AN UNSUPERVISED LEARNING APPROACH TO SEISMIC WAVEFORM CLASSIFICATION VIA REPRESENTATION LEARNING

by

Onur EFE

B.S., Mechatronics Engineering, Yıldız Technical University, 2017

B.S., Electronics and Communication Engineering, Yıldız Technical University, 2021

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Physics

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I want to express my deepest thanks to my advisor, Arkadaş Özakın. He gave me room to embrace my creativity and follow my intuition while getting as excited as I get by the research findings, although he is much more mature academically. Due to the intuitive nature of Machine Learning, it's generally hard to base why you do things in a specific way, and most of the time, there is no way of knowing beforehand if your method will give good results. As a result, I wasted too much time trying things that led to nowhere, and he was patient when I was wasting our precious time with experiments leading nowhere. Besides, his perspective on the research helped me towards the more simple and elegant solutions, while I tended to overcomplicate things.

I want to thank my wife, Rüya, for supporting me in all cases and believing in me more than myself. She was patient when I had stolen countless hours from our lives for academic work. Her existence and support gave me endless motivation and patience to make things work.

I want to thank my family for understanding my circumstances, their patience, and the times I stole from them. And my friends for their sympathy and support. Even when I had temporarily forgotten them due to excessive working hours; they were there whether I asked for them.

ABSTRACT

AN UNSUPERVISED LEARNING APPROACH TO SEISMIC WAVEFORM CLASSIFICATION VIA REPRESENTATION LEARNING

Seismology is an observation-based science and requires well-classified seismic signals to expand its boundaries. One of the most basic classification problems is to separate earthquake-based seismic activity from background noise. Machine learning algorithms are recently introduced as a solution to this problem. Although unsupervised learning-based algorithms have been used for specific cases (certain seismic events, time intervals, geographical regions, etc.), most proposed algorithms are supervised learning-based. Consequently, general-purpose unsupervised learning algorithms have not been developed yet.

Supervised learning-based models may have biases due to the dataset used in training, which can lead to poor results for observational purposes. Considering that the seismic waveforms are labeled in the light of current information, it's clear that training the models using these labels could block potential discoveries.

Developing methods with complementary biases is important to fill the blind spots of supervised learning-based models. Our primary motivation in this thesis is to develop unsupervised learning-based general-purpose detection methods that distinguish earthquake-based seismic activity from background noise. Based on this motivation, deep learning-based methods that can classify waveforms using one or more stations have been developed.

ÖZET

GÖSTERİM ÖĞRENME YOLUYLA SEİSMİK DALGA FORMU SINIFLANDIRMASI İÇİN DENETİMSİZ ÖĞRENME YAKLAŞIMI

Sismoloji, gözlem odaklı bir araştırma alanıdır ve sınırlarını genişletmek için doğru sınıflandırılmış sismik sinyallere ihtiyaç duyar. En temel sınıflandırma problemlerinden biri de önemli sismik aktiviteyi arka plan gürültüsünden ayırmaktır. Yakın zamanda Makine Öğrenmesi algoritmaları bu problemin çözümü için önerilmiştir. Çeşitli özel durumlar (belirli seismik olaylar, zaman dilimleri ve coğrafi bölgeler vb.) için uygulanmış olan denetimsiz öğrenme tabanlı algoritmalar olmakla birlikte, önerilen algoritmaların çoğu denetimli öğrenmeye dayalıdır. Sonuç olarak genel amaçlı kullanıma uygun denetimsiz öğrenme algoritmaları henüz geliştirilmemiştir.

Denetimli öğrenmeye dayalı modeller, eğitimde kullanılan verisetinden kaynaklı eğilimlere sahip olabilir ve bu da gözlemsel amaçlar için kötü sonuçlara yol açabilir. Sismik sinyallerin şuanki bilgilerin ışığında etiketlendiğini düşünürsek modellerin bu etiketlere göre eğitilmesinin potansiyel keşiflerin önüne geçebileceği açıktır.

Tamamlayıcı eğilimlere sahip metotlar geliştirmek denetimli öğrenmeye dayalı modellerin kör noktalarını doldurmak için önemlidir. Bu tezdeki temel motivasyonumuz, arka plan gürültüsü ile deprem kaynaklı sismik aktiviteyi ayırt edebilen denetimsiz öğrenme tabanlı ve genel amaçlı kullanıma uygun yöntemler oluşturmaktır. Bu motivasyon temelinde, bir veya birden fazla istasyon verisi kullanarak sınıflandırma yapabilen Derin Öğrenme tabanlı yöntemler geliştirilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xii
LIST OF SYMBOLS xiii
LIST OF ACRONYMS/ABBREVIATIONS xv
1. INTRODUCTION 1
2. BACKGROUND 4
2.1. Seismology 4
2.1.1. Earthquakes 4
2.1.1.1. Source of the Earthquakes 4
2.1.1.2. Propagation of Seismic Waves 4
2.1.1.3. Earth's Internal Structure 5
2.1.1.4. Instruments 6
2.1.1.5. Earthquake Magnitude Metrics 6
2.1.2. Related Problems of Seismology 7
2.1.2.1. Improving Limit of Detection 7
2.1.2.2. Earthquake Prediction 8
2.2. Feed Forward Neural Networks 8
2.2.1. Training 9
2.2.2. Architectures 10
2.2.2.1. Multilayer Perceptrons 10
2.2.2.2. Convolutional Neural Networks 11
2.2.2.3. Graph Neural Networks 12
3. SINGLE STATION DETECTION 14
3.1. Introduction 14
3.2. Training and Testing Procedures 14

3.2.1. Datasets and Preprocessing 14
3.2.2. Training 15
3.2.2.1. Autoencoder Training 16
3.2.2.2. Autoencoder Ensemble Training 16
3.2.3. Testing 17
3.3. Methods 18
3.3.1. Building Blocks 19
3.3.1.1. CNN Autoencoder 19
3.3.1.2. Cross-covariance Layer 22
3.3.1.3. Maximum Cross-covariance Metric 23
3.3.1.4. Cross-covariance Weighted Average Metric 23
3.3.2. STA/LTA 24
3.3.3. Autocovariance at Latent Space 25
3.3.4. Augmentation Cross-covariances 25
3.3.5. Representation Cross-covariances 27
3.4. Investigating Latent Representations 29
3.4.1. Samples for Randomly Parametrized CNN Autoencoders 29
3.4.2. Samples for Autocovariance Method 30
3.4.3. Samples for Representation Cross-covariances Method 31
3.5. Results 31
3.5.1. Effect of Training Duration on Performance 32
3.5.2. Relationship Between Validation Loss and Performance 34
3.5.3. Performance Metrics and Comparison with the State of the Art 35
3.6. Discussion and Future Work 38
4. MULTI STATION DETECTION 41
4.1. Introduction 41
4.2. Training and Testing Procedures 42
4.2.1. CNN Autoencoder 43
4.2.2. GNN Autoencoder 43
4.2.2.1. Adjacency Matrix 43
4.2.2.2. Loss Function 44

4.3. Building Blocks . . . 45
4.3.1. CNN Autoencoder . . . 45
4.3.2. Cross-covariance of Node Feature Vectors . . . 45
4.3.3. Graph Attention with Translation Alignment . . . 45
4.3.4. Graph Attention with Trainable Alignment . . . 49
4.3.5. Multi-Head Graph Attention . . . 50
4.4. Methods . . . 52
4.4.1. Detection with Network Cross-covariance of Representations . . . 52
4.4.2. Detection and Beamforming with Graph Attention Autoencoder 53
4.4.2.1. Working Principle . . . 53
4.4.2.2. Architecture . . . 54
4.5. Results . . . 54
4.5.1. Beamforming Results . . . 54
4.5.2. Detection Results . . . 55
4.5.2.1. Network Cross-covariance of Representations . . . 55
4.5.2.2. Detection with Graph Attention Autoencoder . . . 56
4.6. Discussion and Future Work . . . 57
5. CONCLUSION . . . 59
REFERENCES . . . 60
APPENDIX A: SAMPLES FOR SINGLE STATION METHODS . . . 65
A.1. Randomly Parametrized Model . . . 65
A.2. Autocovariance Method . . . 69
A.3. Representation Cross-covariances Method . . . 73
APPENDIX B: SAMPLES FOR MULTI STATION METHODS . . . 79
APPENDIX C: ROC-AUC AND VALIDATION LOSS RELATION . . . 85
APPENDIX D: STA/LTA PARAMETER ADJUSTMENT . . . 88

LIST OF FIGURES

Figure 2.1. Multilayer Perceptron with single hidden layer. 10

Figure 3.1. CNN Autoencoder architecture. 19

Figure 3.2. Downsampling layer structure. 20

Figure 3.3. Residual layer structure. 21

Figure 3.4. Upsampling layer structure. 21

Figure 3.5. Diagram of Autocovariance method. 26

Figure 3.6. Diagram of Augmentation Cross-covariances method. 27

Figure 3.7. Diagram of Representation Cross-covariances method. 28

Figure 3.8. ROC-AUC scores concerning training duration. 32

Figure 3.9. ROC-AUC scores concerning training duration for low SNR case. . 33

Figure 3.10. ROC curves. 35

Figure 3.11. ROC curves for low SNR case. 36

Figure 4.1. Network Cross-covariance of representations. 51

Figure 4.2. Graph Attention Autoencoder architecture. 53

Figure 4.3. Network Cross-covariance performance over training duration. . . 55

Figure 4.4. GNN Autoencoder Detector performance concerning training duration. 56

Figure A.1. Randomly parametrized Autoencoder earthquake sample 1. 65

Figure A.2. Randomly parametrized Autoencoder earthquake sample 2. 66

Figure A.3. Randomly parametrized Autoencoder noise sample 1. 67

Figure A.4. Randomly parametrized Autoencoder noise sample 1. 68

Figure A.5. Autocovariance method earthquake sample 1. 69

Figure A.6. Autocovariance method earthquake sample 2. 70

Figure A.7. Autocovariance method noise sample 1. 71

Figure A.8. Autocovariance method noise sample 2. 72

Figure A.9. Representation Cross-covariances method earthquake sample 1. . . 73

Figure A.10. Representation Cross-covariances method earthquake sample 2. . . 74

Figure A.11. Representation Cross-covariances method noise sample 1. 75

Figure A.12. Representation Cross-covariances method noise sample 2. 76

Figure A.13. Representation Cross-covariances method noise sample 3. 77

Figure A.14. Representation Cross-covariances method noise sample 4. 78

Figure B.1. GNN Autoencoder earthquake sample 1. 79

Figure B.2. GNN Autoencoder earthquake sample 2. 80

Figure B.3. GNN Autoencoder earthquake sample 3. 81

Figure B.4. GNN Autoencoder earthquake sample 4. 82

Figure B.5. GNN Autoencoder noise sample 1. 83

Figure B.6. GNN Autoencoder noise sample 2. 84

Figure C.1. Autocovariance ROC-AUC and validation loss. 85

Figure C.2. Augmentation Cross-covariances ROC-AUC and validation loss. . 86

Figure C.3. Representation Cross-covariances ROC-AUC and validation loss. . 87

LIST OF TABLES

Table 2.1. Seismometer codes. 6

Table 3.1. Properties of the STEAD and INSTANCE datasets. 15

Table 3.2. ROC-AUC metrics of models for waveforms with low SNR. 37

Table 3.3. ROC-AUC metrics of models. 38

Table D.1. STA/LTA adjustment parameters and results. 88

LIST OF SYMBOLS

A	Graph Adjacency Matrix
$\mathcal{F}$	Fast Fourier Transform operator
$\mathcal{F}^{-1}$	Inverse Fast Fourier Transform operator
h	Center node representation
$h^{(A)}$	Center node representation at attention space
$h^{(F)}$	Center node representation at fusion space
g	Neighbor node representation
$g^{(A)}$	Neighbor representation at attention space
$g^{(F)}$	Neighbor representation at fusion space
L_{ij}	Alignment operator which aligns node j to node i
M	Classification metric
$\mathbb{R}$	Set of real numbers
$\Re$	Operator taking real part of a complex number
$\mathbf{t}_1$	Unit translation operator
$\mathbf{r}_2$	Unit alignment operator which is trainable
$W^{(A)}$	Attention space projection matrix
$W^{(F)}$	Fusion space projection matrix
x	Input waveform sample
y	Reconstructed waveform sample
$\mathbb{Z}$	Set of integers
α	Attention coefficients for different delays
β_1	Decay rate for first order gradient moving averages
β_2	Decay rate for second order gradient moving averages
ϵ	Small constant for numerical stability of the optimizer
η	Nonlinear activation function
θ	Parameter set
κ	Attention coefficients for neighboring nodes

Λ	Alignment generator at Fourier Basis
ν	Mean value of the Gaussian distribution
σ	Standard Deviation of the Gaussian Distribution
σ_n	Cross-covariance of two multichannel time-series signals
τ	Time delay between two multichannel time-series signals
$\oplus$	Concatenation operator
$*$	Convolution operator

LIST OF ACRONYMS/ABBREVIATIONS

1D	One Dimensional
ANN	Artificial Neural Network
AUC	Area Under the Curve
BN	Batch Normalization
CNN	Convolutional Neural Network
FFT	Fast Fourier Transform
FN	False Negatives
FNN	Feedforward Neural Network
FP	False Positives
FPR	False Positive Rate
GNN	Graph Neural Network
IFFT	Inverse Fast Fourier Transform
MLP	Multi Layer Perceptron
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristics
ROC-AUC	Area Under the Receiver Operating Characteristics Curve
SNR	Signal to Noise Ratio
SOM	Self Organizing Map
SSL	Self Supervised Learning
STA/LTA	Short Term Average/Long Term Average
TN	True Negatives
TP	True Positives
TPR	True Positive Rate

1. INTRODUCTION

Seismic event classification is one of the problems in seismology with broad applications. The problem is solved using classical algorithms with tunable parameters that an analyst adjusts. These algorithms give decent results for high Signal to Noise Ratio (SNR) signals. However, their performance degrades as the event magnitude decreases. Recently, various supervised learning-based machine learning methods are proposed [1–5] which is shown to outperform classical methods. Proposed supervised learning methods are shown to improve the detection performance towards the lower magnitudes and SNRs with very good performance metrics [1]. However, performance measurement of supervised learning-based models can be tricky since it's dependent on the specifics of the training and testing datasets. A recently written review reports significant performance variations of seismic event classification models for a set of training and testing datasets [6]. This problem is called overfit in the literature and is related to the models' lack of generalization abilities. One significant cause of the overfit is related to labeling specifics of the training set as well as its sample distributions. Although the problem can be fixed to some degree by using bigger datasets with quality labels, there are limits to this. The major limiting factor is related to the labeling procedure itself. Since labels are prepared using classical algorithms and human effort, supervised learning-based models have biases similar to the current classical approaches. For observational purposes, it's important to have less or complementary biased methods.

Unsupervised learning-based methods emerge as a candidate solution to the supervision signal-related bias problem. Varios unsupervised learning-based algorithms are applied to different problems of seismology due to mentioned considerations [7–16]. Used algorithms involve dimensionality reduction and classification steps. While dimensionality reduction phases were carried out by manual feature engineering, Principal Component Analysis (PCA), Self Organizing Maps (SOM), deep learning-based methods, and clustering algorithms often solve the classification problem. Recently, [12]

used CNN Autoencoder and K-Means clustering to classify local-teleseismic events while [13] has applied Deep Scattering Network and Gaussian Mixture Clustering approach to separate seismic events from background noise. However, according to our knowledge, all applications of unsupervised learning to seismology are case-specific, and they are not validated to be efficient in cross-domains. Most methods use clustering algorithms, although they are reported to perform poorly on cross-domain cases [17]. Another frequently used method, Self Organizing Maps, also suffers from the same problem and is prone to diverge, introducing further challenges for general application. Due to the mentioned problems, the demand for general-purpose unsupervised learning-based detection tools has not been satisfied yet.

In this thesis, we propose solutions to the unsupervised learning-based seismic event classification problem on a broader scale. The main idea behind these approaches is to focus on designing models that learn the proper way of representing the waveforms from the data so that the classification becomes almost trivial. In this context, instead of using machine learning models at the classification phase, we solve the classification problem by using simple functions and deep autoencoders to find proper representations to make this possible. Our approach involves designing the models, finding suitable representations from the data, and then investigating the obtained representations.

Proposed solutions are grouped as single station and multistation methods. The classical workflow involves applying single-station detection methods to classify seismic events at a seismic station level and then fusing this information by using phase association algorithms to obtain a robust classification label. Due to their part in classical workflow, we aimed to develop reliable single-station detection methods since we believe successful single-station method has its application area. Multistation methods, on the other hand, are less popular. However, we can base our detection methods on first principles, such as earthquake and noise waveforms having different coherency lengths. This allows us to base our methods on more solid grounds, making multistation methods appealing.

We benefit from the Convolutional Neural Network (CNN) Autoencoders in all proposed models. Additionally, we use Graph Neural Network (GNN) Autoencoders for multistation methods, which provide a way to fuse information from the different stations. Autoencoders learn efficient, high-level, and robust representations from the data, making them suitable for our purposes. Besides, they can be used for denoising applications. Especially GNN Autoencoders, which are applied to a network of stations, can eliminate noise signals, favoring the information obtained from other stations. As well as being used for detection purposes, we also show that the proposed method can be used for beamforming applications at regional scales. According to our knowledge, this will also make a novel contribution.

2. BACKGROUND

2.1. Seismology

2.1.1. Earthquakes

2.1.1.1. Source of the Earthquakes. Occasionally, the dynamics of the earth lead to the sudden release of seismic energy, and we call this phenomenon an earthquake. Although deep earthquakes may have different dynamics, the common cause of the energy release depends on the plates of the crust (tectonic plates) moving with different speeds or directions, inducing stress along the crust. The places where two or more tectonic plates meet are called fault zones, and earthquake sources are mostly within these areas. We can model a typical earthquake as a series of rupture events at the fault zone.

We can understand the physics of the phenomena if we consider that the forces giving rise to earthquakes are internal. All forces acting on the fault zones lead to forming pairs with their sum zero. However, torsion is different than zero, giving rise to other stresses (compressional and shear stresses) propagating through the earth medium.

2.1.1.2. Propagation of Seismic Waves. After an earthquake, seismic energy propagates through various layers of the earth according to its radiation pattern and the structural composition of the propagation medium. To understand the propagation dynamics, we must solve the seismic wave equation for the homogenous space case, which is more straightforward. The solution to the seismic wave equation includes two types of waves with different mechanisms: compression and shear waves. While shear waves or S waves are composed of pure shearing (density for a position of space doesn't change), P waves involve both compression and shearing. P waves are also named primary waves because they are faster than S waves around a factor of 1.4. When we

apply boundary conditions the Earth imposes, we see that different solutions also exist. These solutions are called surface waves, which are Love and Rayleigh waves, and their polarization discriminates them. Surface waves propagate are the slowest, but their propagation distance is much higher since they are trapped in the earth's surface.

Even though the accurate solution to the propagation problem requires detailed information about the structural composition of the earth and the use of computational methods, we can get a rough estimate by considering that the earth is spherically symmetric and propagating waves have short wavelengths compared to changes in the composition of soil along the propagation direction. Under this assumption, we can approximate propagating waves to rays as in the optics, which eases to establish intuition. Besides that, Snell's law of refraction can also be applied, which gives crucial insights into the propagation characteristics of the seismic waves when we consider the earth's structure. One of the simplest models of the Earth is based on the fact that pressure increases with depth and, therefore, seismic refractive index. This leads to seismic waves propagating through curved paths even if the earth is flat, making the inverse problem difficult. The rather detailed model takes the earth as a stack of layers with different properties. Seismic wave travel times change discontinuously with location if seismic waves pass through layers with different properties. These discontinuities are called seismic discontinuities and are used to define the boundaries of the layers.

2.1.1.3. Earth's Internal Structure. Information we currently have about the earth's internal structure is partially obtained by studying the propagation of seismic waves. Earth has three main layers: crust, mantle, and core. The crust is the earth's outermost layer and has the lowest thickness. It is composed of rocks with low density. The mantle is the layer below the crust and has the highest thickness. It is of molten and solid parts, while the uppermost part is solid. The core is the innermost layer of the earth and has the highest density. It's composed of an inner and outer core, while both are solid.

2.1.1.4. Instruments. Seismic waves are detected by seismometers capable of measuring the ground acceleration (ground velocity and displacement are obtained by integrating the acceleration). Instruments are labeled according to their frequency range, sensitivity, and measurement axis. Below is the list of widely used seismometer codes and their meanings.

Table 2.1. Seismometer codes.

Frequency range code	Sensitivity code	Direction code
H High frequency	H High gain	N North-South horizontal
B Broadband	N Strong motion	E East-West horizontal
L Low frequency		Z Vertical

Codes are concatenated to form a seismometer label in the order of frequency range, sensitivity, and direction. For example, a sensitive broadband seismometer whose accelerometer measures vertical ground motion is labeled as BHZ.

Instruments with different frequency ranges and sensitivities affect the recorded signal differently. We can eliminate these effects by deconvolving the seismic signal with the instrument response. However, this is not always possible since the instrument response is unknown in some cases.

2.1.1.5. Earthquake Magnitude Metrics. There are different ways to represent the magnitudes of earthquakes. One of the most familiar ones is the Richter scale. It's defined as the logarithm of the maximum amplitude of the seismic wave recorded at a distance of 100 km from the earthquake source. Richter scale is useful to quantify small earthquakes that can only be detected from a single station. For large earthquakes, moment magnitude is used, which is defined as the logarithm of the seismic moment. Since earthquakes are represented by force pairs, it's natural to define a magnitude metric based on moments. The seismic moment has a robust physical basis, and it's defined as the product of the shear modulus, the average slip, and the fault

area. Another magnitude metric is based on energy release, called energy magnitude. It's defined as the logarithm of the energy released by the earthquake.

There are more subjective measures of earthquake magnitude, such as intensity. Intensity is a measure of the earthquake's effects on the surface and is based on the observations of the people. As a result, it's not a physical quantity.

2.1.2. Related Problems of Seismology

2.1.2.1. Improving Limit of Detection. For observational seismology, one of the major problems is discriminating significant seismic activity, especially earthquakes, from background noise. Since recorded seismic data is immense, it is not possible to manually evaluate the whole of it with human effort. In this manner, automatic detection of earthquakes is an active research area with practical applications. Classical algorithms such as STA/LTA, BR-Picker, and Template matching have been proposed and are widely used. However, these algorithms' performance is unsatisfactory for low SNR levels, which is the case for small earthquakes. Guttenberg-Richter law states that the number of detectable earthquakes increases exponentially with decreasing limit of detection as follows

$$\log_{10} N = a - bM, \tag{2.1}$$

while N is the number of earthquakes with magnitude greater than M, a and b are constants. In this manner, improving the detection limit has huge benefits in terms of observational capabilities.

Recently, Machine Learning based methods have been developed that claim to improve the limit of detection. Most of the developed methods are supervised and vulnerable to biases in the training set. The potential of supervised learning-based models for detecting low-magnitude events can be blocked due to dataset labeling either done by automatic algorithms or by a human expert. We believe that the detection

limit can be improved by making a reliable unsupervised detection tool for solving earthquake detection problems on a global scale.

2.1.2.2. Earthquake Prediction. The earthquake prediction problem is one of the most challenging problems of seismology, which has been tried to be solved over decades. Although there have been many studies, no reliable method has been proposed yet [18]. One hypothesis supported by experimental results is that large earthquakes are triggered by a series of small earthquakes, which even smaller ones also trigger. The problem is there is a significant probability that most of these events are too small to be detected. If the hypothesis is true, improving the detection limit can also improve earthquake prediction problems.

2.2. Feed Forward Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by biological neural networks. They are the simulations of interconnected units called neurons whose behavior can be modified with some parameters. Like physical counterparts, neurons take multiple inputs but give only one output. The procedure conducted by a neuron is quite simple. Inputs are summed after being multiplied with weights and passed through a non-linear differentiable function called the activation function. Adjusting the weights is called training of the network.

In the scope of the thesis, we are interested in a kind of ANN called Feedforward Neural Network (FNN). The Feedforward Neural Network is a neural network in which information flows from input to output. It doesn't have a memory or feedback mechanism. This network kind can be modelled as a function $y = f(x, \theta)$ while x, y and θ are the input vector, output vector, and parameter set for adjusting the function.

2.2.1. Training

Feedforward Neural Networks, at their core, are functions whose parameters can be adjusted. We call the procedure of adjusting the parameters training. The training process is formulated as a minimization problem, and the goal is to find the optimal parameters that minimize a function (loss function) that indicates the solution's fitness. The algorithm finding the optimum parameters is called the optimizer. As a result, using the given data, neural networks are trained by the optimizer to find the parameter set that minimizes the loss function.

Training is categorized as unsupervised or supervised learning according to the availability of expected outcomes. In supervised learning, the expected outcome is known, and the loss function is defined according to the difference between the expected and the predicted outcome. Prediction is done by feeding the input data to the network and obtaining the output. The expected outcome is unavailable in unsupervised learning; we only have input. The loss function and the model's architecture are chosen to lead the model to recognize patterns in the data.

As a result, it's more straightforward to train a network with supervised learning since the expected outcome is known. The problem of finding the expected outcome is named labeling when the model is a classifier, and it's not always possible to find quality labels for the data. Unsupervised learning doesn't require labels, but it's more difficult to train since the model should learn the patterns in the data by itself. In this manner, the performance of unsupervised learning is highly dependent on the model's architecture and the loss function. Finding a good architecture and loss function for a given problem is more challenging to obtain decent results compared to supervised learning.

2.2.2. Architectures

The neural network's architecture can be defined as the structure determining which parameters adjust the function $f(x, \theta)$ and in which way. In this manner, architecture is one of the significant determinants of a network's behavior, and it should be chosen carefully considering the specifics of the problem. This part will briefly introduce some of the most common architectures used in the literature.

2.2.2.1. Multilayer Perceptrons. Multilayer Perceptrons (MLPs) are one of the first architectures proposed historically and are used in many applications. They are composed of multiple neurons; each neuron in a layer is connected to all neurons in the previous layer. In Figure 2.1, there is an example of a connections scheme of an MLP with three input, five hidden, and two output neurons.

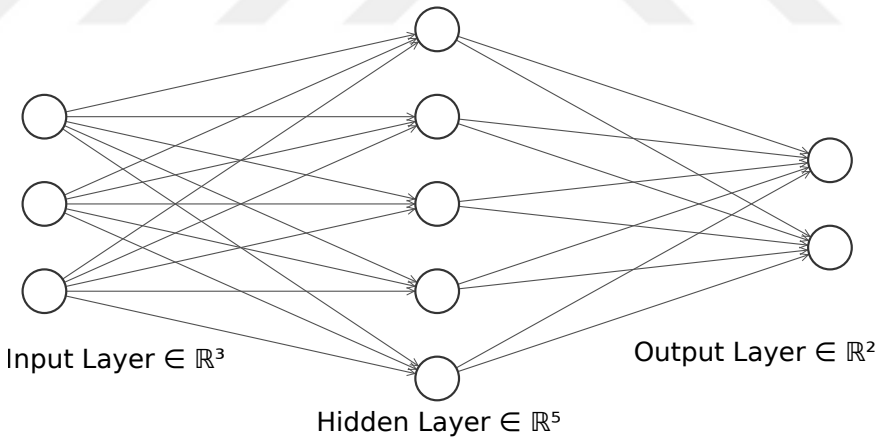


Figure 2.1. Multilayer Perceptron with single hidden layer.

The computation done with neurons in a layer can be expressed as follows

$$h_i^{(l)} = \eta(W_{ij}^{(l)} h_j^{(l-1)} + b_i^{(l)}), \tag{2.2}$$

where $h_i^{(l)}$ is the output of i-th neuron at l-th layer, $W_{ij}^{(l)}$ is the weight of the connection between i-th neuron at l-th layer and j-th neuron at previous layer, $b_i^{(l)}$ is the bias term of i-th neuron at l-th layer and η is the activation function.

2.2.2.2. Convolutional Neural Networks. Convolutional Neural Networks (CNNs) [19] are designed to process data with translational invariant properties. In this manner, they are good candidates for time-series data like seismic signals. CNNs are composed of convolutional layers that involve convolving the input data with some set of filters and then passing the output through a pointwise nonlinear function. Convolutional layers can recognize patterns at their input by convolving the input signals with predefined filters. By repeatedly applying the same procedure, CNNs can learn more complex higher-level structures.

The advantage of using CNNs for data with translational invariant structure comes from the property named weight sharing. CNNs can also be considered a kind of MLP with shared weights. Or in other words, weight matrix $\mathbf{W}^{(l)}$ does have repeating elements for different positions of the input. This property reduces the number of parameters to be learned and increases the network's generalization ability. Convolution is applied to some dimensions of the input tensor and can be multi-dimensional. In the thesis scope, we use 1D convolutional neural networks whose convolution dimension is along the time axis.

1D convolution for l-th hidden layer is defined as follows

$$\mathbf{h}^{l}_{n,f} = \eta\big(\sum_{i,k} F^{l}_{f,i,k} h^{(l-1)}_{i+n,k}\big), \tag{2.3}$$

while $h^{l}_{n,f}$ is the output of the l-th layer at timestep n and channel f. $F^{l}_{f,i,k}$ is the filter tensor at l-th layer. Filter tensor is a 3D tensor with dimensions of the number of filters, filter length along the convolution axis, and number of input channels. η is a nonlinear function that acts on each element of the tensor convolution result separately. Because of this property, it is also called pointwise nonlinearity.

Experiments show that CNNs generalize better with more layers, or in other words, depth. However, when the depth of the layers is too much, training the network becomes difficult since training algorithms depend on gradients to optimize the network's parameters. This problem can be overcome using residual connections [20].

Residual connections provide an alternative path of information flow that can be seen as shortcuts. In this manner, the vanishing gradients problem is tackled, and training deep networks becomes feasible. Although the process is simple, its effect is quite significant. Convolutional layers with residual connections are also called residual layers, and they are used in many applications.

2.2.2.3. Graph Neural Networks. Graph Neural Networks (GNNs) are a kind of neural network designed to process data with graph structure. Graph structure is formed from nodes and links between them. Links have different values indicating the strength of similarity between connected nodes. In this manner, they are good candidates for graph-structured data like social networks, molecular structures, or sensor networks. We apply GNNs to the seismic data obtained from multiple stations.

GNNs are suitable structures for fusing information obtained from very different sources. There are various methods of integrating data, and one of the most efficient methods is to apply the same principles behind CNNs to graphs. Since CNNs perform at regularly sampled signals, using the same operation on different parts of the data is possible. However, GNNs contain data that are sampled irregularly. In other saying, although the sampled data has translational invariance property, sampling may not be homogenous. This problem can be solved by defining the convolution operator in a generalized manner. Since it's outside the scope of the thesis, we refer the reader to [21] for more information.

Graph Attention Networks are inspired by the attention mechanism widely used in sequence-to-sequence models. Sequence-to-sequence models have emerged in the field of NLP (Natural Language Processing), and early attention application was related to language translation problems [22]. The attention mechanism is a way of weighting data elements and fusing them to obtain a new representation. Each element will be emphasized with its relevance to the other elements, while what's relevant will be learned by training the network. Graph Attention Networks are proposed in the same spirit while the relevance of nodes with other nodes form the basis of attention

mechanism [23]. Graph Attention Networks learn the relevance of nodes concerning other nodes and then fuse information according to the learned relevance. As a result, relevant nodes get emphasized or weighted more, while irrelevant ones get filtered out.

Although different kinds of Graph Attention layers are proposed, the general structure is formulated as follows

$$\mathbf{h}_i^{(l+1)} = \eta\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)} \Big), \tag{2.4}$$

where $\mathbf{h}_i^{(l)}$ is the output of i-th node at l-th layer, $\mathcal{N}(i)$ is the set of neighbours of i-th node, $\alpha_{i,j}^{(l)}$ is the attention coefficient between i-th and j-th nodes at l-th layer and $\mathbf{W}^{(l)}$ is the weight matrix at l-th layer. η is the nonlinear activation function. The calculation of attention coefficients is specific to the structure of the attention mechanism.

3. SINGLE STATION DETECTION

3.1. Introduction

In this section, we propose methods to solve seismic event/noise classification problems in a broader scope for single-station. Our approach depends on finding the proper architectures that map waveform signals to another space (latent space) such that features of the earthquakes get emphasized while the noise signals are suppressed. CNN Autoencoders are used to get the mentioned representations since they are efficient, easier to monitor, and keep the data's topological properties (the amount of downsampling and size of filters determine the corruption rate of the topological properties) due to their local kernels. Proposed methods perform comparably to their supervised counterparts in the cross domains (which corresponds to the real-world application) although they are vulnerable to some signals with specific patterns. Findings suggest that the proposed methods have different biases, which also makes them valuable to be used for complementary detection methods.

3.2. Training and Testing Procedures

3.2.1. Datasets and Preprocessing

We have used two different datasets to evaluate the cross-domain performance of the models. Recently proposed STEAD [24] and INSTANCE [25] datasets are selected since they have similar properties and are obtained from different regions, making them suitable for cross-domain evaluation. Besides, waveform characteristics tend to differ due to the epicenter distance of the waveform examples, which is also a desired property of cross-domain assessment. Properties of both datasets are in the Table 3.2.1.

Table 3.1. Properties of the STEAD and INSTANCE datasets.

Dataset	**STEAD**	**INSTANCE**
Channels	E, N, Z	E, N, Z
Earthquake waveforms	1,050,000	1,159,249
Noise waveforms	100,000	132,330
Sampling Rate	100Hz	100Hz
Time Window	60s	120s
Epicenter Distance	$< 350km$	$< 600km$
Region	Global	Italy

To make the data compatible with the review article [6] we have selected the model input time window as 30 seconds. After bandpass filtering ($1-20$Hz) and normalization through the timesteps axis (such that the standard deviation of each channel becomes 1), we take the same procedure as the article [6] crop the waveforms so that at least 2/3 of the earthquake waveforms include one phase arrival time. We randomly choose crop windows without any constraint for the rest of the earthquake waveforms and all noise waveforms. However, at the testing phase, we select all earthquake waveforms such that they are ensured to have onset time similarly [6]. To ensure that the cropped waveform includes the characteristic part of the P or S phase arrival, we used an additional 3 seconds margin. We used the same split ratio for the training part, 0.6. However, due to the application of 5-fold cross-validation, we have chosen validation (0.2 instead of 0.1) and test splits (0.2 instead of 0.3) differently [26].

3.2.2. Training

We have used Both STEAD and INSTANCE datasets for training and testing purposes. K-Fold Cross-validation method is used while $k = 5$ in order to obtain reliable performance metrics. Models are trained for 20 epochs with a batch size of 256 samples, and the lowest validation error epoch is selected. We used ADAM optimizer with learning rate 10^{-4} and $\epsilon = 10^{-7}$ although higher learning rate values were also

stable. Moment estimating moving average filter coefficients β_1 and β_2 are chosen as 0.99 and 0.999. At the same time, we kept other settings as default (we didn't use additional exponential moving average filtering, weight decay, or gradient clipping).

Training took approximately 7.5 minutes for an epoch for the Autoencoder Ensemble, while it took 1.5 minutes for CNN Autoencoder at NVIDIA GTX3090TI GPU. It has been observed that a negligible amount of time is spent on data generation since we have already used preprocessed data.

3.2.2.1. Autoencoder Training. Let $x \in \mathbb{R}^{N \times C}$, $y \in \mathbb{R}^{N \times C}$ denote the input and output of the model while N and C denote the number of timesteps and channels of the waveforms $(3000, 3)$. We train the autoencoder model with the mean squared error loss function. Before calculating the loss means of each channel are subtracted such that $\frac{1}{N}\sum_n x_{nc} = 0$, $\frac{1}{N}\sum_n y_{nc} = 0$. Then the reconstruction loss L_{recons} is calculated as follows

$$L_{recons} = \sqrt{\frac{1}{NC}\sum_{n=0}^{N-1}\sum_{n=0}^{C-1}(x_{nc} - y_{nc})^2}. \tag{3.1}$$

3.2.2.2. Autoencoder Ensemble Training. By jointly training the autoencoder ensemble, we aim to obtain different latent representations of the same input. However, since there aren't any constraints on the latent space, these representations can take any form and be completely different. We use projections for each autoencoder and additional loss term L_{proj} to L_{recons} to prevent this. By preventing gradient flow through the projection layer, we ensure that each autoencoder will learn different latent representations. This method is equivalent to the simultaneous training of projection matrices and CNN Autoencoders with different loss functions.

Let $h^e \in \mathbb{R}^{N_L \times C_L}$ denote projected latent representations of the e-th member of the ensemble while N_L and C_L are latent timesteps and channels. Before calculating the loss, the mean of each channel is subtracted such that $\sum_i h^e_{nc} = 0$ and channel-wise

l2 normalization $\frac{1}{N_L}\sum_n {h_{nc}^e}^2 = 1$ is applied. Projection loss L_{proj} is then calculated as follows

$$L_{proj} = \sqrt{\frac{1}{N_L C_L E(E-1)} \sum_{p=0}^{E-1} \sum_{q=0, p\neq q}^{E-1} \sum_{i=0}^{N_L-1} \sum_{j=0}^{C_L-1} (h_{nc}^p - h_{nc}^q)^2}, \tag{3.2}$$

and the optimizer minimizes the total loss

$$L_{total} = L_{recons} + L_{proj}, \tag{3.3}$$

while the gradient flow through the input of the projection layer is blocked.

Blocking gradient flow results in CNN Autoencoders' gradient updates are not affected from L_{proj}. Showing a set of all CNN Autoencoder parameters as θ, this operation is equivalent to setting $\partial L_{proj}/\partial\theta = 0$ before each parameter update of the training phase. As a result, CNN Autoencoders get trained independently from themselves.

3.2.3. Testing

Tests have been conducted similarly to the review article [6]. However, we have used 5-fold cross-validation while the article only focuses cross-validation on different domains. We discard waveform examples that don't include onset time by the margin of 3 seconds, considering their potential to introduce errors in metric calculations.

We have used the same metric for evaluation as the article, which is the Area Under the Receiver Operating Characteristic curve (ROC-AUC), which is obtained by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. TPR and FPR are defined as follows

$$\begin{aligned} TPR &= \frac{TP}{TP+FN} \\ FPR &= \frac{FP}{FP+TN}, \end{aligned} \tag{3.4}$$

where TP, FP, TN, and FN denote the number of true positives, false positives, true negatives, and false negatives, respectively. The Receiver Operating Characteristic (ROC) curve is obtained by plotting TPR against FPR at various threshold settings.

3.3. Methods

Although currently applied unsupervised learning methods are well suited for observational purposes, they are not suited for classification tasks on a broader scale. In this manner, their strength, lack of supervision-induced bias, is also their weakness. If we summarize the general structure of the algorithms, we can see that they consist of dimensionality reduction and classification methods applied successively. The main obstacle is not the dimensionality reduction step but the classification step, which runs on lower dimensions. We saw that we could eliminate unsupervised classification algorithms running on the latent space using proper classification metrics and dimensionality reduction methods. In this spirit, we propose three methods with increasing complexity and success. We believe these methods can also find applications for noise/signal classification tasks in domains other than seismology.

The first method is based on calculating the autocovariance of the latent representations by themselves. This operation exploits the CNN Autoncoder's filtering capability while measuring the channels' synchronicity. Although simplistic, this method could perform well if the training duration is chosen correctly. However, there is a challenge in selecting the proper training duration.

Another method involves applying random augmentations to raw waveforms and then taking the cross-covariances of their latent representations. The method has similar grounds with Self Supervised Learning (SSL) methods [27], which uses augmentations to obtain robust representations or for classification purposes. In our case, we use time-warping augmentations to measure the representations' robustness. By calculating cross-covariances, we form a metric for seismic event classification between sets of augmented waveform representations.

The last proposed method uses an ensemble of encoder-decoder structures trained jointly to obtain multiple representations of the identical waveform. The idea is similar to the augmentation-based method; however, the procedure affects the training phase in this scheme. This method includes training multiple CNN Autoencoders and linear projection matrices simultaneously. While CNN Autoencoders are trained to reconstruct the input, Projectors are optimized by maximizing cosine similarity between all possible pairs of projections. The idea is to obtain a set of representations in which common features are emphasized and represented coherently while uncommon ones get filtered.

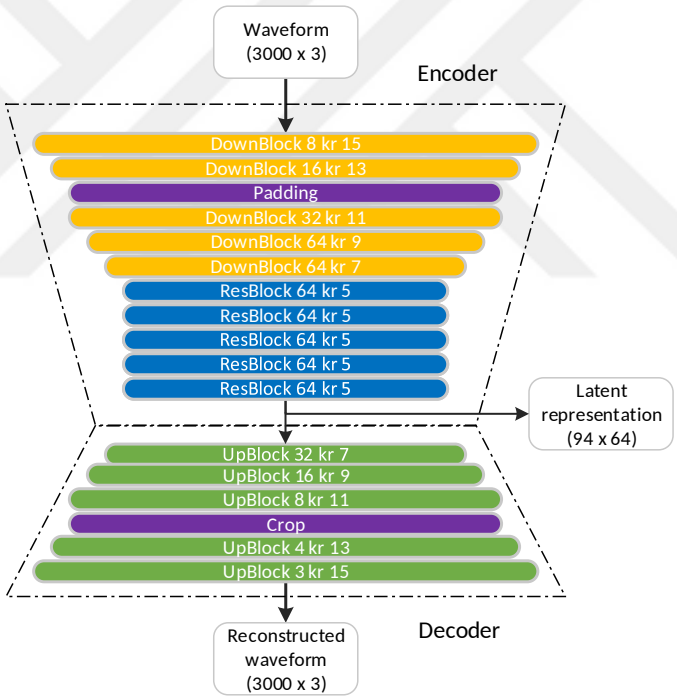


Figure 3.1. Architecture of the CNN Autoencoder used for representation learning.

3.3.1. Building Blocks

3.3.1.1. CNN Autoencoder. All proposed detection methods are based on Convolutional Neural Network (CNN) Autoencoders. We have used CNN Autoencoders due to their equivariance property of the convolutional layers. Although the space of latent representations is different from the space of waveforms, equivariance property

and local filters of the convolutional layers ensure that the topological structure of the signals is preserved to some level, making the covariance a much more stable measure of similarity. The other reason is their efficiency and simplicity compared to sequential architectures such as Recurrent Neural Networks (RNNs).

Used CNN Autoencoders are composed of repeated Downsampling, Residual, and Upsampling layers. Downsampling layers' input dimensions are twice their output along the timesteps axis. After repeatedly applying Downsampling layers, lower dimensional and higher level representation is obtained and fed to the Residual layers. Besides, since some dimensions are not divisible by two, we use padding layers (Reflect Padding is used to decrease edge effects). After reducing the dimensionality, we use Residual layers to get a more abstract and filtered version of the downsampled signal. Finally, the last the residual layer's output is connected to the decoder part. The Decoder comprises Upsampling and Crop layers that retrieve the original waveform dimensions. While Upsampling layers increase the dimensionality, Crop layers remove the padding layers' effects. The architecture of the CNN Autoencoder is shown in Figure 3.1.

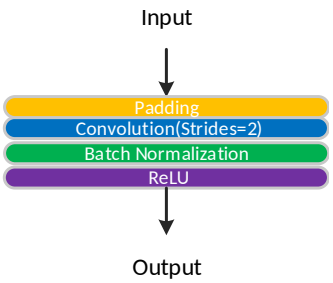


Figure 3.2. Downsampling layer structure.

Downsampling layers comprise stacks containing 1D Convolutions (stride=2), Batch Normalization (BN), and Rectified Linear Unit (ReLU) activation layers. Five downsampling layers with Convolutional filter sizes 15, 13, 11, 9, 7 and counts 8, 16, 32, 64, 64 are used to reduce the length of input waveforms along the timesteps axis from 3000 to 94 while increasing the number of channels from 3 to 64.

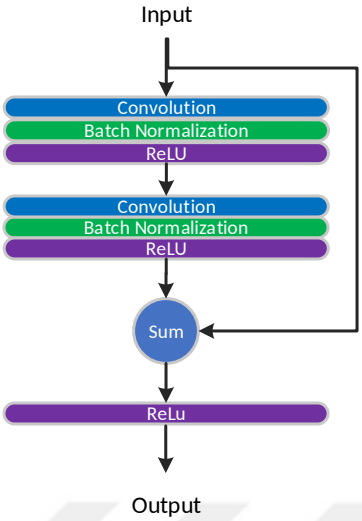


Figure 3.3. Residual layer structure.

Residual layers involve two subsequent identical stacks involving 1D Convolution, Batch Normalization, and ReLU Activation layers. We add skip connections by summing the convolution layer's output with the residual layer's input to be able to train deeper networks. The output of the residual layers is obtained by passing the sum through the ReLU activation function. We use five residual layers with 64 filters of 5 timestep lengths.

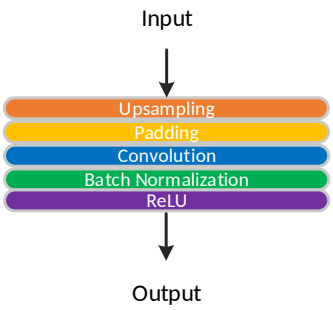


Figure 3.4. Upsampling layer structure.

Upsampling layers comprise 1D Upsampling, 1D Convolution, Batch Normalization, and ReLU activation layers. Five upsampling layers with Convolutional filter sizes 7, 9, 11, 13, 15 and counts 64, 64, 32, 16, and 8 are used to increase the length of

input waveforms along the timesteps axis from 94 to 3000 while increasing the number of channels from 3 to 64.

3.3.1.2. Cross-covariance Layer. The cross-covariance layer is a custom layer that calculates the cross-covariance of two latent representations efficiently by taking advantage of Fast Fourier Transform (FFT). Cross-covariance is calculated between channels of the two representations individually and then averaged over all channels.

Let $g \in \mathbb{R}^{N_L \times C_L}$, $h \in \mathbb{R}^{N_L \times C_L}$ represent two latent representations while N_L, C_L denoting the number of timesteps and channels of the latent space. We condition representations such that the mean of each channel is zero ($\frac{1}{N_L}\sum_n g_{nc} = 0$, $\frac{1}{N_L}\sum_n h_{nc} = 0$). Representations are not strictly normalized since amplitude information is important for detection. However, Batch Normalization layers should be applied to both representations to obtain proper scale. Due to efficiency considerations, cross-covariances are calculated by circular convolutions, which is equivalent to calculating covariance for periodic g_{nc} and h_{nc} ($g_{nc} = g_{n+mN_L,c}$ and $h_{nc} = g_{n+mN_L,c}$ while $n \in [0, N-1]$, $c \in [0, C-1]$ and $m \in \mathbb{Z}$). Instead of circular convolutions, we can use zero padding as well. However, results show that it doesn't improve the performance while increasing the computational cost.

Denoting the layer output as σ_n the cross-covariance operation is equivalent to the following equation

$$\sigma_n = \frac{1}{N_L C_L} \sum_{n'=0}^{N_L-1} \sum_{c=0}^{C_L-1} g_{n',c} h_{n'+n,c} \tag{3.5}$$

while $n \in [-\frac{N_L}{2}, \frac{N_L}{2}-1]$. We calculate the cross-covariance numerically in the frequency domain for efficiency considerations. Discrete frequency index k will correspond to the frequency $f_k = \frac{kf_s}{N_L}$ where $k \in [-\frac{N_L}{2}, \frac{N_L}{2}-1]$ and f_s corresponding to the sampling frequency which is 100Hz. Cross-covariance can be calculated in the frequency domain

as follows

$$\begin{aligned} g_{kc} &= \mathcal{F}_{kn}\{g_{nc}\} \\ h_{kc} &= \mathcal{F}_{kn}\{h_{nc}\} \\ \sigma_n &= \frac{1}{N_L C_L}\mathcal{F}^{-1}_{nk}\{g_{kc}h^*_{kc}\}, \end{aligned} \tag{3.6}$$

where h^*_{kc} denotes the complex conjugate of h_{kc}. It should be noted that g_{kc} and h_{kc} are both complex numbers while σ_n is a real number. It's worth mentioning that both FFT and IFFT operations run for each channel separately, equivalent to considering each channel as a different time series.

3.3.1.3. Maximum Cross-covariance Metric. The filtering property of the Autoencoder is so effective that even calculating the variance of the representation is enough to discriminate between earthquake and noise signals. Experiments show that the amplitude of the cross-covariance vector is much higher for earthquake representations than for noise representations. Therefore, the maximum value of the cross-covariance

$$M = \max_n c_n, \tag{3.7}$$

while M can be used as a metric for classification.

3.3.1.4. Cross-covariance Weighted Average Metric. The metric is calculated by taking the weighted average of cross-covariance by using the window function, which has a maximum around zero lag ($\tau = 0.0$). We have chosen a Gaussian window with $\nu = 0.0$ and $\sigma_0 = 5.0$ seconds while σ_0 is chosen due to empirical factors. Observations show that the window for choosing σ_0 is quite wide. Letting $t_n = \frac{T(n-N/2)}{N}$ as discrete time while T is the 30 second input time interval, we can express weighted average as follows

$$M = \frac{1}{\sqrt{2\pi\sigma_0^2}} \sum_n \exp\left(-\frac{t_n^2}{2\sigma_0^2}\right) c_n, \tag{3.8}$$

while M is the classification metric.

3.3.2. STA/LTA

STA/LTA is a widely used classical method in earthquake detection. It's based on the fact that earthquake phase arrivals are seismic waves carrying energy. During the seismic waves' arrival, the instantaneous power of the seismic signal increases. We can take the time average of the seismic wave energy using windows with different sizes and get their ratio to detect earthquakes.

Selecting two windows is sufficient for detection. If we name the time average of the seismic power through the shorter window as STA and the longer one as LTA, their ratio $\frac{STA}{LTA}$ increases during seismic wave arrival and can be used for detection purposes. We can calculate the STA/LTA ratio as follows

$$\begin{aligned} STA(n) &= \sum_{m=-\frac{N_S}{2}}^{\frac{N_S}{2}-1} \sum_{c=0}^{C-1} x_{(n+m)c}^2 \\ LTA(n) &= \sum_{m=-\frac{N_L}{2}}^{\frac{N_L}{2}-1} \sum_{c=0}^{C-1} x_{(n+m)c}^2 \\ STA/LTA(n) &= \frac{STA(n)}{LTA(n)}, \end{aligned} \tag{3.9}$$

where N_S and N_L are the short and long window sizes, we have chosen $N_S = 350$ (3.5 seconds for 100Hz sampling frequency) and $N_L = 650$ (6.5 seconds) for our experiments. N and C are input timestep and channel counts, while $STA(n)$ and $LTA(n)$ are the short and long-term averages of the signal's instantaneous power. A very small constant value (10^{-27}) is added to the denominator to avoid division by zero.

Since the above method gives a continuous value, we need to threshold it to obtain a binary classifier. Since we use variable thresholds to obtain ROC curves, it's more convenient to convert the output of this method to a classification metric. We take the maximum of the STA/LTA vector for this purpose. Classification metric is defined as follows

$$M = \max_n STA/LTA(n), \tag{3.10}$$

while $n \in [0, N-1]$ and $STA/LTA(n)$ is calculated as in Equation 3.9. Parameters are adjusted by performing a grid search and choosing the best combination. Searched parameter combinations can be found in Table D.1.

3.3.3. Autocovariance at Latent Space

Our observations show that earthquake representations tend to be composed of more long-term features, contrary to noise waveforms. We can measure long-term features' magnitude by calculating the latent representation's autocovariance function. While for noise signals, we obtain a narrow spike at $\tau = 0$, for earthquake signals, we obtain a wider and smoother peak. Therefore, any metric measuring the smoothness and wideness of the autocorrelation function can be used as a classification metric. Besides this, we have seen that performance increases drastically using Autocovariance instead of Autocorrelation. This performance boost is related to the filtering property of Autoencoder. Noise signals do not lead to significant activations compared to earthquake signals. As a result, the autocovariance of an earthquake signal is wider, smoother, and high amplitude.

3.3.4. Augmentation Cross-covariances

Earthquake waveforms are expected to be composed of more robust features such that small changes in the raw waveform don't significantly alter their latent representations. However, a similar kind of property is not expected for noise waveforms. As a result, the cross-correlation of the latent representations of two augmented waveforms is expected to be composed of higher values for earthquake waveforms than noise waveforms. In addition, we exploit the Autoencoder's filtering property by using cross-covariances. Besides, instead of using two augmentations, we use an ensemble of augmentations, calculate their cross-covariances, and use their mean for classification. Our experiments have shown that using an ensemble of augmentations increases stability and performance.

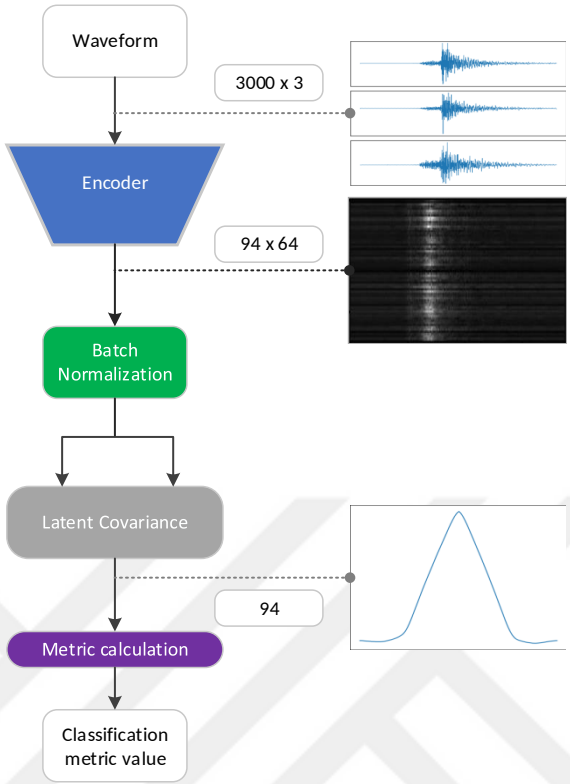


Figure 3.5. Diagram of the Autocovariance method.

Although different augmentations, such as adding noise or phase warping (adding random phase at frequency domain), can be used, our experiments have shown that time-warping augmentation performs well and gives stable results. Time-warping augmentation is accomplished by remapping the time axis of the original waveform by a monotonically increasing function. We obtain the mapping by choosing random deviations from the actual time and interpolating between them by the BSpline algorithm. After finding the interpolating function, the results are discretized.

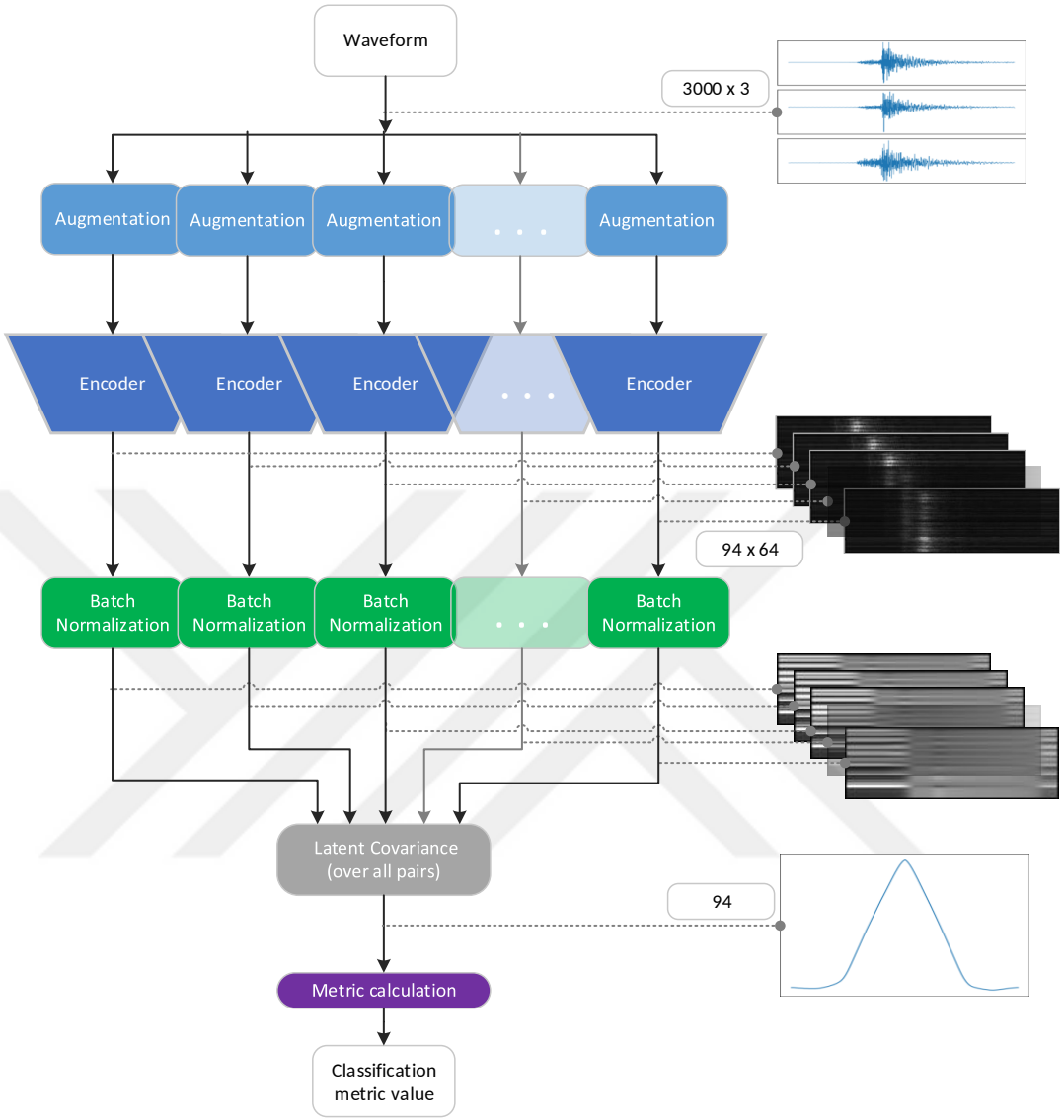


Figure 3.6. Diagram of Augmentation Cross-covariances method.

3.3.5. Representation Cross-covariances

Another way to classify seismic waveforms is to obtain multiple representations from the same waveform and then calculate their cross-covariances. This is done by training an ensemble of Autoencoders. However, since channel encodings can differ from model to model, we need a different structure to map them to another space. We use projection matrices to map different representations to a common feature space.

To train projection matrices accordingly, we put a loss term, the mean distance, over projected representations.

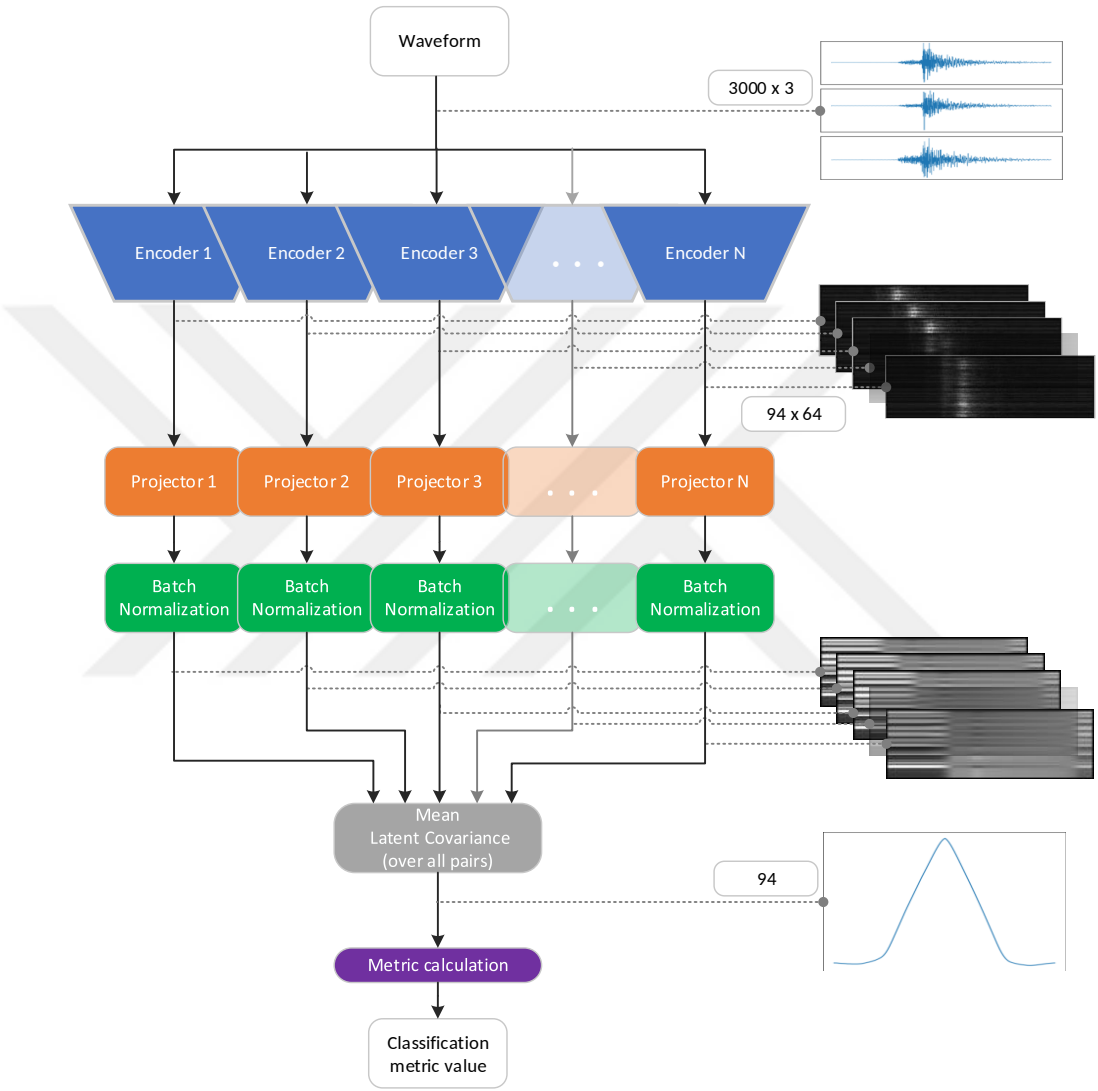


Figure 3.7. Diagram of Representation Cross-covariances method.

The method outperforms other methods and is more robust. One main reason is the additional filtering property induced by projection operations. Intuitively, trying to find a common feature space filters out less frequent and short-term representations. Especially, noise signals are filtered since they are poor in terms of repeated patterns and are encountered less in the training set.

3.4. Investigating Latent Representations

This part involves investigating latent representations obtained by the models. We add this part because the behavior of the models can be understood much better by visual inspection.

3.4.1. Samples for Randomly Parametrized CNN Autoencoders

Our experiments showed that even randomly parametrized models can perform well, especially in STEAD datasets. To investigate this phenomenon, we visualized latent space representations obtained by randomly parametrized models. Figures A.1, A.2, A.3 and A.4 include randomly parametrized models' latent space representations for earthquake and noise waveforms. We see that the regions corresponding to seismic activity get activated at the latent space for randomly parametrized models (Glorot Uniform initialization is used [28]) as well. One of the possible reasons is that seismic activity is composed of broadband continuous frequency spectra, which have a higher possibility of matching with randomly initialized filters. Besides, temporal segments of seismic activity have higher amplitude than other waveform parts. However, this phenomenon can't be explained purely by the amplitude magnitude because the randomly parametrized models' performance is higher (comparable to supervised counterparts for cross-domain cases) than the baseline method.

Investigating the figures further, we can see that the autocovariance of the reconstructed waveform (since we didn't train the model, it's not a reconstruction) is similar to its latent space counterparts, which is interesting. This behavior is related to the ReLU activation function, which acts as an identity function when its input is higher than zero. It's seen that, due to convolutions with random filters, the shape of the autocovariance has some similarities with the Gaussian function. Interestingly, we don't observe this behavior for noise waveforms. Besides, the magnitude of the autocovariance is much lower for noise waveforms, suggesting that channels are not synchronized.

3.4.2. Samples for Autocovariance Method

Representations obtained by the Autocovariance method can be seen in Figures A.5, A.6, A.7 and A.8. We can see easily that activations for earthquake signals are much more regular and organized, while noise signals lack order. Besides, we have observed that channels act synchronously for earthquake signals while there isn't such behavior for noise signals.

Regarding discriminative abilities, we see that the autocovariance of latent space representation is very different in amplitude and shape for earthquake and noise signals. However, checking the input and output of the networks (since the waveforms are normalized, cross-covariance corresponds to cross-correlation), we see that autocovariance for the different seismic activity is hard to discriminate. The shape of the autocovariance for waveforms is much related to their frequency spectrum, which could be misleading for low-frequency seismic activity. As a result, applying autocovariance-based classification to raw waveforms gives poor results. But when applied to latent representations, results are pretty promising.

There are two mechanisms that we consider to be related to performance improvement. The first one is the filtering property of the autoencoder, as mentioned previously. Since it's possible to encode patterns efficiently, under-complete autoencoders tend to selectively represent the patterns they had encountered in their training set. As a result, noise signals without significant patterns get much less represented in the latent space compared to earthquake signals. The second mechanism is related to channels being much more synchronized when seismic activity occurs. As a result, since we take the mean of the autocovariance of each channel, we obtain higher values for earthquake signals.

3.4.3. Samples for Representation Cross-covariances Method

Samples obtained by visualizing latent space projections of different encoders can be seen in Figure A.9, Figure A.10 and Figure A.12, Figure A.11. We see that behavior is very different from the other models. Since we train projection matrices with similarity loss, projection heads must filter out uncommon parts of the representations and emphasize the common parts. Common parts, in most cases, are the long-term features of the representations. Although representations differ from model to model, the ReLU activation function gets activated at regions that involve features to be reconstructed. Consequently, in Figures A.9 and A.10, we see that the areas where there is a significant seismic activity get coded in a way to have the highest contrast compared to the regions where there isn't. Another reason for this segmentation behavior is the channel-wise normalization applied before calculating the cosine similarity between representations. Normalization forces projection heads to find representations that emphasize long-term and the most varying features. In this manner, a kind of contrastive learning [27] mechanism emerges through the normalization procedure along the timesteps axis.

We see that segmentation behavior is specific to earthquake waveform representations, and noise waveform representations are not composed of regular segments. They are not segmented or segmented irregularly as seen in Figures A.9 and A.10.

3.5. Results

We evaluate the proposed methods' performances using the same metric as the review article [6]. ROC-AUC is a reasonable metric since the class imbalance problem does not affect it unless the bias is severe. Besides, it's not threshold-dependent, making it a more objective evaluation metric than threshold-dependent metrics such as accuracy. We have evaluated model performances at different domains concerning training duration and SNR level. This section includes results and analysis related to the models' performances.

3.5.1. Effect of Training Duration on Performance

Evaluation is done for the whole model evolution during training to obtain a more detailed view. Since the training is done unsupervised, validation loss doesn't correspond to how well the models perform, making it harder to choose the best training duration. This factor could mislead model performance measurement dependent on the training duration. We evaluated the models' performance for each epoch to analyze the training behavior.

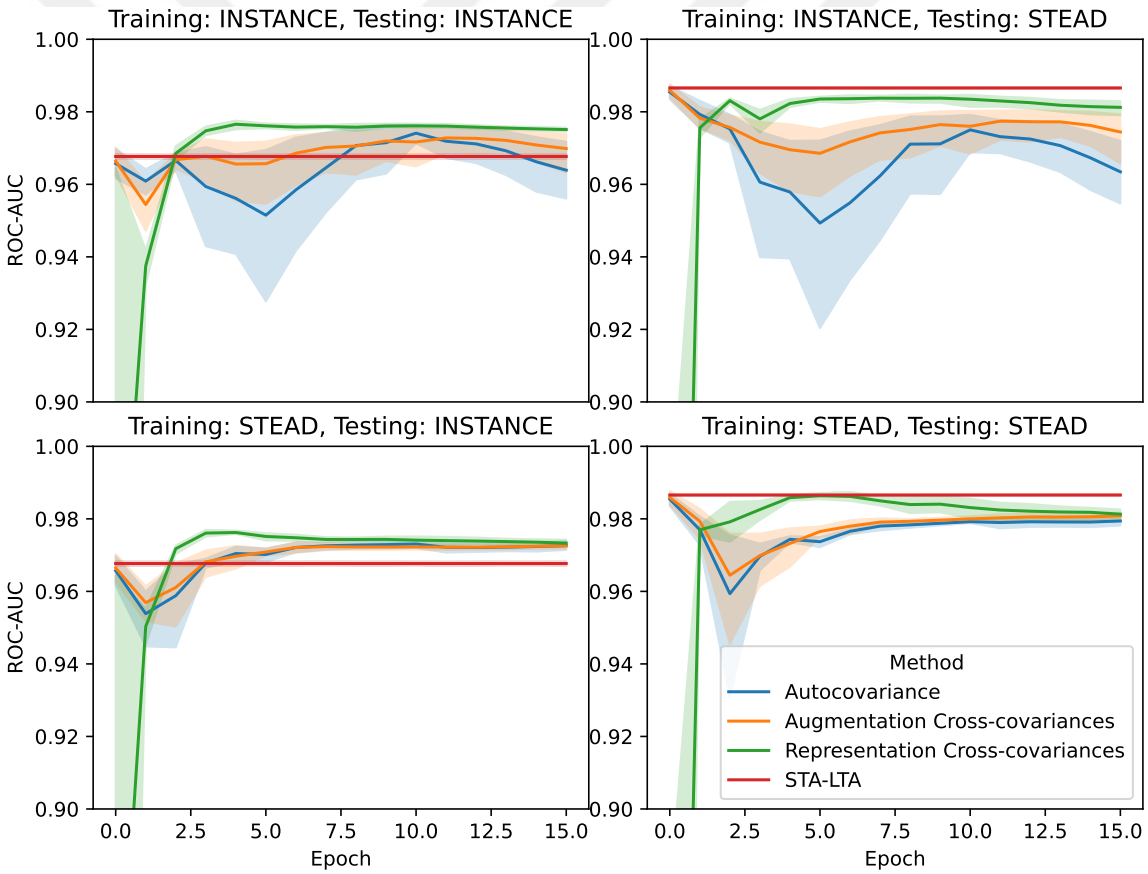


Figure 3.8. ROC-AUC scores concerning training duration.

Figure 3.8 shows that the Representation Cross-covariances method has better stability over training than other methods. It's also better for a small margin and deviates less from split to split. Augmentation Cross-covariances also give decent performance over training, suggesting that ensemble methods could have stability and per-

formance advantages. The Autocovariance method is the most sensitive to training duration among all. Augmentation Cross-covariances and Autocovariance methods have close performances at their best versions, while the Representation Cross-covariances method is better by a small margin for all domains.

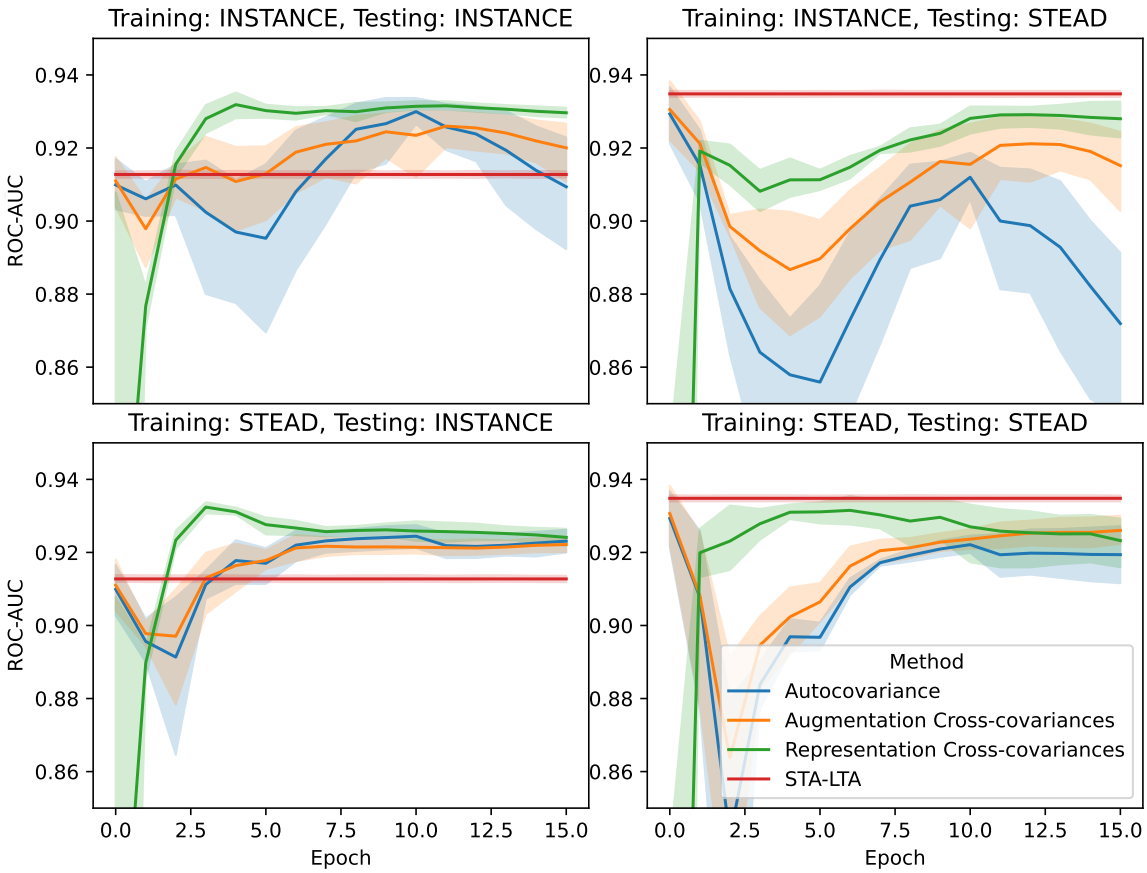


Figure 3.9. ROC-AUC scores concerning training duration for low SNR case.

Additionally, we see an unexpected result for the Autocovariance and Augmentation Cross-covariance methods. Randomly parametrized CNN Autoencoder (CNN Autoencoder at the initial stage of training) performs better than its trained versions for STEAD dataset tests. Besides, the STA/LTA method performs better at the STEAD dataset. We have observed that the models fail due to specific noise waveforms introduced into the STEAD dataset. Noise waveforms involving transient signals tend to be classified as earthquake waveforms, especially for trained models. Our experiments show that this problem may be solved by introducing augmentations such as phase

warping at model input. However, due to the scope of the thesis, we didn't include this method in the evaluation. Problematic samples can be seen in the appendix at Figures A.13 and A.14.

The training behavior of the models is similar for low SNR cases, as seen in Figure 3.9. However, the margin between the STA/LTA method has been increased for INSTANCE dataset in favor of the proposed methods.

3.5.2. Relationship Between Validation Loss and Performance

We have visualized the relationship between the model performances and validation loss on the same plot for each method to investigate the relation between the model performances and validation loss. The Autocovariance method is the most sensitive to training duration, and validation loss doesn't indicate when to stop training, as seen in Figure C.1. Although the method has advantage of being simple; we must show special care to determine the training duration.

Validation loss is not a good indicator for the Augmentation Cross-covariances method either. However, as seen in Figure C.2, performance is much less sensitive to training duration. Besides, training on the STEAD dataset is much more stable. This behavior can be explained by the cleaner waveforms (higher SNR) provided by the STEAD dataset compared to the INSTANCE dataset. By increasing the training duration, we could force the model to represent noise waveforms and earthquake waveforms, degrading its filtering ability. Our experiments suggest that we can eliminate the overfitting problem by adding noise to the input signal. However, to preserve the essence of the method, we excluded this method from the scope of the thesis.

Finally, possibly due to similarity loss between different projections, we see that validation loss is related to performance for the Representation Cross-covariances method as seen in Figure C.3. Besides, the model is more stable concerning training duration. Validation loss increases after a while, which we consider related to models'

learning to represent waveforms on a finer scale, making minimizing similarity loss harder for projection heads. Additionally, the model's performance is much more stable with training duration. However, in this case, experiments show that adding white noise to the input in the training phase doesn't improve the results. The reason may be related to the model's learning contrasts between different timesteps, which could be lost by noise addition.

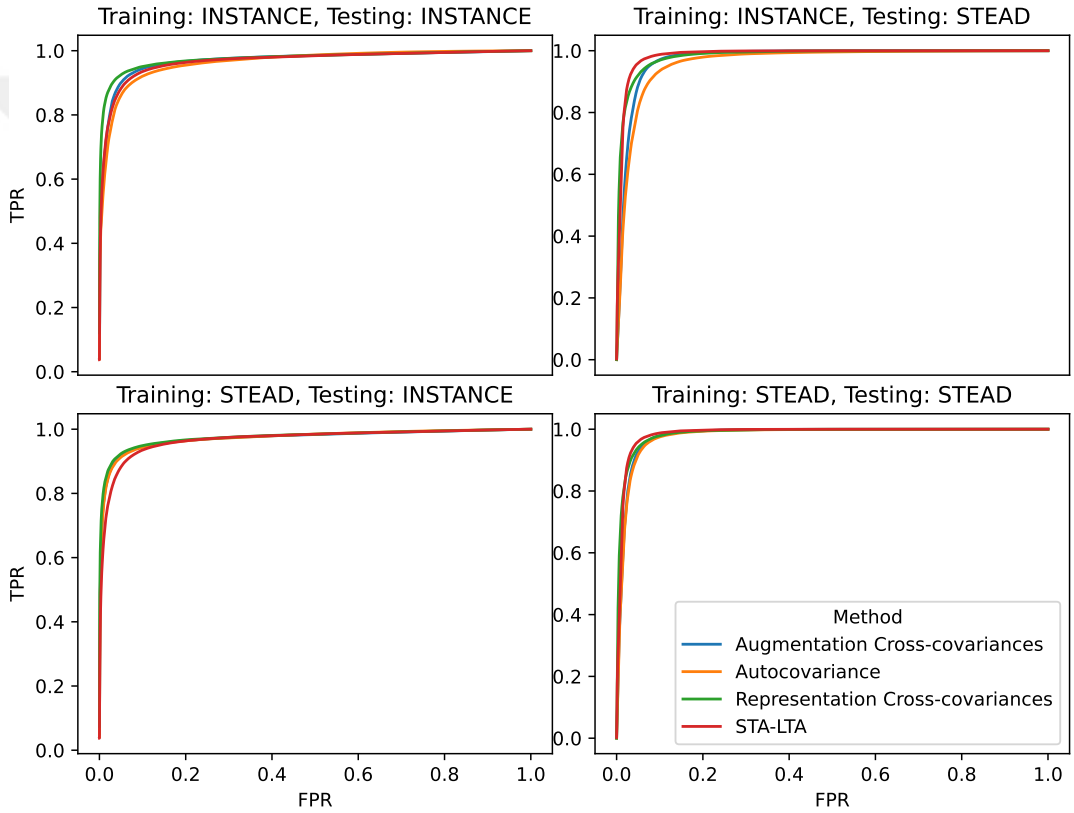


Figure 3.10. ROC curves.

3.5.3. Performance Metrics and Comparison with the State of the Art

The proposed methods have close detection characteristics, although ensemble-based methods have steeper ROC curves. Steeper ROC curves suggest that classifiers can be used effectively for applications requiring low false positive rates. The Representation Cross-covariances method has the steepest ROC curve among all due to

additional filtering mechanisms related to projection heads. Decreasing the SNR level degrades model performances in similar ways.

We see that the Representation Cross-covariances method has better classification performances for all cases, which is followed by Augmentation Cross-covariances method. In addition, performance changes concerning the training set are minimal for ensemble-based methods. This result shows that bias due to the training set is insignificant, and the research aim has been satisfied.

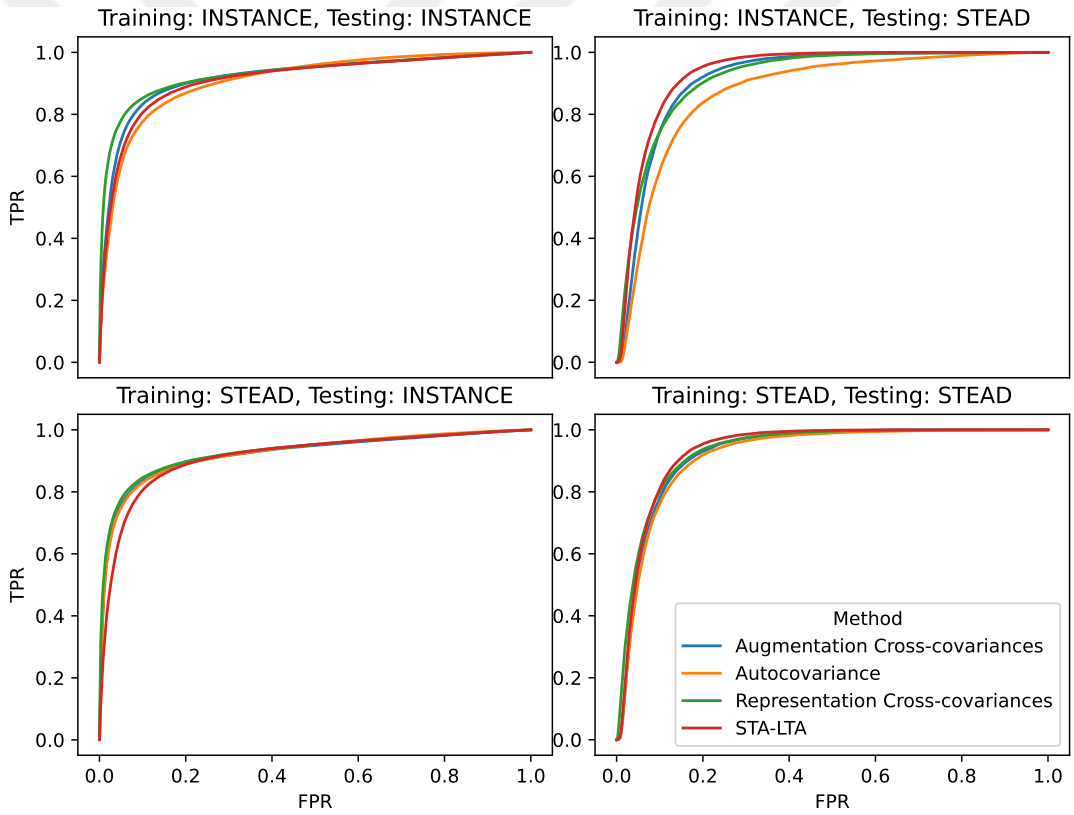


Figure 3.11. ROC curves for low SNR case.

Results are also similar for low SNR (SNR value lower than 15 dB) cases. Unfortunately, we can't compare the results with the State-of-the-Art supervised models at low SNR cases since [6] involves whole dataset evaluations only. Compared with the STA/LTA, we see that STA/LTA performs well in the STEAD dataset, while the proposed methods are better in the INSTANCE dataset. Poor performance in the

STEAD dataset is due to the model's vulnerability to signals with high gradients and gaps.

Table 3.2. Method performances for different training (rows) and testing (columns) datasets for low SNR test sets.

	INSTANCE (Testing)		STEAD (Testing)	
INSTANCE (Training)	Autocovariance	0.910 ± 0.020	Autocovariance	0.873 ± 0.032
	Augmentation Cross-covariance	0.920 ± 0.010	Augmentation Cross-covariance	0.915 ± 0.015
	Representation Cross-covariance	0.930 ± 0.002	Representation Cross-covariance	0.918 ± 0.007
	STA/LTA	0.913 ± 0.001	STA/LTA	0.935 ± 0.001
STEAD (Training)	Autocovariance	0.923 ± 0.004	Autocovariance	0.919 ± 0.010
	Augmentation Cross-covariance	0.922 ± 0.005	Augmentation Cross-covariance	0.926 ± 0.005
	Representation Cross-covariance	0.928 ± 0.003	Representation Cross-covariance	0.934 ± 0.003
	STA/LTA	0.913 ± 0.001	STA/LTA	0.935 ± 0.002

The ROC-AUC score dependence over training sets is minor for normal SNR levels for the proposed methods. Especially the Representation Cross-covariances method's ROC-AUC score change minimally concerning the training set (0.001 for INSTANCE, 0.003 for STEAD dataset tests). This deviation is very close to the uncertainty level. EQTransformer's [1] ROC-AUC score over the training set changes 0.009 for INSTANCE dataset and 0.010 for STEAD dataset tests, while Phasenet's [4] changes 0.023 for INSTANCE and 0.006 for STEAD datasets. Besides, we see that scores are much more balanced. The deviation between test sets is less than 0.01, while it's much more significant (EQTransformer's and Phasenet's ROC-AUC scores change up to 0.043 and 0.059, respectively) for supervised counterparts.

Table 3.3. Method ROC-AUC scores for different training (rows) and testing (columns) datasets. Phasenet and EQTransformer performances are taken from [6].

	INSTANCE (Testing)		STEAD (Testing)	
INSTANCE (Training)	Autocovariance	0.964 ± 0.011	Autocovariance	0.964 ± 0.012
	Augmentation Cross-covariance	0.970 ± 0.007	Augmentation Cross-covariance	0.974 ± 0.009
	Representation Cross-covariance	0.976 ± 0.001	Representation Cross-covariance	0.983 ± 0.002
	STA/LTA	0.969 ± 0.001	STA/LTA	0.987 ± 0.001
	Phasenet	0.964	Phasenet	0.994
	EQTransformer	0.957	EQTransformer	0.990
STEAD (Training)	Autocovariance	0.973 ± 0.002	Autocovariance	0.979 ± 0.002
	Augmentation Cross-covariance	0.973 ± 0.001	Augmentation Cross-covariance	0.981 ± 0.001
	Representation Cross-covariance	0.975 ± 0.001	Representation Cross-covariance	0.986 ± 0.001
	STA/LTA	0.969 ± 0.001	STA/LTA	0.987 ± 0.001
	Phasenet	0.941	Phasenet	1.000
	EQTransformer	0.966	EQTransformer	1.000

3.6. Discussion and Future Work

Our experiments show that the proposed methods have comparable performance to the supervised learning models proposed in the literature. Besides, the proposed methods give similar performances for different training and testing sets, suggesting that they don't have strong biases. Hence, they can be used as general-purpose detection tools reliably besides being used for complementary purposes since they have different detection mechanisms. Although we have applied them for seismic waveform classification, they can be applied to other multidimensional time series data for discriminating background noise from patterns.

In addition to obtaining a general-purpose classifier driven by representation learning, we also show that the common practice of using a trainable classifier after dimensionality reduction may not be the only approach to take. By carefully choosing architecture and loss function, we see that representation learning can take the significant workload of classification such that the classifier can be exchanged by a function (classification metric) as simple as the variance. By eliminating the need to use a trainable classifier, we can obtain methods that don't have the vulnerabilities of standard unsupervised learning classification algorithms. For some reason, research primarily focuses on extracting features manually while leaving the classification task to the unsupervised classifier. However, we have seen that representation learning-based algorithms such as CNN Autoencoders successfully extract the essence of data. Going through the other direction, extracting useful features by learning representation and engineering the classifier may improve performance. We believe that there are opportunities to boost the performances of classical algorithms by running them on high-level representations that are noise-free and robust, thanks to the tools offered by representation learning. In this aspect, we believe our approach may open a new path for developing methods for seismic classification.

Although progress is accomplished, proposed methods have vulnerabilities to certain noise types, which has room for improvement. This can be achieved by using noise augmentation techniques when training the networks. Noise augmentation methods can also be done using generative models or adversarial attacks in the training phase. Another issue to be improved is that models tend to neglect local events while they are very efficient at detecting regional or teleseismic events. This is contrary to the supervised learning-based counterparts, which are observed to be better at detecting local events. This leads to better performance metrics for the INSTANCE dataset (above STA/LTA and supervised counterparts) and poorer performance (comparable or lower than STA/LTA and lower than supervised counterparts) at the STEAD dataset.

Another improvement issue is making the models capable of detecting phase arrival times. Although proposed methods form candidate solutions to general-purpose

unsupervised classification problems, they can't detect phase arrival times or, in other words, can not be used for phase picking. Future work includes developing unsupervised methods such that seismic phase picking is also possible. Representation Cross-covariances method seems to provide segmented representations, which can be a good starting point for phase picking.

4. MULTI STATION DETECTION

4.1. Introduction

In the previous chapter, we focused on performing unsupervised learning-based classification on a single station. However, generic seismic workflow involves the usage of multiple stations, even if the first step is applying single-station detection methods. We can exploit the additional information provided by the similarities of the waveforms received from different stations. Cross-correlation-based earthquake detection algorithms are motivated by the same idea [29–35]. However, they are hindered by the fact that the cross-correlation function is not robust to the nonlinearities, dispersion, and frequency-dependent losses introduced by the propagation medium, which limits their application area to seismic arrays since the stations are very close. In this section, we want to show that if we apply similar algorithms on the latent space instead of raw waveforms, we can eliminate the limits of the classical cross-correlation methods. Cross-correlation at latent space can be used in regional seismic networks with inter-station distances of hundreds of kilometers, while classical cross-correlation is limited to a few kilometers. Besides, we can solve beamforming problems by using a Graph Autoencoder.

All methods proposed in this section work on graph structures, and they perform station-level waveform classification. Graphs are mathematical structures that represent pairwise relations between objects called nodes, and they are convenient to represent the network of stations. In our case, each node corresponds to a station connected to other stations with edges. Edge weights are calculated using intersection distances. Although we could use trainable edge weights, it would be inconvenient for the portability of the method to different domains. We propose three methods with increasing complexity and performance.

The first method takes cross-covariances of representations at each node with its neighbors in addition to autocovariance. CNN Autoencoder for obtaining latent representations are shared for each station. Covariances are then converted to scalars by classification metrics. We take their weighted average since each node has multiple cross-covariances and an autocovariance. At the same time, we take the weights from the adjacency matrix (the diagonal of the matrix is composed of ones).

The second method uses a Graph Neural Network (GNN) Autoencoder topology to solve the detection problem. The proposed GNN Autoencoder uses a novel Graph Attention algorithm, which aligns and sums each node's neighbor node feature vectors. While aligning the neighbor feature vectors, the model learns better ways to detect similarities other than translating and calculating the cosine similarity (cross-covariance). After alignment, the weighted average of the cosine similarity of the center node with its neighbors becomes the detection metric due to ease of calculation.

GNN Autoencoder-based method can be used for beamforming purposes as a side benefit. Beamforming is a technique that enhances the SNR of a signal by fusing information from multiple sensors by aligning and summing them. Classically, alignment is done by using translation operators, which is only effective for dense seismic arrays. Since the proposed GNN Autoencoder aligns the signals at latent space with a learnable operator, it's much more robust to the effects introduced by the propagation medium, making its usage possible for sparse seismic networks. According to our knowledge, this would be a novel contribution as well.

4.2. Training and Testing Procedures

Dataset preparation is more challenging when it comes to multistation detection. STEAD dataset, having too few events and a short time window, was not convenient. Due to this, we have used INSTANCE dataset for training and testing purposes. Each event has been detected at some stations, which is appropriate to be represented as a

graph. We have used a dynamic graph structure since the stations picking the events change. Details of forming the graph structure can be found in section 4.2.2.1.

Since traces in the INSTANCE dataset have different start times, we needed to find a time window for each event and crop the waveforms accordingly. We have selected the start of the time window randomly within the minimum possible interval, encapsulating all start times of the related event.

One major problem was formulating classification problems in the graph language. The most convenient way to achieve that was to make the classification on the node level based manner. However, all nodes naturally correspond to earthquake waveforms since graph structures are formed using the dataset's event information. We added randomly selected noise waveforms to each event in order to train and test the developed methods. Besides, the adjacency matrix is updated accordingly.

We used ADAM optimizer with the same hyperparameters as in section 3.2.2. The batch size is selected as 256 as in the last section. We have trained the GNN Autoencoder model for 40 epochs, while CNN Autoencoder is trained for 20 epochs. The training, validation, and test splits are chosen as 0.6, 0.2, and 0.2. Training took approximately 6.5 minutes for an epoch for the GNN Autoencoder while it had taken 1.5min for CNN Autoencoder at NVIDIA GTX3090TI GPU. Due to time constraints, we postponed the application of K-Fold validation methods.

4.2.1. CNN Autoencoder

CNN Autoencoder is trained similarly to section 3.2.2.1.

4.2.2. GNN Autoencoder

4.2.2.1. Adjacency Matrix. INSTANCE dataset contains a variable number of waveforms recorded at different stations for each event. Representing this with graph for-

malism requires using different graphs for each event. For efficiency considerations, we embed multiple graph structures in a batch-level adjacency matrix, which has a block diagonal form. Each matrix at the diagonal corresponds to a network of stations that recorded the event's seismic signals.

Elements of the batch-level adjacency matrix a_{ij} are calculated by using the distance between stations of waveforms by the following formula

$$a_{ij} = \begin{cases} \exp(-d_{ij}/d_0) & \text{, Waveforms of the same event} \\ 0 & \text{, Waveforms of the different events} \end{cases}, \tag{4.1}$$

while d_0 is a hyperparameter that controls the strength of the relation between nodes, which is set to 50 kilometers in our experiments. As shown in Equation 4.1, loops are allowed in the graph structure since we also want to use the information from the same station. During training, the dropout method is applied to the adjacency matrix to force the network to use information from adjacent nodes. For each batch, randomly chosen elements of the adjacency matrix (with 0.5 probability for each element) are masked (made zero). If all elements of the same row become zero, the element corresponding to the same station is set to one.

4.2.2.2. Loss Function. The loss function used for training is a form of reconstruction loss. Before calculating the loss, the means of each channel are subtracted such that $\frac{1}{N}\sum_n x_{inc} = 0$, $\frac{1}{N}\sum_n y_{inc} = 0$ while $x_i \in \mathbb{R}^{NxC}$, $y_i \in \mathbb{R}^{NxC}$ denote the input and output of the model for the i-th node while N and C denote the number of timesteps and channels of the waveforms $(3000, 3)$. The reconstruction loss L_{recons} is calculated as follows

$$L_{recons} = \sqrt{\frac{1}{VNC}\sum_{v=0}^{V-1}\sum_{n=0}^{N-1}\sum_{c=0}^{C-1}(x_{vnc} - y_{vnc})^2}, \tag{4.2}$$

while V is the number of nodes, N is the number of timesteps, and C is the number of channels.

4.3. Building Blocks

4.3.1. CNN Autoencoder

CNN Autoencoders are used to map the waveform data to a latent space. We used the same CNN Encoder and Decoder architectures as in section 3.1 to investigate the effect of introducing multiple stations for detection.

4.3.2. Cross-covariance of Node Feature Vectors

Classification is done for each node of the graph. We use the neighbors of each node as well as itself for classification. Showing the node to be classified as i and its neighbors as $j \in N_i$, we can express the cross-covariance of the node i with node j as follows

$$\sigma_{ijd} = \sum_{c=0}^{C-1} \sum_{n=0}^{N-1} h_{inc} h_{j(n+d)c}, \tag{4.3}$$

where h_{inc} is the feature vector of the node i at n-th timestep and c-th channel. σ_{ijd} is the cross-covariance between node i and node j for $d \in [-\frac{N}{2}, \frac{N}{2}]$ differential timesteps. Applying it in the frequency domain can also make this operation lighter. Frequency domain implementation is given in the following equation

$$\begin{aligned} H_{vkc} =& \mathcal{F}_{kn}\{h_{vnc}\} \\ \sigma_{ijd} =& \mathcal{F}_{dk}^{-1}\{\sum_{c} H_{ikc} H_{jkc}^{*}\}, \end{aligned} \tag{4.4}$$

while k is the discrete frequency index, c is the channel index, and the H_{vkc} is the frequency domain representation of node v's feature vector.

4.3.3. Graph Attention with Translation Alignment

According to the origin and characteristics of earthquakes, different stations can be similar, which can not be encoded by a static adjacency matrix. As a result, the behavior of the neural network should be changed based on input data. Graph

Attention provides a way to achieve this by exploiting the similarities between nodes and weighting the information according to this. According to our knowledge, currently studied Graph Attention architectures assume that encoded representations can be summed directly without losing expressional power. Or they are concatenated and passed through various layers, increasing the problem's dimensionality proportional to the number of neighbors. Our observations show that although the waveform is encoded, its topological structure and, hence, the alignment matters. Consequently, adding various feature vectors without aligning them leads to poorer representations.

We propose a Graph Attention layer that efficiently solves the alignment problem. As well as weighting the neighbors as in the proposed mechanisms, we also weight different alignments while fusing feature vectors from the neighbors for each node. The alignment problem is solved with translations done efficiently in the frequency domain.

In our method, we first subtract the mean of each channel from the feature vectors. Then, we project feature vectors to fusion and attention spaces by pointwise linear maps (for each timestep acting in the same manner) as given below

$$\begin{aligned} h_{vnc}^{(A)} &= W_{cc'}^{(A)} h_{vnc'} \\ h_{vnc}^{(F)} &= W_{cc'}^{(F)} h_{vnc'}, \end{aligned} \tag{4.5}$$

while $W_{cc'}^{(A)}$ and $W_{cc'}^{(F)}$ are the projection matrices for attention and fusion spaces and $c, c' \in [0, C-1]$. $h_{vnc}^{(A)}$ and $h_{vnc}^{(F)}$ are the attention and fusion space projections of the latent representation of the v-th node.

After mapping the feature vectors, we perform a fusing algorithm consisting of two phases. The first phase involves calculating shift attention weights for different delays for each neighbor. Shift attention weights α_{ijd} are calculated for different delays for the center node i and neighbor node j where $\mathbf{t}_1$ and d are the unit translation operator and $d \in [-\frac{N}{2}, \frac{N}{2} - 1]$ is the amount of shift. Defining dot product for feature vectors a and b as $a \cdot b = \sum_{n,c} a_{nc} b_{nc}$, we can express the shift attention weights as

follows

$$\begin{aligned} \hat{h}_v^{(A)} &= \frac{h_v^{(A)}}{\sqrt{h_v^{(A)} \cdot h_v^{(A)}}} \\ \alpha_{ijd} &= \frac{\exp(\hat{h}_i^{(A)} \cdot (\mathbf{t}_1^d \hat{h}_j^{(A)}))}{\sum_e \exp(\hat{h}_i^{(A)} \cdot (\mathbf{t}_1^e \hat{h}_j^{(A)}))}, \end{aligned} \tag{4.6}$$

while n, c are the timestep and channel indices. $\hat{h}_v^{(A)}$ is the normalized attention space projection of latent representation at node v.

After calculating the shift attention weights for different delays, we can align the feature vector of the node j for both attention and fusion spaces. While we normalize the feature vectors for calculating the shift attention weights, we don't normalize them for the fusion step to preserve the information about the magnitude of the feature vectors. We take the weighted average of translated versions of neighbor latent representations as given below

$$\begin{aligned} g_j^{(A)} &= \sum_d \alpha_{ijd} \mathbf{t}_1^d \hat{h}_j^{(A)} \\ g_j^{(F)} &= \sum_d \alpha_{ijd} \mathbf{t}_1^d h_j^{(F)}, \end{aligned} \tag{4.7}$$

while $g_j^{(A)}$ and $g_j^{(F)}$ represent aligned latent space projections of the j-th neighbor for attention and fusion spaces.

We can reduce the calculational complexity by performing the operations at the frequency domain, which transforms the alignment process to multiplying with a tensor. Translation operator $\mathbf{t}_1$ can be represented by multiplication with a diagonal matrix in the frequency domain if we circularly translate vectors. Taking powers of a diagonal matrix is drastically easier, and the amount of required memory for the backpropagation algorithm reduces significantly. When converted to the frequency domain, shift attention weights α_{ijd} are given as

$$\alpha_{ijd} = \frac{\exp(\hat{H}_i^{(A)} \cdot (\mathbf{t}_1^d \hat{H}_j^{(A)}))}{\sum_e \exp(\hat{H}_i^{(A)} \cdot (\mathbf{t}_1^e \hat{H}_j^{(A)}))} = \frac{\exp(\sum_{k,c} \Re\big[\hat{H}_{ikc}^{(A)} (\Lambda_k^d \hat{H}_{jkc}^{(A)})^*\big])}{\sum_e \exp(\sum_{k,c} \Re\big[\hat{H}_{ikc}^{(A)} (\Lambda_k^e \hat{H}_{jkc}^{(A)})^*\big])}, \tag{4.8}$$

while $\hat{H}_i^{(A)} = \mathcal{F}_{kn}\hat{h}_k^{(A)}$ is the frequency domain representation of the feature vector at node v, and $\Lambda_k = \exp(i2\pi f_s \frac{k}{N})$ expresses a tensor that has eigenvalues of the circulant translation operator. k, f_s, and $\Re$ denote the discrete frequency index, sampling frequency, and the operator taking the real part of a complex number.

After calculating the shift attention coefficients, we can calculate the aligned feature vectors by the following equation in the frequency domain

$$\begin{aligned} G_j^{(A)} &= \sum_d \alpha_{ijd}(\Lambda_k^d)\hat{H}_{jkc}^{(A)} = \hat{H}_{jkc}^{(A)} \sum_d \alpha_{ijd}(\Lambda_k^d) \\ G_j^{(F)} &= \sum_d \alpha_{ijd}(\Lambda_k^d)H_{jkc}^{(F)} = H_{jkc}^{(F)} \sum_d \alpha_{ijd}(\Lambda_k^d), \end{aligned} \tag{4.9}$$

while $G_j^{(A)}$ is the aligned feature vector of the node j at the attention space and $G_j^{(F)}$ is the aligned feature vector of the node j at the fusion space in the frequency domain. We see in Equation 4.9 that $\sum_d \alpha_{ijd}\Lambda_k^d$ factor gets seperated. This improves the performance significantly, and it's one of the main reasons for performing the operations in the frequency domain.

Assigning a symbol L_{ijkc} to $\sum_d \alpha_{ijd}\Lambda_k^d$ for later uses is beneficial. We also name it alignment tensor, the Fourier domain representation of alignment operator L_{ij}. Abstract alignment operator L_{ij} is considered to align the neighbor nodes to the center node such that their cosine similarity is high at the attention space. We added a channel index to its representation for convenience, although it acts similarly for all channels.

As a result, the weighted average of all possible translations with shift attention coefficients can be deduced to element-wise multiplication when performed on the frequency domain. After obtaining aligned versions of the feature vectors, we calculate the neighbor attention weights by the following equation

$$\kappa_{ij} = \frac{\exp(\hat{G}_i^{(A)} \cdot (a_{ij}L_{ij}\hat{G}_j^{(A)}))}{\sum_j \exp(\hat{G}_i^{(A)} \cdot (a_{ij}L_{ij}\hat{G}_j^{(A)}))} = \frac{\exp(\sum_{k,c} \hat{G}_{ikc}^{(A)}(a_{ij}L_{ijkc}\hat{G}_{jkc}^{(A)})^*)}{\sum_j \exp(\sum_{k,c} \hat{G}_{ikc}^{(A)}(a_{ij}L_{ijkc}\hat{G}_{jkc}^{(A)})^*)}, \tag{4.10}$$

while κ_{ij} is the attention weight of node j. a_{ij} is the element of the adjacency matrix denoting the strength of the relation between node i and node j. Dot product is defined as $a \cdot b = \sum_{k,c} a_{kc} b^*_{kc}$ for tensors a and b in this case since we are working in the frequency domain. After calculating the attention weights, we can calculate the layer output by the following equation

$$o_i = \mathcal{F}^{-1}\{\sum_j \kappa_{ij} G_j^{(F)}\}, \tag{4.11}$$

while o_i is the layer output for the i-th node.

Operation given in Equation 4.11 has the same order of complexity as the proposed attention mechanisms in the literature. Besides, loss of information due to compression is eliminated. Another option would be to apply a pairwise attention mechanism followed by position embedding. However, it would require too much computational power, and it's considered unnecessary considering the propagation behavior of seismic waves.

4.3.4. Graph Attention with Trainable Alignment

The alignment operation is done with the shift operator in section 4.3.3. However, the propagation of seismic waveforms is not limited to mere translation, even though feature vectors are higher-level structures. We propose using a learnable operator to encompass a wider set of transformations. This operator will form a generator for the possible set of transformations as in the translation case. h_k denoting feature vector at k-th node we define the generator $\mathbf{r}$ as follows

$$\mathbf{r}_1 h_v = \mathbf{t}_1(h_v + f * h_v), \tag{4.12}$$

while $\mathbf{t}_1$ is the translation operator, which translates the vector by a timestep. $\mathbf{f}$ is the learnable convolution kernel that is defined for each channel, while $*$ is the channel-wise circular convolution operator. Calculating attention coefficients can be done similarly by exchanging the translation operator with the learnable operator $\mathbf{r}_1$. Since the Fourier Transform can diagonalize the circular convolution operator, all operations can be

done much more efficiently in the frequency domain, as in section 4.3.3. However, we have different alignment operators for each channel in this case. We denote the representation of the operator $\mathbf{r}_1$ at Fourier basis as Λ_{kc} instead of Λ_k due to this. Λ_{kc} can be calculated by the following equation

$$\begin{aligned} F_{kc} &= \mathcal{F}_{kn}\{f_{nc}\} \\ \Lambda_{kc} &= \exp(i2\pi f_s \frac{k}{N})(1 + F_{kc}), \end{aligned} \tag{4.13}$$

while F_{kc} is the learnable filter kernel in the frequency domain, and f_s is the sampling frequency in Hz.

As a result, we can express the attention coefficients as follows

$$\alpha_{ijd} = \frac{\exp(\hat{H}_i^{(A)} \cdot (\mathbf{r}_1^d \hat{H}_j^{(A)}))}{\sum_e \exp(\hat{H}_i^{(A)} \cdot (\mathbf{r}_1^e \hat{H}_j^{(A)}))} = \frac{\exp(\sum_{k,c} \Re\big[\hat{H}_{ikc}^{(A)} (\Lambda_{kc}^d \hat{H}_{jkc}^{(A)})^*\big])}{\sum_e \exp(\sum_{k,c} \Re\big[\hat{H}_{ikc}^{(A)} (\Lambda_{kc}^e \hat{H}_{jkc}^{(A)})^*\big])}. \tag{4.14}$$

Rest is also similar to the section 4.3.3. We just need to switch Λ_w with Λ_{wc}, which leads to the following results

$$\begin{aligned} G_j^{(A)} &= \sum_d \alpha_{ijd} (\Lambda_{kc}^d) \hat{H}_{jkc}^{(A)} = \hat{H}_{jkc}^{(A)} \sum_d \alpha_{ijd} (\Lambda_{kc}^d) \\ G_j^{(F)} &= \sum_d \alpha_{ijd} (\Lambda_{kc}^d) H_{jkc}^{(F)} = H_{jkc}^{(F)} \sum_d \alpha_{ijd} (\Lambda_{kc}^d), \end{aligned} \tag{4.15}$$

while $G_j^{(A)}$ and $G_j^{(F)}$ are the aligned neighbor representations at the frequency domain.

The alignment tensor is defined in a similar manner $L_{ijkc} = \sum_d \alpha_{ijd} (\Lambda_{kc}^d)$. As a result, we can express neighbor weights κ_{ij} and the layer output o_i as given in the Equations 4.10 and 4.11 respectively.

4.3.5. Multi-Head Graph Attention

We can independently apply the graph attention mechanism to multiple attention heads and then merge their results. This provides the flexibility to handle features with different propagation characteristics by different attention heads.

By labeling attention projector $W^{(A_h)}$, fusion projectors $W^{(F_h)}$ and the convolutional kernel f^h specific to each head, we can obtain the outputs for all heads by applying the same procedure. We can express the output for each head $h \in [0, H-1]$ while H denotes the number of heads as follows

$$o_i^h = \mathcal{F}^{-1}\{\sum_j \kappa_{ij}^h G_j^{(F_h)}\}, \tag{4.16}$$

while o_i^h is the output of h-th attention head at i-th node.

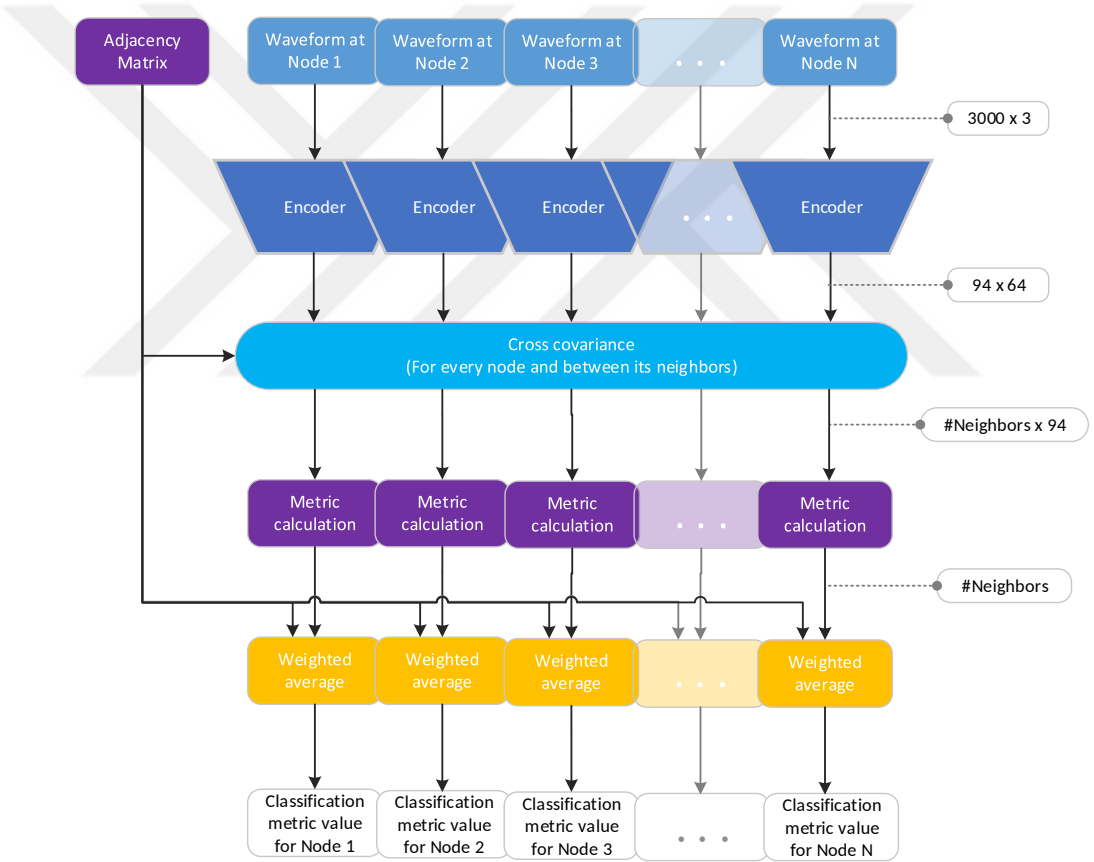


Figure 4.1. Network Cross-covariance of representations.

After calculating the outputs for each head separately, we concatenated them. Then linear transformation $W^{(O)} : R^{C \cdot H} \rightarrow R^O$ is applied to the concatenated vector pointwise to obtain layer output

$$o_{inc} = W_{cd}^{(O)} \cdot \left[o_i^1 \oplus o_i^2 \oplus ... \oplus o_i^H\right]_{idn}, \tag{4.17}$$

while O denotes the number of output channels. o_{inc} is the output tensor for the i-th node having the same dimensionality as the layer input ($n \in [0, N-1], c \in [0, C-1]$).

4.4. Methods

4.4.1. Detection with Network Cross-covariance of Representations

The Autocovariance method at section 3.3.3 can be applied to graph structure easily. In this case, we may exploit the similarities between feature vectors of different stations by taking cross-covariance between them and using it for waveform classification. Since we expect that features of noise waveforms have less coherence length, this could be a good discriminator between earthquake and noise waveforms.

Given the adjacency matrix a_{ij}, we can calculate a metric M_i that can be used for detecting earthquakes for each node i as follows

$$M_i = \sum_j a_{ij} m(\sigma_{ijn}), \tag{4.18}$$

while m is a function that is applied to the cross-covariance of the node i with node j. The function m can be chosen as in the single station case simply by taking the maximum along the timesteps axis as in Equation 3.7 or taking the weighted average as in Equation 3.8. However, the peak of the Gaussian shouldn't be selected as $\tau = 0$ in this case. Instead, it should be selected as the timestep of maximum cross-covariance. This is because we expect the earthquake to be detected at different times in different stations. As a result, we can define the function M as follows

$$\begin{aligned} m_{ij} &= \arg\max_m \sigma_{ijm} \\ M_i &= \frac{1}{\sqrt{2\pi}\sigma_0} \sum_{j,n} a_{ij} \sigma_{ijn} \exp\left(- \frac{(n - m_{ij})^2}{2 f_s \sigma_0^2} \right), \end{aligned} \tag{4.19}$$

where f_s is the sampling frequency, and σ_0 is the standard deviation of the Gaussian. For empirical reasons, we have seen that it's decent to choose σ_0 as 5.0 seconds, although its optimal value depends on the distance between stations. However, it should

be mentioned that even the maximum value of the Cross-covariance also gives a decent performance, so the performance is not related significantly to the value of this parameter. The parameter value can be selected within the range of 1.0-10.0 seconds.

4.4.2. Detection and Beamforming with Graph Attention Autoencoder

4.4.2.1. Working Principle. In section 4.4.1, we have used cross-covariances of a center node between its neighbors as a classification metric. As mentioned, the cross-covariance operation involves calculating the covariance between two representations under different delays. However, we took cross-covariance due to the requirement that we didn't know the translation required to align two waveforms and merely taking covariance would fail.

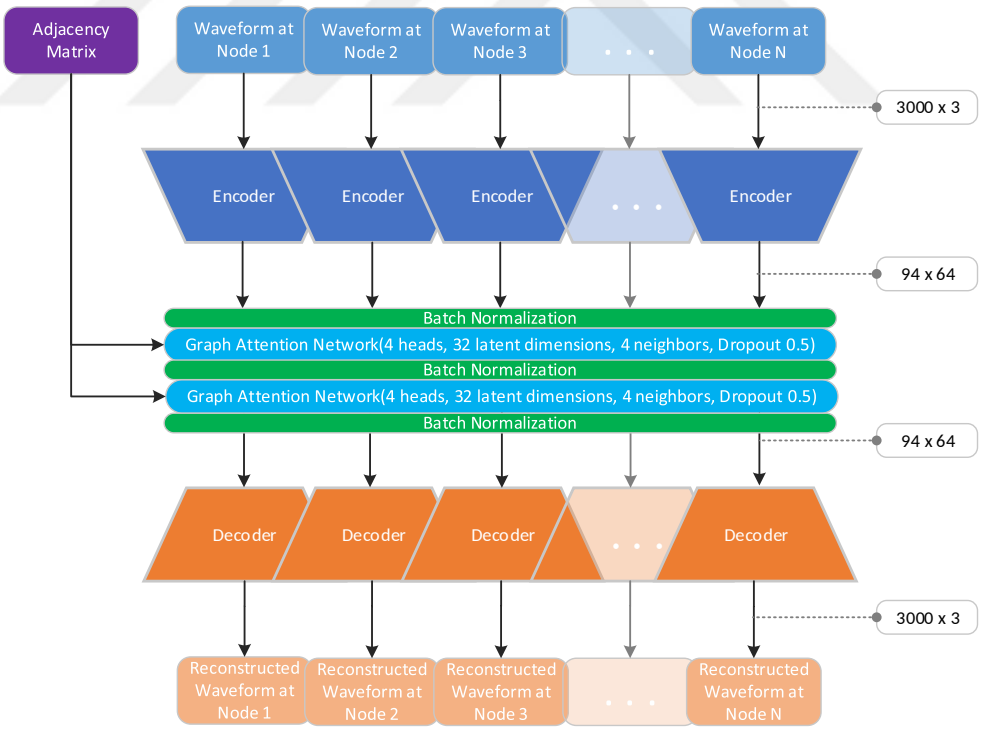


Figure 4.2. Graph Attention Autoencoder architecture.

Introducing Graph Attention with the alignment capability solves this problem. Our experiments have shown that trainable alignment gives better results, which are

described in section 4.3.4. To train the alignment operator in an unsupervised manner, we propose using a Graph Autoencoder architecture composed of CNN Encoder, CNN Decoder, and Graph Attention structures. The model encodes waveforms using CNN Encoders and fuses the information from different nodes with Graph Attention layers. Then, the CNN Decoder reconstructs the original waveform by using the fused representations. After obtaining the aligned neighbor representations, we measure the cosine similarity of the neighbor representations with the center node representation. We get the final detection metric by taking the weighted average of these similarities using the adjacency matrix. The procedure can be expressed as follows

$$M_i = \mathcal{F}^{-1}\{\sum_h \sum_j H_i^{(F_h)} \kappa_{ij}^h G_j^{(F_h)}\}, \tag{4.20}$$

while M_i is the detection metric, $H^{(F_h)}$ and $G^{(F_h)}$ are the representation projections (to the fusion space) of the center node and aligned neighbor nodes at Fourier domains for each head. κ_{ij}^h is the j-th neighbor weight to i-th center node which is specific to attention head h.

4.4.2.2. Architecture. While keeping the encoder and decoder parts similar to the other models, we have introduced two Graph Attention layers with four attention heads. The trainable part of the alignment operator comprises five convolutional filters for each projection channel (projection dimension is chosen as 32). Dropout is applied to the adjacency matrix with the rate of 0.5 in order to force the model to use information from adjacent nodes. The model's architecture is shown in Figure 4.2.

4.5. Results

4.5.1. Beamforming Results

After training for 40 epochs, we fed the test waveforms to the model and observed the outputs of various layers. Interestingly, latent space samples are seen to encode a sort of arrival time, which can be seen in Figure B.1 and Figure B.2. We consider

that the alignment process applied by Graph Attention forces the network to represent arrival times. Besides, it's observed that reconstructions are highly accurate when we don't apply dropout during testing time, as seen in Figure B.1 and Figure B.2. However, applying dropout leads to a significant improvement in the denoising capabilities of the model, as seen in Figure B.3 and Figure B.4.

When we don't apply dropout, since the model is trained to reconstruct the original waveform, it tries to replicate the original waveform even if it's pure noise, which is seen in Figure B.5 and Figure B.6. For denoising purposes, either the dropout method or augmentation techniques (such as injecting noise into the input) should be used.

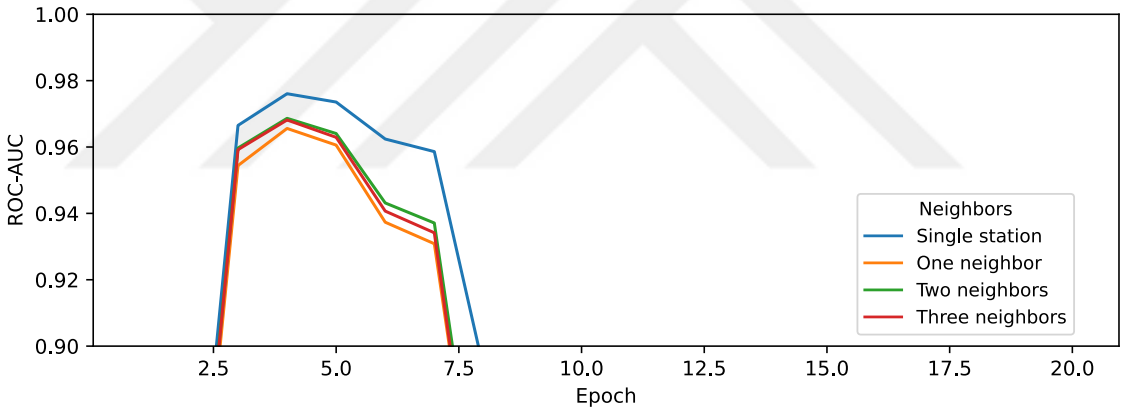


Figure 4.3. Network Cross-covariance performance over training duration.

4.5.2. Detection Results

4.5.2.1. Network Cross-covariance of Representations. When investigating the ROC-AUC score during the course of training, we see that the method works decently only at some point. Except that, performance is very poor. Although the same issue is also encountered with single-station methods, it's not so severe. We consider that the problem is increased noise and CODA content of the waveforms due to the cropping procedure. This is considered to be related to the cropping procedure applied when preprocessing the dataset. Since we are cropping 30.0s time windows out of 120.0s

intervals, most of the waveform samples miss the crucial parts and comprise noise and CODA waves. This affects the method on two levels. One is the reduced filtering capability of the CNN Autoencoder, and the other is the degraded cross-covariance performance.

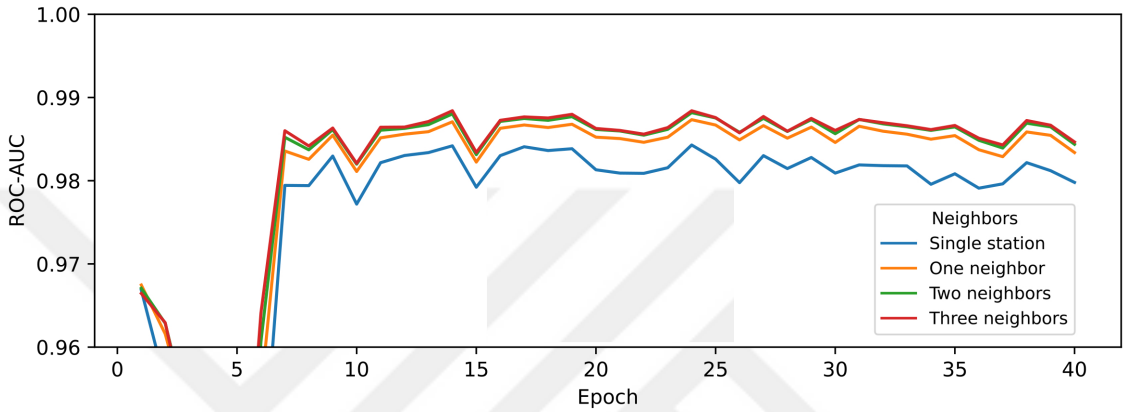


Figure 4.4. GNN Autoencoder Detector performance concerning training duration.

4.5.2.2. Detection with Graph Attention Autoencoder. The use of GNN Autoencoder improved the detection performance significantly compared to merely using cross-covariance. This is related to the selective behavior of the attention mechanism that forced the CNN Encoder to learn the features related to phase arrivals, forming a naturally emerging supervision signal. Figures B.3, B.4, B.1 and B.2 show the effect of the attention mechanism over the latent representation obtained from the CNN Encoder. The latent space samples show that the attention layers emphasize the parts of the latent representation related to phase arrivals even more. Besides, its performance is much more stable during training. Figure 4.4 shows the ROC-AUC score regarding training duration.

It's seen that even for single station detection, performance is higher than the previously proposed methods by the ROC-AUC score of around 0.004. Performance increases with the use of more neighbors.

4.6. Discussion and Future Work

We observed that the multistation cross-covariance method doesn't improve the performance significantly compared to the best single-station method. We consider that the result is related to choosing a global time window for all stations. This leads to a training set in which most cropped waveforms contain Coda waves or noise signals but not phase arrivals. Our result for the no-neighbor case shows that the result is worse than the single stations, suggesting that the CNN Autoencoder learns to replicate noise signals that significantly degrade selective behavior. Using more neighbors makes the performance less dependent on the training duration, but similar behavior is also observed in classical phase association algorithms.

On the other hand, the GNN Autoencoder method gives better results than the single station methods, even when no neighbors are involved during the detection phase. The result is related to Attention Mechanism's selective behavior on some features that can be used for alignment. Latent space samples show that phase arrivals are significantly related to emphasized parts of latent representation obtained on the output of Graph Attention Layers. This opens a new way to train the network with a dataset while there are very few earthquakes. Or a continuous stream of waveform. We believe that this could provide improvement for low SNR signal detection.

Other possible applications involve Machine Learning based beamforming mechanisms. To achieve this, we need to train the GNN Autoencoder as a denoising Autoencoder. This consists of injecting random noise signals into the stations and expecting the model to give the original waveforms. Besides, some gaps can be introduced to waveform samples, another widely encountered corruption in seismic signal recordings. We didn't inject noise or gaps at the current state, but some waveform samples show that denoising properties show themselves even in this case. We will soon train the network this way and obtain accurate performance metrics for SNR improvement.

Due to time constraints and the variety of proposed methods, we couldn't thoroughly evaluate the multistation methods' performance. However, when we compare the performance of the multistation methods tested in the no-neighbor case, we see that multistation training improves the performance in the GNN Autoencoder case. Future work involves forming accurate baselines and comparing the performance of the models with conventional algorithms. Besides, the method of introducing noisy nodes to graph structure may have some flaws. The most convenient way would be to create a dataset for this purpose and check if the obtained results are compatible with the current case.

5. CONCLUSION

We believe the thesis has achieved its primary goal of developing reliable unsupervised methods for classification tasks. Although possible application areas have not been discovered yet, we consider that the proposed classification algorithms apply to different noise/signal classification tasks. In addition to that, we have developed a new method for sensor networks to improve the SNR ratio for challenging configurations. In order to achieve that, we have proposed a novel Graph Attention structure that may contribute to Machine Learning research.

Although we have obtained promising results, we believe we may improve the limitations of the proposed methods further. Additionally, more comprehensive benchmarking procedures should be used to evaluate multistation methods. Future work will focus on improving the limitations of the proposed methods and testing them in more challenging settings.

REFERENCES

1. Mousavi, S. M., W. L. Ellsworth, W. Zhu, L. Y. Chuang and G. C. Beroza, "Earthquake Transformer—An Attentive Deep-Learning Model for Simultaneous Earthquake Detection and Phase Picking", *Nature Communications*, Vol. 11, No. 1, pp. 3952–3963, 2020.

2. Soto, H. and B. Schurr, "DeepPhasePick: A Method for Detecting and Picking Seismic Phases from Local Earthquakes Based on Highly Optimized Convolutional and Recurrent Deep Neural Networks", *Geophysical Journal International*, Vol. 227, No. 2, pp. 1268–1294, 2021.

3. Woollam, J., A. Rietbrock, A. Bueno and S. De Angelis, "Convolutional Neural Network for Seismic Phase Classification, Performance Demonstration over a Local Seismic Network", *Seismological Research Letters*, Vol. 90, No. 2A, pp. 491–502, 2019.

4. Zhu, W. and G. C. Beroza, "PhaseNet: A Deep-Neural-Network-Based Seismic Arrival-Time Picking Method", *Geophysical Journal International*, Vol. 216, No. 1, pp. 261–273, 2019.

5. Ross, Z. E., M. Meier, E. Hauksson and T. H. Heaton, "Generalized Seismic Phase Detection with Deep Learning", *Bulletin of the Seismological Society of America*, Vol. 108, No. 5A, pp. 2894–2901, 2018.

6. Münchmeyer, J., J. Woollam, A. Rietbrock, F. Tilmann, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović, A. Michelini, J. Saul and H. Soto, "Which Picker Fits My Data? A Quantitative Evaluation of Deep Learning Based Seismic Pickers", *Journal of Geophysical Research: Solid Earth*, Vol. 127, No. 1, p. e2021JB023499, 2022.

7. Chen, Y., "Fast Waveform Detection for Microseismic Imaging using Unsupervised Machine Learning", *Geophysical Journal International*, Vol. 215, No. 2, pp. 1185–1199, 2018.

8. Chen, Y., "Automatic Microseismic Event Picking via Unsupervised Machine Learning", *Geophysical Journal International*, Vol. 222, No. 3, pp. 1750–1764, 2020.

9. Duque, A., K. Gonzalez, N. Perez, D. Benitez, F. Grijalva, R. Lara-Cueva and M. Ruiz, "Exploring the Unsupervised Classification of Seismic Events of Cotopaxi Volcano", *Journal of Volcanology and Geothermal Research*, Vol. 403, p. 107009, 2020.

10. Huang, W., "Seismic Signal Recognition by Unsupervised Machine Learning", *Geophysical Journal International*, Vol. 219, No. 2, pp. 1163–1180, 2019.

11. Johnson, C. W., Y. Ben-Zion, H. Meng and F. Vernon, "Identifying Different Classes of Seismic Noise Signals using Unsupervised Learning", *Geophysical Research Letters*, Vol. 47, No. 15, p. e2020GL088353, 2020.

12. Mousavi, S. M., W. Zhu, W. Ellsworth and G. Beroza, "Unsupervised Clustering of Seismic Signals using Deep Convolutional Autoencoders", *IEEE Geoscience and Remote Sensing Letters*, Vol. 16, No. 11, pp. 1693–1697, 2019.

13. Seydoux, L., R. Balestriero, P. Poli, M. d. Hoop, M. Campillo and R. Baraniuk, "Clustering Earthquake Signals and Background Noises in Continuous Seismic Data with Unsupervised Deep Learning", *Nature Communications*, Vol. 11, No. 1, pp. 3972–3983, 2020.

14. Carniel, R., A. D. Jolly and L. Barbui, "Analysis of Phreatic Events at Ruapehu Volcano, New Zealand using a New SOM Approach", *Journal of Volcanology and Geothermal Research*, Vol. 254, pp. 69–79, 2013.

15. Köhler, A., M. Ohrnberger and F. Scherbaum, "Unsupervised Pattern Recognition in Continuous Seismic Wavefield Records using Self-Organizing Maps", *Geophysical Journal International*, Vol. 182, No. 3, pp. 1619–1630, 2010.

16. Kuyuk, H., E. Yildirim, E. Dogan and G. Horasan, "An Unsupervised Learning Algorithm: Application to the Discrimination of Seismic Events and Quarry Blasts in the Vicinity of Istanbul", *Natural Hazards and Earth System Sciences*, Vol. 11, No. 1, pp. 93–100, 2011.

17. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, Massachusetts, 2016.

18. Shearer, P. M., *Introduction to Seismology*, Cambridge University Press, New York, 2019.

19. LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.

20. He, K., X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Nevada, 2016.

21. Kipf, T. N. and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *arXiv preprint arXiv:1609.02907*, 2016.

22. Bahdanau, D., K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", *arXiv preprint arXiv:1409.0473*, 2014.

23. Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph Attention Networks", *arXiv preprint arXiv:1409.0473*, 2018.

24. Mousavi, S. M., Y. Sheng, W. Zhu and G. C. Beroza, "Stanford Earthquake Dataset

(STEAD): A Global Data Set of Seismic Signals for AI", *IEEE Access*, Vol. 7, pp. 179464–179476, 2019.

25. Michelini, A., S. Cianetti, S. Gaviano, C. Giunchi, D. Jozinović and V. Lauciani, "INSTANCE–The Italian Seismic Dataset for Machine Learning", *Earth System Science Data*, Vol. 13, No. 12, pp. 5509–5544, 2021.

26. Woollam, J., J. Münchmeyer, F. Tilmann, A. Rietbrock, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović, A. Michelini, J. Saul and H. Soto, "SeisBench—A Toolbox for Machine Learning in Seismology", *Seismological Research Letters*, Vol. 93, No. 3, pp. 1695–1709, 2022.

27. Liu, X., F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang and J. Tang, "Self-Supervised Learning: Generative or Contrastive", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 35, No. 1, pp. 857–876, 2021.

28. Glorot, X. and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks", *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, Sardini, 2010.

29. Schaff, D. P., "Semiempirical Statistics of Correlation-Detector Performance", *Bulletin of the Seismological Society of America*, Vol. 98, No. 3, pp. 1495–1507, 2008.

30. Schaff, D. P., "Broad-Scale Applicability of Correlation Detectors to China Seismicity", *Geophysical Research Letters*, Vol. 36, No. 11, 2009.

31. Schaff, D. P. and F. Waldhauser, "One Magnitude Unit Reduction in Detection Threshold by Cross-Correlation Applied to Parkfield (California) and China seismicity", *Bulletin of the Seismological Society of America*, Vol. 100, No. 6, pp. 3224–3238, 2010.

32. Savard, G. and M. G. Bostock, "Detection and Location of Low-Frequency Earth-

quakes using Cross-Station Correlation", *Bulletin of the Seismological Society of America*, Vol. 105, No. 4, pp. 2128–2142, 2015.

33. Gibbons, S. J. and F. Ringdal, "The Detection of Low Magnitude Seismic Events using Array-Based Waveform Correlation", *Geophysical Journal International*, Vol. 165, No. 1, pp. 149–166, 2006.

34. Gibbons, S. J., M. B. Sórensen, D. B. Harris and F. Ringdal, "The Detection and Location of Low Magnitude Earthquakes in Northern Norway using Multi-Channel Waveform Correlation at Regional Distances", *Physics of the Earth and Planetary Interiors*, Vol. 160, No. 3-4, pp. 285–309, 2007.

35. Bergen, K. J. and G. C. Beroza, "Earthquake Fingerprints: Extracting Waveform Features for Similarity-Based Earthquake Detection", *Pure and Applied Geophysics*, Vol. 176, pp. 1037–1059, 2019.

APPENDIX A: SAMPLES FOR SINGLE STATION METHODS

This appendix involves visualizations of latent space representations of the waveform samples as well as model inputs and outputs for the single station methods.

A.1. Randomly Parametrized Model

There, we have the samples, which are obtained by using an untrained CNN Autoencoder model.

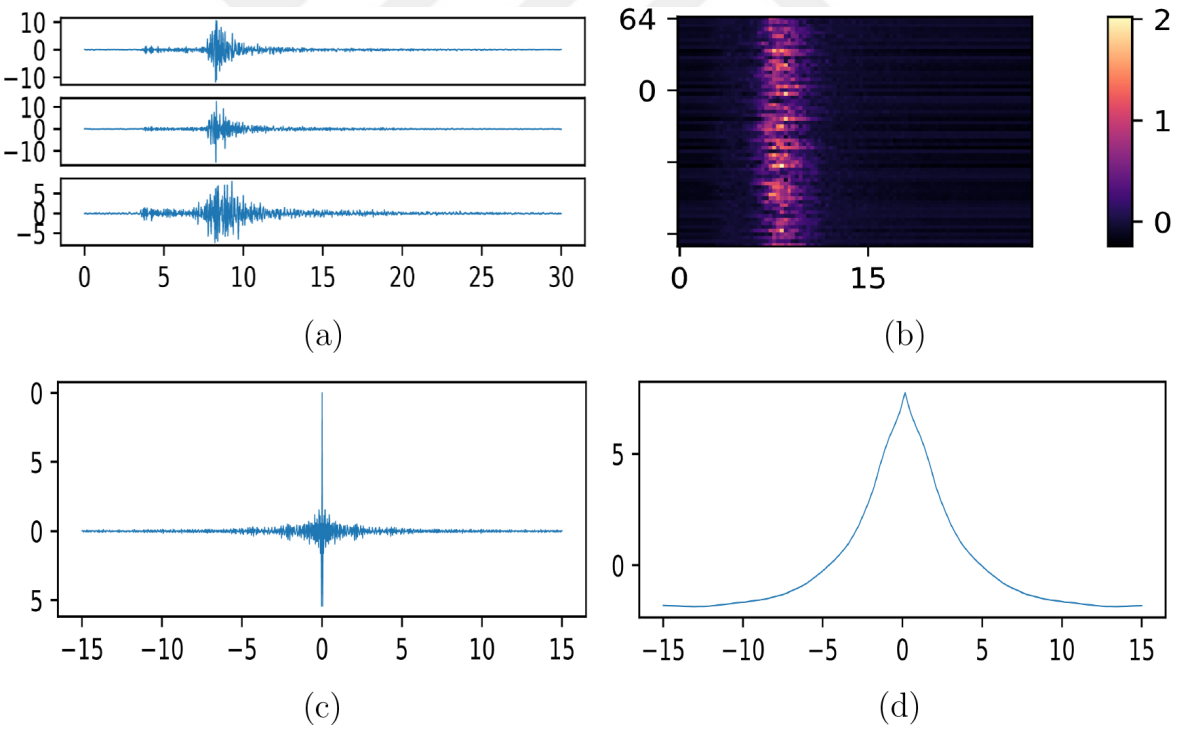


Figure A.1. Earthquake sample 1 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

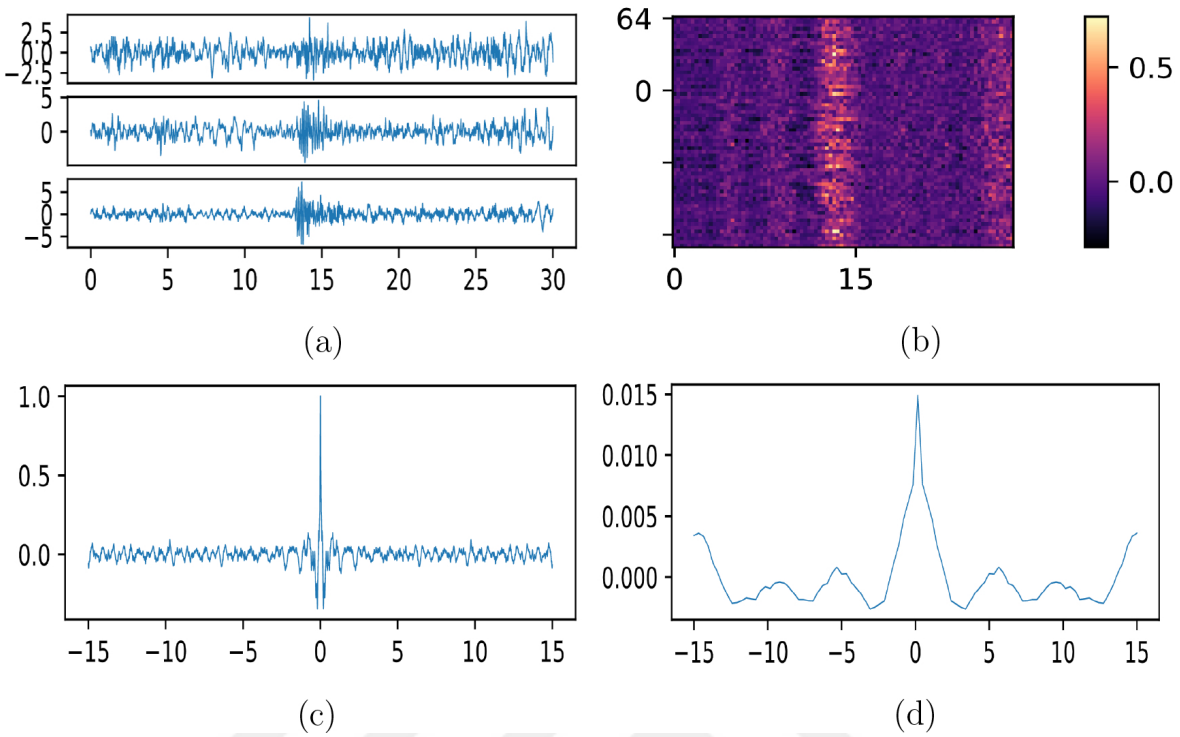


Figure A.2. Earthquake sample 2 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

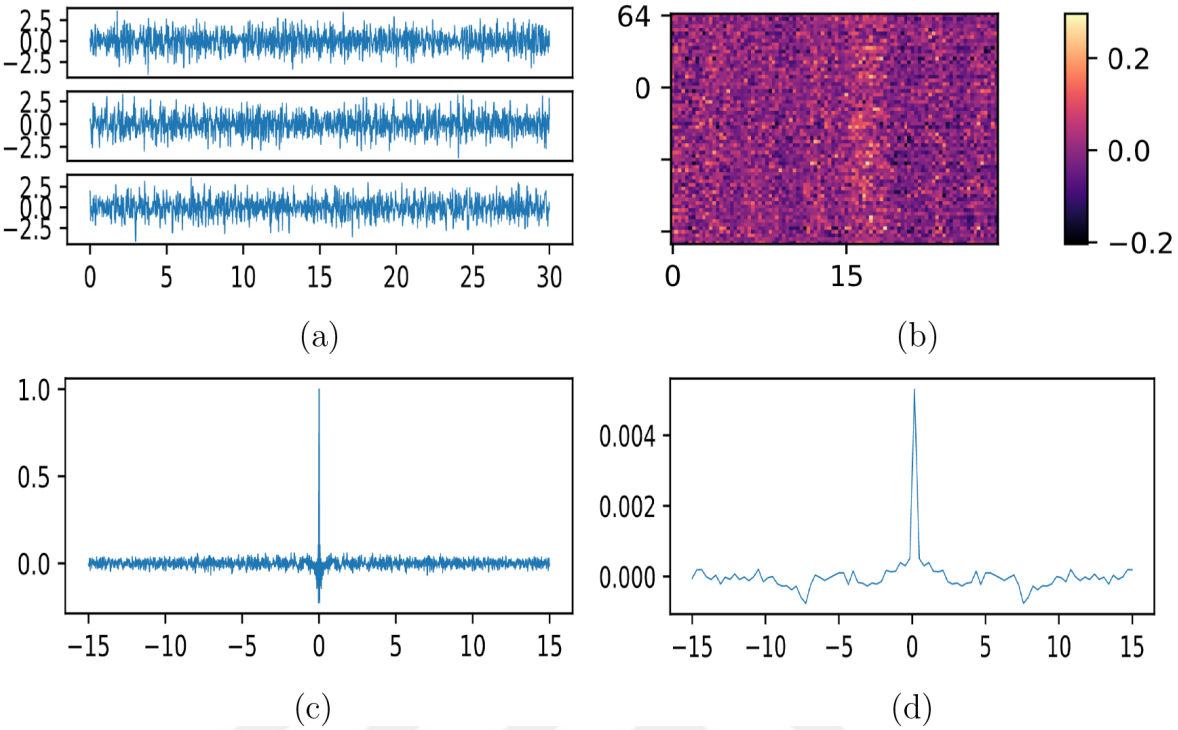


Figure A.3. Noise sample 1 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

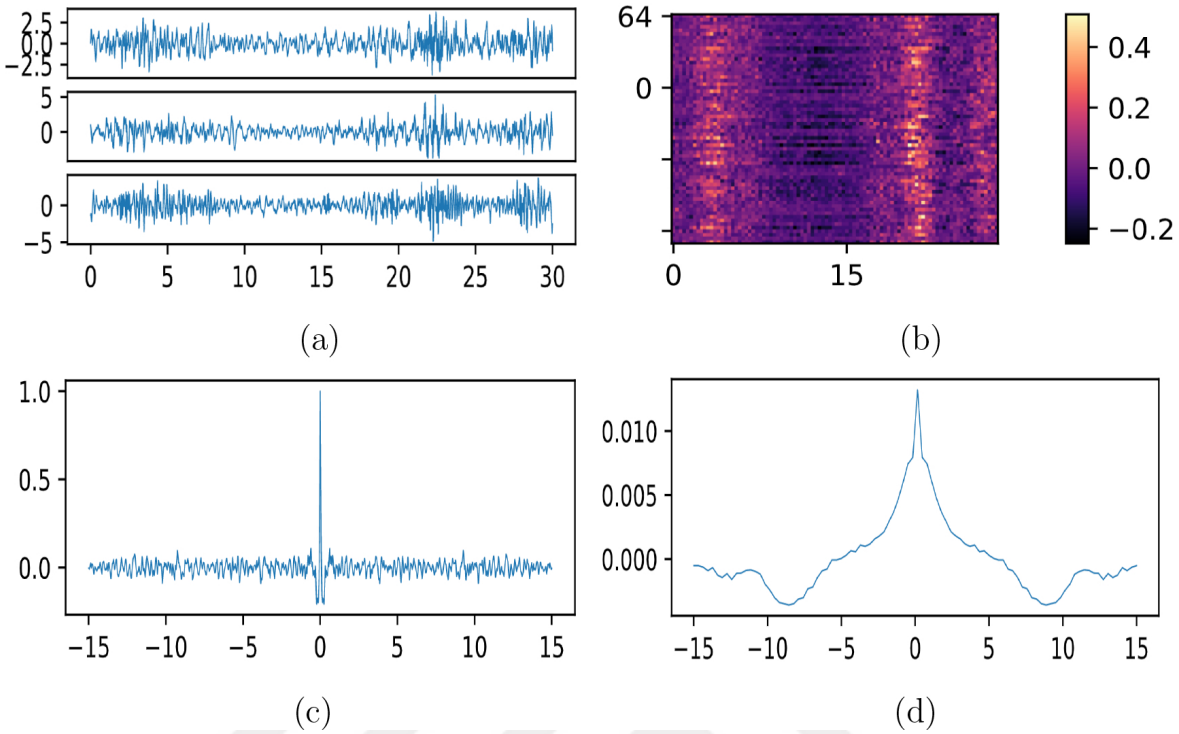


Figure A.4. Noise sample 2 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

A.2. Autocovariance Method

There, we have the samples, which are obtained by using a trained CNN Autoencoder model.

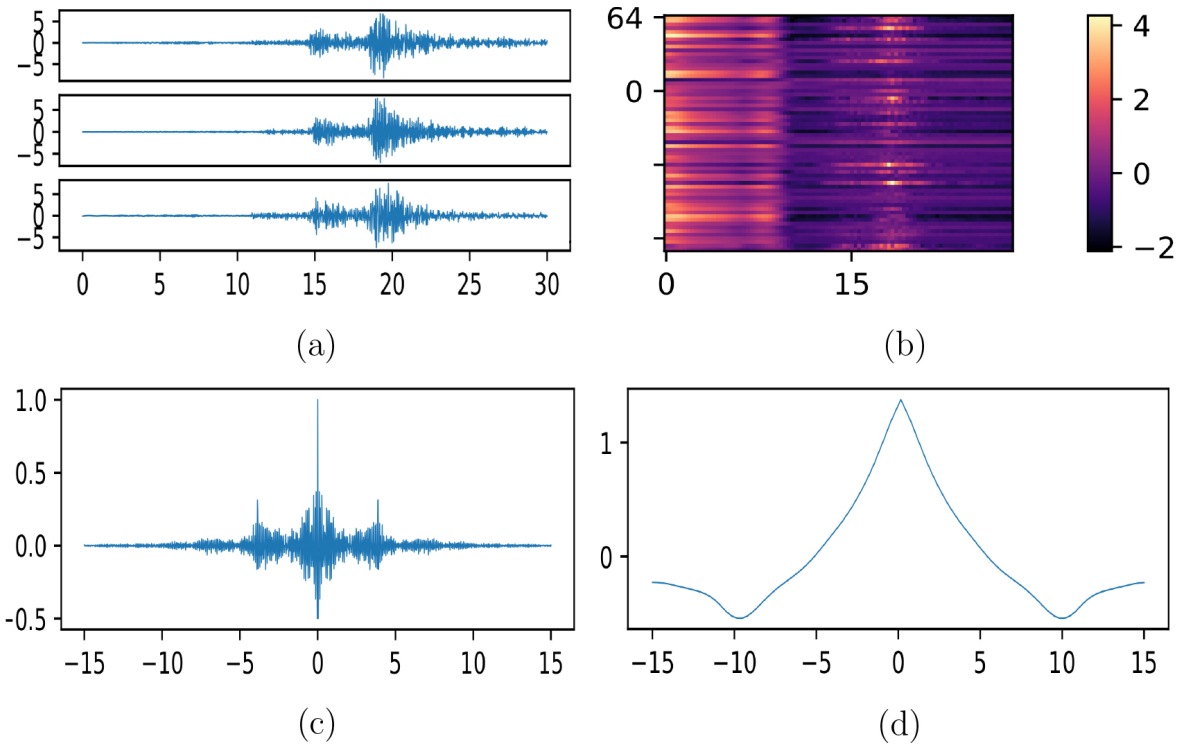


Figure A.5. Earthquake sample 1 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

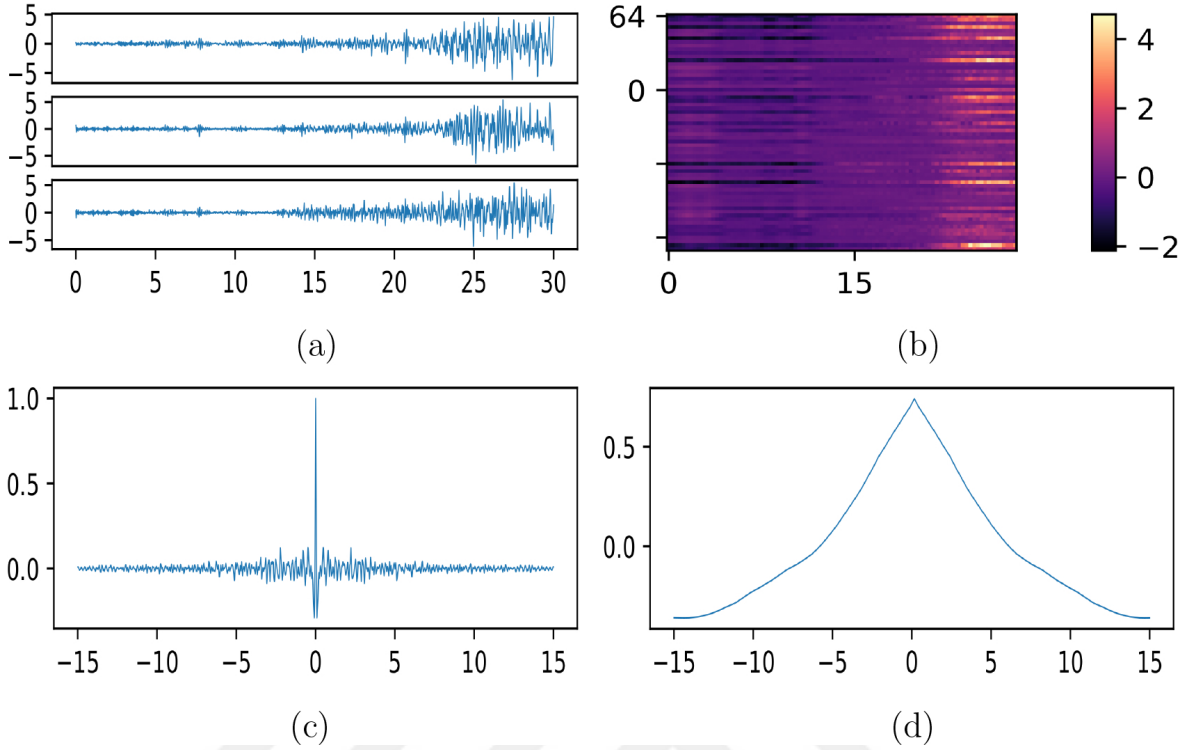


Figure A.6. Earthquake sample 2 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

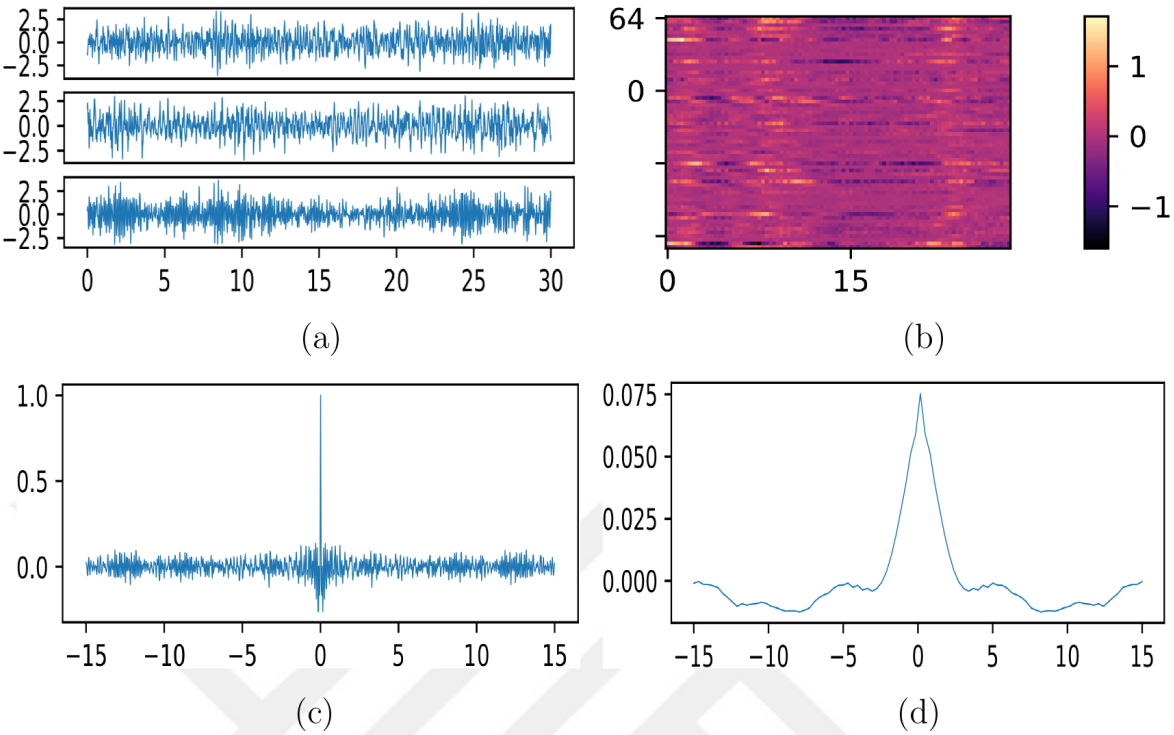


Figure A.7. Noise sample 1 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

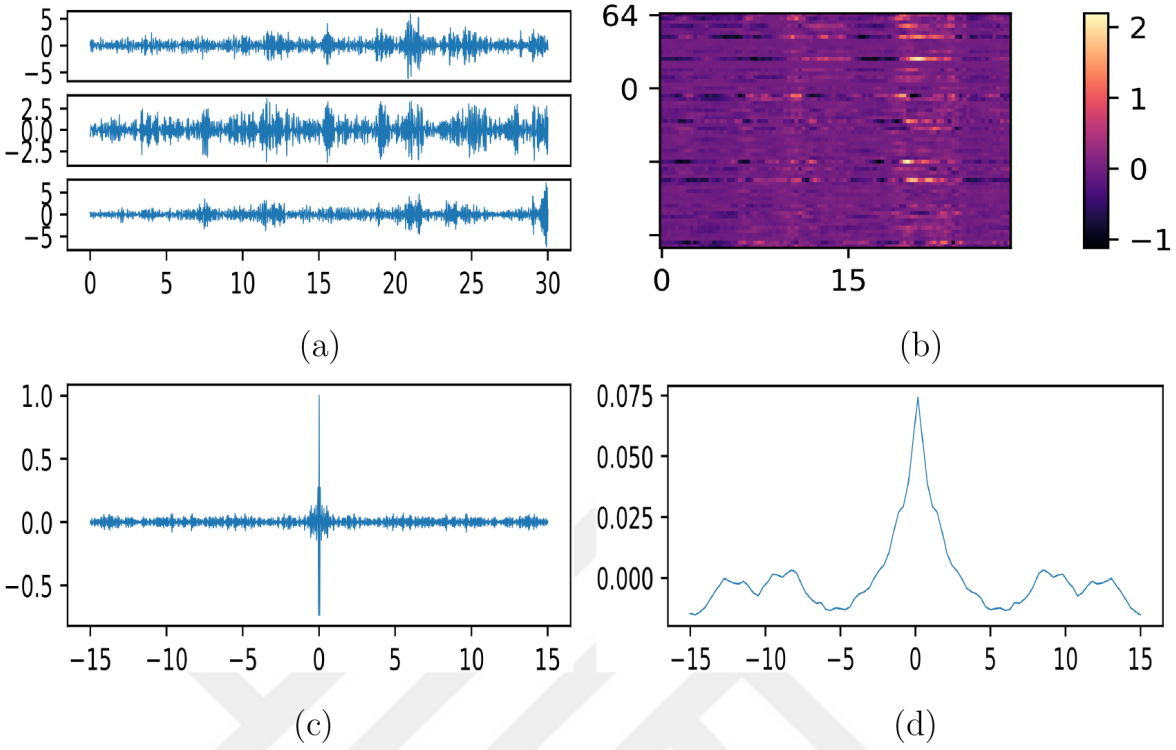


Figure A.8. Noise sample 2 (a) three-channel waveform, (b) latent space representation, (c) waveform Autocovariance, (d) latent space Autocovariance.

A.3. Representation Cross-covariances Method

There, we have the samples obtained using an ensemble of CNN Autoencoders and projectors for each.

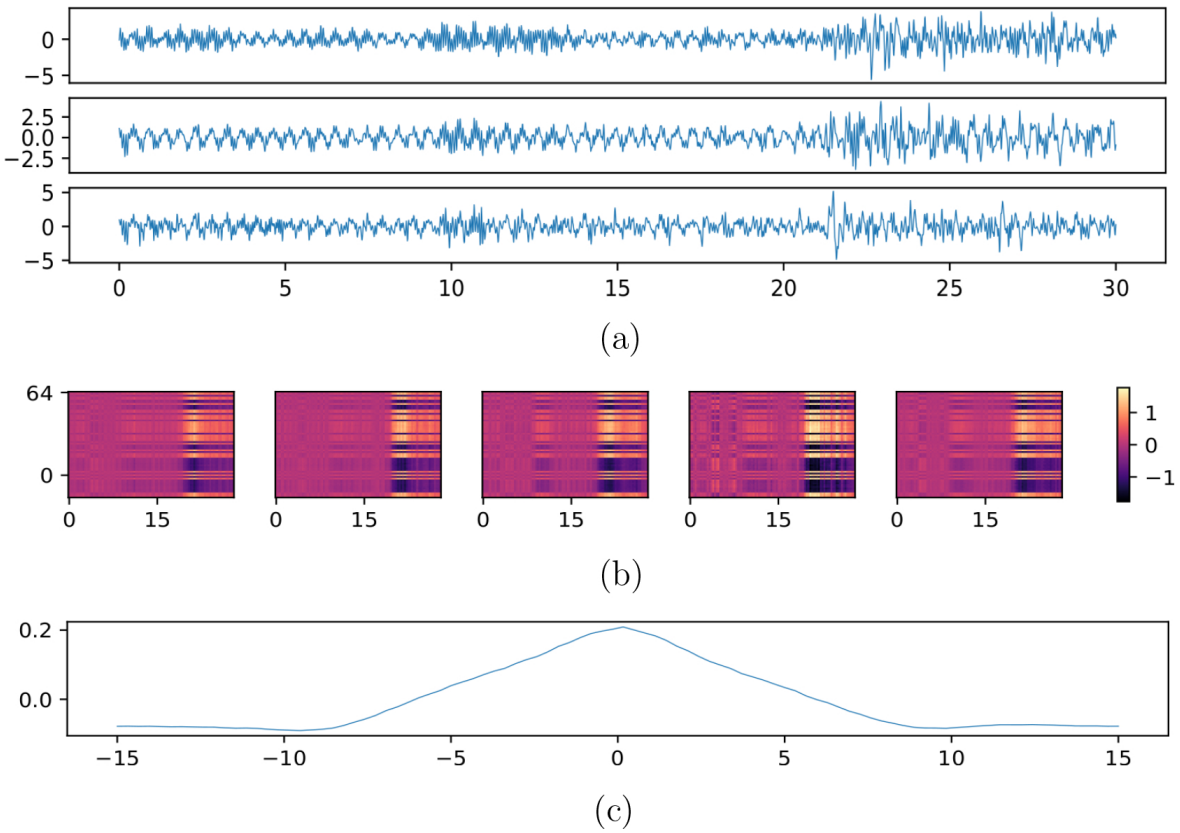


Figure A.9. Earthquake sample 1 (a) three-channel waveform, (b) latent space representations for ensemble members, (c) mean latent space Cross-covariance.

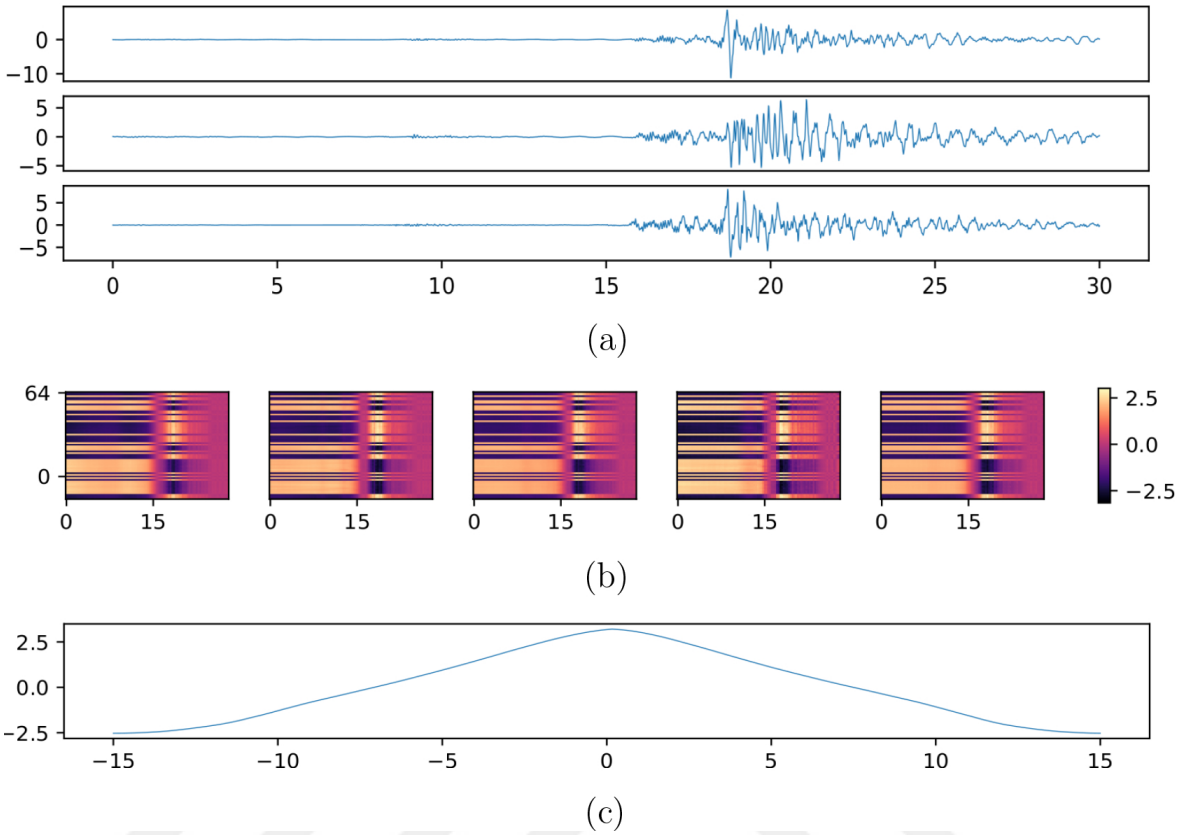


Figure A.10. Earthquake sample 2 (a) three-channel waveform, (b) latent space representations for ensemble members, (c) mean latent space Cross-covariance.

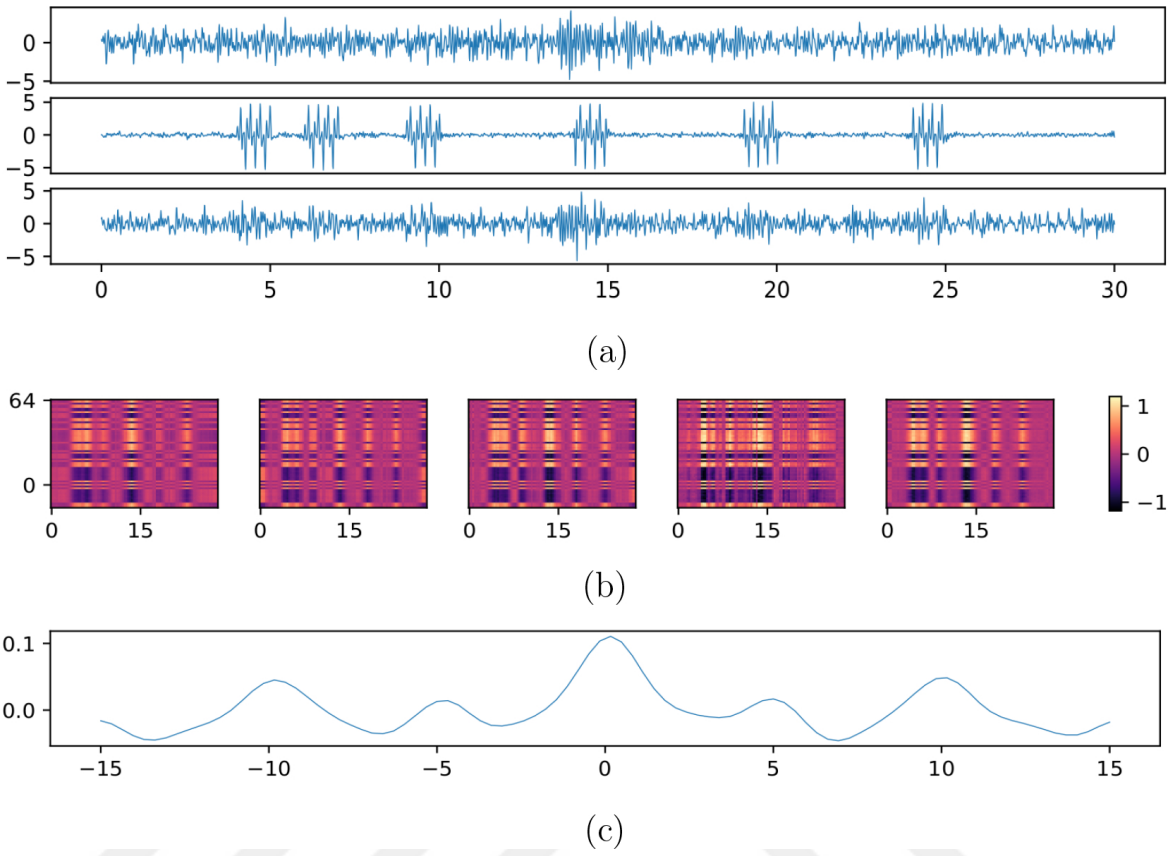
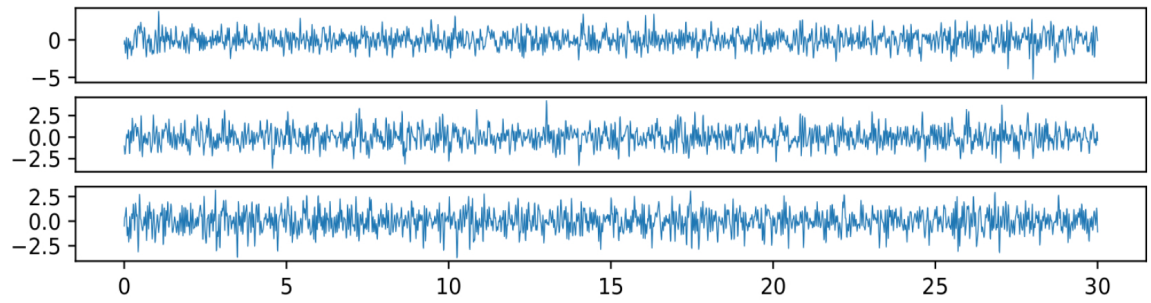
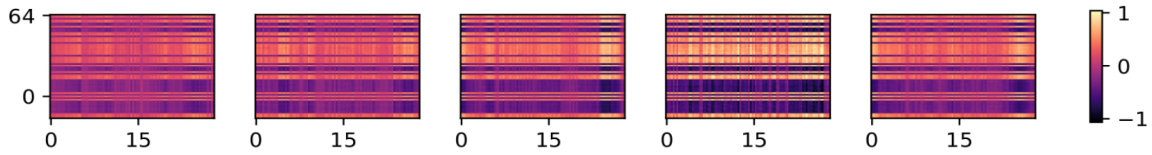


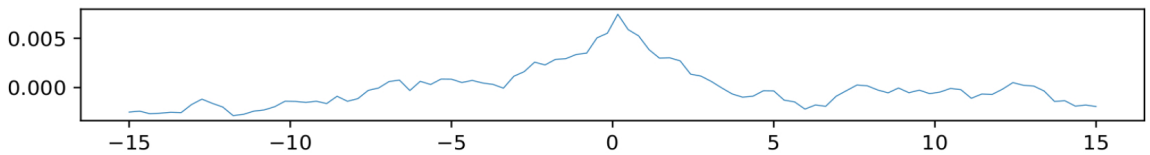
Figure A.11. Noise sample 1 (a) three-channel waveform, (b) latent space representations for ensemble members, (c) mean latent space Cross-covariance.



(a)



(b)



(c)

Figure A.12. Noise sample 2 (a) three-channel waveform, (b) latent space representations for ensemble members, (c) mean latent space Cross-covariance.

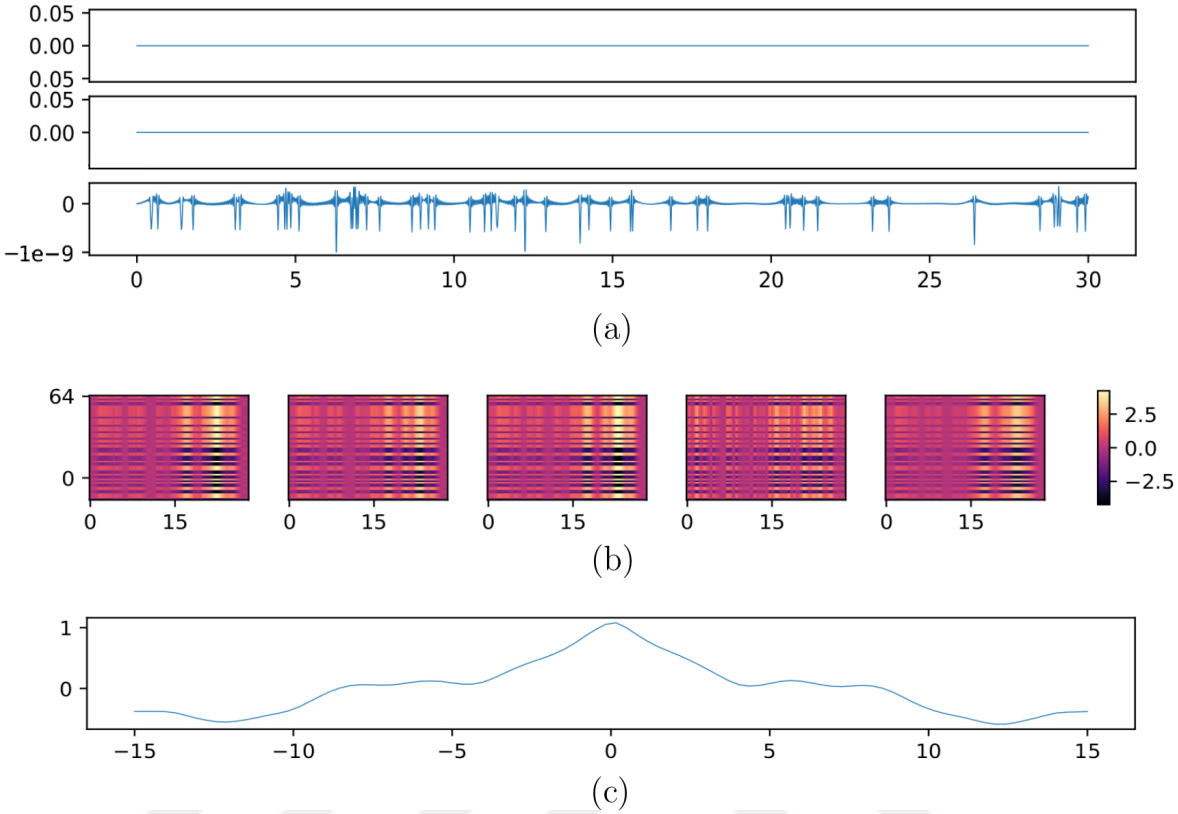


Figure A.13. Noise sample 3 (a) three-channel waveform, (b) latent space representations for ensemble members, (c) mean latent space Cross-covariance.

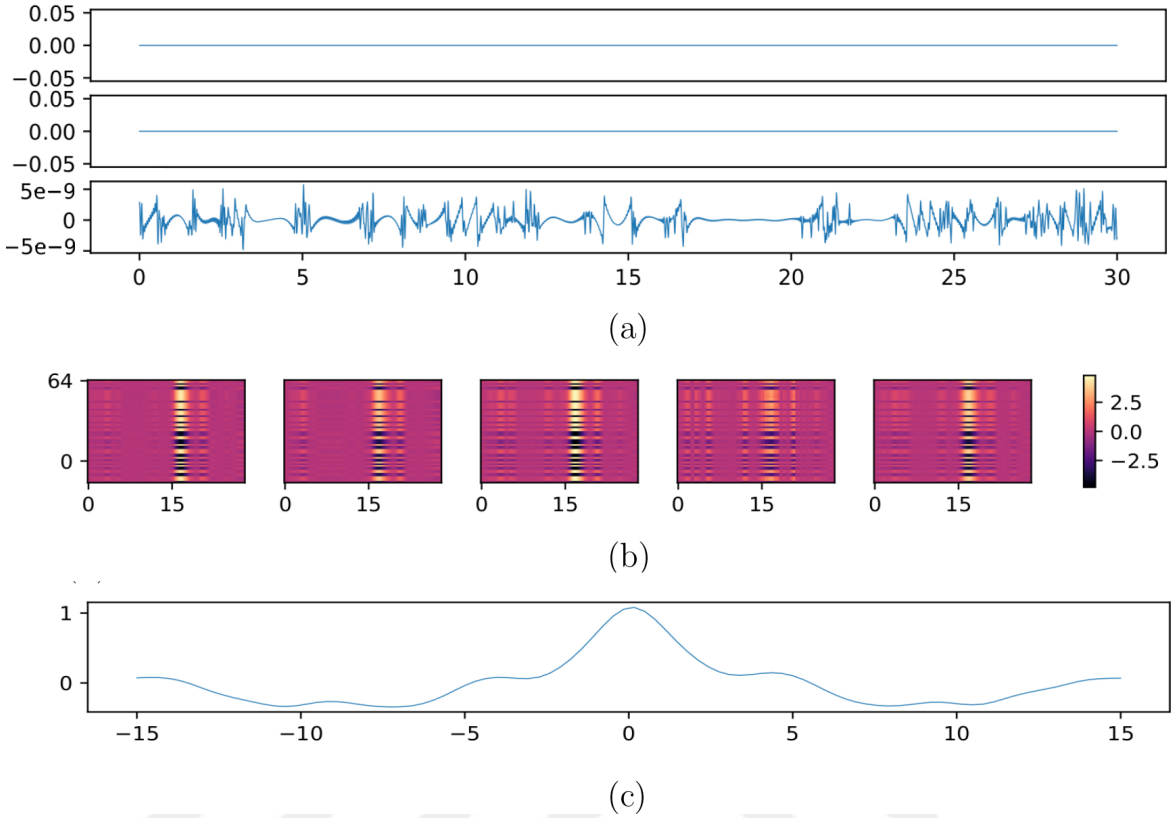


Figure A.14. Noise sample 4 (a) three-channel waveform, (b) latent space representations for ensemble members, (c) mean latent space Cross-covariance.

APPENDIX B: SAMPLES FOR MULTI STATION METHODS

This appendix includes visualizations of the latent space representations of the waveform samples and model input and output for the GNN Autoencoder model.

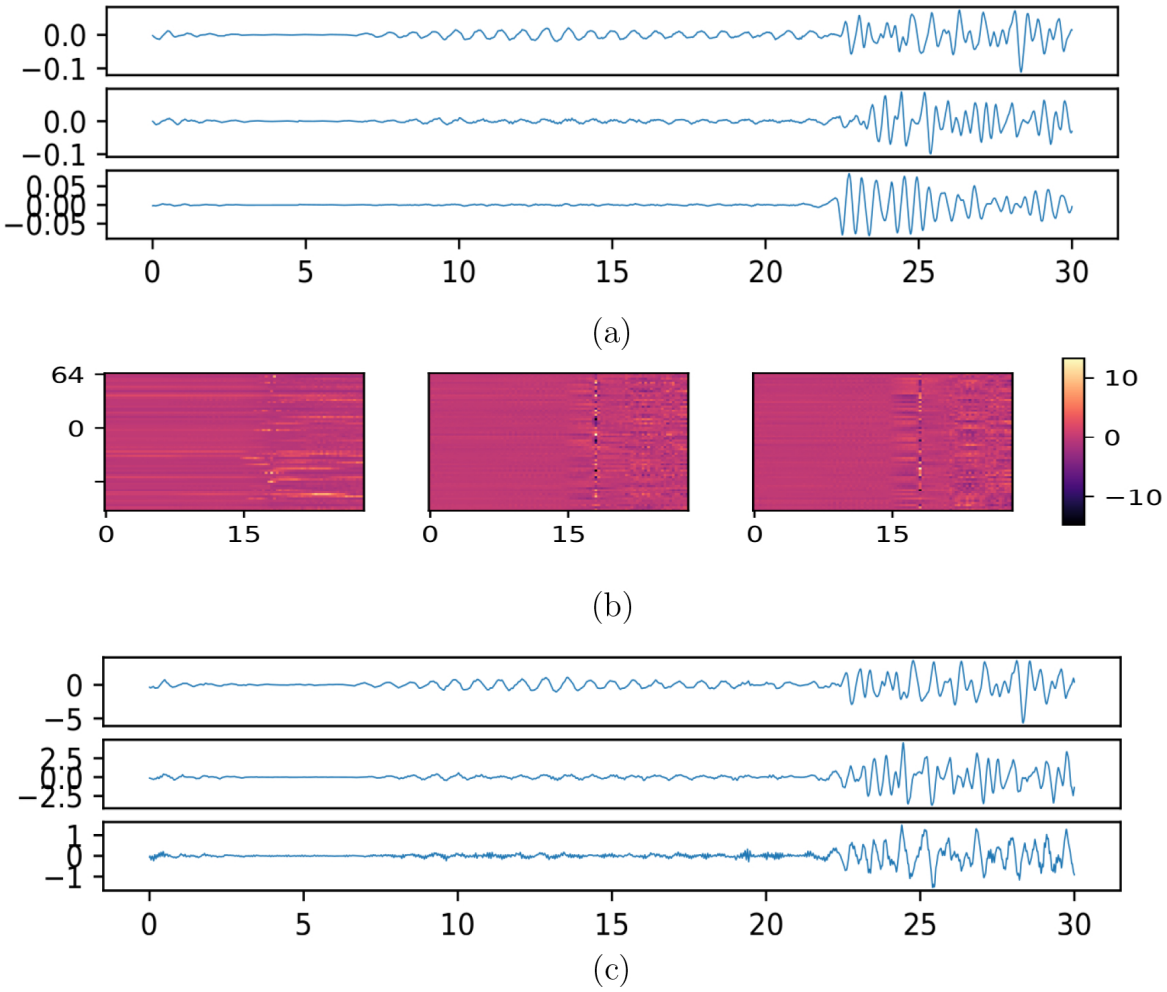


Figure B.1. Earthquake sample 1 (a) three-channel waveform, (b) latent space representations for CNN Autoencoder, first and second attention layers from left to right, (c) three-channel waveform reconstruction.

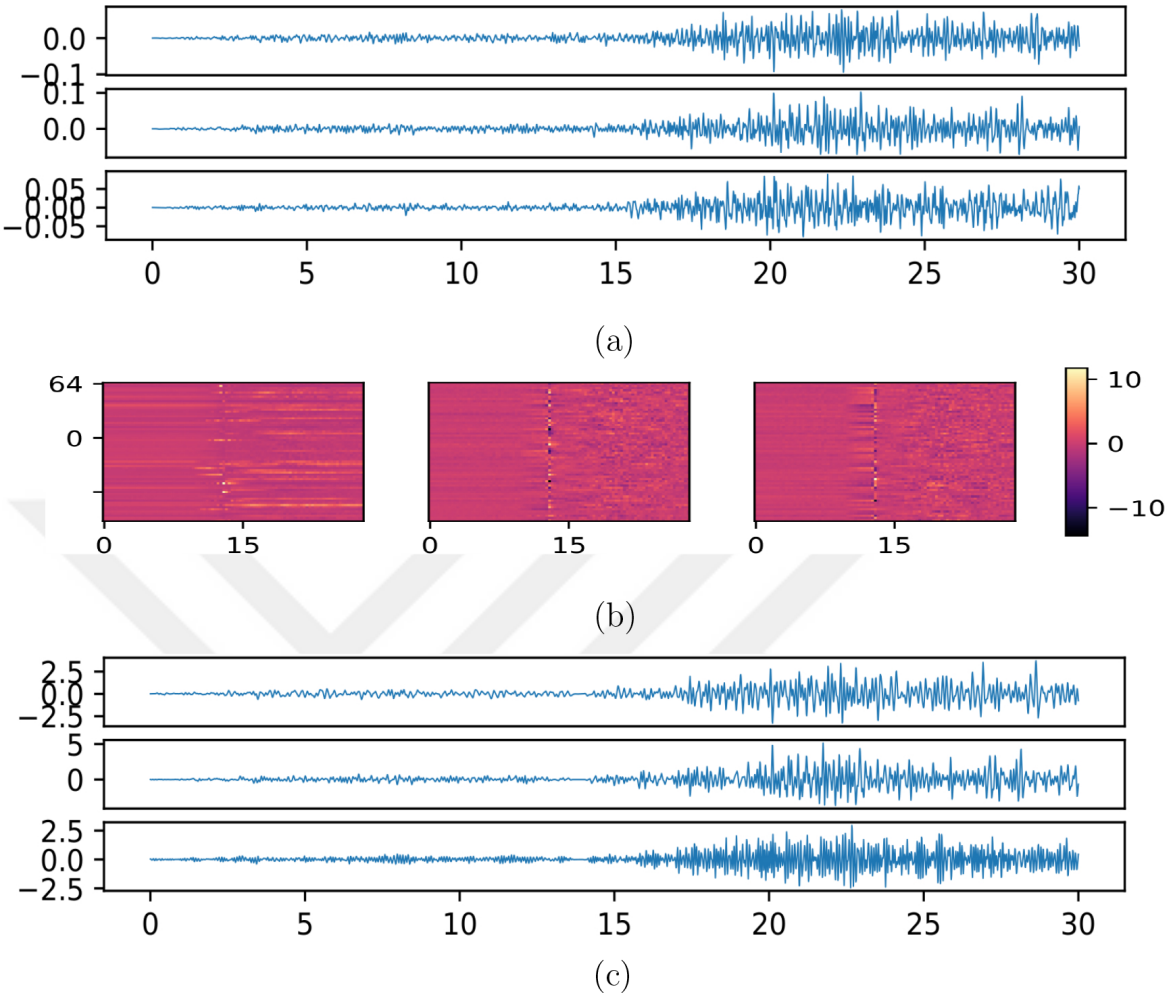


Figure B.2. Earthquake sample 2 (a) three-channel waveform, (b) latent space representations for CNN Autoencoder, first and second attention layers from left to right, (c) three-channel waveform reconstruction.

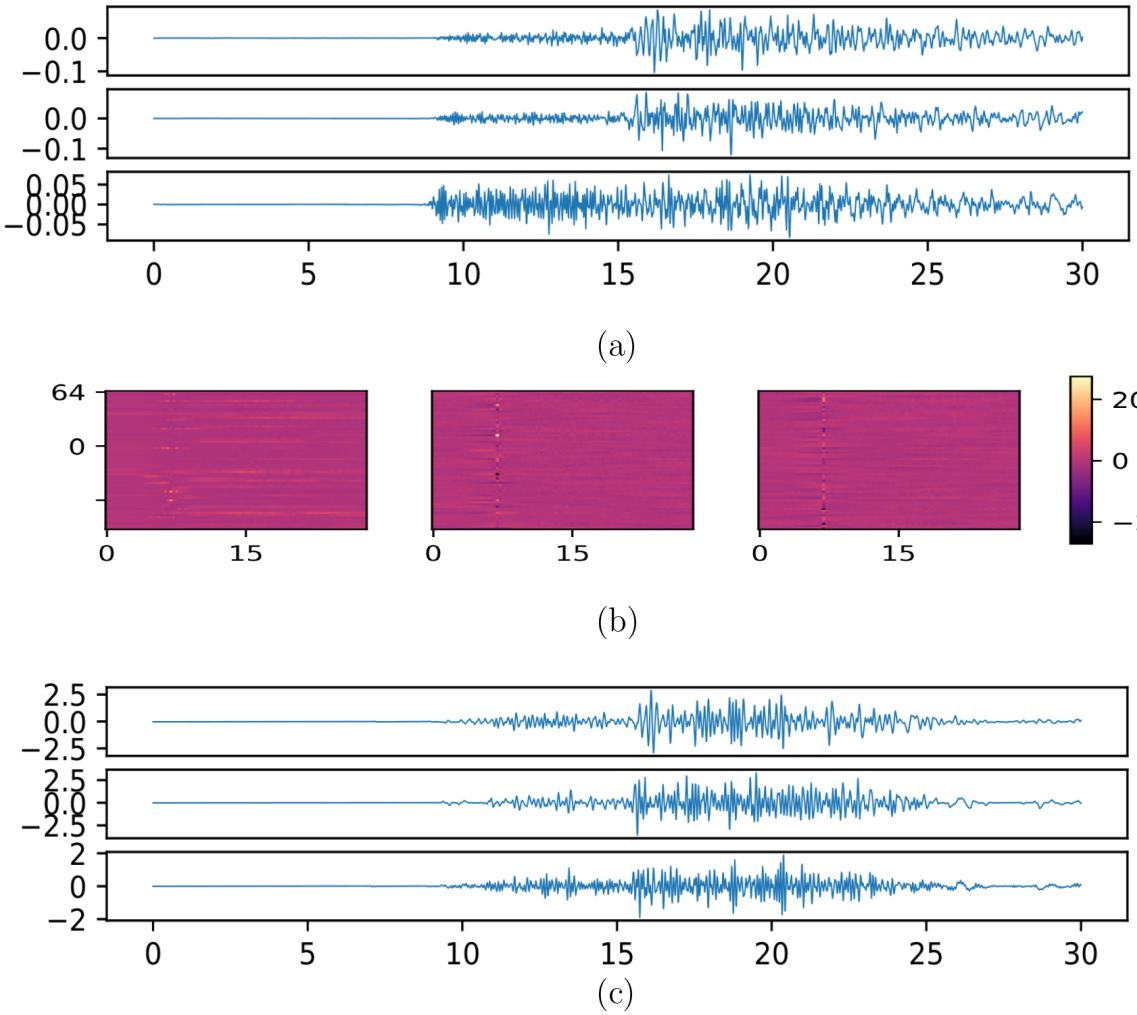


Figure B.3. GNN Autoencoder earthquake sample 3 (at masked mode) (a) three-channel waveform, (b) latent space representations for CNN Autoencoder, first and second attention layers from left to right, (c) three-channel waveform reconstruction.

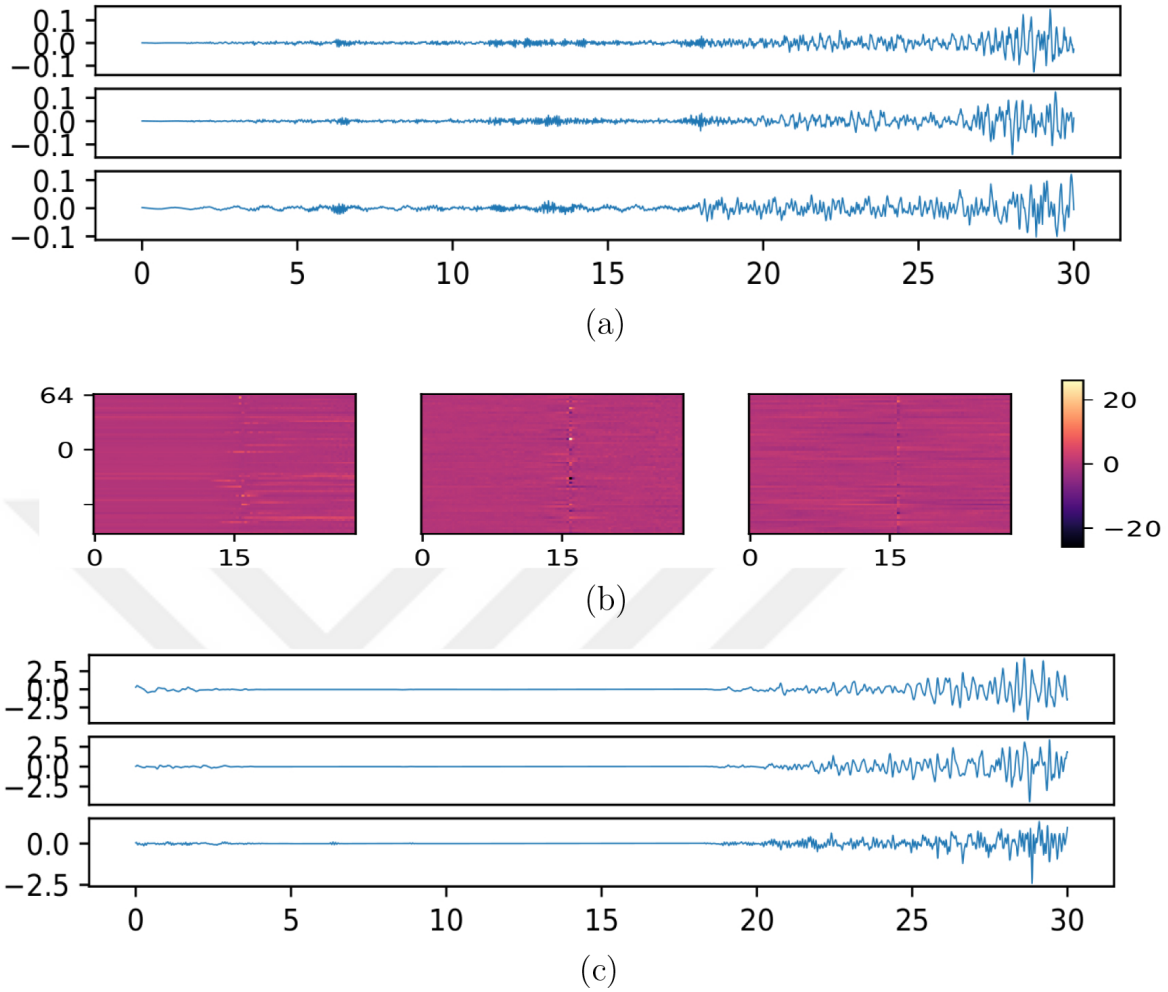


Figure B.4. GNN Autoencoder earthquake sample 4 (at masked mode) (a) three-channel waveform, (b) latent space representations for CNN Autoencoder, first and second attention layers from left to right, (c) three-channel waveform reconstruction.

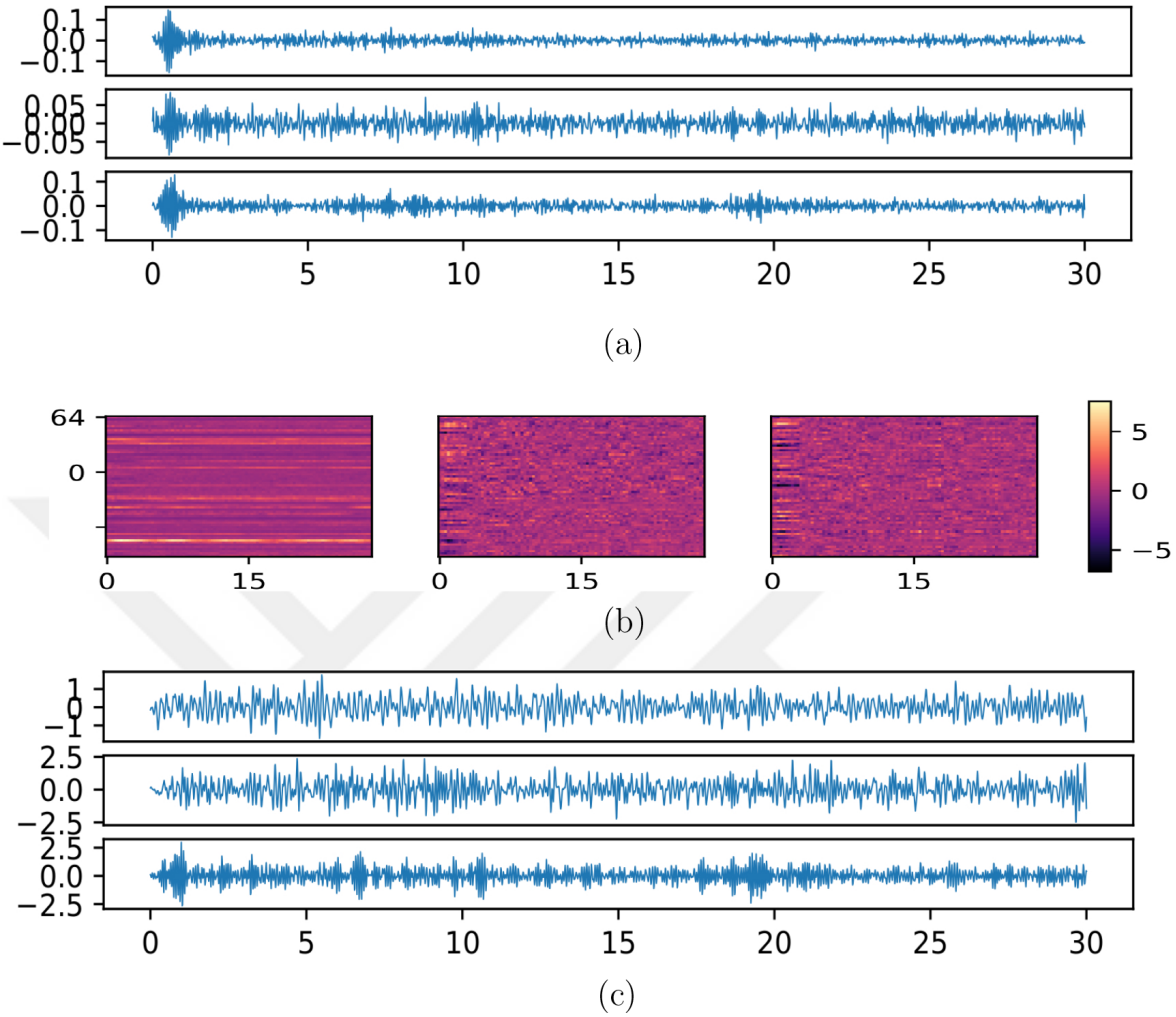


Figure B.5. GNN Autoencoder noise sample 1 (a) three-channel waveform, (b) latent space representations for CNN Autoencoder, first and second attention layers from left to right, (c) three-channel waveform reconstruction.

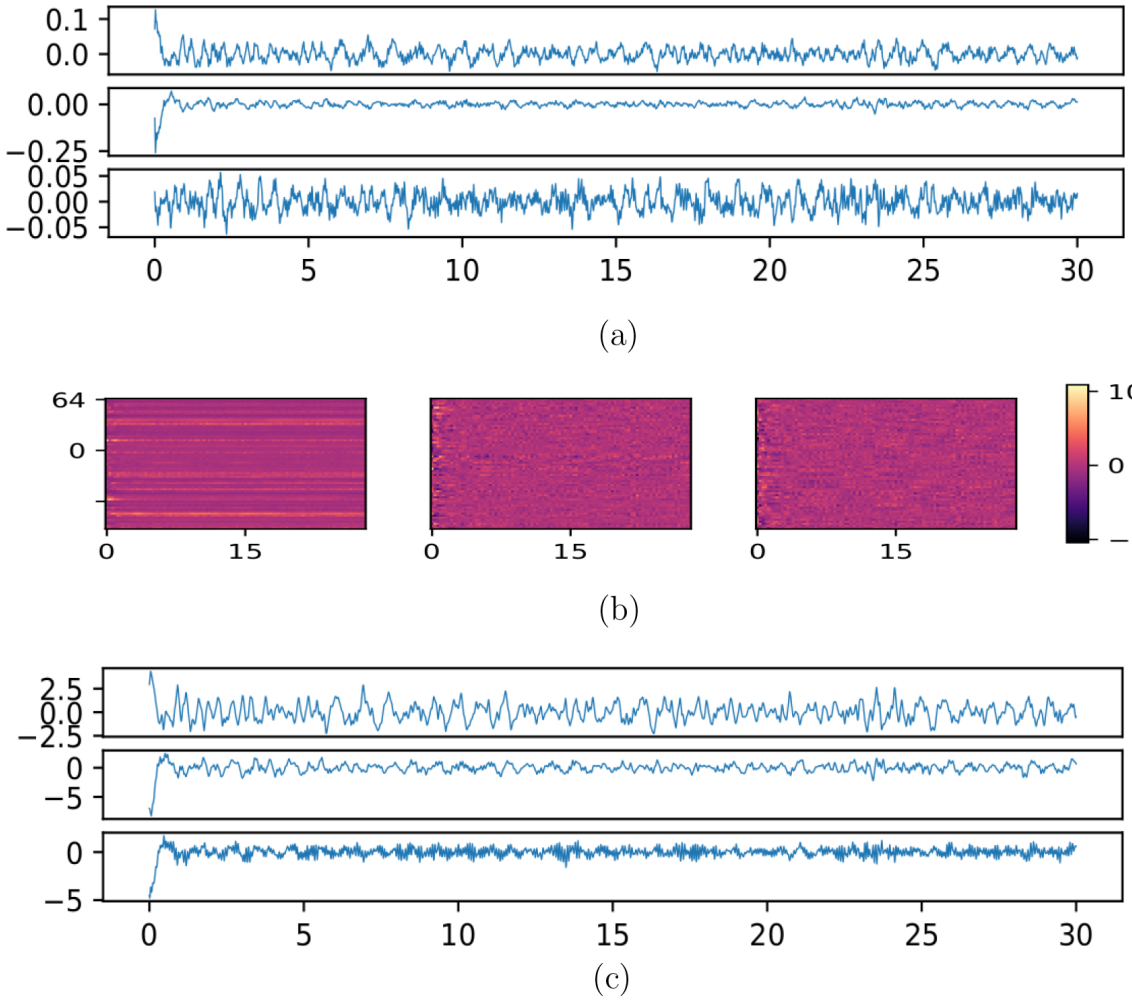


Figure B.6. GNN Autoencoder noise sample 2 (a) three-channel waveform, (b) latent space representations for CNN Autoencoder, first and second attention layers from left to right, (c) three-channel waveform reconstruction.

APPENDIX C: ROC-AUC AND VALIDATION LOSS RELATION

This appendix involves the ROC-AUC score and validation loss relation for single-station methods.

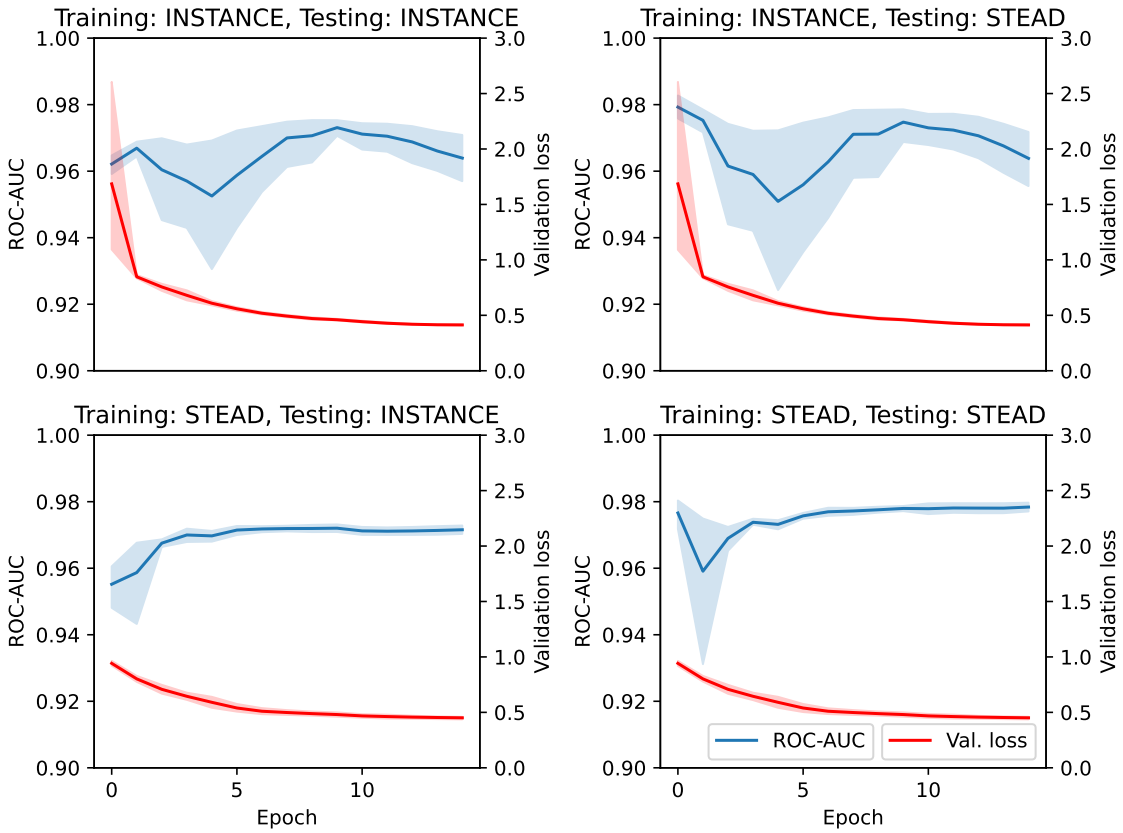


Figure C.1. Autocovariance ROC-AUC and validation loss vs. training duration.

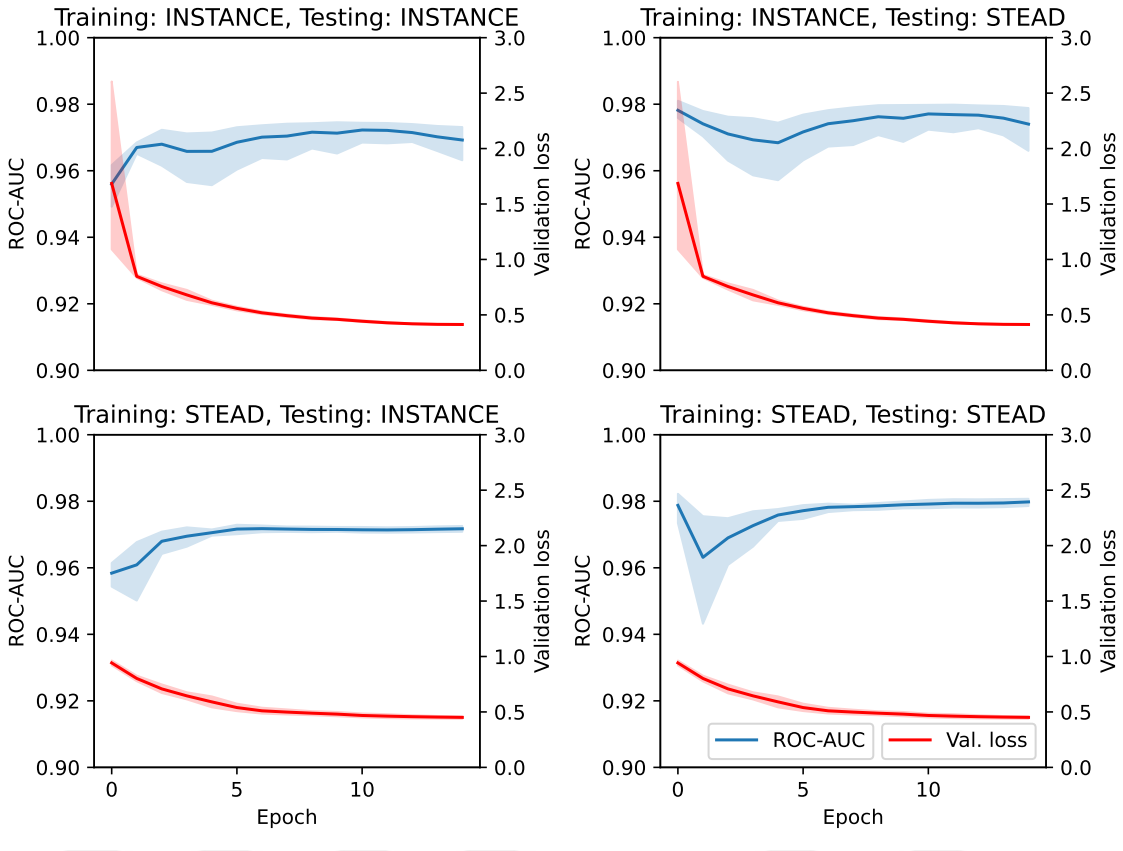


Figure C.2. Augmentation Cross-covariances ROC-AUC score and validation loss vs. training duration.

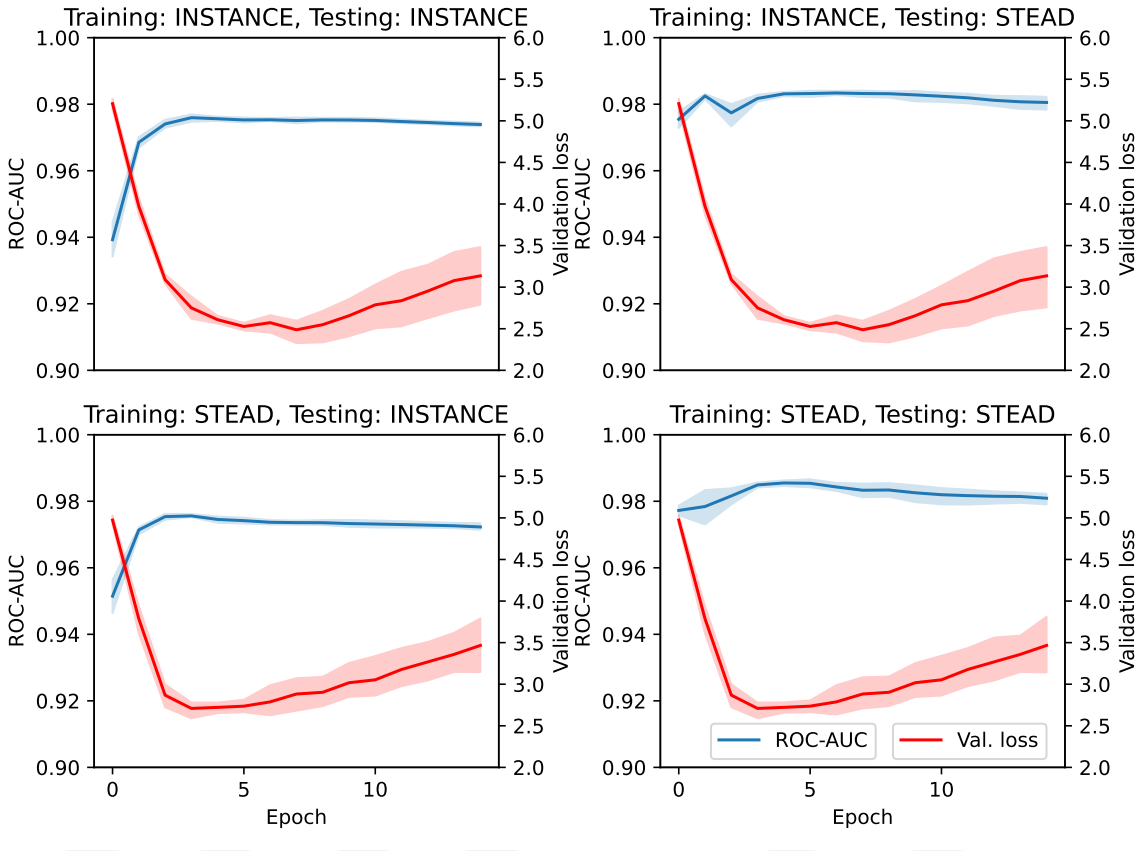


Figure C.3. Representation Cross-covariances ROC-AUC score and validation loss vs. training duration.

APPENDIX D: STA/LTA PARAMETER ADJUSTMENT

This appendix includes the STA/LTA methods searched parameters and corresponding performances.

Table D.1. STA/LTA adjustment parameters and results.

Short term window	Long term window	INSTANCE AUC	STEAD AUC
1.000	2.000	0.945 ± 0.001	0.978 ± 0.001
1.000	4.000	0.953 ± 0.001	0.983 ± 0.001
1.500	3.000	0.957 ± 0.001	0.985 ± 0.001
1.500	6.000	0.957 ± 0.001	0.982 ± 0.001
2.000	4.000	0.962 ± 0.001	0.986 ± 0.001
2.000	6.000	0.965 ± 0.001	0.976 ± 0.001
2.000	8.000	0.961 ± 0.001	0.880 ± 0.001
2.500	4.000	0.963 ± 0.001	0.982 ± 0.001
2.500	5.000	0.966 ± 0.001	0.985 ± 0.001
2.500	7.500	0.966 ± 0.001	0.936 ± 0.001
3.000	4.500	0.965 ± 0.001	0.982 ± 0.001
3.000	6.000	0.968 ± 0.001	0.987 ± 0.001
3.000	7.500	0.968 ± 0.001	0.956 ± 0.001
3.500	5.000	0.966 ± 0.001	0.983 ± 0.001
3.500	6.500	0.969 ± 0.001	0.987 ± 0.001
3.500	8.000	0.968 ± 0.001	0.954 ± 0.001
4.000	5.000	0.966 ± 0.001	0.980 ± 0.001
4.000	6.500	0.969 ± 0.001	0.986 ± 0.001
4.000	8.000	0.969 ± 0.001	0.974 ± 0.001
4.500	6.000	0.968 ± 0.001	0.983 ± 0.001
4.500	7.000	0.970 ± 0.001	0.986 ± 0.001
4.500	8.000	0.970 ± 0.001	0.985 ± 0.001