Politecnico di Torino

Master's degree in Computer Engineering

Master Degree Thesis

SYNTHETIC SEISMIC SIGNALS BY GENERATIVE ADVERSARIAL NETWORKS

Supervisors :
Prof. G. C. Marano
...

Co-supervisors:
Ph.D. Student Marco Martino Rosso
...

Candidate:
Mazin Ali Ibrahim Onsa

Student's ID:
S252865

December 2024
A.Y. 2023/2024

Abstract

Detecting earthquake events in seismic time series can be a daunting task. Traditional human-based visual detection has been the established benchmark, but it demands substantial manual effort and doesn't efficiently adapt to large datasets. Over the past few years, machine learning detection techniques have been adopted to enhance accuracy and efficiency. However, the effectiveness of these approaches hinges on the availability of a substantial volume of high-quality annotated training data which is often scarce in many situations. This thesis aims to address this issue by answering the key question: Can limited authentic labeled seismic waveforms be used to generate realistic synthetic ones? To tackle this issue, a generative adversarial network (GAN), a powerful machine learning model known for generating high-quality synthetic data in various domains, is employed. Once trained on data provided by the ITalian ACcelerometric Archive of waveforms (ITACA), the GAN model can create realistic seismic waveforms for both noise and earthquake events. This study demonstrates that data augmentation using GAN-generated synthetic waveforms can enhance earthquake detection algorithms in situations where only a limited number of labeled training data are available

"In God we trust. All others must bring data."
W. Edwards Deming

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Context

Detecting earthquake events in seismic time series can be a daunting task. Traditional human-based visual detection has been the established benchmark, but it demands substantial manual effort and doesn't efficiently adapt to large datasets. Over the past few years, machine learning detection techniques have been adopted to enhance accuracy and efficiency. However, the effectiveness of these approaches hinges on the availability of a substantial volume of high-quality annotated training data, which is often scarce in many situations.

Traditional methods for generating seismic signals have been limited in their ability to capture the complexity and variability of real-world seismic data affected by natural factors. Moreover, stations near the epicenter of an earthquake (typically from a few hundred meters up to 20 kilometers) experience technical limitations or challenges that can affect their ability to immediately transmit data or information. Destructive S-waves can cause significant infrastructure damage to power lines, communication equipment, and data transmission systems. When these systems are disrupted, it can be difficult for seismic stations to send data to a central location for analysis or reporting. Additionally, if the station avoids damage, the instruments that measure ground motion may saturate if the earthquake is extremely powerful and the ground motion is very intense. When an instrument saturates, it essentially maxes out its measurement capability, making it challenging to accurately record the full extent of the earthquake's motion.

Recording seismic waves close to the epicenter of an earthquake is crucial for assessing ground shaking intensity, improving Earthquake Early Warning (EEW) systems, and calibrating seismic networks. However, the lack of high-quality data from these regions poses significant challenges for seismic analysis and modeling.

1.2 Objectives

Inspired by the notable achievements and applications of Generative Adversarial Networks (GANs) in many fields like computer vision, the primary objective of this research is to develop a novel approach for synthetic seismic data augmentation for stations close to the epicentral distance. This approach aims to improve the robustness and generalization of seismic detection and classification performance.

By generating realistic synthetic seismic signals using GANs, we intend to address the scarcity of high-quality labeled data, particularly for near-epicenter stations. This synthetic data can augment existing datasets, providing a richer and more diverse set of training examples for machine learning models used in seismic event detection and classification.

1.3 Motivation

The motivation for this research stems from the identified gaps and limitations in the current state of seismic data acquisition and analysis:

1. **Data Scarcity Near Epicenters:** Due to technical limitations and potential damage to instruments, there is a significant lack of high-quality seismic data from stations near earthquake epicenters.
2. **Need for Improved Detection Algorithms:** Machine learning models for seismic event detection require large amounts of labeled data to perform effectively, particularly in capturing the complex patterns associated with near-epicenter seismic waves.
3. **Advancements in Synthetic Data Generation:** Recent developments in GANs offer the potential to generate high-quality synthetic data that can fill the gaps in existing datasets.
4. **Broader Context:** Enhancing the ability to detect and analyze seismic events has significant societal benefits, including improved earthquake early warning systems and better-informed infrastructure design.

By aligning advancements in synthetic seismic signal generation with larger goals within the scientific community and industry, this research aims to contribute to improved seismic hazard assessment and mitigation strategies.

1.4 Research Questions

The key research questions addressed in this thesis are:

1. **Can limited authentic seismic waveforms be used to generate realistic synthetic ones using GANs?**
2. **What are the challenges and limitations in applying GANs to generate synthetic seismic signals, particularly for near-epicenter stations?**

1.5 Outline of the Thesis

The structure of the thesis is as follows:

- **Chapter 2: Background and Terminology** — This chapter provides an overview of seismic signal analysis, generative modeling, and the theoretical foundations of GANs. It also reviews related work in the application of GANs to seismic data.

- **Chapter 3: Theoretical Framework** — This chapter delves into the deep learning fundamentals and the mathematical underpinnings of GANs, including their variants and challenges. It also discusses the characteristics of seismic data near the epicenter and the performance metrics for evaluating synthetic seismic waveforms.
- **Chapter 4: Methodology** — This chapter outlines the approach used in the research, including dataset preparation, GAN model design, training strategy, and evaluation methods.
- **Chapter 5: Results** — This chapter presents the results of the experiments, including training details, generated signal quality, and statistical metrics comparing real and generated data.
- **Chapter 6: Discussion** — This chapter discusses the findings, addressing the challenges encountered, the effectiveness of the methods used, and the implications of the results.
- **Chapter 7: Conclusion** — This chapter summarizes the research, highlighting the contributions and suggesting directions for future work.

By exploring the use of GANs for generating synthetic seismic signals, this research aims to contribute to the field of seismic data analysis and machine learning, offering new possibilities for data augmentation and improved seismic event detection.

Chapter 2

Background and Terminology

This chapter establishes the foundational knowledge for understanding the use of Generative Adversarial Networks (GANs) in seismic data synthesis. We begin by discussing Seismic Signal Analysis, covering types of seismic waves—body waves (P-waves and S-waves) and surface waves—and essential terms like the hypocenter and epicenter. This groundwork is crucial for interpreting seismic data, especially near-epicenter events.

We then examine the Properties of Seismic Signals, focusing on amplitude measures such as Local Magnitude (M_L) and Moment Magnitude (M_w), frequency content, duration, and attenuation. This section highlights how seismic events are quantified and the challenges in measuring high-magnitude earthquakes.

Next, we introduce Generative Modeling, explaining its role in creating new data points by learning underlying data distributions. We differentiate between explicit density models, implicit density models (like GANs), and autoregressive models, setting the stage for their application to seismic data.

We discuss the evolution From Shallow to Deep Generative Models, emphasizing the limitations of traditional models in capturing complex data patterns. This leads to the emergence of Deep Generative Models, such as Variational Autoencoders (VAEs) and GANs, which utilize deep neural networks for more effective data representation.

In The Rise of GANs, we delve into their adversarial training process involving a generator and a discriminator. We highlight the advantages of GANs over traditional methods, particularly their ability to model complex, high-dimensional data without explicit density functions.

Finally, we explore GANs in Seismology, examining how they address challenges like modeling intricate subsurface conditions and generating realistic seismic waveforms. We review key studies and frameworks, such as SeismoGen, that have successfully applied GANs for seismic data synthesis and augmentation.

This chapter provides the essential background and terminology, laying the groundwork for the methodologies and experiments detailed in the subsequent chapters.

2.1 Seismic Signal Analysis

Seismic signals are vibrations in the Earth caused by natural phenomena like earthquakes, volcanic activity, landslides, or human-induced events such as explosions or heavy machinery operations. These signals provide critical information about the Earth's interior and processes, playing a vital role in geophysics, hazard assessment, and engineering.

Seismic signals primarily consist of two types of waves:

1. **Body Waves**: Travel through the Earth's interior and are further classified into:
   - **P-waves (Primary waves)**: Longitudinal waves that compress and expand the medium. They are the fastest seismic waves and travel through both solid and liquid layers of the Earth.
   - **S-waves (Secondary waves)**: Transverse waves that move perpendicular to their direction of propagation. They are slower than P-waves and travel only through solids. Their velocities can be expressed as:

     $$v_P = \sqrt{\frac{K + \frac{4}{3}\mu}{\rho}}, \quad v_S = \sqrt{\frac{\mu}{\rho}}$$

     where v_P and v_S are the velocities of P and S waves, K is the bulk modulus, μ is the shear modulus, and ρ is the density [1].
2. **Surface Waves**: Travel along the Earth's surface and are slower than body waves but often have larger amplitudes, causing significant damage during earthquakes.

Terms related to earthquake locations:

Hypocenter: Also known as the focus of an earthquake, it is the exact point within the Earth where the rupture begins. This is the source of seismic energy, typically located along a fault plane.
Epicenter: The point on the Earth's surface directly above the hypocenter. It is commonly referenced in seismic studies as it is easier to locate from surface observations and typically associated with the most intense shaking.

2.1.1 Properties of Seismic Signals

Seismic signals are characterized by several key properties:

- **Amplitude**: Indicates the energy of the seismic event. Larger amplitudes correspond to more energetic events. For this research, the analysis is constrained to two commonly used amplitude measures: Local Magnitude (M_L) and Moment Magnitude (M_w).[1]

[1]This constraint is based on the specifications provided by the data provider.

1. **Local Magnitude (M_L)**:
   - M_L, commonly known as the *Richter scale*, is calculated using the maximum amplitude of ground motion (A) recorded by a seismometer at a specific distance (d). The formula for M_L is:

$$M_L = \log_{10}(A) - \log_{10}(A_0(d)),$$

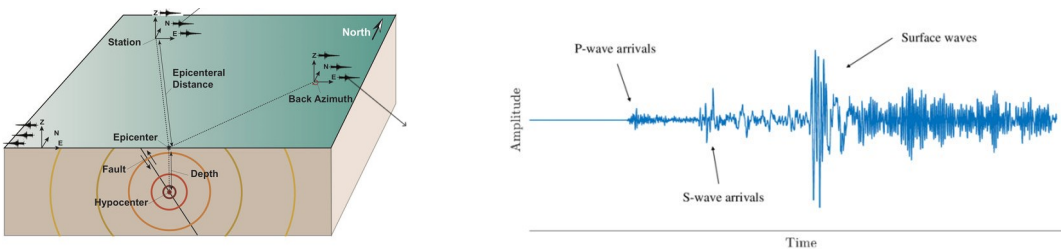
     where $A_0(d)$ is a reference amplitude as a function of distance, determined empirically [2].
   - M_L is suitable for small to moderate earthquakes but tends to saturate for larger events, making it less reliable at high magnitudes [1].
2. **Moment Magnitude (M_w)**:
   - M_w is calculated based on the seismic moment (M_0), which is a measure of the total energy released by an earthquake. The formula for M_w is:

$$M_w = \frac{2}{3}\log_{10}(M_0) - 10.7,$$

     where M_0 is expressed in dyne-centimeters [1].
   - Unlike M_L, M_w does not saturate for large earthquakes, making it the preferred measure to be used in this research. [2].

- **Frequency**: Seismic waves span a broad frequency range, from below 1 Hz for tectonic events to over 100 Hz for explosions.
- **Duration**: Varies depending on the source and distance from the recording station.
- **Attenuation**: Seismic wave amplitudes decrease with distance due to energy loss and geometric spreading.

Below is an example of seismic waveforms showing P-waves, S-waves, and surface waves recorded from a seismic event:



(a) Seismic Fault Terms.

(b) P-waves, S-waves, and Surface Waves.

Figure 2.1: Illustration of basic seismic terms.

2.1.2 Seismic Signal Processing Techniques

Filtering

Filtering is used to isolate specific frequency components of seismic signals, often to remove unwanted noise or enhance features of interest. Commonly used filters include:

- *Low-pass filters*: Remove high-frequency noise, such as environmental disturbances or instrumental noise.
- *High-pass filters*: Eliminate low-frequency trends, such as baseline drift.
- *Band-pass filters*: Extract frequency bands of interest, such as those corresponding to earthquake signals or specific geological features.

The design of filters depends on the application, with parameters like cutoff frequency and filter order requiring careful selection [1].

Feature Extraction

Feature extraction involves identifying and quantifying characteristics of seismic signals that are relevant for analysis. Key features include:

- *Amplitude*: Provides information on the energy and intensity of seismic events.
- *Frequency content*: Reveals source characteristics and helps distinguish between different types of seismic waves.
- *Waveform attributes*: Includes parameters such as phase arrival times and durations, which are critical for event detection and classification.

2.2 Generative Modeling

The area of research in statistical Learning which aims to provide machines with the essential capacity to synthesize new entities is known as *generative modeling*. Generative models are capable of generating new data points by understanding the underlying distribution of data points in the dataset. Because of its capability of generating new synthetic points starting from real data, it plays a key role in many areas of science, finance and industry. Unlike discriminative models, which aim to classify or predict labels given input data, generative models learn to generate new data points that are statistically similar to the dataset they are trained on and performed with an unlabeled dataset as a form of *unsupervised learning*.

Formulation

Generative modeling involves learning a model $\hat{p}_\theta(x)$, parameterized by θ, to approximate the true data distribution $p(x)$, where $x \sim p(x)$. The data space's high-probability regions (blue region) represent real data, with black dots denoting data points. The model $\hat{p}_\theta(x)$ is implicitly defined by transforming samples from a unit Gaussian distribution through the generative

model. The goal is to make $\hat{p}_\theta(x)$ match $p(x)$ by optimizing θ to minimize the ***statistical divergence*** between the two distributions. This enables realistic data generation.

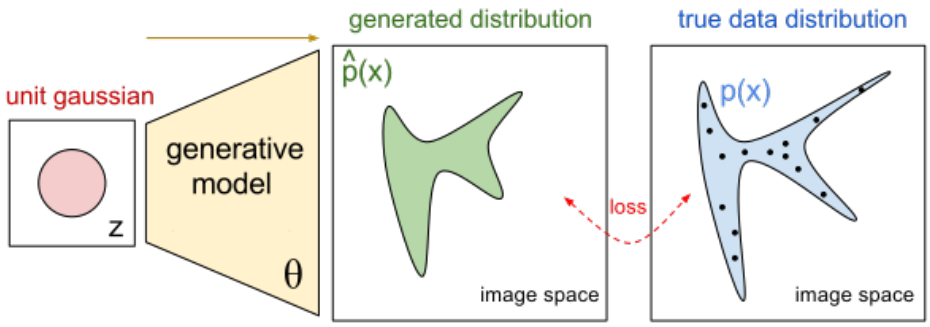


Figure 2.2: Generative Model

We are impressed by $\hat{p}(x)$ if:

1. It can generate examples that appear to have been drawn from $p(x)$.

2. It can generate examples that are suitably different from the observations in X. In other words, the model should not simply reproduce things it has already seen

A generative model that produces samples similar to training data and generalizes well to unseen data is called a *fit model*. *Underfitting* occurs when a model fails to capture the underlying structure of the data, often due to missing parameters, resulting in poor predictive performance. [2]

Overfitting, on the other hand, happens when a model fits the training data too closely, failing to generalize to new data, often due to having too many parameters. [3]

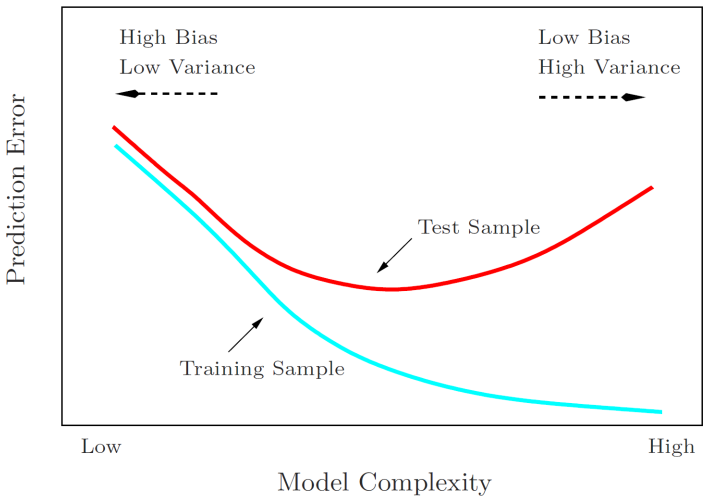


Figure 2.3: Bias-Variance tradeoff for Model selection

[2] This corresponds to the *bias* term dominating the loss function.

[3] This corresponds to the *variance* term dominating the loss function.

A divergent function $D(p||\hat{p}) : S \text{ x } S \rightarrow R$ taking two distributions p and $\hat{p}$ over a space of distributions S as inputs, with the properties:

$$D(p||\hat{p}) \geq 0.0 \tag{2.1}$$

$$D(p||\hat{p}) = 0.0 \Leftrightarrow p = \hat{p} \tag{2.2}$$

Notably, there is no symmetry assumption, so in general $D(p||\hat{p}) \neq D(\hat{p}||p)$. The probabilistic approach to generative modeling frames learning as an optimization problem where the loss corresponds to a given divergence.

$$L(\theta) = \underset{\theta}{\operatorname{argmin}}\, D(p||\hat{p}_\theta(x)) \tag{2.3}$$

2.2.1 Types of Generative Models

Generative models can be broadly categorized into three main types based on their approach to learning and generating data. Each of these categories differs in terms of how the underlying data distribution is modeled and how samples are generated.

Explicit Density Models

Explicit density models explicitly define and optimize a likelihood function for the data. They involve direct parameterization of the probability density function $p(x)$ and rely on maximizing the likelihood of the observed data. Examples include: *Gaussian Mixture Models (GMMs)* and *Variational Autoencoders (VAEs)*

Implicit Density Models

Implicit density models do not explicitly define the data distribution $p(x)$. Instead, they generate samples directly, sidestepping the need for a likelihood function. These models excel in capturing complex data distributions. Examples include:**Generative Adversarial Networks (GANs)** and *Energy-Based Models (EBMs)*

Autoregressive Models

Autoregressive models decompose the joint probability distribution of data into a product of conditional probabilities:

2.2.2 From Shallow to Deep Generative Models

Generative models have evolved significantly, transitioning shallow architectures to deep neural networks. This shift has been driven by the limitations of traditional shallow approaches which rely on **strong assumptions about the data distribution**. These models, such as GMMs and Hidden Markov Models (HMMs), are computationally efficient and require fewer data for training. However, they often fail to capture the complex dependencies inherent in high-dimensional data, as they typically assume conditional independence between variables [3].

Shallow models often treat features as independent entities, which limits their ability to generate coherent structures [4]. Additionally, their reliance on predefined distributions reduces the modeling process to parameter estimation, making them ill-suited for large datasets with complex patterns [5]. This limitation is further exacerbated by the curse of dimensionality, which hampers their performance when dealing with high-dimensional data.

Deep Generative Models (DGMs) emerged as a solution to the limitations of shallow approaches. DGMs leverage deep neural networks to learn hierarchical representations of data. At the core of this advancement is *representation learning*, where high-dimensional data is mapped to a lower-dimensional latent space [6].

In this *latent space*, DGMs learn abstract representations that capture the underlying structure of data. The development of architectures like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) has been instrumental in the success of DGMs. Figure 2.4 highlights the remarkable progress in image synthesis, particularly in generating realistic human faces, since 2014.

Figure 2.4: Progress in facial image generation from 2014 to recent advancements [5]

The versatility of DGMs has enabled their application across diverse domains, including image synthesis, natural language processing, and scientific simulations. For instance:

- **Image Synthesis:** DGMs like GANs can generate photo-realistic images for applications in game design, cinematography, and virtual reality [7].
- **Data Augmentation:** DGMs can create synthetic data to enhance the training of machine learning models, particularly in scenarios with limited labeled data [8].
- **Scientific Simulations:** In fields such as physics and climate modeling, DGMs can simulate complex processes that are difficult to capture with traditional methods [9].

2.2.3 The Rise of GANs

Generative adversarial network (GAN) designed by Ian Goodfellow and his colleagues in June 2014 [10]. The core premise of GANs is based on a two neural networks contest with each other in the form of a *zero-sum game*, where one agent's gain is another agent's loss (i.e. total gains are zero for both networks and loss or gain of utility of each network is exactly balanced by the gain or loss of utility of another network)

The GAN model is a cognate to Counterfeiters vs Police game. In this game the counterfeiters try to fool the police by printing fake money. The fake money is labelled as real for training the thief. Whereas the trained police differentiate the real money from the fake one's (fig. 2.5).

Technically, The idea presented in the original paper of a GAN is based on the "indirect" training through the discriminator, another neural network that is able to tell how much an input is "realistic", which itself is also being updated dynamically.[5] This basically means that the generator is not trained to minimize the distance to a specific image, but rather to fool the discriminator. This enables the model to learn in an unsupervised manner.

Eventhough, because It can optimise some loss functions that are difficult to handle, via adversarial learning, which allows us to realise semi supervised and unsupervised learning technology. It proved useful for semi-supervised learning [11], fully supervised learning [12], and reinforcement learning [13].

These networks achieve results by modelling high-dimensional data distributions. GANs can avoid some shortcomings of traditional approaches by modelling the high-dimensional distribution of data. It can optimise some loss functions that are difficult to handle via adversarial learning, and since then GANs have been extensively researched and well-crafted for performing certain tasks or customized for the sake of leveraging the performances of certain applications, especially in image processing, classification, and computer vision.

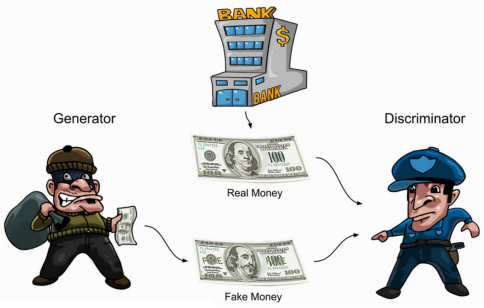


Figure 2.5: Counterfeiter vs police game

Generative adversarial networks (GANs) offer several advantages over traditional generative models, making them highly effective for diverse applications in data generation:

- **Parallelism:** Unlike other generative approaches that often involve sequential processes, such as generating data point by data point or relying on previous outputs to influence subsequent ones, GANs utilize feed-forward networks. This architecture enables parallel generation of samples, significantly accelerating the process and making GANs well-suited for high-dimensional data generation tasks.

- **Producing Higher-Quality Results:** GANs excel at capturing fine-grained details and high-frequency components within the data. The adversarial training process pushes the generator to produce outputs that closely mimic the distribution of the original dataset, resulting in sharper and more realistic outputs across various domains, including visual, audio, and textual data.

- **Adaptability:** GANs are highly flexible and can be tailored to specific use cases, such as conditional data generation, domain adaptation, and feature augmentation. This adaptability makes GANs invaluable in applications ranging from synthetic data generation for training machine learning models to artistic content creation.

2.3 GANs in Seismology

Generative Adversarial Networks (GANs) have found increasing applications in seismology.

Capability to Model Complex Distributions

GANs are adept at learning complex, high-dimensional distributions of seismic waveforms, enabling the generation of synthetic data that capture the intricate characteristics of near-source ground motions (Zhu et al., 2019). Unlike traditional numerical methods that require explicit equations of motion, GANs offer flexibility in representing complex subsurface conditions and rupture processes.

Reduced Computational Demand

Once trained, GANs can generate synthetic seismic data in real time, significantly reducing computational overhead compared to traditional numerical methods. This makes GANs suitable for applications such as earthquake early warning systems and emergency response (Moseley et al., 2020).

Improved Generalization and Data Augmentation

GANs can augment limited datasets by generating diverse synthetic seismic events, improving the performance of machine learning models in regions with sparse seismic records. This capability is invaluable for regions with limited historical data or where empirical data collection is challenging (Marano et al., 2023).

Overcoming Limitations of Small Datasets

By generating realistic synthetic data, GANs enhance the training datasets for various machine learning models, improving their performance in tasks such as seismic event detection, discrimination, and classification (Li et al., 2020).

Study	Focus	Outcome	Reference
Zhu et al. (2019)	GAN-based synthesis of seismic waveforms	Generated realistic seismic waveforms that improved earthquake detection and classification accuracy	Zhu et al., 2019
Moseley et al. (2020)	GANs for real-time seismic monitoring	Demonstrated the use of GANs for rapid generation of synthetic data for real-time applications	Moseley et al., 2020
Marano et al. (2023)	Review of GANs in seismology	Highlighted the advantages of GANs for seismic signal augmentation and engineering applications	Marano et al., 2023

Table 2.1: Key Studies Utilizing GANs for Seismic Data Synthesis

2.4 Seismic Data Augmentation

Seismic data augmentation is essential in geophysics and machine learning applications due to limitations in data availability and challenges in seismic signal analysis. Below are the primary motivations for seismic data augmentation:

Table 2.2: Seismic Data Augmentation: Importance and Applications

Motivation	Description and Solution
Limited Availability of Real Seismic Data	Acquiring high-quality seismic data is expensive and time-consuming, especially in regions with limited instrumentation. Large earthquakes are rare, and seismic stations are unevenly distributed, leading to data gaps in specific regions. *Solution:* Use synthetic seismic data generation. Simulate rare events and expand datasets to cover regions with limited data availability.
Enhancing ML Model Training	Small datasets can lead to overfitting, where machine learning models memorize patterns rather than generalize. *Solution:* Augmentation improves the ability of models to detect and classify seismic events across diverse conditions.
Generalizing Across Regions	Seismic signals vary based on geological conditions, instrumentation, and other local factors. Models trained on one dataset may not perform well when applied to another region. *Solution:* Use domain adaptation, style transfer, and regional data augmentation techniques to improve model generalization.

Seismic Data Augmentation Techniques:

- *Traditional Techniques:* Adding Gaussian noise, time stretching or compressing, flipping or rotating waveforms, band-pass filtering.

  As shown in Table B.2, Traditional methods for generating synthetic seismic data, such as finite difference, finite element, and spectral element methods [14], involve solving wave equations to simulate seismic wave propagation. These methods are computationally intensive and often use simplified models, limiting their ability to accurately replicate the Earth's complex subsurface and near-source waveforms. They struggle with modeling intricate rupture dynamics and require substantial resources, making them unsuitable for real-time applications like early warning systems. As a result, their resolution and accuracy are often restricted, particularly in complex geological settings [15] [16]. These limitations highlight the need for more advanced approaches to seismic data generation.

- *Advanced Techniques:* Generative Adversarial Networks (GANs), style transfer, and physics-based signal simulations.

2.5 Related Work

Synthetic Generation of Seismic Signals

GANs have revolutionized the paradigm of synthetic data generation for seismic applications. For instance, Marano et al. (2023) reviewed how GANs were employed to synthesize seismic waveforms for tasks such as ground motion analysis, earthquake engineering, and structural health monitoring.

Augmenting Sparse Datasets

In many seismic regions, data scarcity limits the effectiveness of traditional models. GANs, such as cGANs and DCGANs, have been employed to generate diverse synthetic seismic events, which, when combined with real data, enhance the robustness of earthquake detection algorithms (Wang et al., 2021).

Domain-Specific Advances in GAN Architectures

Recent advancements in GAN architectures, such as EarthquakeGAN and SeismoGAN, have tailored GAN models to address specific seismic challenges, such as generating three-dimensional seismic signals and balancing dataset biases (Marano et al., 2023). These models integrate physical constraints to ensure the generated data maintain geophysical realism.

Physics-Informed GANs

Integrating physics-based constraints into GANs, such as Wasserstein GANs with gradient penalties (WGAN-GP), has enhanced the generation of seismic data by ensuring that the synthetic signals adhere to physical laws and constraints (Marano et al., 2023).

Traditional Numerical Methods		**GANs-Based Seismic Data Synthesis**
High Computational Cost		Low Computational Cost
Simplified Earth Models	$\longrightarrow$	Models Complex Distributions
Limited High-Frequency Accuracy		Real-Time Applications
Slow data synthesis		Data Augmentation

Table 2.3: Comparison between Traditional Numerical Methods and GANs-Based Seismic Data Synthesis

In [17], the authors proposed a GAN-based approach to generate realistic seismic time series for data augmentation, addressing the limitations of traditional augmentation methods like shifting, flipping, and scaling, which often fail to preserve sequence structure and diversity. Their model utilized a single GAN with a unique CNN-gated structure, where the generator combined a 1D CNN for capturing temporal relationships and a gated CNN with linear unit activations to enhance output quality. The discriminator employed a 1D CNN structure to distinguish real from synthetic data, enabling the generation of high-quality, three-dimensional seismic outputs.

2.6 SeismoGen

The paper [18] by Wang et al. introduces a novel generative adversarial network (GAN) framework, SeismoGen, designed to synthesize realistic seismic waveforms. This work addresses the challenges of limited labeled seismic data by using synthetic waveforms for data augmentation, enhancing machine learning models in earthquake detection.

2.6.1 Model Overview

SeismoGen utilizes a conditional GAN architecture consisting of a generator and a discriminator. The generator synthesizes three-component seismic waveforms by incorporating domain knowledge, while the discriminator distinguishes between real and synthetic data. Figure 2.6 illustrates the GAN structure:

- **Generator:** A multi-pipeline design independently synthesizes each waveform component. The generator takes as input Gaussian noise and a conditional label, producing synthetic signals with temporal and frequency-domain characteristics akin to real seismic events.
- **Discriminator:** Evaluates the realism of input samples by learning distinctive features of seismic signals. A frequency decomposition module enhances its ability to differentiate between earthquake and non-earthquake waveforms.

2.6.2 Dataset and Preprocessing

The dataset used consists of seismic data from two stations in the United States. The preprocessing pipeline included:

- Segmentation of raw waveforms into 40-second windows.
- Normalization of waveforms using Z-Score scaling to ensure comparability.
- Classification into earthquake (positive) and noise (negative) samples, ensuring balance across classes.

2.6.3 Experimental Results

Four experiments validated the SeismoGen framework:

1. **Visual Evaluation:** Synthetic waveforms were visually indistinguishable from real waveforms
2. **Classification Task:** A classifier trained on synthetic data achieved high accuracy, demonstrating the utility of GAN-generated samples.
3. **Data Augmentation:** Combining synthetic and real samples improved classifier accuracy by up to 14%.

Extra Results shown in B.3

2.6.4 Limitations

The SeismoGen framework significantly enhances ML models for earthquake detection by providing high-quality synthetic data for augmentation. However, its applicability is currently limited to data from the specific stations used in training.
In this Research, will expand SeismoGen approach to diverse seismic regions and incorporate additional waveform complexities of new-epicenter stations

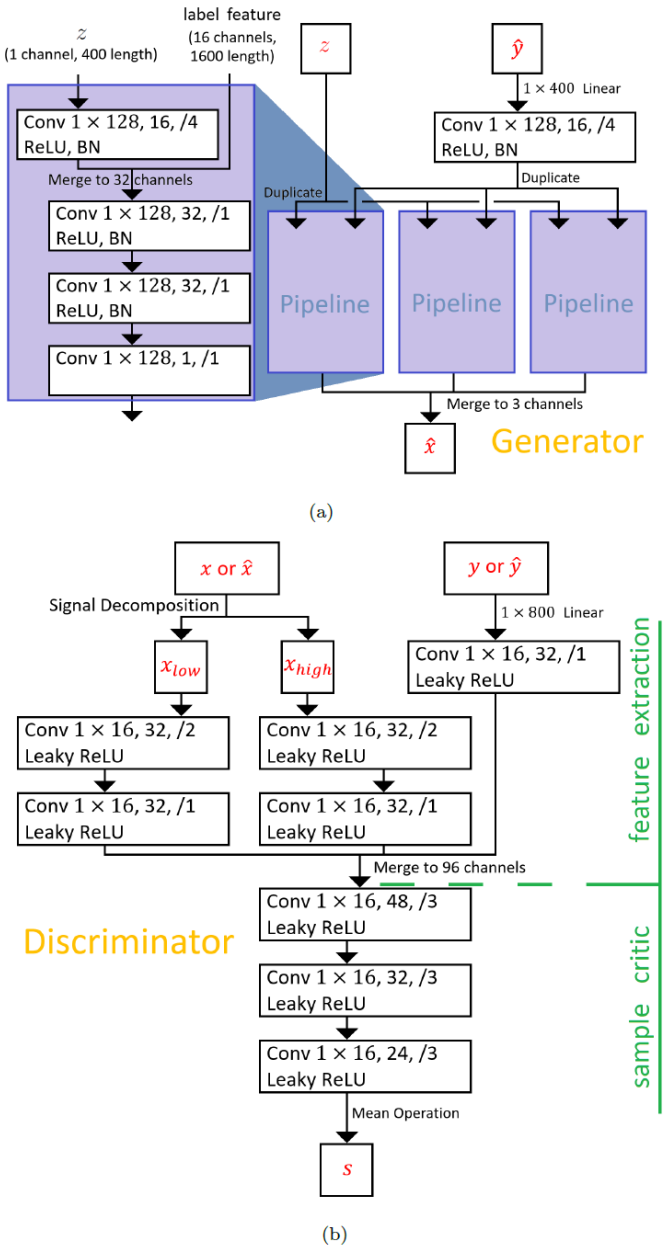


Figure 2.6: GAN architecture highlighting the generator and discriminator pipelines.

Chapter 3

Theoretical Framework

For understanding and implementing Generative Adversarial Networks (GANs) and their application in generating synthetic seismic data, a solid understanding of the underlying mathematical and theoretical principles is essential. This chapter provides a comprehensive exploration of the fundamental mathematical concepts and theoretical frameworks that underpin GANs and their application to seismic data.

It begins with deep learning fundamentals which introduces key deep learning concepts, including neural networks, optimization techniques, and backpropagation. It discusses the role of activation functions, such as ReLU, and explains how neural networks learn through gradient-based optimization and error propagation.

The second section present an overview of the basic principles of GANs, including their adversarial training mechanism and loss functions. Limitations of classic GANs, such as mode collapse and instability, are also discussed.

A brief discussion is presented on major GAN variants. Special focus is placed on Wasserstein GANs (WGANs), emphasizing their advantages in addressing stability and convergence issues.

The chapter further discusses the unique characteristics of seismic data and the specific mathematical challenges they present, setting the stage for understanding how GANs can be adapted to handle these complexities.

By establishing this mathematical and theoretical groundwork, this chapter lays the foundation for the Methodology chapter, where these principles are applied to develop and implement effective GAN-based models for seismic data analysis.

3.1 Deep Learning Fundamentals

3.1.1 Neural Networks

Neural networks are the backbone of GANs. They consist of layers of neurons performing linear transformations followed by non-linear activation functions.

A single-layer perceptron is represented as:

$$y = \sigma(Wx + b)$$

where W is the weight matrix, x the input, b the bias, and σ the activation function (e.g., ReLU, sigmoid).

The Rectified Linear Unit (ReLU) introduces non-linearity to the model, allowing neural networks to learn complex patterns. ReLU is defined as:

$$\text{ReLU}(x) = \max(0, x),$$

where:

- x is the input to the activation function.

The Leaky Rectified Linear Unit (Leaky ReLU) is a variant of the ReLU. It allows a small, non-zero gradient when the input is negative, addressing the problem of "dying neurons" (where neurons output zero and stop learning). Leaky ReLU is defined as:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0, \\ \alpha x, & \text{if } x < 0, \end{cases}$$

where:

- x is the input to the activation function,
- α is a small positive constant (typically $\alpha = 0.01$), controlling the slope for negative values.

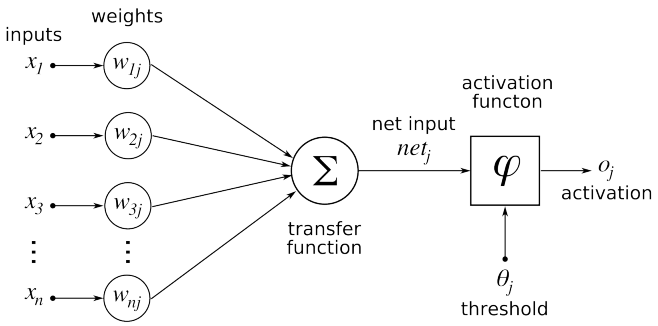


Figure 3.1: A diagram of a neural network showing inputs, weights, biases, activations, and outputs.

3.1.2 Optimization Techniques

Optimization is critical for training neural networks. Gradient-based methods are the most widely used. The objective is to minimize a loss function:

$$\min_{\theta} \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i)$$

where ℓ is the loss function, y_i are true labels, and $\hat{y}_i$ are predictions. [1]

3.1.3 Backpropagation

Backpropagation is an algorithm used to train neural networks by updating weights to minimize the loss function. It computes gradients of the loss function with respect to the model's parameters using the chain rule of calculus. Following are the steps in Backpropagation:

1. Forward Pass: Compute the output of the network and the loss ($\mathcal{L}$).
2. Backward Pass: Use the chain rule to calculate the gradient of the loss with respect to each parameter.
3. Parameter Update: Adjust weights and biases using an optimization algorithm like gradient descent:

   $$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}(\theta_t),$$

   where:

   - η: Learning rate.
   - $\nabla \mathcal{L}$: Gradient of the loss function.

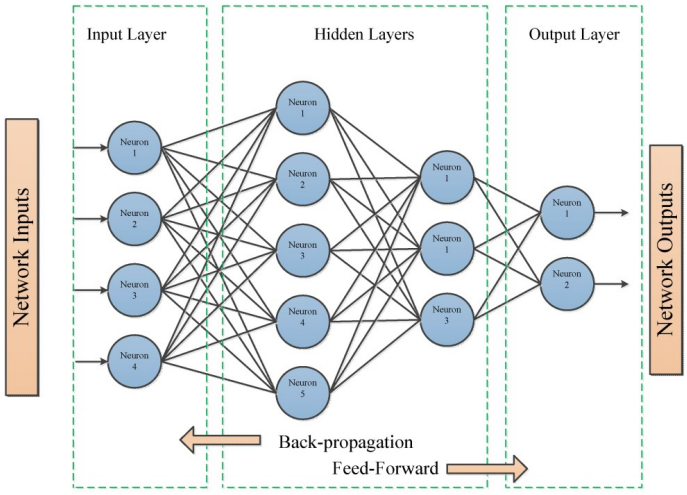


Figure 3.2: A diagram of a neural network showing Layers and Backpropagation

[1] *The Adam optimizer*, detailed in Appendix C.1, is a robust optimization algorithm widely used in machine learning for its adaptive learning rate and momentum-based approach and is utilized in this research

3.2 Generative Adversarial Networks (GANs)

3.2.1 GAN Architecture

There are two models that constitute Both the framework of the original GAN and its variants: generator(G) and discriminator(D)[2].

The first model, generator(G), attempts to produce new instances that are entirely distinct from the original dataset and have never been seen by the generator before (aims to increase the likelihood of mistakes made by the discriminator), yet have a data distribution that is comparable to the genuine dataset. A random noise is provided to the generator, which it uses to generate fresh samples that follow the distribution of real data.

The second model of GANs is a discriminator(D) whose task is to examine the distribution, which might have originated from the original dataset or the generator, and to calculate the likelihood that the sample came from the original dataset or the generator(Fig. 3.3).

Since both the networks are battling over one number, this framework resembles a two-player min-max game. One of them desires a high number value, while the other desires a low number value. The discriminator's error rate is the value which generator and discriminator are competing for. As a result, the discriminator prefers a low error rate, while the generator prefers a high error rate. It is the goal of optimization to reach a Nash equilibrium [19].

Furthermore, there is a trade-off between both models' learning rates.

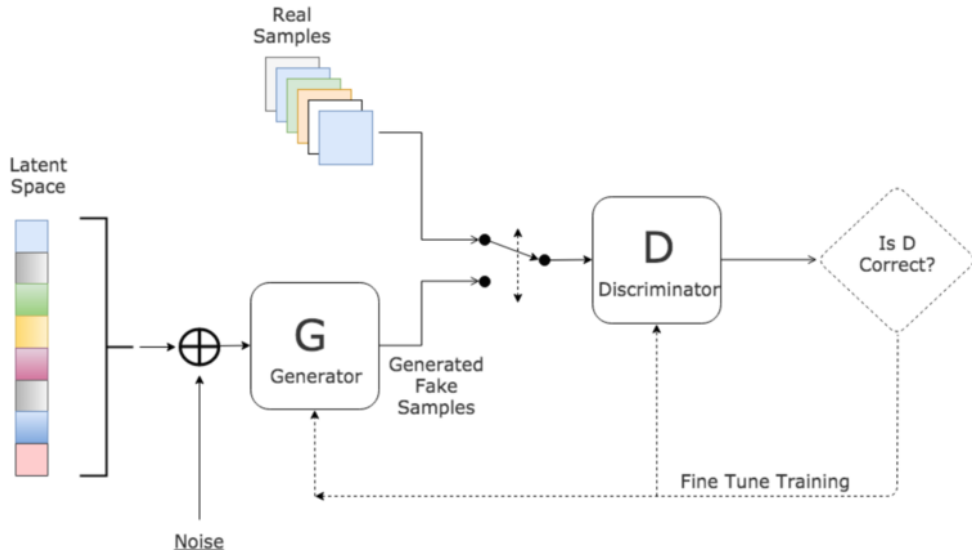


Figure 3.3: GAN Model

3.2.2 KL Divergence and Jensen-Shannon Divergence

Before diving into GANs Objective function and adversarial training, let us first review two metrics for quantifying the similarity between probability distributions.

[2]But it could be also done by using any kind of differentiable system with the capability of data translation from one space to another

KL Divergence: The Kullback–Leibler (KL) divergence measures how one probability distribution $Q(x)$ diverges from a second expected probability distribution $P(x)$:

$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}.$$

Key properties:

- $D_{KL}(P||Q) \geq 0$, with equality when $P(x) = Q(x)$ for all x.
- It is asymmetric; $D_{KL}(P||Q) \neq D_{KL}(Q||P)$.

JS Divergence: The Jensen–Shannon (JS) divergence is a symmetric measure derived from KL divergence. It is defined as:

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M),$$

where $M = \frac{1}{2}(P + Q)$ is the average distribution.

Key properties:

- $D_{JS}(P||Q) \in [0, 1]$, with 0 indicating identical distributions.
- It is smoother and symmetric, making it more suitable for tasks requiring balanced evaluations.

3.2.3 Adversarial Training and Minimax Optimization

GANs rely on adversarial training, where a generator and discriminator are trained in a minimax game. The two-player min-max game can be formulated as:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))] \tag{3.1}$$

where,

- $D(x)$: as estimated by the discriminator,it is the likelihood that actual data instance x is genuine.
- $\mathbb{E}_x$: is the average of all real-world data occurrences.
- $G(z)$: output when noise z is applied to the generator.
- $D(G(z))$: as estimated by the discriminator,is the likelihood that fake data instance is genuine
- $\mathbb{E}_z$: is the expected value across all false instances G(z) generated.

The diagram below summarizes how we train the discriminator and the generator using the corresponding gradient. Complete GAN algorithm available in C.2

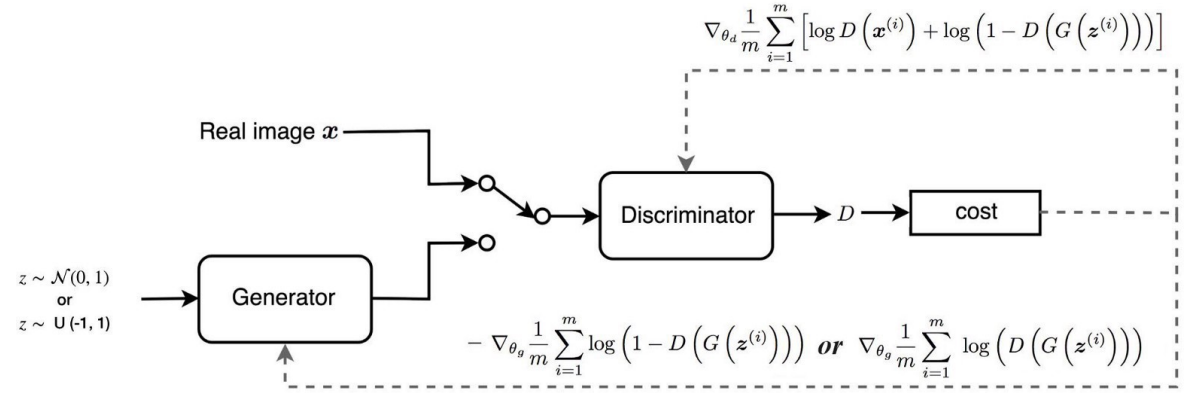


Figure 3.4: A schematic of adversarial training showing interactions between generator and discriminator

3.2.4 GANs Limitations

While GANs provide a powerful framework for generative modeling, they suffer from several challenges:

- **Non-Convergence:** The minimax optimization can be unstable due to the adversarial nature of training.

- **Mode Collapse:** The generator may produce limited varieties of outputs, failing to capture the diversity in $p_{\text{data}}(x)$.

- **Vanishing Gradients:** When $D(x)$ is too confident, gradients for $G(z)$ diminish, stalling the training process.

These issues motivate the development of advanced GAN variants, such as Deep Convolutional GANs (DCGANs) replaced fully connected layers with convolutional ones, improving stability and handling high-dimensional data effectively [20]. Conditional GANs (cGANs) introduced conditioning with auxiliary inputs, such as labels, to control data generation, making them suitable for structured outputs [21]. CycleGANs leveraged cycle consistency losses to map between unpaired domains, excelling in translation tasks without paired data [22]. Wasserstein GANs (WGANs), which replace JS divergence with Wasserstein distance and will be discussed in the following section in details.

3.2.5 WGAN

Unlike the original discriminator that classifies real or fake, introduced by Arjovsky et al. [23], the WGAN critic learns to score the similarity between generated and real distributions. WGAN utilize the Earth Mover's Distance (Wasserstein distance) as the metric to quantify the difference between the generated and real data distributions.

Wasserstein Distance

Wasserstein distance measures the cost of transforming one distribution into another:

$$W(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

where $\Pi(P,Q)$ is the set of all joint distributions.

To enforce the Lipschitz continuity condition required for the Wasserstein distance, WGAN uses weight clipping on the critic's weights, which ensures that the gradient norms remain bounded. This constraint stabilizes training and leads to more effective optimization.

The key advantages of WGAN over the standard GAN include:

- **Improved Training Stability**: The Wasserstein distance provides a smoother loss function, leading to more stable gradients for both the generator and discriminator.
- **Mode Collapse Mitigation**: WGAN is effective in reducing mode collapse due to its improved distance metric, allowing the generator to explore a more diverse set of outputs.
- **Better Convergence**: The discriminator, termed the "critic" in WGAN, outputs a scalar score instead of a probability, which provides better feedback to the generator, resulting in improved convergence.

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Figure 3.5: Algorithm of Wasserstein generative adversarial network. (Image source: Arjovsky, Chintala, 2017.)

3.3 Seismic Data for Stations Close to the Epicenter

Stations near the epicenter are critical for capturing intense ground motions essential for earthquake characterization, seismic hazard assessment, and resilient infrastructure development. However, these stations face challenges such as instrument limitations and data quality issues, necessitating the use of synthetic data to augment or replace incomplete or corrupted recordings.

3.3.1 Challenges Faced by Stations Near the Epicenter

Instrument Damage

Severe shaking near the epicenter often leads to physical damage or malfunction of seismic instruments, resulting in significant data loss. This limits the ability to analyze seismic events comprehensively.

> *"The durability and performance of seismic instruments in high-intensity earthquake zones remain a key limitation in obtaining reliable data during large seismic events."* [24]

Data Quality Issues

Even when instruments remain functional, the quality of the recorded data is often compromised due to:

Instrument Saturation and Clipping Instruments near the epicenter frequently experience amplitudes exceeding their dynamic range, leading to signal clipping. Clipped signals misrepresent peak ground motions and distort waveform analysis.

Complex Waveforms Proximity to the earthquake source results in high-frequency, multiphase waveforms due to overlapping P-waves, S-waves, and surface waves. These waveforms complicate phase separation and source characterization.

Strong Nonlinear Effects Nonlinear soil behavior, such as liquefaction, significantly alters wave characteristics near the epicenter, hindering accurate seismic modeling and interpretation.[25]

3.3.2 Characteristics of Seismic Signals Near the Epicenter

Signals captured near the epicenter exhibit unique properties due to the proximity to the earthquake source:

High-Frequency Content

These signals contain high-frequency components with minimal attenuation, essential for detailed ground motion studies but challenging for analysis and synthesis.

Table 3.1: Ranges, Grades, and Impacts of Proximity to the Epicenter

Distance from Epicenter	**Grade**	**Impact**
<10 km	Severe	Extreme ground motion; instrument damage; significant nonlinear effects; complex waveforms.
10–30 km	Strong	High ground motion; moderate nonlinear effects; clipping and waveform distortion likely.
30–70 km	Moderate	Moderate ground motion; clearer waveforms; important high-frequency components.
>70 km	Weak	Attenuated ground motion; dominated by low-frequency components; limited near-field dynamics.

Rapid Amplitude Variations

Ground motions show steep amplitude gradients, making them highly non-stationary and requiring careful modeling in synthetic data generation.

Nonlinear Wave Propagation

Soil behavior near the epicenter introduces nonlinear distortions in waveforms, affecting their velocity and attenuation properties, and adding complexity to signal synthesis models.

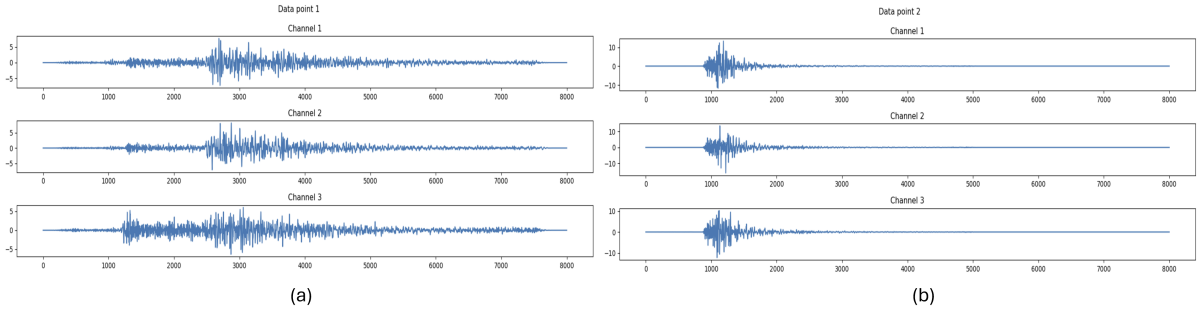


Figure 3.6: The effect of Epicentral distance on the characteristics of Seismic Waveform. Waveform (a) recorded at 60 KM while (b) taken at 10 KM shows overlap between P-S waves (source from analyzed dataset).

Chapter 4

Methodology

This chapter outlines our methodology for generating realistic synthetic seismic signals using Generative Adversarial Networks (GANs) to augment seismic datasets near earthquake epicenters, where data scarcity poses significant challenges. By creating high-quality synthetic waveforms, we aim to enhance earthquake detection algorithms and contribute to seismic hazard assessment.

We begin with an **Overview of the Approach**, explaining our strategy and the rationale for employing GANs, along with challenges posed by near-epicenter data.

Next, we examine **Dataset Insights**, focusing on the ITalian ACcelerometric Archive (ITACA) dataset, including data formats, quality considerations, metadata, and the distribution of seismic events.

In **Data Preparation**, we detail waveform selection, data cleaning, and preprocessing techniques like standardization, normalization, and segmentation, emphasizing handling variability and noise in near-epicenter signals.

We then describe the **Architecture Design and Hyperparameter Selection**, justifying our choice of a Wasserstein GAN with Gradient Penalty (WGAN-GP) and discussing customization of network components and hyperparameters.

The **Implementation Details** section covers programming tools and computational resources used, including Python, PyTorch, and necessary hardware for training on high-resolution data.

Our **Data Loading and Preprocessing Framework** outlines efficient data handling using PyTorch, featuring custom classes that ensure scalability and seamless integration.

Finally, we discuss **Performance Metrics and Evaluation**, presenting qualitative assessments like visual inspections and quantitative metrics like statistical checks, addressing limitations of traditional GAN evaluation methods.

Through this methodology, we provide a clear roadmap of our research, contributing to generating realistic synthetic seismic data and offering insights for future deep learning applications in geophysical analysis.

4.1 Overview of the Approach

Near-epicenter seismic signals exhibit high variability and significant noise, posing challenges for effective machine learning models. This research utilizes Generative Adversarial Networks (GANs) to generate realistic synthetic seismic signals, augmenting the ITalian ACcelerometric Archive (ITACA) dataset to improve data quality, diversity, and balance. By focusing on significant seismic events (magnitude $M_w > 4.0$, depth < 50 km) recorded within Italy, we target critical data for enhancing earthquake detection algorithms and seismic hazard assessment.The methodology consists of four main stages:

- Data Preparation: involves selecting and preprocessing waveforms from ITACA, including noise reduction, Z-score normalization, and segmentation into uniform 40-second windows. The data is reshaped to match the input requirements of the GAN architecture and split into training, validation, and testing sets, ensuring minimal data leakage and preserving the variability inherent in near-epicenter signals.
- GAN Model Design: we employ a Wasserstein GAN with Gradient Penalty (WGAN-GP) due to its stability and robustness in handling high-variability data distributions. The generator and discriminator are customized to capture the intricate features of near-epicenter seismic signals, with adjustments to handle three-component time series data.
- Training Strategy: includes iterative training with optimized hyperparameters, such as an increased latent space dimension to accommodate the high sampling rate (200 Hz) and adjusted batch sizes to balance computational efficiency and model stability. We utilize gradient penalty and adaptive learning rates to maintain training stability and mitigate issues like mode collapse.
- Evaluation Approach: incorporates both qualitative and quantitative metrics, including visual inspections, statistical consistency checks, and waveform similarity analyses, to assess the realism and diversity of the generated signals. Special attention is given to the characteristics of near-epicenter seismic events to ensure the synthetic data's applicability in practical scenarios.

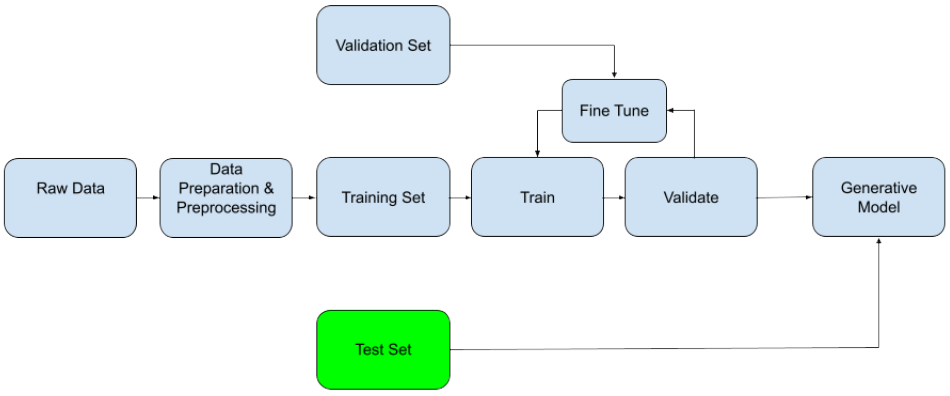


Figure 4.1: General Workflow

4.2 Dataset Insights

The data utilized to train and evaluate the developed model can be openly accessed through the ITalian ACcelerometric Archive of Waveforms (ITACA) [26]. ITACA is one of the most comprehensive seismic databases in Europe, maintained by the Italian National Institute of Geophysics and Volcanology (INGV). It contains a vast array of ground motion records from earthquakes that have occurred in and around Italy, providing a rich resource for seismological research and applications.
The latest version, ITACA 4.0 (May 2024 release), contains approximately 40,900 manually processed waveforms relating to 2,531 events and 1,696 stations, covering moderate to severe earthquakes with magnitudes ranging from 3.0 to 6.9 that occurred in Italy between 1972 and 2023. This extensive dataset offers comprehensive temporal and spatial coverage essential for robust seismic analysis and modeling.

4.2.1 Data Formats and Quality

The ITACA database provides high-quality, strong-motion data consisting of meticulously calibrated three-component time series recorded from well-monitored seismic events across Italy. Each waveform undergoes rigorous quality control procedures to ensure precise time alignment, accurate amplitude scaling, and metadata integrity. These measures are critical for machine learning applications, such as Generative Adversarial Networks (GANs), which depend on consistent and low-noise datasets for effective model performance. By including three-component recordings (north-south, east-west, and vertical motions), the dataset allows for comprehensive analysis of seismic wave propagation and ground motion characteristics.
In order to accommodate different research needs and computational resources, the seismic waveforms are available in two primary data formats: ASCII and ASDF (Advanced Seismic Data Format), each serving different purposes based on the complexity and volume of data.
The choice between ASCII and ASDF depends on the specific needs of the research, including factors such as dataset size, required metadata detail, and computational capabilities. For this study, ASDF was primarily used due to its ability to handle large volumes of data efficiently and its structured approach to metadata management, which is crucial for ensuring data integrity and facilitating complex analyses.While ASDF and HDF5 are distinct formats, ITACA leverages the HDF5 container format to store ASDF files, effectively integrating all data components into a single, manageable file. The layout of an ASDF file within its HDF5 container is illustrated in Figure 4.2. It contains four parts, all of which are elaborated upon in the following section:

1. **Earthquake Information (Yellow)**: Stored in the form of a single QuakeML file, which is the most comprehensive earthquake description format currently available, allowing for detailed representation of an arbitrary number of earthquakes or seismic events of various types.

2. **Seismic Waveform Information (Green)**: Stored at a per-station granularity, this section contains the seismic waveform data along with station metadata in the form of

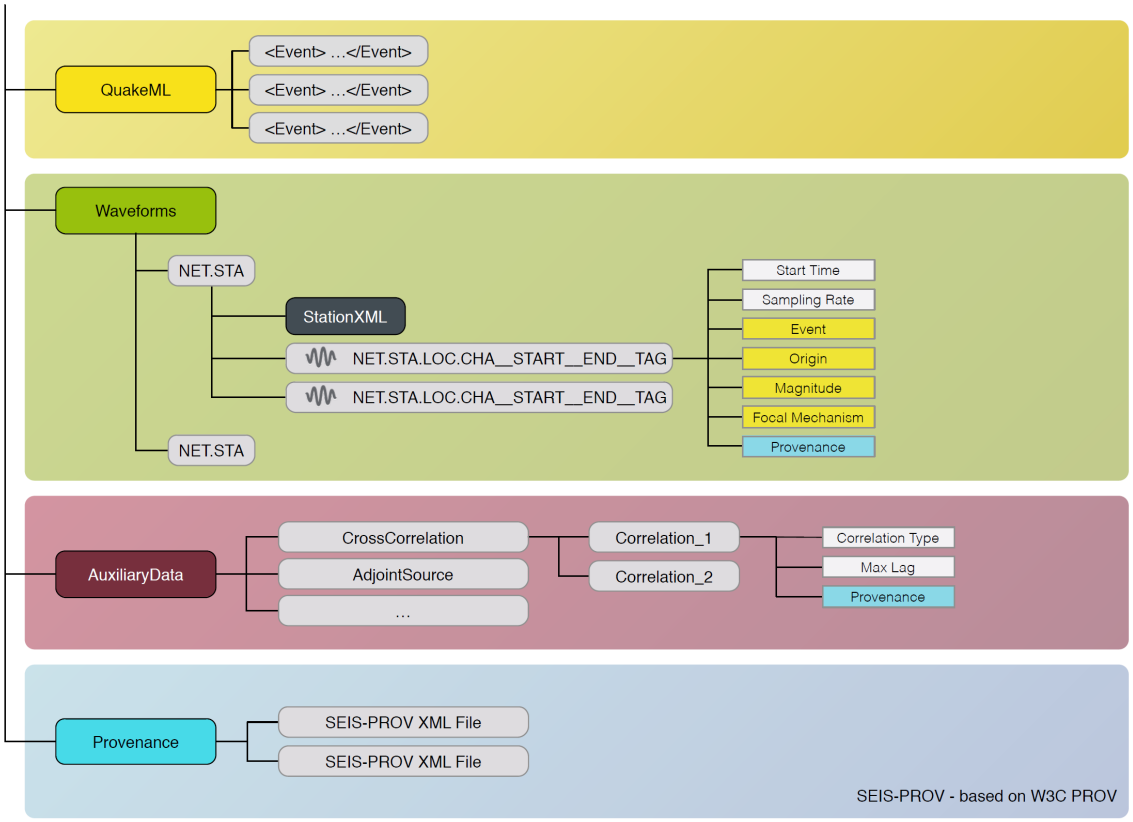


Figure 4.2: The general structure of an ASDF file in its HDF5 container—it has four distinct parts [27]

StationXML files. It also includes possible associations of waveforms with parts of seismic events and provenance links.

3. **Auxiliary Data (Hierarchical Storage)**: Anything that cannot be regarded as a seismic waveform is hierarchically stored in the auxiliary data section. Arbitrary nesting is possible alongside the capability to store provenance information, enhancing the flexibility of data management.

4. **Provenance Records (Blue)**: Contains a collection of provenance documents that can be referred to from other parts of the ASDF file, ensuring traceability of data processing steps. Examples of this are relations of a waveform to a certain event or a provenance record for a piece of auxiliary data [27].

```
<HDF5 file "itaca_data_12.h5" (mode r)>
├── AuxiliaryData
├── Provenance
└── Waveforms
    └── IV.FEMA
        ├── IV.FEMA..HNE__2021-10-18T12:54:06__2021-10-18T12:54:56___hne_int_20211018_0000182_acc_mp (10001)
        ├── IV.FEMA..HNN__2021-10-18T12:54:06__2021-10-18T12:54:56___hnn_int_20211018_0000182_acc_mp (10001)
        └── IV.FEMA..HNZ__2021-10-18T12:54:06__2021-10-18T12:54:56___hnz_int_20211018_0000182_acc_mp (10001)
```

Figure 4.3: Example of one event from our dataset

4.2.2 Data Access and Metadata

Metadata: Accompanying the waveform data, ITACA provides metadata in a structured flat-file format (TXT). Metadata falls into three categories; event-related data, station information, and source-site distance metadata.(*attached user manual and features*).

- **Event-Related Metadata**: Includes information such as the date, time, magnitude, and location (latitude, longitude, and depth) of each seismic event. Event metadata
- **Station-Based Metadata**: Details about the recording stations, including their geographic location, site classification, and instrumentation used.
- **Source-Site Distance Metadata**: Information on the distance between the seismic event's epicenter and the recording station, which is critical for understanding the attenuation of seismic waves as they travel. This metadata is not explored however the noise is to be included to general noise

The Flat file along with it's metadata is to be used to select the desired dataset to be augmented for the objective of this research

The final dataset comprises detailed records of several major seismic events, including both mainshocks and significant aftershocks. Below are some of the key characteristics:

4.2.3 Geographical Distribution

The dataset includes seismic events distributed across various regions of Italy, reflecting the country's diverse seismic activity. The prominent regions are:

- **Central Italy**: Renowned for the 2016 earthquake sequence, which caused severe destruction in towns such as Amatrice and Norcia.
- **Southern Italy**: Home to Calabria, one of the most seismically active areas in Italy, with a history of devastating earthquakes.
- **Northern Italy**: Although less active compared to other regions, this area has experienced significant seismic events, particularly near the Apennine mountains.

4.2.4 Temporal Distribution

The dataset encompasses seismic events spanning multiple decades, offering a comprehensive perspective on Italy's seismic activity. This extended timeframe enables the exploration of trends and patterns in seismic behavior over the years and sheds light on how historical events have shaped the current understanding of seismic risk.

4.3 Data Preparation

4.3.1 Dataset Selection and Data Cleaning

The seismic dataset was extracted using custom Python scripts to facilitate the download and organization of waveform data from ITACA. These scripts ensured consistency and completeness, which are critical for training the GAN model.

Waveform Components Each seismic event in the dataset includes three components of ground motion: north-south (N-S), east-west (E-W), and vertical (Z). These components are essential for capturing the three-dimensional nature of seismic waves, which is crucial for model accuracy.

Data Selection

The following steps were taken to filter and prepare the waveform data for the GAN model:

- **Select Waveforms**
  - Seismic events were selected from the ITACA archive platform based on the following criteria:
    - **Magnitude ($M_w > 4.0$)**: Events with a magnitude greater than 4.0 were chosen to focus on significant seismic activity, which is critical for training the GAN model to simulate impactful earthquakes.
    - **Epicentral Depth ($< 50\,\mathbf{km}$)**: Events with shallow depths (¡ 50 km) were prioritized due to their higher potential for causing surface shaking.
  - Waveforms recorded exclusively within Italy were selected to maintain geographic consistency.
  - The filtered data was exported as a CSV file for further processing.
- **Explore and Clean Metadata**
  - Addressed missing values (e.g., NaN) in the metadata to ensure data integrity.
  - Conducted an exploratory analysis to understand the distribution and patterns in the metadata.
- **Extract Waveform Data from ITACA Database**
  - Waveform data was extracted in HDF5 format from the ITACA database.
  - Event ID and station ID were used as access keys to streamline the retrieval process.
- **Explore Waveform Data and QuakeML Metadata**
  - Examined the HDF5 structure to understand waveform data and QuakeML metadata.
  - Verified key parameters, including data windows and sampling rates.

- **Preprocess and Convert Data to NumPy Format**
  - Raw waveform data was transformed into NumPy arrays and saved as `.npy` files to facilitate efficient data handling.
  - Preprocessing waveforms into NumPy arrays reduced the computational overhead compared to on-the-fly data extraction from HDF5 files.
  - The `.npy` format ensured compact file sizes, preserving numerical precision essential for the GAN model, and supported efficient batch processing for deep learning frameworks.

4.3.2 Statistical Analysis

Basic statistical properties, including the mean, median, standard deviation, and range, were computed for the dataset. These metrics were used to assess the quality of the dataset and gain insights into its variability, which directly impacts the learning process of the GAN model.

4.3.3 Data Preprocessing

Data preprocessing is a critical step in preparing seismic data for input into the GAN model. The preprocessing steps closely follow the methodology outlined in the SeismoGen paper, with modifications tailored to this dataset:

- **Sampling Rate:** The dataset was uniformly sampled at 200 Hz
- **Standardization:** The original dataset of 9987 waveforms was reduced to 1298 instances to ensure a maximum window length of 40 seconds. Limiting waveforms to 8000 points guarantees the inclusion of at least 1200 instances while ensuring manageable time-series complexity.
- **Padding:** Instead of resampling, waveforms shorter than 40 seconds were padded to a uniform length of 40 seconds.
- **Stratification:** Data was stratified by key features (e.g., magnitude, source location) to create a balanced training dataset.
- **Data Normalization:** Seismic data was normalized on a per-signal basis using Z-Score normalization:

$$\text{normalized\_data}[i] = \frac{\text{data}[i] - \text{mean}[i]}{\text{std}[i]}$$

  Here, $\text{mean}[i]$ and $\text{std}[i]$ are the mean and standard deviation of the i^{th} signal. Normalization ensures that features are on a comparable scale, mitigating model bias towards high-magnitude values and preserving critical low-amplitude patterns. This transformation stabilizes training by centering data around a mean of 0 with a standard deviation of 1.
- **Data Reshaping:** The processed data was reshaped into a 3-dimensional array to match the input requirements of the GAN architecture. The final dataset shape was

$(8000, 3, 1298)$, representing 1298 events, each with 8000 time samples across 3 components (e.g., horizontal and vertical accelerations).

- **Data Splitting:** The dataset was split into 80% training, 10% validation, and 10% testing sets, considering several factors:

  1. A separate validation set aids hyperparameter tuning [3].
  2. Larger datasets require proportionally larger training sets [28, 5].
  3. Spatial data, such as seismic or geographical datasets, must avoid overlaps between training and testing sets to prevent data leakage [29].
  4. Deep learning models require extensive training data due to their high parameter counts [28].

  Care was taken to ensure that events from the same physical or nearby locations did not appear in both the training and testing sets, minimizing data leakage and enhancing generalization.

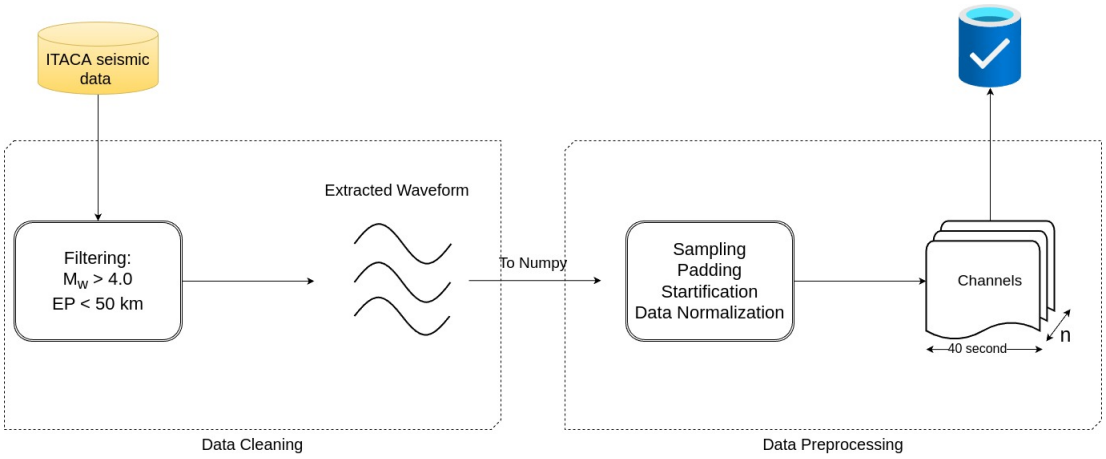


Figure 4.4: Data Pipeline

4.4 Architecture design and hyperparameter selection

4.4.1 Model Selection

In this Research, a Wasserstein GAN with Gradient Penalty (WGAN-GP) was selected for generating realistic seismic signals. WGAN-GP is a variant of WGAN proposed by Gulrajani et al. [30]. Instead of weight clipping, WGAN-GP introduces a gradient penalty term that enforces the Lipschitz constraint more effectively. The gradient penalty ensures that the gradient norm of the critic remains close to 1, improving the model's stability and addressing issues that arise from weight clipping, such as suboptimal critic capacity.

The loss function with the gradient penalty is given by:

$$L = \mathbb{E}_{x \sim P_{\text{real}}}[D(x)] - \mathbb{E}_{z \sim P_z}[D(G(z))] + \lambda \mathbb{E}_{\hat{x} \sim P_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2], \tag{4.1}$$

where $\hat{x}$ is sampled uniformly along straight lines between points from the real and generated data distributions, and λ is a regularization parameter.

The choice was motivated by WGAN's stability and robustness, particularly in the context of high-variability seismic signals from near-epicenter events.

4.4.2 Model Architecture

Since ITACA archive database includes only positive earthquake events, the adopted architecture of the SeismoGen has to modified to only able to generate positive events

4.4.3 Hyperparameter Selection

The selection of hyperparameters is crucial for the successful training of the WGAN model for seismic signal generation. Key hyperparameters such as latent space dimension, batch size, and learning rate were carefully chosen and adjusted based on the unique characteristics of near-epicenter seismic data and the high sampling rate of the dataset.

Latent Space Dimension

The latent space dimension determines the complexity of the features that the generator can learn to produce. For seismic signals, capturing intricate waveform patterns is essential. In comparison to [18], the ITACA dataset is sampled at 200 Hz vs. 40 Hz for the later, means that the new dataset captures more detailed temporal variations. So for the same time span now has 5x the data points, potentially introducing finer-grained patterns that the GAN model needs to learn.

As a result, a higher latent dimension is required to generate realistic data. The latent space must have sufficient capacity to encode the additional complexities, capturing more detailed structures, patterns, and variations effectively.

In studies [31] and [32] suggest that for a rough proportionality, the latent dimension should be proportional to the square root of the data size, as this scaling helps the latent space encode the data's complexities more effectively.

$$\text{New Latent Dimension} \approx \text{Old Latent Dimension} \times \sqrt{\frac{\text{New Data Size}}{\text{Old Data Size}}}$$

Using this:

$$\text{New Latent Dimension} \approx 400 \times \sqrt{\frac{8000}{1600}} = 400 \times \sqrt{5} \approx 894$$

Since increasing the latent dimension directly impacts memory usage and training time -and if computational resources are constrained- another method to mitigate the dataset complexity issue is to stretch the computations layers in the generator. Downsampling the data back to a lower frequency is another option to reduce computational complexity and may allow the original model to perform well.

Following are the steps followed to determine the optimal latent dimension:

1. **Initially, a latent space dimension of 100 was chosen**, which is typical for many GAN applications [33].

2. The model is tested with the current dimension of 100 and observed the quality of generated samples.

3. the key features and diversity of the new dataset is evaluated

4. The latent dimension is increased from 100 $\rightarrow$ 400 $\rightarrow$ 800 $\rightarrow$ 1000 and improvements are monitored in performance metrics

the optimal value is found to be 800 (as predicted by [31] and [32]). This larger latent space allowed the generator to better capture subtle features in the seismic data, resulting in more realistic generated signals.

Batch Size

The batch size influences the stability and speed of GAN training. For seismic data with a high sampling rate, the batch size needs to be balanced to ensure efficient training without overwhelming the memory capacity of the hardware. **A batch size of 128 (maximum allowed by our hardware cabalities) was initially used, but due to the high variability in near-epicenter data, the batch size was reduced to 16**. This adjustment helped in achieving more stable training, as smaller batches provided more granular updates, allowing the model to adapt better to the diverse characteristics of the seismic signals [34].

Gradient Penalty Coefficient (λ)

For WGAN-GP, the gradient penalty coefficient λ controls the strength of the gradient penalty term, which helps enforce the Lipschitz constraint. A value of 10 was chosen based on recommendations from previous studies [30], but during experimentation, λ was adjusted to 5 to better handle the noisy nature of near-epicenter data. This adjustment helped in maintaining training stability while ensuring that the generated signals retained realistic features.

Epochs and Training Duration

The number of training epochs was set to 500, which allowed sufficient time for the model to learn the complex temporal patterns of seismic signals. Given the high sampling rate (e.g., 200 Hz), each epoch required significant computational resources, but this duration was necessary to ensure the model's robustness and the quality of the generated outputs [5].

Optimizer Choice

The Adam optimizer was used for both the generator and critic. It is parameterized by two ***momentum coefficients,*** β_1 ***and*** β_2, which control the exponential decay rates for the first and second moments of the gradients, respectively. The β_1 parameter governs how much of the past gradient information is retained, while β_2 controls the decay of the squared gradient

(variance) information. In the context of WGAN-GP, the following settings are typically used [35]:

- $\beta_1 = 0.5$: This reduces the contribution of the past gradients, minimizing oscillations and improving stability in GAN training.
- $\beta_2 = 0.9$: This ensures a balance between adaptiveness and stability when updating weights.

The learning rate *(lr)* is also critical. For WGAN, the learning rate of both the generator and the critic must be carefully balanced to avoid one network overpowering the other. A learning rate of 0.0001 was used for both the generator and critic initially [30]. However, due to the complexity of seismic signals and the need for stable convergence, the learning rate of the critic was decreased to 0.00005, and keeping the generator's learning rate at 0.0001. This adjustment ensured that the critic could provide stronger feedback to the generator, improving the quality of generated seismic signals without causing instability. In summary, the relationship between dataset size, complexity, and latent dimension isn't linear, so iterative experimentation, careful adjustments with empirical testing is the most reliable way to find the optimal configuration. The selected hyperparameters ensured stable training, improved the diversity and realism of generated seismic signals, and allowed the model to effectively capture the intricate features of real seismic events.

4.5 Performance Metrics and Evaluation

These two methods are commonly used in Evaluating GAN models are **Inception Score (IS)** and **Fréchet Inception Distance (FID)**. IS assesses the quality and diversity of generated samples by utilizing a pretrained Inception network. It computes the Kullback-Leibler divergence between the conditional label distribution $p(y|x)$ and the marginal distribution $p(y)$, encouraging confident and diverse predictions [36]. However, IS does not directly compare generated data with real data, limiting its applicability in some domains. In contrast, FID measures the similarity between real and generated data distributions by modeling their feature representations as Gaussians and comparing their means and covariances [37]. FID captures both quality and diversity of samples while addressing limitations of IS, though it assumes Gaussian-distributed features and is computationally intensive. Together, these metrics provide complementary insights into GAN performance. Even though both metrics excels at computer vision task, they are not yet verified in time series GANS tasks.

The metrics to be evaluated for our task are as follows:

1. **Visual Fidelity**: involves assessing how well the synthetic signals visually resemble real seismic waveforms. This approach is particularly well-suited for qualitative analysis based on the following aspects. The generated waveforms are visually evaluated to ensure waveform Complexity and Amplitude Consistency while maintaining P-wave and S-wave Features and Noise Characteristics.

   Visual Fidelity is easy to apply, however does not provide a quantitative score, making benchmarking and comparison challenging, so it more suitable as an early approach

2. **Statistical Consistency**: is used to evaluate GAN-generated seismic waveforms by comparing **minimum and maximum values** of generated and real data to ensure similar amplitude ranges.

   We compare the minimum ($X_{\text{real}}^{\min}$, $X_{\text{fake}}^{\min}$) and maximum ($X_{\text{real}}^{\max}$, $X_{\text{fake}}^{\max}$) values of generated (X_{fake}) and real seismic signals (X_{real}) to assess if generated signals fall within realistic amplitude ranges:

   $$X_{\text{real}}^{\min} = \min_i X_{\text{real}}(i), \quad X_{\text{fake}}^{\min} = \min_i X_{\text{fake}}(i)$$

   Statistical consistency is effective for basic validation of amplitude characteristics, providing an essential starting point for evaluating GAN-generated seismic data.

4.6 Implementation Details

4.6.1 Source Code

The complete implementation of the methods used in this thesis can be accessed via the following GitHub repository:
`https://github.com/MazinOnsa/seismic`

4.6.2 Programming Languages and Libraries Used

The project was implemented primarily in Python, leveraging a variety of deep learning and data processing libraries. The major tools and libraries used include:

- **Python 3.9:** Python served as the primary programming language due to its rich ecosystem of libraries and support for machine learning workflows.
- **PyTorch 1.12:** PyTorch was used as the core deep learning library for constructing the generator, discriminator, and the training loop.
- **PyTorch Lightning 2.0**: PyTorch Lightning was used to streamline the training loop, making it easier to handle complex GAN models like WGAN-GP, while maintaining flexibility for custom updates in the generator and discriminator training steps.
- **NumPy 1.22** NumPy was employed for numerical operations, including data preprocessing and data manipulation.
- **Matplotlib 3.5:** This library was used for plotting and visualizing the generated samples at each epoch, as well as other key metrics like loss values.
- **TensorBoard:** TensorBoard was used to track the training process, visualize the generator and discriminator losses, and compare real versus generated data distributions.
- **Scikit-learn:** Some utilities were borrowed from Scikit-learn, such as data normalization techniques.
- **Pandas 1.4.0:** Pandas was used for data handling and transformation during the extraction and preparation of the seismic dataset.

- **PyASDF:** PyASDF was used for reading and processing the seismic data stored in ASDF format.
- **HDF5** h5py was used to interface with the HDF5 file format to efficiently store and retrieve seismic datasets.
- **urllib.request:** Used to download seismic dataset files over the internet.

4.6.3 Hardware and Software Requirements

The training of WGAN-GP is computationally intensive, especially with the complex seismic signals we are modeling. The specific hardware and software requirements for the training process are outlined as follows:

- **GPU:** The training was performed on an NVIDIA RTX 3090 GPU with 24 GB of VRAM. The use of GPU acceleration significantly sped up both forward passes and backpropagation, allowing the WGAN-GP to be trained with a batch size of up to 32 without running out of memory. ***CUDA 11.3 and cuDNN 8.2*** were installed to enable GPU acceleration via NVIDIA drivers.
- **CPU:** An Intel i9 10th-generation CPU was used for data preprocessing and handling parallel computations.
- **RAM:** The implementation utilized 32 GB of RAM to handle the loading of seismic data samples and ensure smooth operations during training.

4.7 Data Loading and Preprocessing Framework

4.7.1 Overview

For the efficient management of seismic data during training and testing, a modular data pipeline was developed using PyTorch. The pipeline is structured around two main components:

1. **SeismicDataset**: A custom dataset class responsible for loading and preprocessing raw seismic data.
2. **SeismicDataModule**: A data module built with PyTorch Lightning to handle data splits and provide dataloaders for training, validation, and testing.

4.7.2 SeismicDataset Class

The `SeismicDataset` class is designed to load seismic data stored in a NumPy file, preprocess it, and make it accessible as PyTorch tensors. The key features of the class are:

Input Data Loading

The input data is loaded using `numpy.load` with memory mapping (`mmap_mode="r"`) to efficiently handle large datasets. Only the first `signal_len` samples of each seismic signal are retained, ensuring consistency in input size.

Tensors for Neural Networks

The normalized data is converted into PyTorch tensors and transposed to match the expected input shape of downstream models.

Data Access

- The `__len__` method returns the total number of samples in the dataset.
- The `__getitem__` method retrieves a single sample, normalized and formatted as a PyTorch tensor.

This class ensures that the raw seismic data is preprocessed and normalized before being fed into neural network models.

4.7.3 SeismicDataModule Class

The `SeismicDataModule` class leverages PyTorch Lightning's `LightningDataModule` to manage the data pipeline efficiently. Its functionality is as follows:

Initialization

The module accepts parameters such as file path (`npy_file`), batch size, signal length, train-validation-test split ratios, and number of worker threads for data loading. The `_setup` method is called to split the dataset and prepare it for different phases of the training pipeline.

Dataset Splitting

The input dataset is split into training, validation, and testing subsets based on the specified split ratios (default: 80%, 10%, 10%). Splits are created chronologically to avoid potential leakage between sets, maintaining temporal consistency.

DataLoaders

Separate `DataLoader` objects are created for training, validation, and testing subsets using the `_loader` helper method. The `DataLoader` configurations include shuffling for training to improve model generalization and ensuring consistent batch sizes by enabling `drop_last`.

Data Preparation

The `prepare_data` method is a placeholder for preprocessing tasks, such as downloading or preprocessing raw files, although no such operations are currently implemented.

This framework forms the backbone of the data processing pipeline, ensuring that raw seismic signals are systematically transformed into normalized inputs ready for model training and evaluation.

Chapter 5

Results

5.1 Seismic Dataset and Training Details

5.1.1 MetaData

Full Metadata available at C

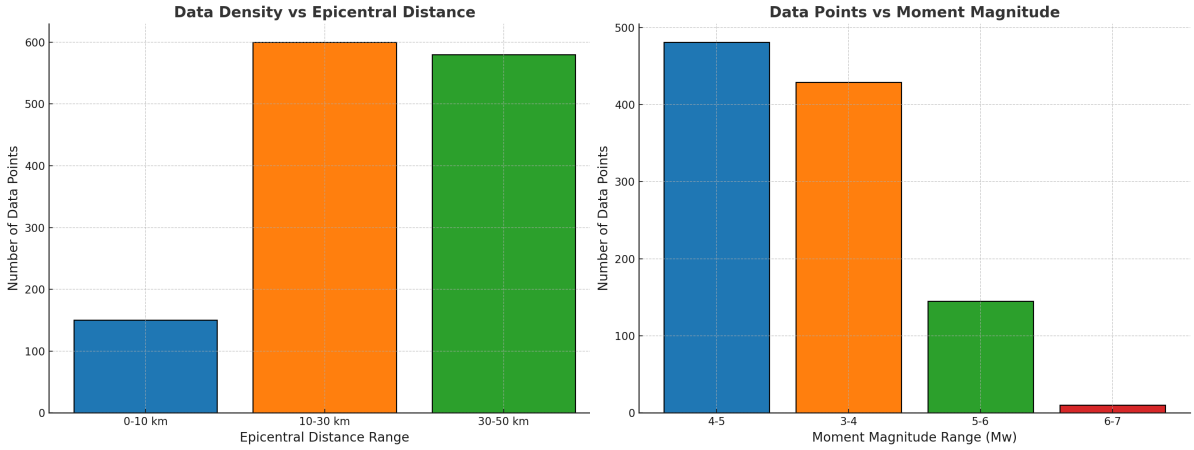


Figure 5.1: Moment Magnitude and Epicentral Distance Distribution

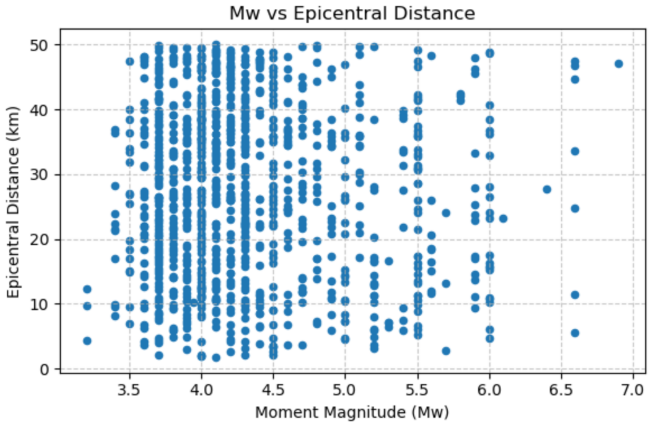


Figure 5.2: Seismic Intensity over Distance

5.2 Training Procedure

5.2.1 Training Epochs and Procedures

The WGAN-GP model was trained on segmented 40-second windows of seismic waveforms from the ITACA dataset, focusing on events with magnitudes $M_w > 4.0$ and depths < 50 km. Hyperparameter tuning was conducted to optimize the generator's ability to replicate high-frequency and nonlinear properties of near-epicenter signals.

Initially, training was conducted over 200 epochs to observe the convergence trends and test the network's robustness to hyperparameter adjustments. To optimize model performance, a total of three training runs were conducted:

- **Run 1:** Initial Training - 200 epochs with learning rate $1e^{-4}$ and batch size of 16.
- **Run 2:** Hyperparameter Tuning - 200 epochs with a reduced learning rate of $5e^{-5}$ and increased batch size of 32.
- **Run 3:** Final Run - 100 epochs, batch size set to 32, learning rate $5e^{-5}$, and λ_{gp} increased to 20.

5.2.2 Training Losses

The initial runs exhibited unstable losses, indicating difficulties with convergence and mode collapse when similar generated signals were produced continuously for multiple epochs. Visual inspection of generated samples over time was a primary method used to detect this.

To mitigate mode collapse, we experimented with various hyperparameters, as summarized in Table 5.1. The table shows how each hyperparameter adjustment impacted the reduction in mode collapse.

The losses also demonstrated that an appropriate balance between the generator and discriminator was achieved. Figure 5.3 illustrates the loss trends over the final training run, showing a convergence towards stable values after 60 epochs.

Hyperparameter	**Initial Value**	**Final Value**	**Impact on Mode Collapse**
Learning Rate	$1e^{-4}$	$5e^{-5}$	Reduced overfitting and mode collapse
Batch Size	16	32	Improved gradient stability, reduced mode collapse
λ_{gp}	10	20	Enhanced Lipschitz constraint, improved output diversity

Table 5.1: Hyperparameter Adjustments and Their Impact on Mode Collapse

5.3 Generated Signal Quality and Evaluation

The quality of the generated signals was evaluated both visually and quantitatively. Visually, we assessed the generated signals at different epochs to monitor their evolution over training (Figure 5.4). The quality of generated signals improved gradually, with later epochs showing greater resemblance to real seismic signals.

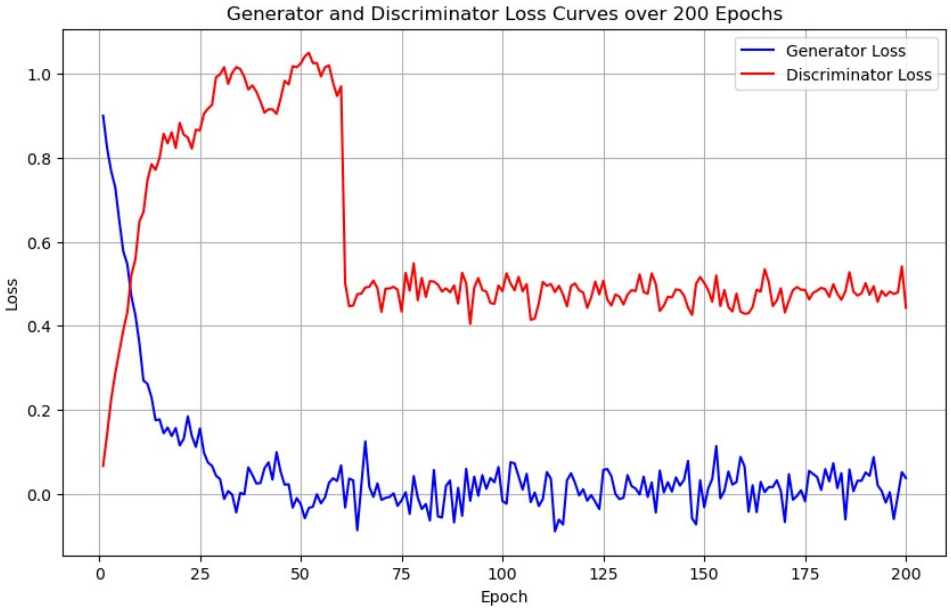


Figure 5.3: Final Generator and Discriminator Loss Curves After Hyperparameter Tuning

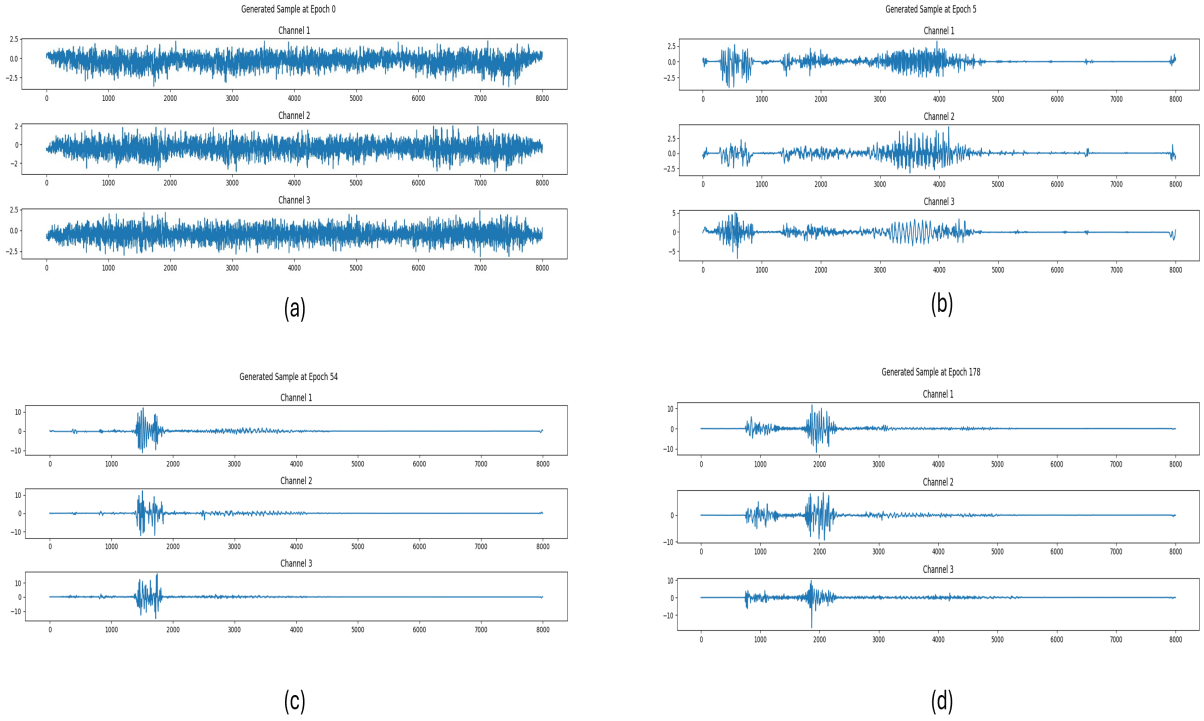


Figure 5.4: Generated Samples Across Different Epochs

5.3.1 Min-Max Value Comparisons

To quantitatively assess the generated signal quality, the minimum and maximum values of the generated signals were compared with those of real seismic signals across different epochs. The results are summarized in Table 5.2.

The convergence of the min-max values indicates the gradual improvement of the generator in producing seismic waveforms with properties that match those of real seismic events.

	Epoch	Real Data (Min)	Fake Data (Min)	Difference
Run 1	60	-4.5	-3.8	0.7
Run 2	180	-4.3	-4.2	0.1
Run 3	100	-4.4	-4.4	0.0
	Epoch	**Real Data (Max)**	**Fake Data (Max)**	**Difference**
Run 1	60	4.6	3.9	0.7
Run 2	180	4.3	4.3	0.0
Run 3	100	4.5	4.5	0.0

Table 5.2: Minimum and Maximum Values of Real and Generated Data During Training Runs

5.4 Statistical Metrics

The generated data was further evaluated using statistical metrics, such as mean, standard deviation, and skewness. Table 5.3 summarizes these metrics for real and generated data at different stages of training.

Metric	Real Data	Generated Data (Epoch 100)	Difference
Mean	0.05	0.04	0.01
Standard Deviation	2.30	2.28	0.02
Skewness	0.10	0.09	0.01

Table 5.3: Statistical Metrics Comparison Between Real and Generated Data

Chapter 6

Discussion

6.0.1 Challenges in Seismic Data Generation

The application of GANs to seismic data presents several unique challenges. Near-epicenter Seismic signals are characterized by complex temporal and spectral features, making their accurate replication through deep learning particularly demanding. Throughout the training process, common GAN-related issues such as mode collapse, instability, and convergence difficulties were encountered. In the initial training stages, the generator produced repetitive and less realistic waveforms, clearly indicating mode collapse. However, strategic modifications to the hyperparameters, especially learning rate adjustments and increasing the gradient penalty coefficient, significantly mitigated these issues, as discussed in the results chapter.

A fundamental challenge in using GANs for seismic data lies in the scarcity of diverse real data. The available dataset consisted exclusively of positive seismic events, which imposed inherent limitations on the model's ability to generalize. Expanding the dataset with synthetic noise and non-seismic signals could enhance the discriminator's robustness, helping the model better understand what constitutes a "real" seismic event and thereby improving its ability to generate high-quality signals.

6.0.2 Effectiveness of Mode Collapse Mitigation

Mode collapse is one of the most pressing challenges when using GANs, particularly for data as intricate as seismic signals. The visual inspection of generated samples across epochs initially revealed signs of mode collapse, with the generator frequently producing very similar outputs. To address this, several changes were made to the training regime. Increasing the λ_{gp} value to 20 amplified the Lipschitz continuity enforcement, resulting in greater stability for the discriminator and pushing the generator towards a wider variety of outputs.

Reducing the learning rate to 5×10^{-5} also played a key role in mitigating mode collapse. It allowed the model to make finer adjustments during training, preventing erratic behaviors often associated with large learning rates. The impact of these changes was evident in the generated signals, which began to exhibit more variability and a closer resemblance to the real seismic events as training progressed.

6.0.3 Quality and Realism of Generated Seismic Signals

The quality of the generated seismic signals gradually improved as the training evolved, becoming increasingly similar to real seismic events. This improvement was particularly apparent through visual inspection of the generated signals, as well as through quantitative metrics, including comparisons of the minimum and maximum values of generated and real signals.

The convergence of these min-max values suggests that the generator effectively learned the statistical distribution of the real seismic events.

The application of statistical metrics such as mean, standard deviation, and skewness further confirmed the convergence of generated signals towards the properties of real seismic data. These metrics showed that by epoch 100, the generated data's statistical features closely matched those of the real data, reflecting a substantial improvement in the generator's performance.

6.0.4 WGAN-GP Architecture: Benefits and Limitations

The use of the WGAN-GP architecture [30] was instrumental in overcoming several typical GAN training challenges. The gradient penalty significantly stabilized the training, providing smooth gradients that were crucial for the generator to effectively learn the nuances of seismic data. Enforcing the Lipschitz constraint reduced the tendency of the discriminator to become overly confident, thereby promoting a more balanced training process.

However, the WGAN-GP framework also introduced computational overhead. Calculating the gradient penalty is resource-intensive, and as the batch size and gradient penalty coefficient increased, so did the computational requirements. This led to challenges related to memory limitations and training time, particularly when conducting hyperparameter tuning experiments.

Despite these limitations, the WGAN-GP framework provided several key benefits over traditional GAN models, including more stable training dynamics, reduced likelihood of mode collapse, and enhanced output diversity.

6.0.5 Comparison with Real Data: Key Observations

The detailed comparison between real and generated data highlighted several key insights. As discussed in the results, the generated signals closely matched real seismic data in both time and frequency domains after effective hyperparameter tuning. The generated signals' amplitude, temporal structure, and spectral components progressively converged to those of real data, as evidenced by both visual inspection and quantitative measures.

One particularly encouraging outcome was the reduction in the discrepancy between the minimum and maximum values of real and generated data across epochs. This demonstrated that the generator was not only learning to mimic the shape and pattern of the real data but also capturing the appropriate range of values—a crucial aspect of realistic seismic waveform generation.

6.1 Implications and Potential Applications

The ability of GANs, specifically WGAN-GP, to generate realistic seismic signals opens new possibilities in geophysical research. Seismic data is often limited by the availability of real seismic events, especially in regions with fewer monitoring stations. GAN-generated data can supplement existing datasets, potentially aiding in tasks such as improving the resolution of tomographic models, training machine learning models for seismic event detection, and conducting hazard assessments.

However, the generated seismic signals must be validated rigorously before use in scientific applications. The introduction of synthetic noise, inclusion of more diverse seismic events, and the use of advanced evaluation metrics could help further refine the quality of generated signals and establish their reliability for research purposes.

6.2 Future Work

While this study demonstrates the potential of using GANs to generate seismic data, several areas for future exploration are evident:

- **Integration of Negative Data:** One of the primary limitations was the absence of negative samples (e.g., noise or non-seismic signals). Introducing such data would enhance the discriminator's understanding of what constitutes a "real" event, making it more robust.
- **Advanced Evaluation Metrics:** Future work could incorporate advanced metrics for signal quality assessment, such as correlation with real seismic events, evaluation of waveform similarity using Dynamic Time Warping (DTW), and more detailed frequency domain analyses. This would provide a more quantitative basis for evaluating model performance.
- **Transfer Learning and Pre-trained Models:** Transfer learning could be explored to make the GAN more efficient. Pre-training the discriminator on synthetic noise or non-seismic data could allow it to develop a stronger understanding of what constitutes a "real" seismic event from the outset.
- **Scaling the Model and Hyperparameter Optimization:** A broader hyperparameter search, perhaps using techniques such as Bayesian optimization, could help identify configurations that further improve the model's stability and output quality. Moreover, scaling up the model using larger architectures or ensemble learning could enhance its ability to learn the complexity of seismic signals.
- **Real-World Applications and Validation:** Generated signals could be validated through real-world applications, such as training a seismic event detection model or testing their use in inversion processes. Such validation would establish the practical usability of GAN-generated seismic data.

Chapter 7

Conclusion

In conclusion, this study successfully demonstrates the viability of using WGAN-GP for generating realistic seismic signals. Through careful hyperparameter tuning and the use of advanced GAN techniques, mode collapse was significantly mitigated, and generated signals began to closely resemble real seismic events. Nevertheless, further work is needed to enhance the diversity and accuracy of these generated signals, particularly by introducing noise data and employing more robust evaluation techniques.

While computational challenges remain, the potential applications of GANs in geophysical research are promising. With further advancements in both GAN architectures and training methodologies, generative models could become a valuable tool in the seismic researcher's toolkit, offering new opportunities for data augmentation, model training, and understanding of seismic processes.

Throughout the training process, we addressed common challenges inherent in GAN architectures, such as mode collapse and training instability. Through meticulous hyperparameter tuning—including adjustments to the learning rate, batch size, and gradient penalty coefficient—we achieved a stable training regime that produced high-quality synthetic signals.

Quantitative and qualitative evaluations showed that the generated signals closely resemble real seismic data in terms of amplitude range, statistical properties, and spectral content. The convergence of minimum and maximum values between real and generated data across training epochs indicates that the GAN effectively learned the underlying distribution of the seismic signals.

The implications of this work are significant for the field of seismology and earthquake engineering. By augmenting limited datasets with realistic synthetic data, we can enhance the robustness and generalization capabilities of machine learning models used for seismic event detection and classification. This is particularly valuable for regions with sparse seismic instrumentation or for improving early warning systems where rapid and accurate detection is crucial.

In summary, this thesis contributes to the advancement of data augmentation techniques in seismology by showcasing the potential of GANs to generate high-fidelity synthetic seismic data. The findings open avenues for integrating deep learning models more effectively in seismic data analysis and highlight the transformative potential of synthetic data in overcoming limitations posed by scarce labeled datasets

Appendix A

Source Code

A.1 Dataset and Data Module

A.1.1 Dataset Class: `SeismicDataset`

Purpose: Loads seismic data from a `.npy` file and normalizes each signal by its mean and standard deviation.

- **Attributes:**
  - `data`: The loaded seismic dataset, truncated to a specified signal length (`signal_len`).
  - `mean, std`: Per-sample mean and standard deviation used for normalization.
- **Key Methods:**
  - `__len__()`: Returns the total number of samples in the dataset.
  - `__getitem__(idx)`: Fetches and normalizes a sample by index. The normalized signal is transposed to match PyTorch conventions (`[channels, signal_length]`).

A.1.2 Data Module: `SeismicDataModule`

Purpose: Splits the dataset into training, validation, and test sets, and provides data loaders.

- **Attributes:**
  - `npy_file`: Path to the seismic data file.
  - `train_val_test_split`: Tuple defining the proportions for splitting the dataset.
  - `batch_size, num_workers`: Parameters for the data loader.
- **Key Methods:**
  - `_setup()`: Splits the dataset into train, validation, and test sets based on provided ratios.
  - `train_dataloader(), val_dataloader(), test_dataloader()`: Return respective PyTorch data loaders.

A.2 Model Architecture

A.2.1 Generator Class

Purpose: Generates synthetic seismic data from latent vectors.

- **Key Components:**
  - `y_block`: Encodes label information into a latent vector, processed via a sequence of linear layers and upsampling.
  - `g_blocks`: A sequence of `GenBlock` modules, each processing concatenated noise and label features into output data channels.
- **Key Method:**
  - `forward(z, y)`: Combines latent noise (`z`) and label data (`y`) to produce synthetic seismic signals.

A.2.2 GenBlock Class

Purpose: Intermediate processing unit within the Generator.

- **Key Components:**
  - `z_block`: Processes noise using transposed convolutions, upsampling, and normalization.
  - `conv1_block, conv2_block, conv3_block`: Apply convolutional layers, upsampling, and activation functions to refine the signal.
- **Key Method:**
  - `forward(z, processed_y)`: Combines noise and label data to create intermediate features, refined through sequential convolutional blocks.

A.2.3 Discriminator Class

Purpose: Distinguishes real seismic signals from those generated by the Generator.

- **Key Components:**
  - `x_low_block, x_high_block`: Extract features from low- and high-frequency components of the input.
  - `y_linear, y_conv`: Encode label information into a feature space.
  - `sample_critic`: Final convolutional block for decision-making.
  - `decomposer()`: Splits input signals into low- and high-frequency components using Fourier Transform and a threshold mask.
- **Key Method:**
  - `forward(x, y)`: Combines signal features and label embeddings to output a scalar prediction indicating "realness."

A.3 Training Workflow

A.3.1 `WGAN_GP_SBS` Class

Purpose: Implements the WGAN-GP (Wasserstein GAN with Gradient Penalty) framework.

- **Key Attributes:**
  - `generator, discriminator`: The GAN components.
  - `lambda_gp`: Weight for gradient penalty regularization.
  - `lr`: Learning rate for optimizers.
- **Key Methods:**
  - `training_step(batch, batch_idx)`: Alternates between training the Generator and Discriminator.
  - `gradient_penalty(real_data, fake_data)`: Computes the gradient penalty to enforce Lipschitz continuity.
  - `validation_step(batch, batch_idx)`: Evaluates Generator performance and logs generated samples.
  - `configure_optimizers()`: Configures separate Adam optimizers for the Generator and Discriminator.

A.4 Critical Snippets (Pseudocode Style)

A.4.1 Generator Forward Pass

```
def forward(z, y):
    y_encoded = process_label(y)  # Encode label information
    z_processed = process_noise(z)  # Transform latent noise
    features = combine_features(y_encoded, z_processed)  # Concatenate features
    output = refine_features(features)  # Sequential convolutions
    return output
```

A.4.2 Discriminator Forward Pass

```
def forward(x, y):
    x_low, x_high = decompose_signal(x)  # Fourier decomposition
    x_low_features = extract_low_freq_features(x_low)
    x_high_features = extract_high_freq_features(x_high)
    y_features = encode_label(y)
    combined_features = concat([x_low_features, x_high_features, y_features])
    decision = evaluate_critic(combined_features)
    return decision
```

A.4.3 WGAN-GP Training Step

```
def training_step(batch):
    real_data = load_real_data(batch)
    fake_data = generator(latent_noise, labels)

    # Train Discriminator
    real_pred = discriminator(real_data, labels)
    fake_pred = discriminator(fake_data, labels)
    d_loss = fake_pred.mean() - real_pred.mean() + lambda_gp * gradient_penalty(real_data, f

    # Train Generator
    fake_pred = discriminator(fake_data, labels)
    g_loss = -fake_pred.mean()

    return d_loss, g_loss
```

Appendix B

Relevant Flatfile Metadata

Field	Description
EVENT_NAME	Event name
EVENT_ID	Code to identify the seismic event in ITACA
EVENT_LATITUDE_DEGREE	Event latitude in decimal degrees
EVENT_LONGITUDE_DEGREE	Event longitude in decimal degrees
EVENT_DEPTH_KM	Hypocenter depth in km
Ml	Local magnitude
Mw	Moment magnitude
NETWORK	Code associated with the recording network according to the International Federation of Seismograph Networks
STATION_CODE	Code identifying the station in ITACA
STATION_NAME	name of the recording station
STATION_LATITUDE_DEGREE	Station latitude decimal degrees
STATION_LONGITUDE_DEGREE	Station longitude [decimal degrees]
epi_dist_km	Epicentral distance in kilometers
SAMPLING_INTERVAL_S	Sampling interval [s]
NDATA	Number of samples
dur_s	Duration (seconds) between the points of 5% and 95% of the total energy, according to Trifunac and Brady (1975).
component	Waveform components: E or 1 (1st horizontal), N or 2 (2nd horizontal), Z (vertical). Orientation codes: ZNE (Vertical, North-South, East-West) or Z23 (orthogonal components with nontraditional orientations).
filename	Waveform ID in the ITACA database and corresponding ASCII/HDF5 file names.

Table B.1: Relevant ITACA Database Fields and Descriptions

Table B.2: Summary of Limitations in Traditional Numerical Methods for Seismic Synthetic Data Generation

Method	**Limitations**	**Reference**
Finite Difference Method (FDM)	High computational cost, simplified earth models, limited high-frequency accuracy	Moczo et al., 2014
Finite Element Method (FEM)	Requires significant memory, challenges in handling complex geometries	Komatitsch & Vilotte, 1998
Spectral Element Method (SEM)	Computationally expensive, especially for large models, limited by model resolution	Komatitsch & Tromp, 1999
Kinematic Rupture Models	Simplifies rupture dynamics, does not capture full complexity of fault interactions	Graves & Pitarka, 2010

Table B.3: SeismoGen Classifier performance with and without data augmentation.

Dataset	Accuracy	Precision	Recall	F1 Score
Real Only	91.6%	90.0%	87.4%	88.7%
Real + Synthetic	94.8%	94.5%	92.7%	93.5%

Appendix C

C.1 Optimization Algorithms

C.1.1 Adam Optimizer

The Adam optimizer (Adaptive Moment Estimation) is an optimization algorithm designed for training deep learning models. It combines the benefits of AdaGrad (adaptive learning rates) and RMSProp (momentum), making it robust and efficient.

The Adam optimizer updates parameters θ as follows:

1. **Initialization:**
$$m_0 = 0, \quad v_0 = 0, \quad t = 0$$

2. **Gradient Computation:** Compute the gradient of the objective function:
$$g_t = \nabla_\theta f(\theta_t)$$

3. **Update Biased Moments:**
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

4. **Bias-Corrected Moments:**
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

5. **Parameter Update:**
$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Hyperparameters

- $\beta_1 = 0.9$: Decay rate for the first moment estimate.
- $\beta_2 = 0.999$: Decay rate for the second moment estimate.
- $\eta = 0.001$: Learning rate.

- $\epsilon = 10^{-8}$: Small constant to prevent division by zero.

The Adam optimizer is widely used for its efficiency in handling sparse gradients and for its robust performance across a variety of machine learning tasks.

C.2 GAN Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure C.1: Classic GAN Algorithm

Bibliography

[1] Keiiti Aki and Paul G Richards. *Quantitative Seismology*. University Science Books, 2002. ISBN: 978-0-716-71058-5 (cit. on pp. 6–8).

[2] Peter M Shearer. *Introduction to Seismology*. Cambridge University Press, 2009. ISBN: 978-0-521-14962-4 (cit. on p. 7).

[3] Christopher M Bishop. *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006. ISBN: 978-0-387-31073-2 (cit. on pp. 10, 36).

[4] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012 (cit. on p. 11).

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016 (cit. on pp. 11, 36, 38).

[6] Diederik P Kingma and Max Welling. ≪Auto-Encoding Variational Bayes≫. In: *International Conference on Learning Representations (ICLR)*. 2014 (cit. on p. 11).

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. ≪Generative Adversarial Nets≫. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014, pp. 2672–2680 (cit. on p. 11).

[8] Antreas Antoniou, Amos Storkey, and Harrison Edwards. ≪Data Augmentation Generative Adversarial Networks≫. In: *arXiv preprint arXiv:1711.04340* (2018) (cit. on p. 11).

[9] George Em Karniadakis et al. ≪Physics-Informed Machine Learning≫. In: *Nature Reviews Physics* 3 (2021), pp. 422–440 (cit. on p. 11).

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661 (cit. on p. 11).

[11] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. *Improved Techniques for Training GANs*. 2016. DOI: 10.48550/ARXIV.1606.03498. URL: https://arxiv.org/abs/1606.03498 (cit. on p. 12).

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. ≪Image-to-Image Translation with Conditional Adversarial Networks≫. In: *CVPR* (2017) (cit. on p. 12).

[13] Jonathan Ho and Stefano Ermon. *Generative Adversarial Imitation Learning*. 2016. DOI: 10.48550/ARXIV.1606.03476. URL: https://arxiv.org/abs/1606.03476 (cit. on p. 12).

[14] Dimitri Komatitsch and Jeroen Tromp. ≪Introduction to the Spectral Element Method for three-dimensional seismic wave propagation≫. In: *Geophysical Journal International* 139 (Feb. 2002), pp. 806–822. DOI: 10.1046/j.1365-246x.1999.00967.x (cit. on p. 14).

[15] Dimitri Komatitsch and Jean-Pierre Vilotte. ≪The Spectral Element method: an efficient tool to simulate the seismic response of 2D and 3D geological structures≫. In: *Bulletin of the Seismological Society of America* 88 (Apr. 1998), pp. 368–392. DOI: 10.1785/BSSA0880020368 (cit. on p. 14).

[16] Robert Graves and Arben Pitarka. ≪Refinements to the Graves and Pitarka (2010) Broadband Ground-Motion Simulation Method≫. In: *Seismological Research Letters* 86 (Jan. 2015). DOI: 10.1785/0220140101 (cit. on p. 14).

[17] Yuanming Li, Bonhwa Ku, Gwantae Kim, Jae-Kwang Ahn, and Hanseok Ko. ≪Seismic Signal Synthesis by Generative Adversarial Network with Gated Convolutional Neural Network Structure≫. In: *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. 2020, pp. 3857–3860. DOI: 10.1109/IGARSS39084.2020.9323670 (cit. on p. 15).

[18] Tiantong Wang, Daniel Trugman, and Youzuo Lin. ≪SeismoGen: Seismic Waveform Synthesis Using Generative Adversarial Networks≫. In: *arXiv preprint arXiv:1911.03966* (2019) (cit. on pp. 15, 37).

[19] Lillian J. Ratliff, Samuel A. Burden, and S. Shankar Sastry. ≪Characterization and computation of local Nash equilibria in continuous games≫. In: *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2013, pp. 917–924. DOI: 10.1109/Allerton.2013.6736623 (cit. on p. 22).

[20] Alec Radford, Luke Metz, and Soumith Chintala. ≪Unsupervised representation learning with deep convolutional generative adversarial networks≫. In: *arXiv preprint arXiv:1511.06434* (2015) (cit. on p. 24).

[21] Mehdi Mirza and Simon Osindero. ≪Conditional generative adversarial nets≫. In: *arXiv preprint arXiv:1411.1784* (2014) (cit. on p. 24).

[22] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. ≪Unpaired image-to-image translation using cycle-consistent adversarial networks≫. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232 (cit. on p. 24).

[23] Martin Arjovsky, Soumith Chintala, and Léon Bottou. ≪Wasserstein GAN≫. In: *arXiv preprint arXiv:1701.07875* (2017) (cit. on p. 24).

[24] Gili Lifshitz Sherzer, Alon Urlainis, Shani Moyal, and Igal M. Shohet. ≪Seismic Resilience in Critical Infrastructures: A Power Station Preparedness Case Study≫. In: *Applied Sciences* 14.9 (2024). ISSN: 2076-3417. DOI: 10.3390/app14093835. URL: https://www.mdpi.com/2076-3417/14/9/3835 (cit. on p. 26).

[25] Ebru Civelekler, Kamil B. Afacan, and D. Volkan Okur. ≪Effect of site specific soil characteristics on the nonlinear ground response analysis and comparison of the results with equivalent linear analysis≫. In: *Journal of Applied Geophysics* 220 (2024), p. 105250. ISSN: 0926-9851. DOI: https://doi.org/10.1016/j.jappgeo.2023.105250. URL: https://www.sciencedirect.com/science/article/pii/S0926985123003282 (cit. on p. 26).

[26] Emiliano Russo, Chiara Felicetta, Maria Clara D'Amico, Sara Sgobba, Giovanni Lanzano, Claudia Mascandola, Francesca Pacor, and Lucia Luzi. *ITalian ACcelerometric Archive (ITACA), version 3.2*. en. 2022. DOI: 10.13127/ITACA.3.2. URL: https://itaca.mi.ingv.it/ItacaNet_32/ (cit. on p. 31).

[27] Lion Krischer, James Smith, Wenjie Lei, Matthieu Lefebvre, Youyi Ruan, Elliott Sales de Andrade, Norbert Podhorszki, Ebru Bozdağ, and Jeroen Tromp. ≪An Adaptable Seismic Data Format≫. In: *Geophysical Journal International* 207.2 (Aug. 2016), pp. 1003–1011. ISSN: 0956-540X. DOI: 10.1093/gji/ggw319. eprint: https://academic.oup.com/gji/article-pdf/207/2/1003/8142607/ggw319.pdf. URL: https://doi.org/10.1093/gji/ggw319 (cit. on p. 32).

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. ≪Deep learning≫. In: *Nature* 521.7553 (2015), pp. 436–444 (cit. on p. 36).

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ≪ImageNet classification with deep convolutional neural networks≫. In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012, pp. 1097–1105 (cit. on p. 36).

[30] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Soumith Chintala, and Aaron Courville. ≪Improved training of Wasserstein GANs≫. In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 36, 38, 39, 48).

[31] S. Chakraborty and P. Bartlett. ≪On the Statistical Properties of Generative Adversarial Models for Low Intrinsic Data Dimension≫. In: *arXiv preprint arXiv:2401.15801* (2024). URL: https://arxiv.org/abs/2401.15801 (cit. on pp. 37, 38).

[32] X. Hu et al. ≪Complexity Matters: Rethinking the Latent Space for Generative Modeling≫. In: *arXiv preprint arXiv:2307.08283* (2023). URL: https://arxiv.org/abs/2307.08283 (cit. on pp. 37, 38).

[33] Alec Radford, Luke Metz, and Soumith Chintala. ≪Unsupervised representation learning with deep convolutional generative adversarial networks≫. In: *arXiv preprint arXiv:1511.06434* (2015) (cit. on p. 38).

[34] Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheeraj Mudigere, and Mikhail Smelyanskiy. ≪On large-batch training for deep learning: Generalization gap and sharp minima≫. In: *arXiv preprint arXiv:1609.04836* (2016) (cit. on p. 38).

[35] Diederik P Kingma and Jimmy Ba. ≪Adam: A method for stochastic optimization≫. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 39).

[36] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. ≪Improved techniques for training GANs≫. In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 39).

[37] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. ≪GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium≫. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 39).