it is challenging to define mathematically the exact decision rules used by analysts to recognize specific signals of interest, deep neural networks can implicitly learn these decision rules from manual labels. For example, the PhaseNet model after training can learn to pick S-phases as effectively as P-phases, while conventional algorithms have failed to define an effective characteristic function for S-phase. Second, deep neural networks run very quickly once trained, allowing us to efficiently apply deep learning models to processing large data sets that have not been studied before to detect many more similar signals or to verify whether the signals exists under different conditions. These detected signals could offer new insights into a broad range of scientific problems. In sum, deep learning has become an important tool for studying earthquakes, and it will be more widely applied to solving various scientific problems in Geophysics and Earth Sciences in general.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., ... Zheng, X. (2016). Tensorflow: A system for large-scale machine learning, In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.

Abma, R., & Claerbout, J. (1995). Lateral prediction for noise attenuation by tx and fx techniques. *Geophysics*, *60*(6), 1887–1896.

Abramowitz, M., & Stegun, I. A. (1964). *Handbook of mathematical functions with formulas, graphs, and mathematical tables* (Vol. 55). US Government printing office.

Adler, A., Araya-Polo, M., & Poggio, T. (2021). Deep learning for seismic inverse problems: Toward the acceleration of geophysical analysis workflows. *IEEE Signal Processing Magazine*, *38*(2), 89–119.

Akazawa, T. (2004). Technique for Automatic Detection of Onset Time of P- and S-Phases in Strong Motion Records. *13 th World Conference on Earthquake Engineering*, (786), 786.

Al-Ismail, F., Ellsworth, W. L., & Beroza, G. C. (2020). Empirical and Synthetic Approaches to the Calibration of the Local Magnitude Scale, ML, in Southern Kansas. *Bulletin of the Seismological Society of America*, *110*(2), 689–697.

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

Allen-Zhu, Z., Li, Y., & Song, Z. (2019). A Convergence Theory for Deep Learning via Over-Parameterization. *arXiv:1811.03962 [cs, math, stat]*.

Arora, S., Cohen, N., & Hazan, E. (2018). On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization. *arXiv:1802.06509 [cs]*.

Asnaashari, A., Brossier, R., Garambois, S., Audebert, F., Thore, P., & Virieux, J. (2013). Regularized Seismic Full Waveform Inversion with Prior Model Information. *GEOPHYSICS*, *78*(2), R25–R36.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Baer, M., & Kradolfer, U. (1987). An automatic phase picker for local and teleseismic events. *Bulletin of the Seismological Society of America*, *77*(4), 1437–1445.

Bai, C.-y., & Kennett, B. (2000). Automatic phase-detection and identification by full use of a single three-component broadband seismogram. *Bulletin of the Seismological Society of America*, *90*(1), 187–198.

Baillard, C., Crawford, W. C., Ballu, V., Hibert, C., & Mangeney, A. (2014). 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.

Barnier, G., Biondi, E., & Biondi, B. (2018). Full waveform inversion by model extension. In *SEG Technical Program Expanded Abstracts 2018* (pp. 1183–1187). Society of Exploration Geophysicists.

Barnier, G., Biondi, E., & Clapp, R. (2019). Waveform inversion by model reduction using spline interpolation. In *SEG Technical Program Expanded Abstracts 2019* (pp. 1400–1404). Society of Exploration Geophysicists.

Barrett, S., & Beroza, G. (2014). An Empirical Approach to Subspace Detection. *Seismological Research Letters*, *85*, 594–600.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2017). Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, *18*(1), 5595–5637.

Bekara, M., & Van der Baan, M. (2009). Random and coherent noise attenuation by empirical mode decomposition. *Geophysics*, *74*(5), V89–V98.

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning, In *Proceedings of ICML workshop on unsupervised and transfer learning*.

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

Ben-Zion, Y., Vernon, F. L., Ozakin, Y., Zigone, D., Ross, Z. E., Meng, H., White, M., Reyes, J., Hollis, D., & Barklage, M. (2015). Basic data features and results from a spatially dense seismic array on the San Jacinto fault zone. *Geophysical Journal International*, *202*(1), 370–380.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, *1*(3), 81–84.

Beroza, G. C. (1991). Near-source modeling of the Loma Prieta earthquake: Evidence for heterogeneous slip and implications for earthquake hazard. *Bulletin of the Seismological Society of America*, *81*(5), 1603–1621.

Beroza, G. C., & Spudich, P. (1988). Linearized inversion for fault rupture behavior: Application to the 1984 Morgan Hill, California, earthquake. *Journal of Geophysical Research: Solid Earth*, *93*(B6), 6275–6296.

Beskardes, G., Hole, J., Wang, K., Michaelides, M., Wu, Q., Chapman, M., Davenport, K., Brown, L., & Quiros, D. (2018). A comparison of earthquake backprojection imaging methods for dense local arrays. *Geophysical Journal International*, *212*(3), 1986–2002.

Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., & Wassermann, J. (2010). ObsPy: A Python toolbox for seismology. *Seismological Research Letters*, *81*(3), 530–533.

Billette, F. J., & Brandsberg-Dahl, S. (2005). The 2004 BP Velocity Benchmark, In *67th EAGE Conference & Exhibition*, European Association of Geoscientists & Engineers.

Biondi, B., & Almomin, A. (2014). Simultaneous inversion of full data bandwidth by tomographic full-waveform inversion. *Geophysics*, *79*(3), WA129–WA140.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York, Springer ZSCC: 0047397.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*.

Bonar, D., & Sacchi, M. (2012). Denoising seismic data using the nonlocal means algorithm. *Geophysics*, *77*(1), A5–A8.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177–186). Springer.

Bozdağ, E., Trampert, J., & Tromp, J. (2011). Misfit functions for full waveform inversion based on instantaneous phase and envelope measurements. *Geophysical Journal International*, *185*(2), 845–870.

Bradley, A. M. (2013). *PDE-constrained optimization and the adjoint method* (tech. rep.). Technical Report. Stanford University. https://cs. stanford. edu/˜ ambrad ....

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

Burstedde, C., & Ghattas, O. (2009). Algorithmic Strategies for Full Waveform Inversion: 1D Experiments. *GEOPHYSICS*, *74*(6), WCC37–WCC46.

Cao, D., & Liao, W. (2015). A computational method for full waveform inversion of crosswell seismic data using automatic differentiation. *Computer Physics Communications*, *188*, 47–58.

Cao, S., & Chen, X. (2005). The second-generation wavelet transform and its application in denoising of seismic data. *Applied geophysics*, *2*(2), 70–74.

Cao, Y., & Gu, Q. (2019). Generalization Error Bounds of Gradient Descent for Learning Over-Parameterized Deep ReLU Networks. *arXiv:1902.01384 [cs, math, stat]*.

Chai, C., Maceira, M., Santos-Villalobos, H. J., Venkatakrishnan, S. V., Schoenball, M., Zhu, W., Beroza, G. C., Thurber, C., & Team, E. C. (2020). Using a Deep Neural Network and

Transfer Learning to Bridge Scales for Seismic Phase Picking. *Geophysical Research Letters*, *47*(16), e2020GL088651.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Chen, T., Xu, B., Zhang, C., & Guestrin, C. (2016). Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Chen, W., Xie, J., Zu, S., Gan, S., & Chen, Y. (2017). Multiple-Reflection Noise Attenuation Using Adaptive Randomized-Order Empirical Mode Decomposition. *IEEE Geosci. Remote Sensing Lett.*, *14*(1), 18–22.

Chen, Y. (2017). Fast dictionary learning for noise attenuation of multidimensional seismic data. *Geophysical Journal International*, *209*(1), 21–31.

Chen, Y. (2018). Non-stationary least-squares complex decomposition for microseismic noise attenuation. *Geophysical Journal International*, *213*(3), 1572–1585.

Chen, Y., & Fomel, S. (2015). Random noise attenuation using local signal-and-noise orthogonalization. *Geophysics*, *80*(6), WD1–WD9.

Chen, Y., Huang, W., Zhang, D., & Chen, W. (2016). An open-source matlab code package for improved rank-reduction 3D seismic data denoising and reconstruction. *Computers & Geosciences*, *95*, 59–66.

Chen, Y., Ma, J., & Fomel, S. (2016). Double-sparsity dictionary for seismic noise attenuation. *Geophysics*, *81*(2), V103–V116.

Chen, Y., & Ma, J. (2014). Random noise attenuation by fx empirical-mode decomposition predictive filtering. *Geophysics*, *79*(3), V81–V91.

Chen, Y., Zhang, D., Jin, Z., Chen, X., Zu, S., Huang, W., & Gan, S. (2016). Simultaneous denoising and reconstruction of 5-D seismic data via damped rank-reduction method. *Geophysical Journal International*, *206*(3), 1695–1717.

Chen, Y., Zhou, Y., Chen, W., Zu, S., Huang, W., & Zhang, D. (2017). Empirical low-rank approximation for seismic noise attenuation. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(8), 4696–4711.

Chu, C., & Stoffa, P. L. (2012). Implicit finite-difference simulations of seismic wave propagation. *Geophysics*, *77*(2), T57–T67.

Clarke, L., Glendinning, I., & Hempel, R. (1994). The MPI message passing interface standard. In *Programming environments for massively parallel distributed systems* (pp. 213–218). Springer.

Cockett, R., Kang, S., Heagy, L. J., Pidlisecky, A., & Oldenburg, D. W. (2015). SimPEG: An open source framework for simulation and gradient based parameter estimation in geophysical applications. *Computers & Geosciences*, *85*, 142–154.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine learning*, *15*(2), 201–221.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data, In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Cui, X., Goel, V., & Kingsbury, B. (2015). Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(9), 1469–1477.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database, In *2009 IEEE conference on computer vision and pattern recognition*. Ieee.

Deng, L., & Platt, J. C. (2014). Ensemble deep learning for speech recognition, In *Fifteenth Annual Conference of the International Speech Communication Association*.

DeVries, P. M., Viégas, F., Wattenberg, M., & Meade, B. J. (2018). Deep learning of aftershock patterns following large earthquakes. *Nature*, *560*(7720), 632.

DeVries, T., & Taylor, G. W. (2017a). Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*.

DeVries, T., & Taylor, G. W. (2017b). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Dickey, J., Borghetti, B., Junek, W., & Martin, R. (2020). Beyond Correlation: A Path-Invariant Measure for Seismogram Similarity. *Seismological Research Letters*, *91*(1), 356–369.

Dietz, L. (2002). Notes on Configuring BINDER_EW: Earthworm's Phase Associator. *URL http://www. isti2. com/ew/ovr/binder_setup. html*.

Dokht, R. M., Kao, H., Visser, R., & Smith, B. (2019). Seismic event and phase detection using time–frequency representation and convolutional neural networks. *Seismological Research Letters*, *90*(2A), 481–490.

Donoho, D. L., & Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, *90*(432), 1200–1224.

Donoho, D. L., & Johnstone, J. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *biometrika*, *81*(3), 425–455.

Draelos, T. J., Ballard, S., Young, C. J., & Brogan, R. (2015). A new method for producing automated seismic bulletins: Probabilistic event detection, association, and location. *Bulletin of the Seismological Society of America*, *105*(5), 2453–2467.

Duarte, M. (2015). Notes on Scientific Computing for Biomechanics and Motor Control. GitHub.

Esser, E., Guasch, L., van Leeuwen, T., Aravkin, A. Y., & Herrmann, F. J. (2018). Total variation regularization strategies in full-waveform inversion. *SIAM Journal on Imaging Sciences*, *11*(1), 376–406.

Fadaee, M., Bisazza, A., & Monz, C. (2017). Data augmentation for low-resource neural machine translation. *arXiv preprint arXiv:1705.00440*.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The annals of statistics*, 209–230.

Fichtner, A. (2010). *Full seismic waveform modelling and inversion*. Springer Science & Business Media.

Fichtner, A., Bunge, H.-P., & Igel, H. (2006). The adjoint method in seismology: I. Theory. *Physics of the Earth and Planetary Interiors*, *157*(1-2), 86–104.

Fischer, T., Kühn, D., & Roth, M. (2020). Microseismic events on the Åknes rockslide in Norway located by a back-projection approach. *Journal of Seismology*, *24*(1), 55–74.

Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons.

Friberg, P., Lisowski, S., Dricker, I., & Hellman, S. (2010). Earthworm in the 21st century, In *EGU General Assembly Conference Abstracts*.

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, *321*, 321–331.

Gaci, S. (2014). The use of wavelet-based denoising techniques to enhance the first-arrival picking on seismic traces. *IEEE Transactions on Geoscience and Remote Sensing*, *52*(8), 4558–4563.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning, In *international conference on machine learning*.

Gebraad, L., Boehm, C., & Fichtner, A. (2020). Bayesian Elastic Full-Waveform Inversion Using Hamiltonian Monte Carlo. *Journal of Geophysical Research: Solid Earth*, *125*(3), e2019JB018428.

Gentili, S., & Michelini, A. (2006). Automatic picking of P and S phases using a neural tree. *Journal of Seismology*, *10*(1), 39–63.

Gharti, H. N., Oye, V., Roth, M., & Kühn, D. (2010). Automated microearthquake location using envelope stacking and robust global optimizationAutomated microearthquake location. *Geophysics*, *75*(4), MA27–MA46.

Gibbons, S. J., Kværna, T., Harris, D. B., & Dodge, D. A. (2016). Iterative strategies for aftershock classification in automatic seismic processing pipelines. *Seismological Research Letters*, *87*(4), 919–929.

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

Girshick, R. (2015). Fast R-CNN. *arXiv:1504.08083 [cs]*.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation, In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets, In *Advances in neural information processing systems*.

Griewank, A., & Walther, A. (2000). Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)*, *26*(1), 19–45.

Grote, M. J., & Sim, I. (2010). Efficient PML for the wave equation. *arXiv preprint arXiv:1001.0319*.

Guitton, A. (2012). Blocky Regularization Schemes for Full-Waveform Inversion. *Geophysical Prospecting*, *60*(5), 870–884.

Guitton, A., Ayeni, G., & Díaz, E. (2012). Constrained full-waveform inversion by model reparameterization. *Geophysics*, *77*(2), R117–R127.

Gutenberg, B. (1956). The energy of earthquakes. *Quarterly Journal of the Geological Society*, *112*(1-4), 1–14.

Gutenberg, B., & Richter, C. F. (1944). Frequency of earthquakes in California. *Bulletin of the Seismological society of America*, *34*(4), 185–188.

Han, J., & van der Baan, M. (2015). Microseismic and seismic denoising via ensemble empirical mode decomposition and adaptive thresholding. *Geophysics*, *80*(6), KS69–KS80.

Harris, D. B., & Dodge, D. A. (2011). An Autonomous System for Grouping Events in a Developing Aftershock Sequence. *Bulletin of the Seismological Society of America*, *101*(2), 763–774.

Hartzell, S. H., & Heaton, T. H. (1983). Inversion of strong ground motion and teleseismic waveform data for the fault rupture history of the 1979 Imperial Valley, California, earthquake. *Bulletin of the Seismological Society of America*, *73*(6A), 1553–1583.

Hauksson, E., Yang, W., & Shearer, P. M. (2012). Waveform relocated earthquake catalog for southern California (1981 to June 2011). *Bulletin of the Seismological Society of America*, *102*(5), 2239–2244.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning, In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, *21*(9), 1263–1284.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN, In *Proceedings of the IEEE international conference on computer vision*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition, In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

He, Q., & Wang, Y. (2021). Reparameterized full-waveform inversion using deep neural networks. *Geophysics*, *86*(1), V1–V13.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of tricks for image classification with convolutional neural networks, In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hennenfent, G., & Herrmann, F. J. (2006). Seismic denoising with nonuniformly sampled curvelets. *Computing in Science & Engineering*, *8*(3), 16–25.

Hore, A., & Ziou, D. (2010). Image quality metrics: PSNR vs. SSIM, In *2010 20th international conference on pattern recognition*. IEEE.

Hu, W., Abubakar, A., & Habashy, T. M. (2009). Simultaneous Multifrequency Inversion of Full-Waveform Seismic Data. *GEOPHYSICS*, *74*(2), R1–R14.

Hu, W., Chen, J., Liu, J., & Abubakar, A. (2018). Retrieving low wavenumber information in FWI: An overview of the cycle-skipping phenomenon and solutions. *IEEE Signal Processing Magazine*, *35*(2), 132–141.

Hu, W., Jin, Y., Wu, X., & Chen, J. (2021). Progressive transfer learning for low-frequency data prediction in full waveform inversion. *Geophysics*, *86*(4), 1–82.

Huang, W., Wang, R., & Chen, Y. (2018). Regularized non-stationary morphological reconstruction algorithm for weak signal detection in microseismic monitoring: methodology. *Geophysical Journal International*, *213*(2), 1189–1211.

Huang, W., Wang, R., Chen, Y., Li, H., & Gan, S. (2016). Damped multichannel singular spectrum analysis for 3D random noise attenuation. *Geophysics*, *81*(4), V261–V270.

Huang, W., Wang, R., Yuan, Y., Gan, S., & Chen, Y. (2016). Signal extraction using randomized-order multichannel singular spectrum analysis. *Geophysics*, *82*(2), V69–V84.

Huang, W., Wang, R., Zu, S., & Chen, Y. (2017). Low-frequency noise attenuation in seismic and microseismic data using mathematical morphological filtering. *Geophysical Journal International*, *211*(3), 1318–1340.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics* (pp. 492–518). Springer.

Hughes, T. W., Williamson, I. A., Minkov, M., & Fan, S. (2019). Wave physics as an analog recurrent neural network. *Science Advances*, *5*(12).

Inbal, A., Clayton, R. W., & Ampuero, J.-P. (2015). Imaging widespread seismicity at midlower crustal depths beneath Long Beach, CA, with a dense seismic array: Evidence for a depth-dependent earthquake size distribution. *Geophysical Research Letters*, *42*(15), 6314–6323.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, In *International conference on machine learning*. PMLR.

Ishii, M., Shearer, P. M., Houston, H., & Vidale, J. E. (2005). Extent, duration and speed of the 2004 Sumatra–Andaman earthquake imaged by the Hi-Net array. *Nature*, *435*(7044), 933–936.

Johnson, C. E., Lindh, A., & Hirshorn, B. (1997). Robust regional phase association.

Kalita, M., Kazei, V., Choi, Y., & Alkhalifah, T. (2019). Regularized full-waveform inversion with automated salt flooding. *Geophysics*, *84*(4), R569–R582.

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.

Kazei, V., Ovcharenko, O., Plotnitskii, P., Peter, D., Zhang, X., & Alkhalifah, T. (2021). Mapping full seismic waveforms to vertical velocity profiles by deep learning. *Geophysics*, *86*(5), 1–50.

Kendall, A., & Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 5574–5584). Curran Associates, Inc.

Kikuchi, M., & Kanamori, H. (1982). Inversion of complex body waves. *Bulletin of the Seismological Society of America*, *72*(2), 491–506.

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

Kirsch, A., Van Amersfoort, J., & Gal, Y. (2019). Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, *32*, 7026–7037.

Kiser, E., & Ishii, M. (2013). Hidden aftershocks of the 2011 Mw 9.0 Tohoku, Japan earthquake imaged with the backprojection method. *Journal of Geophysical Research: Solid Earth*, *118*(10), 5564–5576.

Kissling, E., Kradolfer, U., & Maurer, H. (1995). Program VELEST user's guide-Short Introduction. *Institute of Geophysics, ETH Zurich*.

Kissling, E., Ellsworth, W., Eberhart-Phillips, D., & Kradolfer, U. (1994). Initial reference models in local earthquake tomography. *Journal of Geophysical Research: Solid Earth*, *99*(B10), 19635–19646.

Klein, F. W. (2002). *User's guide to HYPOINVERSE-2000, a Fortran program to solve for earthquake locations and magnitudes* (tech. rep.). US Geological Survey.

Komatitsch, D., & Martin, R. (2007). An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, *72*(5), SM155–SM167.

Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, *30*(1), 25–36.

Kremers, S., Fichtner, A., Brietzke, G., Igel, H., Larmat, C., Huang, L., & Käser, M. (2011). Exploring the potentials and limitations of the time-reversal imaging of finite seismic sources. *Solid Earth*, *2*(1), 95–105.

Kreps, J., Narkhede, N., Rao, J., et al. (2011). Kafka: A distributed messaging system for log processing, In *Proceedings of the NetDB*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks, In *Advances in neural information processing systems*.

Krüger, F., & Ohrnberger, M. (2005). Tracking the rupture of the $M_w = 9.3$ Sumatra earthquake over 1,150 km at teleseismic distance. *Nature*, *435*(7044), 937–939.

Küperkoch, L., Meier, T., Lee, J., & Friederich, W. (2010). Automated determination of P-phase arrival times at regional and local distances using higher order statistics. *Geophysical Journal International*, *181*(2), 1159–1170.

Langet, N., Maggi, A., Michelini, A., & Brenguier, F. (2014). Continuous kurtosis-based migration for seismic event detection and location, with application to Piton de la Fournaise Volcano, La Reunion. *Bulletin of the Seismological Society of America*, *104*(1), 229–246.

Lay, T., Ammon, C. J., Kanamori, H., Koper, K., Sufri, O., & Hutko, A. (2010). Teleseismic inversion for rupture process of the 27 February 2010 Chile (Mw 8.8) earthquake. *Geophysical Research Letters*, *37*(13).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Touresky, D., Hinton, G., & Sejnowski, T. (1988). A theoretical framework for back-propagation, In *Proceedings of the 1988 connectionist models summer school*. CMU, Pittsburgh, Pa: Morgan Kaufmann.

Li, D., & Harris, J. M. (2018). Full Waveform Inversion with Nonlocal Similarity and Model-Derivative Domain Adaptive Sparsity-Promoting Regularization. *Geophysical Journal International*, *215*(3), 1841–1864.

Li, D., Xu, K., Harris, J. M., & Darve, E. (2019). Time-lapse Full Waveform Inversion for Subsurface Flow Problems with Intelligent Automatic Differentiation. *arXiv preprint arXiv:1912.07552*.

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2017). Visualizing the Loss Landscape of Neural Nets.

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

Li, Y. E., & Demanet, L. (2016). Full-Waveform Inversion with Extrapolated Low-Frequency Data. *GEOPHYSICS*, *81*(6), R339–R348.

Li, Z., Meier, M.-A., Hauksson, E., Zhan, Z., & Andrews, J. (2018). Machine learning seismic wave discrimination: Application to earthquake early warning. *Geophysical Research Letters*, *45*(10), 4773–4779.

Li, Z., Peng, Z., Hollis, D., Zhu, L., & McClellan, J. (2018). High-resolution seismic event detection using local similarity for Large-N arrays. *Scientific reports*, *8*(1), 1646.

Lienert, B. R., Berg, E., & Frazer, L. N. (1986). HYPOCENTER: An earthquake location method using centered, scaled, and adaptively damped least squares. *Bulletin of the Seismological Society of America*, *76*(3), 771–783.

Liu, G., Chen, X., Du, J., & Wu, K. (2012). Random noise attenuation using f-x regularized nonstationary autoregression. *Geophysics*, *77*(2), V61–V69.

Liu, L., Ma, J., & Plonka, G. (2018). Sparse graph-regularized dictionary learning for suppressing random seismic noise. *Geophysics*, *83*(3), V215–V231.

Liu, M., Zhang, M., Zhu, W., Ellsworth, W. L., & Li, H. (2020). Rapid characterization of the July 2019 Ridgecrest, California, earthquake sequence from raw seismic data using machine-learning phase picker. *Geophysical Research Letters*, *47*(4), e2019GL086189.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *arXiv:1512.02325 [cs]*, *9905*, 21–37.

Liu, W., Cao, S., & Chen, Y. (2016). Seismic Time-Frequency Analysis via Empirical Wavelet Transform. *IEEE Geosci. Remote Sensing Lett.*, *13*(1), 28–32.

Liu, X., Ben-Zion, Y., & Zigone, D. (2016). Frequency domain analysis of errors in cross-correlations of ambient seismic noise. *Geophysical Supplements to the Monthly Notices of the Royal Astronomical Society*, *207*(3), 1630–1652.

Liu, Y., Fomel, S., & Liu, C. (2015). Signal and noise separation in prestack seismic data using velocity-dependent seislet transform. *Geophysics*, *80*(6), WD117–WD128.

Liu, Y., & Sen, M. K. (2009). A practical implicit finite-difference method: examples from seismic modelling. *Journal of Geophysics and Engineering*, *6*(3), 231–249.

Liu, Y., Li, Y., Lin, H., & Ma, H. (2014). An amplitude-preserved time–frequency peak filtering based on empirical mode decomposition for seismic random noise reduction. *IEEE Geoscience and Remote Sensing Letters*, *11*(5), 896–900.

Lomax, A., Michelini, A., Curtis, A., & Meyers, R. (2009). Earthquake location, direct, global-search methods. *Encyclopedia of complexity and systems science*, *5*, 2449–2473.

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.

Lomax, A., Virieux, J., Volant, P., & Berge-Thierry, C. (2000). Probabilistic earthquake location in 3D and layered models. In *Advances in seismic event location* (pp. 101–134). Springer.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation, In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Louboutin, M., Lange, M., Luporini, F., Kukreja, N., Witte, P. A., Herrmann, F. J., Velesko, P., & Gorman, G. J. (2019). Devito (v3.1.0): an embedded domain-specific language for finite differences and geophysical exploration. *Geoscientific Model Development*, *12*(3), 1165–1187.

Loveman, D. B. (1993). High performance fortran. *IEEE Parallel & Distributed Technology: Systems & Applications*, *1*(1), 25–42.

Luenberger, D. G., & Ye, Y. (1984). *Linear and nonlinear programming* (Vol. 2). Springer.

Luo, Y., & Schuster, G. T. (1991). Wave-equation Traveltime Inversion. *GEOPHYSICS*, *56*(5), 645–653.

Ma, Y., & Hale, D. (2013). Wave-equation reflection traveltime inversion with dynamic warping and full-waveform inversion. *Geophysics*, *78*(6), R223–R233.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models, In *Proc. icml*. Citeseer.

Marmanis, D., Datcu, M., Esch, T., & Stilla, U. (2015). Deep learning earth observation classification using ImageNet pretrained networks. *IEEE Geoscience and Remote Sensing Letters*, *13*(1), 105–109.

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

McBrearty, I. W., Delorey, A. A., & Johnson, P. A. (2019). Pairwise association of seismic arrivals with convolutional neural networks. *Seismological Research Letters*, *90*(2A), 503–509.

McBrearty, I. W., Gomberg, J., Delorey, A. A., & Johnson, P. A. (2019). Earthquake arrival association with backprojection and graph theory. *Bulletin of the Seismological Society of America*, *109*(6), 2510–2531.

Melchior, P., & Goulding, A. D. (2018). Filling the Gaps: Gaussian Mixture Models from Noisy, Truncated or Incomplete Samples. *Astronomy and Computing*, *25*, 183–194.

Meng, L., Ampuero, J.-P., Stock, J., Duputel, Z., Luo, Y., & Tsai, V. (2012). Earthquake in a maze: Compressional rupture branching during the 2012 Mw 8.6 Sumatra earthquake. *Science*, *337*(6095), 724–726.

Mikołajczyk, A., & Grochowski, M. (2018). Data augmentation for improving deep learning in image classification problem, In *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE.

Mosser, L., Dubrule, O., & Blunt, M. J. (2020). Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. *Mathematical Geosciences*, *52*(1), 53–79.

Mousavi, S. M., & Beroza, G. C. (2019). Bayesian-Deep-Learning Estimation of Earthquake Location from Single-Station Observations. *arXiv preprint arXiv:1912.01144*.

Mousavi, S. M., & Beroza, G. C. (2020). A Machine-Learning Approach for Earthquake Magnitude Estimation. *Geophysical Research Letters*, *47*(1).

Mousavi, S. M., Ellsworth, W. L., Zhu, W., Chuang, L. Y., & Beroza, G. C. (2020). Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nature communications*, *11*(1), 1–12.

Mousavi, S. M., & Langston, C. A. (2016a). Adaptive noise estimation and suppression for improving microseismic event detection. *Journal of Applied Geophysics*, *132*, 116–124.

Mousavi, S. M., & Langston, C. A. (2017). Automatic noise-removal/signal-removal based on general cross-validation thresholding in synchrosqueezed domain and its application on earthquake data. *Geophysics*, *82*(4), V211–V227.

Mousavi, S. M., Langston, C. A., & Horton, S. P. (2016). Automatic microseismic denoising and onset detection using the synchrosqueezed continuous wavelet transform. *Geophysics*, *81*(4), V341–V355.

Mousavi, S. M., Langston, C. A., Mostafa Mousavi, S., & Langston, C. A. (2016). Hybrid seismic denoising using higher-order statistics and improved wavelet block thresholding. *Bulletin of the Seismological Society of America*, *106*(4), 1380–1393.

Mousavi, S. M., Sheng, Y., Zhu, W., & Beroza, G. C. (2019). STanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI. *IEEE Access*.

Mousavi, S. M., Zhu, W., Ellsworth, W., & Beroza, G. (2019). Unsupervised clustering of seismic signals using deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, *16*(11), 1693–1697.

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. *Scientific reports*, *9*(1), 1–14.

Mousavi, S. M., & Langston, C. (2016b). Fast and novel microseismic detection using time-frequency analysis. In *SEG Technical Program Expanded Abstracts 2016* (pp. 2632–2636). Society of Exploration Geophysicists.

Müller, R., Kornblith, S., & Hinton, G. E. (2019). When does label smoothing help?, In *Advances in Neural Information Processing Systems*.

Naghizadeh, M. (2012). Seismic data interpolation and denoising in the frequency-wavenumber domain. *Geophysics*, *77*(2), V71–V80.

Nakata, N., & Beroza, G. C. (2016). Reverse time migration for microseismic sources using the geometric mean as an imaging condition. *Geophysics*, *81*(2), KS51–KS60.

Nakata, N., Beroza, G., Sun, J., & Fomel, S. (2016). Migration-based passive-source imaging for continuous data. In *SEG Technical Program Expanded Abstracts 2016* (pp. 2607–2611). Society of Exploration Geophysicists.

NCEDC. (2014). Northern California earthquake data center. UC Berkeley Seismological Laboratory.

Neelamani, R., Baumstein, A. I., Gillard, D. G., Hadidi, M. T., & Soroka, W. L. (2008). Coherent and random noise attenuation using the curvelet transform. *The Leading Edge*, *27*(2), 240–248.

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation, In *Proceedings of the IEEE international conference on computer vision*.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks, In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Oropeza, V., & Sacchi, M. (2011). Simultaneous seismic data denoising and reconstruction via multichannel singular spectrum analysis. *Geophysics*, *76*(3), V25–V32.

Ovcharenko, O., Kazei, V., Kalita, M., Peter, D., & Alkhalifah, T. (2019). Deep learning for low-frequency extrapolation from multioffset seismic data. *Geophysics*, *84*(6), R989–R1001.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Park, Y., Mousavi, S. M., Zhu, W., Ellsworth, W. L., & Beroza, G. C. (2020). Machine-learning-based analysis of the Guy-Greenbrier, Arkansas earthquakes: A tale of two sequences. *Geophysical Research Letters*, *47*(6), e2020GL087032.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch.

Patton, J. M., Guy, M. R., Benz, H. M., Buland, R. P., Erickson, B. K., & Kragness, D. S. (2016). *Hydra–the National Earthquake Information Center's 24/7 Seismic Monitoring, Analysis, Catalog Production, Quality Analysis, and Special Studies Tool Suite*. US Department of the Interior, US Geological Survey.

Peng, Z., & Zhao, P. (2009). Migration of early aftershocks following the 2004 Parkfield earthquake. *Nature Geoscience*, *2*(12), 877–881.

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Permuter, H., Francos, J., & Jermyn, I. (2006). A study of Gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognition*, *39*(4), 695–706.

Perol, T., Gharbi, M., & Denolle, M. (2018). Convolutional neural network for earthquake detection and location. *Science Advances*, *4*(2), e1700578.

Picozzi, M., Bindi, D., Spallarossa, D., Di Giacomo, D., & Zollo, A. (2018). A Rapid Response Magnitude Scale for Timely Assessment of the High Frequency Seismic Radiation. *Scientific Reports*, *8*(1), 8562.

Plessix, R.-E. (2006). A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, *167*(2), 495–503.

Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade* (pp. 55–69). Springer.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640 [cs]*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*.

Ren, Y., Xu, X., Yang, S., Nie, L., & Chen, Y. (2020). A physics-based neural-network way to perform seismic full waveform inversion. *IEEE Access*, *8*, 112266–112277.

Reynolds, D. A., & Rose, R. C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, *3*(1), 72–83.

Richardson, A. (2018a). Generative adversarial networks for model order reduction in seismic full-waveform inversion. *arXiv preprint arXiv:1806.00828*.

Richardson, A. (2018b). Seismic full-waveform inversion using deep learning tools and techniques. *arXiv preprint arXiv:1801.07232*.

Richter, C. F. (1935). An instrumental earthquake magnitude scale. *Bulletin of the seismological society of America*, *25*(1), 1–32.

Roden, J. A., & Gedney, S. D. (2000). Convolution PML (CPML): An efficient FDTD implementation of the CFS–PML for arbitrary media. *Microwave and optical technology letters*, *27*(5), 334–339.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Miccai*, 234–241.

Ross, Z. E., & Ben-Zion, Y. (2014). Automatic picking of direct P, S seismic phases and fault zone head waves. *Geophysical Journal International*, *199*(1), 368–381.

Ross, Z. E., Cochran, E. S., Trugman, D. T., & Smith, J. D. (2020). 3D fault architecture controls the dynamism of earthquake swarms. *Science*, *368*(6497), 1357–1361.

Ross, Z. E., Idini, B., Jia, Z., Stephenson, O. L., Zhong, M., Wang, X., Zhan, Z., Simons, M., Fielding, E. J., Yun, S.-H., et al. (2019). Hierarchical interlocked orthogonal faulting in the 2019 Ridgecrest earthquake sequence. *Science*, *366*(6463), 346–351.

Ross, Z. E., Meier, M.-A., & Hauksson, E. (2018). P wave arrival picking and first-motion polarity determination with deep learning. *Journal of Geophysical Research: Solid Earth*, *123*(6), 5120–5129.

Ross, Z. E., Meier, M.-A., Hauksson, E., & Heaton, T. H. (2018). Generalized Seismic Phase Detection with Deep Learning. *arXiv preprint arXiv:1805.01075*.

Ross, Z. E., Trugman, D. T., Azizzadenesheli, K., & Anandkumar, A. (2020). Directivity modes of earthquake populations with unsupervised learning. *Journal of Geophysical Research: Solid Earth*, *125*(2), e2019JB018299.

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

Ross, Z. E., Yue, Y., Meier, M.-A., Hauksson, E., & Heaton, T. H. (2019). PhaseLink: A deep learning approach to seismic phase association. *Journal of Geophysical Research: Solid Earth*, *124*(1), 856–869.

Rubinstein, J. L., & Beroza, G. C. (2007). Full waveform earthquake location: Application to seismic streaks on the Calaveras fault, California. *Journal of Geophysical Research: Solid Earth*, *112*(B5).

Rücker, C., Günther, T., & Wagner, F. M. (2017). pyGIMLi: An open-source library for modelling and inversion in geophysics. *Computers & Geosciences*, *109*, 106–123.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211–252.

Salamon, J., & Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, *24*(3), 279–283.

Sambridge, M., Rickwood, P., Rawlinson, N., & Sommacal, S. (2007). Automatic differentiation in geophysical inverse problems. *Geophysical Journal International*, *170*(1), 1–8.

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.

Saurel, J.-M., Retailleau, L., Zhu, W., Issartel, S., Satriano, C., & Beroza, G. C. (2021). *Implementation of a new real time seismicity detector for the Mayotte crisis.*

SCEDC. (2013). Southern California Earthquake Center. *Caltech. Dataset.*

Schaff, D. P., Bokelmann, G. H., Ellsworth, W. L., Zanzerkia, E., Waldhauser, F., & Beroza, G. C. (2004). Optimizing correlation techniques for improved earthquake location. *Bulletin of the Seismological Society of America*, *94*(2), 705–721.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, *42*(3), 1–21.

Settles, B. (2009). *Active learning literature survey* (tech. rep.). University of Wisconsin-Madison Department of Computer Sciences.

Seydoux, L., Balestriero, R., Poli, P., De Hoop, M., Campillo, M., & Baraniuk, R. (2020). Clustering earthquake signals and background noises in continuous seismic data with unsupervised deep learning. *Nature communications*, *11*(1), 1–12.

Shan, H., Ma, J., & Yang, H. (2009). Comparisons of wavelets, contourlets and curvelets in seismic denoising. *Journal of Applied Geophysics*, *69*(2), 103–115.

Shelly, D. R. (2020). A high-resolution seismic catalog for the initial 2019 Ridgecrest earthquake sequence: Foreshocks, aftershocks, and faulting complexity. *Seismological Research Letters*, *91*(4), 1971–1978.

Shelly, D. R., Beroza, G. C., & Ide, S. (2007). Non-volcanic tremor and low-frequency earthquake swarms. *Nature*, *446*(7133), 305–307.

Shen, H., & Shen, Y. (2021). Array-Based Convolutional Neural Networks for Automatic Detection and 4D Localization of Earthquakes in Hawai 'i. *Seismological Research Letters*.

Sheng, Y., Denolle, M. A., & Beroza, G. C. (2017). Multicomponent C3 Green's Functions for Improved Long-Period Ground-Motion PredictionMulticomponent C3 Green's Functions for Improved Long-Period Ground-Motion Prediction. *Bulletin of the Seismological Society of America*, *107*(6), 2836–2845.

Shin, H.-C., Tenenholtz, N. A., Rogers, J. K., Schwarz, C. G., Senjem, M. L., Gunter, J. L., Andriole, K. P., & Michalski, M. (2018). Medical image synthesis for data augmentation and anonymization using generative adversarial networks, In *International workshop on simulation and synthesis in medical imaging*. Springer.

Si, X., & Yuan, Y. (2018). Random noise attenuation based on residual learning of deep convolutional neural network. In *SEG Technical Program Expanded Abstracts 2018* (pp. 1986–1990). Society of Exploration Geophysicists.

Siahsar, M. A. N., Gholtashi, S., Abolghasemi, V., & Chen, Y. (2017). Simultaneous denoising and interpolation of 2D seismic data using data-driven non-negative dictionary learning. *Signal Processing*, *141*, 309–321.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), 484–489.

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. (2003). Best practices for convolutional neural networks applied to visual document analysis., In *Icdar*.

Simons, M., Minson, S. E., Sladen, A., Ortega, F., Jiang, J., Owen, S. E., Meng, L., Ampuero, J.-P., Wei, S., Chu, R., Helmberger, D. V., Kanamori, H., Hetland, E., Moore, A. W., & Webb, F. H. (2011). The 2011 magnitude 9.0 Tohoku-Oki earthquake: Mosaicking the megathrust from seconds to centuries. *science*, *332*(6036), 1421–1425.

Sleeman, R., & Van Eck, T. (1999). Robust automatic P-phase picking: An on-line implementation in the analysis of broadband seismogram recordings, In *Physics of the Earth and Planetary Interiors*.

Somala, S. N., Ampuero, J.-P., & Lapusta, N. (2018). Finite-fault source inversion using adjoint methods in 3D heterogeneous media. *Geophysical Journal International*, *214*(1), 402–420.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

SSA. (2021). SSA 2021 Annual Meeting. *Seismological Research Letters*, *92*(2B), 1213–1479.

Su, J., Liu, M., Zhang, Y., Wang, W., Li, H., Yang, J., Li, X., & Zhang, M. (2021). High resolution earthquake catalog building for the 21 May 2021 Yangbi, Yunnan, M S 6.4 earthquake sequence using deep-learning phase picker. *Chinese Journal of Geophysics*, *64*(8), 2647–2656.

Sun, H., & Demanet, L. (2020). Extrapolated Full-Waveform Inversion with Deep Learning. *GEOPHYSICS*, *85*(3), R275–R288.

Suzuki, W., Aoi, S., Sekiguchi, H., & Kunugi, T. (2011). Rupture process of the 2011 Tohoku-Oki mega-thrust earthquake (M9. 0) inverted from strong-motion data. *Geophysical Research Letters*, *38*(7).

Symes, W. W. (2007). Reverse time migration with optimal checkpointing. *Geophysics*, *72*(5), SM213–SM221.

Symes, W. W. (2008). Migration Velocity Analysis and Waveform Inversion. *Geophysical Prospecting*, *56*(6), 765–790.

Symes, W. W. (2015). IWAVE structure and basic use cases. *THE RICE INVERSION PROJECT*, 85.

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., & Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, *35*(5), 1299–1312.

Tan, Y. J., Waldhauser, F., Ellsworth, W. L., Zhang, M., Zhu, W., Michele, M., Chiaraluce, L., Beroza, G. C., & Segou, M. (2021). Machine-Learning-Based High-Resolution Earthquake Catalog Reveals How Complex Fault Structures Were Activated during the 2016–2017 Central Italy Sequence. *The Seismic Record*, *1*(1), 11–19.

Tang, G., & Ma, J. (2011). Application of total-variation-based curvelet shrinkage for three-dimensional seismic data denoising. *IEEE geoscience and remote sensing letters*, *8*(1), 103–107.

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

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

Thurber, C. H. (1985). Nonlinear earthquake location: theory and examples. *Bulletin of the Seismological Society of America*, *75*(3), 779–790.

Tian, Y., Li, Y., & Yang, B. (2014). Variable-Eccentricity Hyperbolic-Trace TFPF for Seismic Random Noise Attenuation. *IEEE Trans. Geoscience and Remote Sensing*, *52*(10), 6449–6458.

Tselentis, G.-A., Martakis, N., Paraskevopoulos, P., Lois, A., & Sokos, E. (2012). Strategy for automated analysis of passive microseismic data based on S-transform, Otsu's thresholding, and higher order statistics. *Geophysics*, *77*(6), KS43–KS54.

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation, In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2018). Deep Image Prior. *arXiv:1711.10925 [cs, stat]*.

Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., & Kulić, D. (2017). Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks, In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*.

Vanacore, E. A., Joyce, J., Ten Brink, U., Fielding, E. J., & Lopez-Venegas, A. (2021). Double Difference Relocations of the 2020 Southwestern Puerto Rico Seismic Sequence. *SSA2021 Annual Meeting*.

Versteeg, R. (1994). The Marmousi experience: Velocity model determination on a synthetic complex data set. *The Leading Edge*, *13*(9), 927–936.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders, In *Proceedings of the 25th international conference on Machine learning*.

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

Vlasenko, A., Köhl, A., & Stammer, D. (2016). The efficiency of geophysical adjoint codes generated by automatic differentiation tools. *Computer Physics Communications*, *199*, 22–28.

Wald, D. J., Helmberger, D. V., & Hartzell, S. H. (1990). Rupture process of the 1987 Superstition Hills earthquake from the inversion of strong-motion data. *Bulletin of the Seismological Society of America*, *80*(5), 1079–1098.

Waldhauser, F. (2001). hypoDD–A program to compute double-difference hypocenter locations.

Waldhauser, F., & Ellsworth, W. L. (2000). A double-difference earthquake location algorithm: Method and application to the northern Hayward fault, California. *Bulletin of the seismological society of America*, *90*(6), 1353–1368.

Waldhauser, F., & Schaff, D. P. (2008). Large-scale relocation of two decades of Northern California seismicity using cross-correlation and double-difference methods. *Journal of Geophysical Research: Solid Earth*, *113*(B8).

Walker, K. T., Ishii, M., & Shearer, P. M. (2005). Rupture details of the 28 March 2005 Sumatra Mw 8.6 earthquake imaged with teleseismic P waves. *Geophysical Research Letters*, *32*(24).

Walter, J. I., Ogwari, P., Thiel, A., Ferrer, F., & Woelfel, I. (2021). EasyQuake: Putting machine learning to work for your regional seismic network or local earthquake study. *Seismological Society of America*, *92*(1), 555–563.

Wang, B., Wu, R., Chen, X., & Li, J. (2015). Simultaneous seismic data interpolation and denoising with a new adaptive method based on dreamlet transform. *Geophysical Journal International*, *201*(2), 1180–1192.

Wang, J., Xiao, Z., Liu, C., Zhao, D., & Yao, Z. (2019). Deep learning for picking seismic arrival times. *Journal of Geophysical Research: Solid Earth*, *124*(7), 6612–6624.

Wang, K., Zhu, W., Ellsworth, W. L., & Beroza, G. C. (2019). Earthquake Detection in Develocorder Films: An Image-based Detection Neural Network for Analog Seismograms, In *AGU Fall Meeting 2019*. AGU.

Wang, R., Schmandt, B., Zhang, M., Glasgow, M., Kiser, E., Rysanek, S., & Stairs, R. (2020). Injection-Induced Earthquakes on Complex Fault Zones of the Raton Basin Illuminated by Machine-Learning Phase Picker and Dense Nodal Array. *Geophysical Research Letters*, *47*(14), e2020GL088168.

Weber, B., Becker, J., Hanka, W., Heinloo, A., Hoffmann, M., Kraft, T., Pahlke, D., Reinhardt, J., Saul, J., & Thoms, H. (2007). SeisComP3—Automatic and interactive real time data processing, In *Geophysical Research Abstracts*.

Wishart, J. (1928). The generalised product moment distribution in samples from a normal multivariate population. *Biometrika*, 32–52.

Witte, P., Louboutin, M., Lensink, K., Lange, M., Kukreja, N., Luporini, F., Gorman, G., & Herrmann, F. J. (2018). Full-waveform inversion, part 3: Optimization. *The Leading Edge*, *37*(2), 142–145.

Witte, P. A., Loubouting, M., & Herrmann, F. J. (2020). *JUDI4Flux: Seismic modeling for deep learning* (Version v0.1.1). Zenodo.

Woollam, J., Rietbrock, A., Leitloff, J., & Hinz, S. (2020). HEX: Hyperbolic Event eXtractor, a Seismic Phase Associator for Highly Active Seismic Regions. *Seismological Research Letters*, *91*(5), 2769–2778.

Wu, R.-S., Luo, J., & Wu, B. (2014). Seismic envelope inversion and modulation signal model. *Geophysics*, *79*(3), WA13–WA24.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wu, Y., Lin, Y., & Zhou, Z. (2018). Inversionet: Accurate and Efficient Seismic-Waveform Inversion with Convolutional Neural Networks, In *SEG Technical Program Expanded Abstracts 2018*, Anaheim, California, Society of Exploration Geophysicists.

Wu, Y., Lin, Y., Zhou, Z., Bolton, D. C., Liu, J., & Johnson, P. (2018). DeepDetect: A cascaded region-based densely connected network for seismic event detection. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(1), 62–75.

Wu, Y., & McMechan, G. A. (2019). Parametric Convolutional Neural Network-Domain Full-Waveform Inversion. *GEOPHYSICS*, *84*(6), R881–R896.

Wu, Y., & McMechan, G. A. (2020). CNN-boosted full-waveform inversion. In *SEG Technical Program Expanded Abstracts 2020* (pp. 1526–1530). Society of Exploration Geophysicists.

Xu, K., & Darve, E. (2020). ADCME: Learning spatially-varying physical fields using deep neural networks. *arXiv preprint arXiv:2011.11955*.

Xu, Y., Koper, K. D., Sufri, O., Zhu, L., & Hutko, A. R. (2009). Rupture imaging of the Mw 7.9 12 May 2008 Wenchuan earthquake from back projection of teleseismic P waves. *Geochemistry, Geophysics, Geosystems*, *10*(4).

Yang, F., & Ma, J. (2019). Deep-Learning Inversion: A next-Generation Seismic Velocity Model Building Method. *GEOPHYSICS*, *84*(4), R583–R599.

Yang, S., Hu, J., Zhang, H., & Liu, G. (2021). Simultaneous Earthquake Detection on Multiple Stations via a Convolutional Neural Network. *Seismological Society of America*, *92*(1), 246–260.

Yang, W., & Ben-Zion, Y. (2010). An algorithm for detecting clipped waveforms and suggested correction procedures. *Seismological Research Letters*, *81*(1), 53–62.

Yang, Y., Engquist, B., Sun, J., & Hamfeldt, B. F. (2018). Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion. *Geophysics*, *83*(1), R43–R62.

Yeck, W. L., Patton, J. M., Johnson, C. E., Kragness, D., Benz, H. M., Earle, P. S., Guy, M. R., & Ambruz, N. B. (2019). GLASS3: A Standalone Multiscale Seismic Detection AssociatorGLASS3: A Standalone Multiscale Seismic Detection Associator. *Bulletin of the Seismological Society of America*, *109*(4), 1469–1478.

Yeck, W. L., Patton, J. M., Ross, Z. E., Hayes, G. P., Guy, M. R., Ambruz, N. B., Shelly, D. R., Benz, H. M., & Earle, P. S. (2021). Leveraging Deep Learning in Global 24/7 Real-Time Earthquake Monitoring at the National Earthquake Information Center. *Seismological Society of America*, *92*(1), 469–480.

Yee, K. (1966). Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on antennas and propagation*, *14*(3), 302–307.

Yi, X., Walia, E., & Babyn, P. (2019). Generative adversarial network in medical imaging: A review. *Medical image analysis*, 101552.

Yoon, C. (2021). A High-Resolution View of the 2020 Puerto Rico Earthquake Sequence With Machine Learning. *SSA2021 Annual Meeting*.

Yoon, C. E., O'Reilly, O., Bergen, K. J., & Beroza, G. C. (2015). Earthquake detection through computationally efficient similarity search. *Science advances*, *1*(11), e1501057.

Yu, E., Acharya, P., Bhaskaran, A., Chen, S., Andrews, J. R., Thomas, V., Ross, Z. E., Hauksson, E., & Clayton, R. W. (2019). *Cloud Computing and Big Data – Using the Southern California*

Earthquake Data Center (SCEDC) and the Southern California Seismic Network (SCSN) Products and Services for Earthquake Research.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., et al. (2010). Spark: Cluster computing with working sets. *HotCloud*, *10*(10-10), 95.

Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. (2013). Discretized streams: Fault-tolerant streaming computation at scale, In *Proceedings of the twenty-fourth ACM symposium on operating systems principles*.

Zhan, Z. (2020). Distributed acoustic sensing turns fiber-optic cables into sensitive seismic antennas. *Seismological Research Letters*, *91*(1), 1–15.

Zhang, C., & van der Baan, M. (2018). Multicomponent microseismic data denoising by 3D shearlet transform. *Geophysics*, *83*(3), A45–A51.

Zhang, J., Hao, J., Zhao, X., Wang, S., Zhao, L., Wang, W., & Yao, Z. (2016). Restoration of clipped seismic waveforms using projection onto convex sets method. *Scientific Reports*, *6*, 39056.

Zhang, M., Ellsworth, W. L., & Beroza, G. C. (2019). Rapid earthquake association and location. *Seismological Research Letters*, *90*(6), 2276–2284.

Zhang, M., & Wen, L. (2015). An effective method for small event detection: match and locate (M&L). *Geophysical Journal International*, *200*(3), 1523–1537.

Zhang, P., Han, L., Xu, Z., Zhang, F., & Wei, Y. (2017). Sparse blind deconvolution based low-frequency seismic data reconstruction for multiscale full waveform inversion. *Journal of Applied Geophysics*, *139*, 91–108.

Zhang, X., Chen, H., Zhang, W., Tian, X., & Chu, F. (2021). Generalized neural network trained with a small amount of base samples: Application to event detection and phase picking in downhole microseismic monitoring. *Geophysics*, *86*(5), 1–66.

Zhang, X., Zhang, J., Yuan, C., Liu, S., Chen, Z., & Li, W. (2020). Locating induced earthquakes with a network of seismic stations in Oklahoma via a deep learning method. *Scientific Reports*, *10*(1), 1–12.

Zhang, Y., Feng, W., Xu, L., Zhou, C., & Chen, Y. (2009). Spatio-temporal rupture process of the 2008 great Wenchuan earthquake. *Science in China Series D: Earth Sciences*, *52*(2), 145–154.

Zhang, Z., Mei, J., Lin, F., Huang, R., & Wang, P. (2018). Correcting for salt misinterpretation with full-waveform inversion. In *SEG Technical Program Expanded Abstracts 2018* (pp. 1143–1147). Society of Exploration Geophysicists.

Zheng, J., Harris, J. M., Li, D., & Al-Rumaih, B. (2020). SC-PSNET: A deep neural network for automatic P-and S-phase detection and arrival-time picker using 1C recordings. *Geophysics*, *85*(4), U87–U98.

Zheng, J., Lu, J., Peng, S., & Jiang, T. (2018). An automatic microseismic or acoustic emission arrival identification scheme with deep recurrent neural networks. *Geophysical Journal International*, *212*(2), 1389–1397.

Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2017). Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*.

Zhou, P., Ellsworth, W., Yang, H., Tan, Y., Beroza, G., & Chu, R. (2021). Machine learning facilitated earthquake detections near the Weiyuan shale gas reservoir, Sichuan, China. *Earth and Planetary Physics*.

Zhou, Y., Li, S., Zhang, D., & Chen, Y. (2017). Seismic noise attenuation using an online subspace tracking algorithm. *Geophysical Journal International*, *212*(2), 1072–1097.

Zhou, Y., Shi, C., Chen, H., Xie, J., Wu, G., & Chen, Y. (2017). Spike-like blending noise attenuation using structural low-rank decomposition. *IEEE Geoscience and Remote Sensing Letters*, *14*(9), 1633–1637.

Zhou, Y., Yue, H., Kong, Q., & Zhou, S. (2019). Hybrid event detection and phase-picking algorithm using convolutional and recurrent neural networks. *Seismological Research Letters*, *90*(3), 1079–1087.

Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, *23*(4), 550–560.

Zhu, L., Peng, Z., McClellan, J., Li, C., Yao, D., Li, Z., & Fang, L. (2019). Deep learning for seismic phase detection and picking in the aftershock zone of 2008 Mw7. 9 Wenchuan Earthquake. *Physics of the Earth and Planetary Interiors*, *293*.

Zhu, L., Liu, E., & McClellan, J. H. (2015). Seismic data denoising through multiscale and sparsity-promoting dictionary learning. *Geophysics*, *80*(6), WD45–WD57.

Zhu, L., Liu, E., & McClellan, J. H. (2017). Sparse-Promoting Full-Waveform Inversion Based on Online Orthonormal Dictionary Learning. *GEOPHYSICS*, *82*(2), R87–R107.

Zhu, W., & Beroza, G. C. (2018). PhaseNet: A Deep-Neural-Network-Based Seismic Arrival Time Picking Method. *arXiv preprint arXiv:1803.03211*.

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

Zhu, W., Ian, W. M., Mostafa, M. S., L., E. W., & Beroza, G. C. (2021). Earthquake Phase Association using a Bayesian Gaussian Mixture Model, In *2021 SSA Annual Meeting*.

Zhu, W., McBrearty, I. W., Mousavi, S. M., Ellsworth, W. L., & Beroza, G. C. (2021). Earthquake Phase Association using a Gaussian Mixture Model. *SSA2021 Annual Meeting*.

Zhu, W., Mousavi, S. M., & Beroza, G. C. (2019). Seismic signal denoising and decomposition using deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(11), 9476–9488.

Zhu, W., Tai, K. S., Mousavi, S. M., & Beroza, G. C. (2019). An end-to-end earthquake monitoring method for joint earthquake detection and association using deep learning, In *AGU Fall Meeting 2019*. AGU.

Zhu, W., Xu, K., Darve, E., & Beroza, G. C. (2020). A General Approach to Seismic Inversion with Automatic Differentiation. *arXiv:2003.06027 [physics]*.

Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., & Le, Q. V. (2020). Rethinking pre-training and self-training. *arXiv preprint arXiv:2006.06882*.