Tunisian Republic
Ministry of Higher Education and Scientific Research
Carthage University - National Engineering School of Carthage

Submitted to

National Engineering School of Carthage

In partial fulfillment of the requirements for the degree of

National Engineering Diploma in Mechatronics

by

Soufiene Sellami

FAULTDIFF: A Conditional Diffusion Approach to Mitigate Synthetic-Real Disparities in Seismic Fault Detection

Defended on 09/10/2024 in front of the committee composed of:

Mrs. Afef Elloumi	President
Mrs. Salwa Elloumi	Examiner
Mrs. Samia Hachmi	Supervisor
Mr. Chih-Yang Lin	Mentor

A Graduation Project made at

CVIT LAB

Acknowledgments

All thanks and praises be to Allah (SWT), and may peace and blessings be upon the Prophet Muhammad (PBUH), his family, his companions, and his followers.

I would like to express my sincere gratitude to my advisor, Prof. Chih-Yang Lin, and my co-advisor, Prof. Isack Farady, for their unrelenting and unwavering support and guidance throughout the entirety of my thesis. Their understanding, continual encouragement, motivation, and intellectual support have been invaluable in shaping the success of this work.

I also wish to extend my deepest appreciation to my university advisor, Ms. Samia Hachmi, for her insightful suggestions at the very beginning of this research, which significantly contributed to focusing the thesis work and guiding its development.

To my parents and family members, I am profoundly grateful for their prayers, love, support, compassion, advice, and encouragement, which have served as a constant source of motivation throughout this journey.

Abstract

Seismic fault detection is a crucial process in geophysical exploration, playing a pivotal role in identifying subsurface discontinuities for resource extraction and risk assessment. The field has evolved significantly, transitioning from traditional manual interpretation by geological experts to advanced deep learning models. However, the limited availability of annotated seismic data presents a significant challenge for training these models. To address this, synthetic seismic data is often employed, though existing methods frequently struggle with poor generalization to real-world datasets due to discrepancies in statistical and structural properties.

This thesis introduces FaultDiff, a conditional diffusion model designed to generate high-fidelity synthetic seismic images. By conditioning the synthesis process on fault segmentation and edge maps, FaultDiff captures detailed geological features with enhanced realism. Extensive experiments demonstrate that FaultDiff outperforms state-of-the-art generative models, such as Pix2Pix and ControlNet, in key data quality metrics. Moreover, fault detection models trained on FaultDiff-generated data achieve segmentation performance comparable to those trained on field seismic data, effectively bridging the synthetic-to-real gap.

Keywords— Seismic, Fault Detection, Deep Learning, Generative, Diffusion

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AP	Average Precision.
AUC	Area Under the Curve.
BF16	Brain Float 16.
CNN	Convolutional Neural Network.
DDIM	Denoising Diffusion Implicit Model.
DDPM	Denoising Diffusion Probabilistic Model.
GAI	Generative AI.
GAN	Generative Adversarial Network.
GELU	Gaussian Error Linear Unit.
MLP	Multilayer Perceptron.
ResNet	Residual Network.
RMSprop	Root Mean Square Propagation.
SAM	Segment Anything Model.
SD	Stable Diffusion.
SGD	Stochastic Gradient Descent.
SiLU	Sigmoid Linear Unit.
SVM	Support Vector Machine.
VAE	Variational Autoencoder.

Introduction

Geological exploration and monitoring have been revolutionized by advancements in seismic interpretation techniques. These methods offer crucial insights into subterranean structures by analyzing seismic wave propagation through Earth's layers. While primarily associated with oil and gas exploration, seismic interpretation also plays a vital role in earthquake risk assessment and civil engineering project planning. A key challenge in this field is the precise identification of faults—fractures or weak zones in the Earth's crust that significantly impact seismic wave behavior and subsequent geological interpretations.

Traditionally, fault detection has relied heavily on human expertise, introducing elements of subjectivity and inconsistency. The emergence of artificial intelligence and machine learning presents promising opportunities to enhance fault detection through automation and improved accuracy. However, these advanced techniques, particularly deep learning approaches, require extensive, well-annotated datasets to function optimally. This requirement poses a significant hurdle in seismology, where obtaining labeled data is resource-intensive. As a result, many current models struggle with performance issues, often relying on synthetic datasets that may not adequately represent the intricacies of real-world seismic data.

Our research aims to tackle these challenges by developing an innovative approach to synthetic seismic image generation. We introduce FaultDiff, a novel conditional denoising diffusion model designed to create high-quality synthetic seismic images. The primary goal of FaultDiff is to generate synthetic data that closely mimics the characteristics of real seismic information, thereby providing a more reliable foundation for training fault detection algorithms.

This thesis is structured to provide a comprehensive overview of our research:

1. The opening chapter introduces the host organization and outlines the research motivations and potential contributions.
2. Chapter two provides context by exploring seismic fault detection fundamentals, including data characteristics, detection methodologies, and the role of synthetic data.
3. The third chapter offers a background on generative AI, exploring models such as VAEs, GANs, and Diffusion Models, with a focus on Diffusion as the foundation for our approach. It provides context for understanding the methods used in our solution..

4. Chapter four presents FaultDiff, our proposed method, along with experimental designs, results, and a thorough analysis of our findings.

5. The concluding section summarizes key contributions and suggests avenues for future research in this domain.

Chapter 1

Context and Aims of the Research

Introduction

In this first chapter, we will begin by introducing the host company where we conducted our research internship. We will then present the key motivations that led to the development of our proposed approach then outlining the main contributions of this thesis work.

1.1 Presentation of Host Company

This research was conducted at the CVIT (Computer Vision and Interactive Technology) Lab, situated within the Department of Mechanical Engineering at National Central University, Taiwan. The CVIT Lab, under the direction of Prof. Chih-Yang Lin, is distinguished for its cutting-edge research in computer vision, machine learning, deep learning, image processing, big data analysis, and the design of surveillance systems.

The CVIT Lab's research portfolio is extensive and diverse, supported by multiple grants from the Ministry of Science and Technology, with over 100 papers in highly-cited international conferences and journals. Current projects include the development of lightweight deep learning models with lifelong learning capability and security robustness, few-shot learning for fruit ripeness classification, automatic surface defect detection for industrial applications, and plant disease detection and classification using deep learning frameworks. These projects showcase the lab's commitment to both theoretical advancements and practical applications in the field of computer vision and machine learning.

1.2 Research Statement

1.2.1 Motivation

Recent developments in deep learning methods have demonstrated considerable promise in seismic interpretation tasks. These methods leverage large neural networks trained on substantial datasets to identify and analyze patterns within seismic images. However their effectiveness in fault segmentation has been limited. The intricacies of fault structures and their subtle manifestations in seismic images pose challenges that current deep learning models often struggle to address. Furthermore, the scarcity of annotated fault data exacerbates these difficulties. Labeling seismic faults is a labor-intensive process that requires substantial expertise and can be prohibitively time-consuming [3]. As a result, the availability of high-quality labeled data is a major bottleneck in advancing fault detection technologies.

Synthetic data has become a viable alternative to address the limitations of real data. A study conducted by Lin et al. [24] discovered that over 50% of deep learning techniques utilized synthetic data. Nonetheless, models trained on synthetic data frequently lack the ability to generalize effectively [4, 11]. This problem is caused by differences in the statistical and structural characteristics between current synthetic data and real field data. Figure 1.1 shows a comparison of these differences, highlighting the visual disparity between real and synthetic seismic images. Consequently, synthetic data may not fully represent the complexity and variability of actual seismic images, resulting in suboptimal performance when training fault detection models. To overcome these challenges, recent advancements in generative models offer promising solutions. Models such as GANs (Generative Adversarial Networks) and diffusion models have been explored for various seismic applications, including resolution enhancement, reconstruction, and inversion [25, 42, 43, 15]. GANs, for instance, have been employed to generate high-quality synthetic images by learning to match the distribution of real data; however, their training process can be complex and unstable. In contrast, diffusion models have emerged as a more effective alternative due to their simpler denoising process and superior performance in generating high-quality data [9, 34]. Despite their advantages, the application of diffusion models to generate synthetic seismic images remains unexplored.

One notable approach is demonstrated by Ferreira et al. [12], who utilized Pix2Pix a GAN-based model, to generate synthetic seismic images from sketches. While their work was innovative, it was limited by the quality of the input sketches and did not fully explore the potential of the generated data for specific seismic interpretation tasks.
This highlights a gap in current research, suggesting that diffusion models could offer a more effective means of generating high-fidelity synthetic seismic data. By leveraging diffusion models, it is possible to bridge the gap between synthetic and real seismic datasets, ultimately enhancing the performance of fault detection algorithms and contributing to more accurate and reliable seismic interpretations.

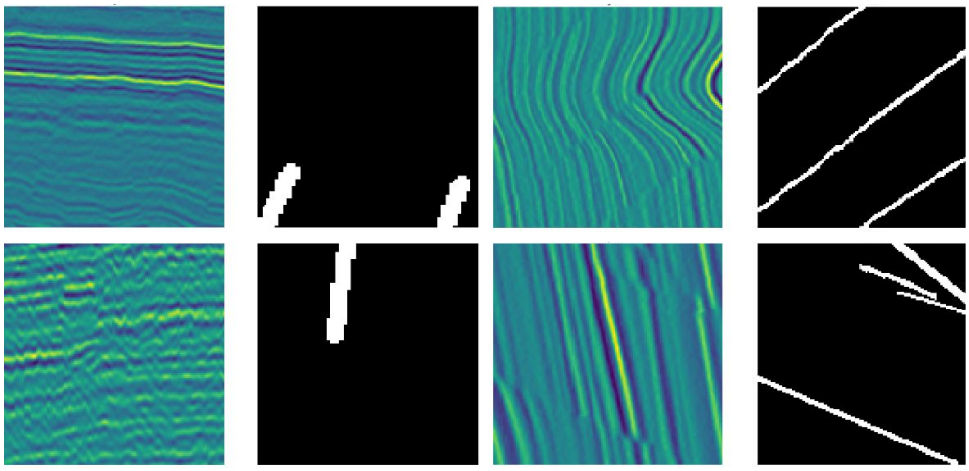

Figure 1.1: Comparison of 2D seismic real (left) and synthetic (right) data

1.2.2 Contribution

In this thesis, we introduce a novel conditional denoising diffusion model specifically designed for generating synthetic seismic data for fault detection. Our model is conditioned on fault segmentation maps and edge features extracted using Canny edge detection, enhancing its ability to capture complex seismic characteristics.

We demonstrate that our approach outperforms state-of-the-art generative models in key quality metrics for synthetic data. Additionally, we validate the fidelity of the generated seismic images to their corresponding fault masks through segmentation. Our results show that models trained on our synthetic data perform comparably to those trained on real data.

To our knowledge, this is the first application of diffusion models to seismic data synthesis, achieving high quality and fidelity. We hope this work will inspire further research and address the limitations of synthetic data in seismic interpretation.

Conclusion

This chapter has provided an overview of the context and objectives of our thesis research. We began by introducing the host company where we conducted our internship project.We then highlighted the three key motivations behind our work: the scarcity of annotated seismic fault data, the limitations of current synthetic data methods, and the potential of GAI (Generative Artificial Intelligence) techniques to address these challenges.

Finally, we summarized the core contributions of our research, focusing on the introduction of a novel conditional denoising diffusion model designed to generate high-quality synthetic seismic data. This innovative approach aims to bridge the gap between real and synthetic

data, enhancing the effectiveness of fault detection models.

In the next chapter, we will delve into the background of seismic fault detection, discussing the nature of seismic data, traditional and modern fault detection techniques, and the role and limitations of synthetic data in this field.

Chapter 2

Seismic Fault Detection

Introduction

The identification and analysis of faults in seismic data form a crucial cornerstone in the realm of geophysical exploration and interpretation. This chapter delves into the multifaceted landscape of seismic fault detection, tracing its roots in geophysics and its branches into deep learning. We embark on this exploration by unraveling the complexities of seismic data acquisition, providing readers with the essential context for understanding how faults manifest within seismic imagery. Our journey then charts the course of fault detection methodologies, from the painstaking process of manual interpretation to the advent of sophisticated computational approaches. Special emphasis is placed on the pivotal role of synthetic seismic data in the development and refinement of fault detection algorithms, shedding light on both its transformative potential and inherent limitations.

2.1 Seismic Data

The interpretation of seismic data is the final stage of a complex workflow that includes data acquisition, processing, and imaging. This workflow, shown in Figure 2.1, transforms raw seismic measurements into meaningful images that can be analyzed for geological features such as faults.

2.1.1 Seismic Data Acquisition

Seismic data acquisition is the foundation of geophysical exploration, providing the raw information needed to analyze subsurface structures. One of the most used techniques is Seismic Reflection: The process involves generating seismic waves using controlled sources, such as dynamite blasts, air guns, or vibrators, which propagate through the Earth's subsurface layers. As these waves travel, they encounter different geological formations,

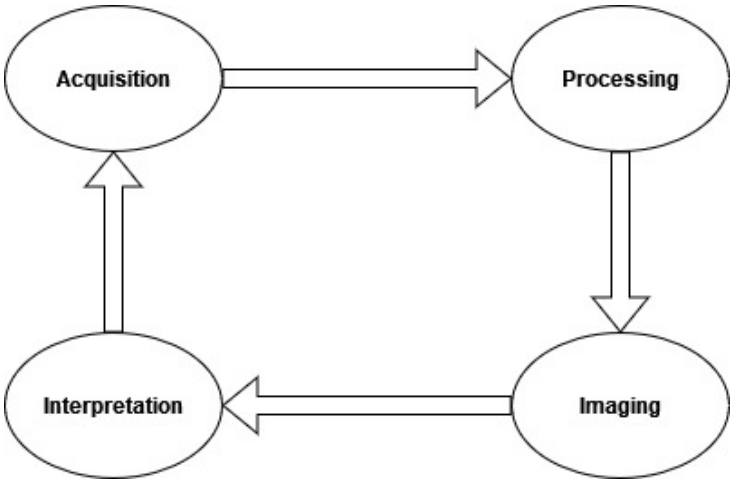

Figure 2.1: Seismic Exploration Workflow

each with its own density and elastic properties. When seismic waves hit a boundary between two layers, a portion of the wave energy is reflected back to the surface while the rest continues to propagate, as illustrated in Figure 2.2. These reflected waves are recorded by an array of sensors, known as geophones or hydrophones, distributed along the surface or ocean floor. The recorded signals are then processed to create a seismic dataset, often visualized in the form of seismic sections, time-depth profiles, or three-dimensional volumes. The quality and resolution of seismic data depend on several factors, including the type and arrangement of sources and receivers, the recording parameters, and the geological properties of the surveyed area.

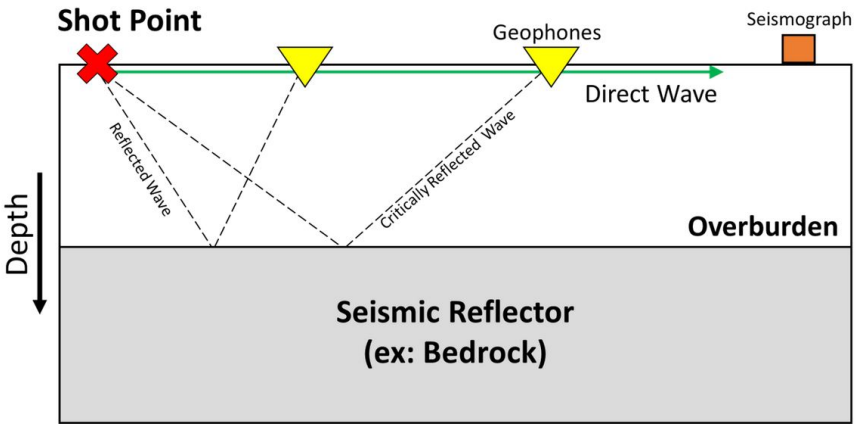

Figure 2.2: Sketch of seismic reflection

2.1.2 Seismic Processing: Denoising and Interpolation

After data acquisition, the first step is denoising, which aims to remove unwanted noise from the recorded seismic signals. Noise can originate from various sources, such as

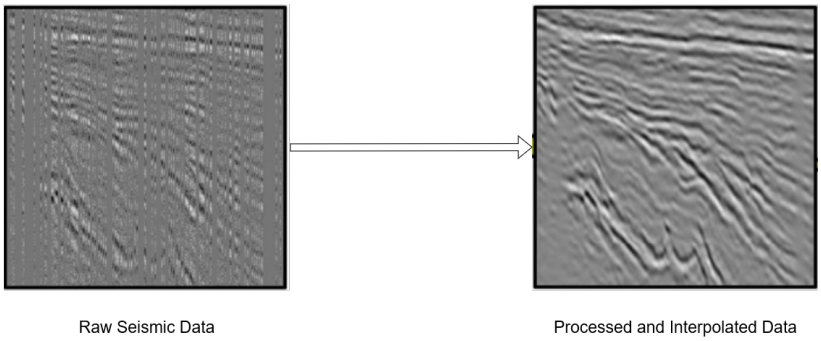

Figure 2.3: Seismic data processing

environmental disturbances, equipment limitations, or human activities. Denoising techniques, such as frequency filtering, wavelet transforms, and adaptive noise suppression are applied to enhance the signal-to-noise ratio, making the seismic reflections more distinct and interpretable.

Interpolation follows denoising, where missing or sparse data points are estimated to create a more continuous and complete seismic dataset. Seismic data often suffer from gaps due to irregularities in sensor placement or challenging terrain conditions. Interpolation techniques, such as nearest-neighbor interpolation, and multi-dimensional filtering are used to reconstruct missing traces and fill in gaps, ensuring a uniform data, as shown in Figure 2.3, for subsequent analysis.

2.1.3 Seismic Data Inversion: Migration and Imaging

Once the seismic data has been processed, the next critical step is inversion, which transforms the recorded seismic signals into a representation of the subsurface. Inversion is essential for interpreting seismic data, as it adjusts for distortions and produces images that more accurately reflect the true geometry and composition of subsurface layers.

Two key components of the inversion process are migration and imaging. Migration is the process that corrects the positions of seismic reflections to their true locations in the subsurface. When seismic waves travel through the Earth, their reflections are initially recorded at the positions of the source and receiver based on the travel time of the waves. However, these recorded positions do not necessarily correspond to the actual location where the reflections originated. Migration repositions these events to their correct locations in the subsurface, accounting for the complex geometry of the layers and the varying velocities of seismic waves through different materials. This step is crucial for producing accurate images, as it "re-locates" seismic events to their true positions where they occurred beneath the surface. Effective migration relies on an accurate estimate of the subsurface velocity to correctly reposition the reflections.

Following migration, the next step is imaging, where the corrected seismic signals are used to create high-resolution visualizations of the subsurface. The goal of imaging is to convert these migrated signals into a comprehensive 3D seismic volume as depicted in Figure

2.4, often referred to as a 3D seismic cube. This involves compiling multiple seismic traces collected across a grid of sensors into a volumetric representation of the subsurface. The imaging process allows for the construction of detailed, three-dimensional models.

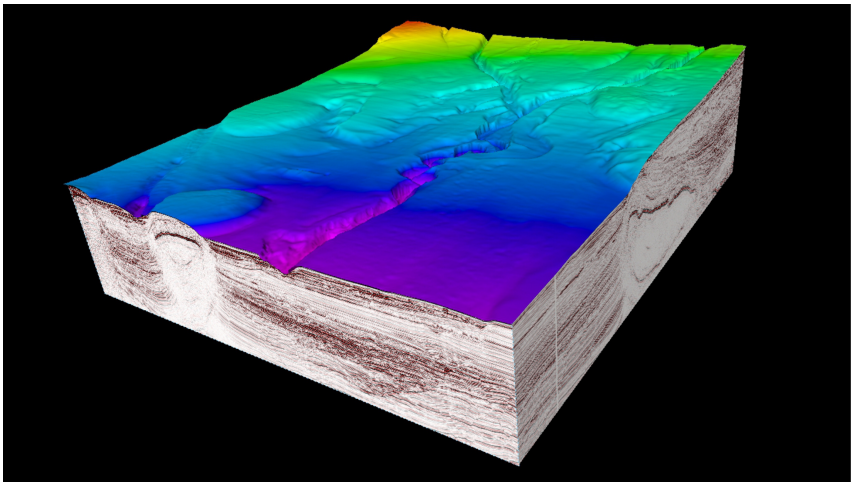

Figure 2.4: 3D seismic cube

The 3D seismic volume, or cube, generated through the imaging process provides a detailed representation of the subsurface in three spatial dimensions: inline, crossline, and depth (Z) as shown in Figure 2.5. The inline direction corresponds to the orientation of the seismic survey lines, typically running parallel to the primary survey axis. Inlines provide vertical slices through the seismic volume, offering a view along this direction. Perpendicular to this, the crossline direction captures seismic data across the survey grid, offering vertical slices at right angles to the inline direction. Together, inline and crossline sections allow for a comprehensive horizontal examination of the subsurface features. The depth (Z) direction, which represents the vertical axis of the 3D volume, extends downward from the surface into the Earth. It is often recorded in terms of two-way travel time, the time it takes for a seismic wave to travel to a reflector and back to the surface, which can be converted to depth using velocity models. This three-dimensional framework enables geologists and geophysicists to analyze the subsurface structures from multiple perspectives, facilitating a more accurate interpretation of geological features such as faults, stratigraphic boundaries, and rock formations, ultimately enhancing the understanding of subsurface complexities for exploration and development activities.

2.2 Seismic Faults

Faults are fractures or discontinuities in the Earth's crust where blocks of rock have moved relative to each other. They form due to tectonic forces such as compression, tension, or shearing, which cause stress to accumulate in the crust until it is released in the form of a fault. Faults are critical features in geology and geophysics because they can serve as conduits for fluid flow, traps for hydrocarbons, or potential sites for earthquakes.

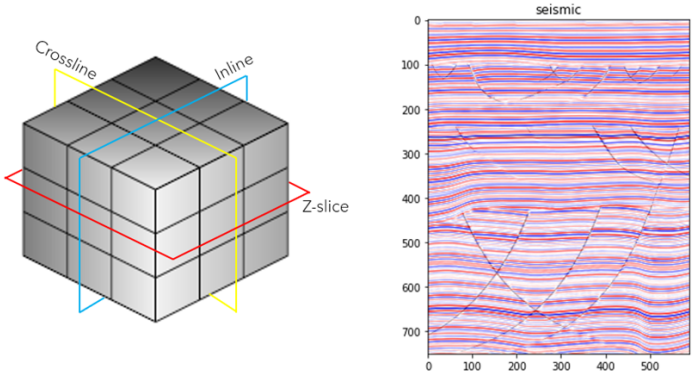

Figure 2.5: 3D visualization of seismic data: left - 3D cube representation (inline, crossline, and z-slice); right - 2D seismic section (inline slice)

In seismic data, faults manifest as sharp breaks or offsets in the continuity of seismic reflections, typically appearing as linear or curvilinear features. These disruptions result from the abrupt displacement of rock layers along the fault plane, leading to variations in the seismic signal that can be detected and interpreted by geophysicists. In reality, faults are not simply planar structures, but rather 3D rock bodies with complex internal structures and properties. Figure 2.6 shows how faults manifest both in reality and in seismic data.

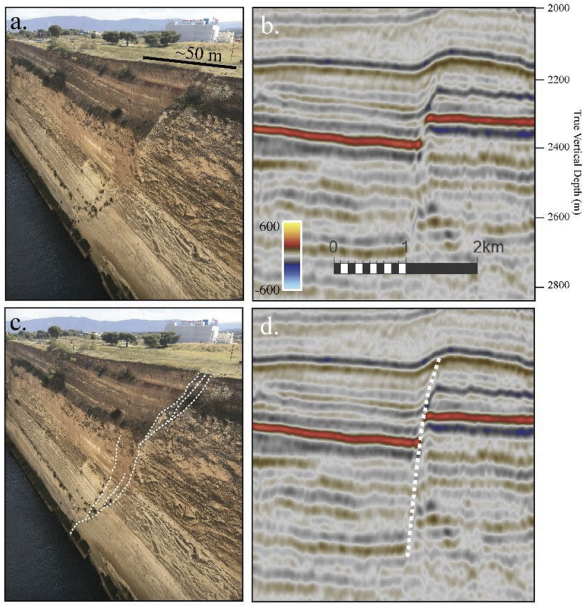

Figure 2.6: a. An image of a fault in the Corinth region of Greece; b. An image of a fault in 2D seismic data; c. and d. Images of a. and b., respectively, with fault interpretations overlaid (from [8])

2.3 Early Approaches for Seismic Fault Detection

The field of seismic fault detection has undergone a significant evolution over the past decades, transitioning from manual interpretation methods to sophisticated automated algorithms. Traditional approaches relied heavily on the expertise of skilled geoscientists who would visually inspect 2D seismic sections to identify discontinuities and offsets in reflectors. The advent of 3D seismic data in the 1980s introduced new visualization techniques such as time-slice analysis, where interpreters could examine horizontal slices through the seismic volume to identify linear features suggestive of fault planes.

The development of seismic attributes in the 1990s marked a significant advancement in fault detection capabilities. Coherence [5] quantifies the similarity of waveforms or trace shapes within a small volume of traces. Faults appear as zones of low coherence, making them more easily identifiable. The coherence attribute quickly became a standard tool in fault interpretation workflows, with subsequent refinements improving its resolution and noise tolerance. Curvature [28], which measures the structural shape of the seismic data, proved particularly effective in highlighting subtle flexures and small-scale faults that might be overlooked in conventional amplitude displays. Dip and azimuth attributes [6], representing the magnitude and direction of the plane tangent to a seismic event, provided valuable information for identifying structural features including faults. Figure 2.7 illustrates different attributes.

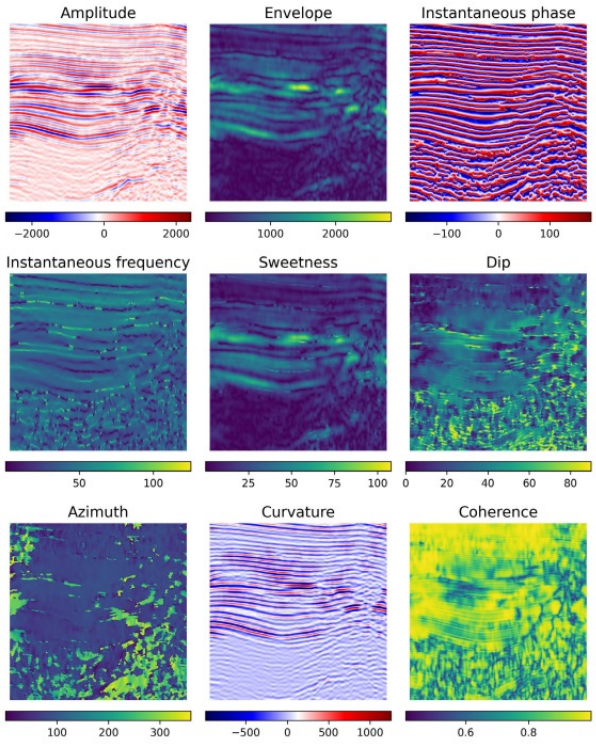

Figure 2.7: Visualisation of different seismic attributes

This progress led to the emergence of attribute-based methods: Albinhassne et al. [2] applied the Hough Transform on 2D coherence slices. Later, the Hough Transform method was extended to 3D [47]. Pedersen et al. [26] proposed the ant tracking algorithm. This approach combines multiple attributes and uses a swarm intelligence algorithm to trace fault surfaces through the data. Filtering techniques, such as the use of 3D Sobel filters[11], were also employed to enhance the detection of fault features and suppress noise in the seismic data. These attribute-based methods significantly improved fault detection efficiency and reduced subjectivity compared to purely manual interpretation. However, they often required careful parameter tuning and post-processing to achieve optimal results, and the interpretation of attribute volumes still relied heavily on human expertise.

2.4 Deep Learning for Seismic Fault Detection

2.4.1 Overview

In the nascent stages of applying machine learning to seismic fault detection, researchers primarily viewed the problem through the lens of binary classification. This approach involved training algorithms to categorize seismic data into two distinct classes: fault-present(1) or fault-absent(0). Traditional machine learning methods such as SVM (Support Vector Machines) [10] and MLP (Multi-Layer Perceptrons) [35] were the workhorses of this era. To apply this approach to entire seismic volumes, researchers employed a sliding window technique, dividing a whole image into fault and no fault patches, classifying small patches of data and combining the results to create a binary volume highlighting fault locations. While innovative, this method was computationally expensive and struggled with large datasets, often proving impractical for rapid fault detection as the size of seismic data grew.

Even as the field progressed and CNNs (Convolutional Neural Networks) entered the scene, researchers initially continued to frame the problem as a classification task [46, 40, 16]. CNNs, with their ability to automatically learn relevant features from image-like data, seemed like a natural fit for seismic imagery. However, while this approach yielded improvements over traditional methods, it still fell short in capturing the intricate spatial nature of fault structures.

The real paradigm shift came when researchers began to re-conceptualize seismic fault detection as a binary segmentation problem. This transition marked a significant leap forward in the field's approach to fault detection. Segmentation, unlike classification, aims to classify each pixel or voxel in the input data, as outlined in Figure 2.8.
This shift to segmentation opened the door to more sophisticated deep learning architectures, chief among them the U-Net. Originally developed for biomedical image segmentation, the U-Net architecture proved remarkably effective when adapted for seismic data[41]. Its structure, consisting of an encoder(contracting path) to capture context and a symmetric decoder(expanding path) for precise localization, aligned well with the

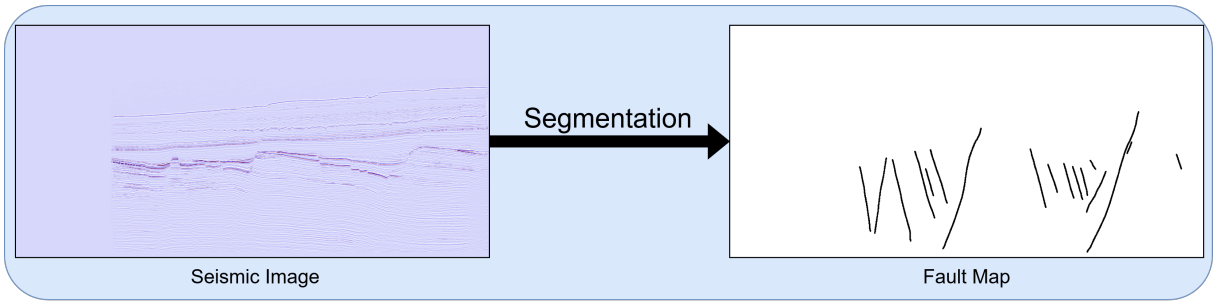

Figure 2.8: Visualization of seismic image segmentation: The left panel shows the original seismic image, while the right panel displays the corresponding binary fault map generated through segmentation

needs of seismic fault detection. As the field continued to evolve, researchers began incorporating even more advanced techniques. ViT (Vision Transformers), which had shown promise in computer vision tasks, were integrated into U-Net architectures. This hybrid approach [33, 37, 45] combined the strengths of CNNs in capturing local features with the ability of transformers to model long-range dependencies, potentially offering a more comprehensive understanding of fault structures.

The latest frontier in this evolution involves the use of foundational models. These are large-scale models pre-trained on vast and diverse datasets, which can then be fine-tuned for specific tasks. In [7] Chen et al. fine tune SAM (Segment Anything Model) using adapters technique to detect fault. This approach holds the promise of leveraging knowledge gained from broader domains to improve performance and generalization in the specialized field of fault detection.

However, as the sophistication of these deep learning approaches has grown, so too has their hunger for data. Herein lies a significant challenge in the field of seismic fault detection: the scarcity of annotated data. Deep learning models typically require large amounts of labeled data to train effectively and generalize well to new, unseen examples.

Yet in the domain of seismic fault detection, researchers face a stark reality – only two annotated field datasets are available for training and testing models, and among these, only one [4] provides the pixel-level labels crucial for training segmentation models. This data scarcity poses a formidable obstacle to the development and evaluation of deep learning models for seismic fault detection. It has spurred researchers to explore various techniques to maximize the utility of limited data, including data augmentation, transfer learning, semi-supervised learning approaches, and synthetic data generation.

2.4.2 Synthetic Data for Seismic Fault Detection

In 2018, Wu et al. [39] introduced FaultSeg3D, a 3D U-Net model trained on synthetic data for seismic fault detection. Their simulation-based approach to generate the dataset addressed the scarcity of annotated seismic data and has since become influential in the field. The FaultSeg3D dataset and model architecture have been widely adopted by

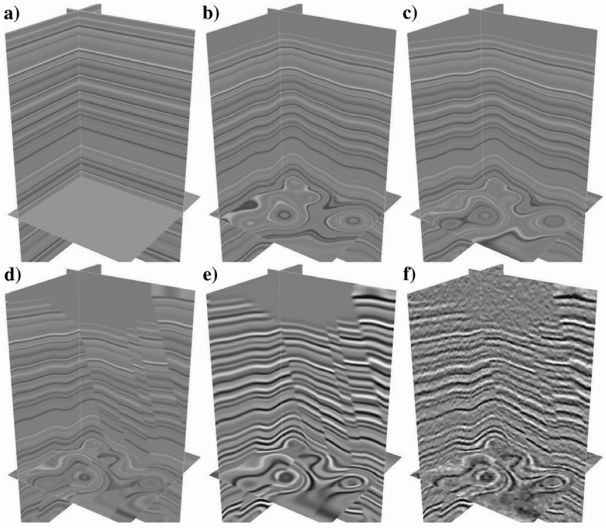

Figure 2.9: Method to create synthetic seismic data by Wu et al. [39]

researchers, demonstrating the value of synthetic data in seismic interpretation tasks.

The synthetic data generation workflow in FaultSeg3D begins with the creation of a 1D horizontal reflectivity model $r(x, y, z)$. This model is generated by assigning random values between -1 and 1 to each spatial coordinate (x, y, z), as shown in Figure 2.9.a, The reflection model is expressed by:

$$r(x, y, z) = \text{RandomValue}\,(-1, 1)\,. \tag{2.1}$$

Next, folding structures are introduced (see Figure 2.9.b) by applying a vertical shear defined by:

$$s_1(x, y, z) = a_0 + \frac{1.5z}{z_{\max}} \sum_{k=1}^{N} b_k \exp\left(-\frac{(x - c_k)^2 + (y - d_k)^2}{2\sigma_k^2}\right). \tag{2.2}$$

Here, a_0 is a constant offset, b_k are the amplitudes of the Gaussian functions, and c_k, d_k, and σ_k determine the centers and spread of these Gaussian functions, respectively. The parameter $z_{\max}$ represents the maximum depth. The function combines multiple 2D Gaussian components with a linear scaling factor, introducing a vertical attenuation of the folding effects.

For each point (x, y, z), the new depth is computed as:

$$z_{\text{new}} = z + s_1(x, y, z). \tag{2.3}$$

The reflectivity values at these new depths are interpolated using sinc interpolation :

$$r(x, y, z_{\text{new}}) = \sum_i \sum_j \sum_k r(x_i, y_j, z_k) \cdot \text{sinc}\left(\frac{x - x_i}{\Delta x}\right) \cdot \text{sinc}\left(\frac{y - y_j}{\Delta y}\right) \cdot \text{sinc}\left(\frac{z_{\text{new}} - z_k}{\Delta z}\right). \tag{2.4}$$

This results in a folded model $r_{\text{folded}}(x, y, z)$ with smooth and realistic folding features.

Further complexity is added by introducing planar shearing, shown in Figure 2.9.c, defined by:

$$s_2(x, y, z) = e_0 + fx + gy, \tag{2.5}$$

where e_0, f, and g are randomly selected parameters. Applying these planar shifts to the folded model results in a new reflectivity model:

$$r_{\text{new}}(x, y, z) = r(x, y, z) + s_2(x, y, z). \tag{2.6}$$

Faults are then incorporated into the model (Figure 2.9.d). Fault displacements are defined using either Gaussian or linear functions. For a Gaussian distribution:

$$d_{\text{Gaussian}}(x, y) = A \exp\left(-\frac{(x - x_0)^2 + (y - y_0)^2}{2\sigma^2}\right), \tag{2.7}$$

and for a linear distribution:

$$d_{\text{Linear}}(x, y) = A + B \cdot (x - x_0) + C \cdot (y - y_0). \tag{2.8}$$

Here, A represents the maximum displacement, and B and C determine the gradient. The fault displacement $d(x, y)$ is added to the vertical position of the reflectivity model to produce:

$$r_{\text{faulted}}(x, y, z) = r(x, y, z + d(x, y)). \tag{2.9}$$

The next step involves convolving the folded and faulted model with a Ricker wavelet to generate a 3D seismic image (Figure 2.9.e). The Ricker wavelet is defined by:

$$w(t) = \left(1 - \frac{t^2}{2\sigma^2}\right) \exp\left(-\frac{t^2}{2\sigma^2}\right), \tag{2.10}$$

where σ is related to the peak frequency of the wavelet.

The convolution yields the seismic image:

$$\text{Seismic Image}(x, y, z) = r_{\text{faulted}}(x, y, z) * w(t). \tag{2.11}$$

Finally, random Gaussian noise is added to simulate real conditions. The noise is modeled from a Gaussian distribution:

$$\text{Noise}(x, y, z) \sim \mathcal{N}(0, \sigma^2_{\text{noise}}), \tag{2.12}$$

and the final seismic image (Figure 2.9.f) is obtained by:

$$\text{Seismic Image}_{\text{final}}(x, y, z) = \text{Seismic Image}(x, y, z) + \text{Noise}(x, y, z). \tag{2.13}$$

This approach is efficient and memory-conserving due to its reliance on simulations. However, models trained on this synthetic data often underperform when applied to field seismic data [11]. For instance, An et al. [4] demonstrated in their study using the Thebe dataset, a large-scale field dataset with expert labeling, that models trained solely on synthetic data, such as the original FaultSeg3D, often underperform when applied to field seismic data. Their experiments showed a substantial performance gap, with an AP (Average Precision) difference of 0.33 between the model trained on synthetic data and one retrained on real data.

The observed performance disparity can be traced back to fundamental shortcomings in simulation-based methods. Primarily, synthetic data tends to overly simplify fault geometries, failing to capture the full spectrum of fault sizes, orientations, and complexities—such as irregular shapes and varied displacement patterns—typical of real field data. Additionally, the noise models used in synthetic datasets often fall short; these models typically do not encompass the diverse range of noise types and patterns present in actual seismic data, leading to a higher incidence of false positives when models trained on synthetic data are deployed in real-world scenarios. Moreover, synthetic datasets may not accurately reflect the geological variability and stratigraphic complexities found in real seismic environments, which can impair model performance. Fault detection is a notably unbalanced task, where negative non-fault points greatly outnumber positive fault points. Research by Zhang et al. [45] has shown that this imbalance is less severe in synthetic data compared to real data as detailed in Figure 2.10.

Several studies [36, 20, 38] have attempted to align synthetic data with real data by modifying the simulation process. Nevertheless, these approaches still face limitations, such as inadequate representation of complex fault shapes and imperfect noise modeling. According to a review by Lei et al. [24], over half of the studies on fault detection relied on synthetic data, underscoring the need for developing more realistic synthetic datasets to improve models performance.

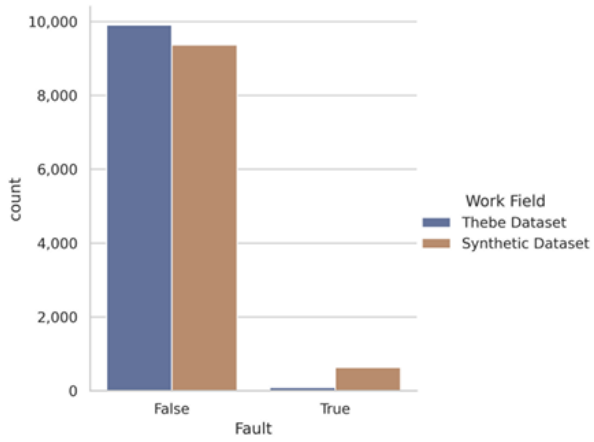

Figure 2.10: The distribution of labels in synthetic data (FaultSeg3D) and field data (Thebe)

Conclusion

This chapter has provided a comprehensive overview of seismic fault detection, exploring the complexities of seismic data acquisition and processing, and tracing the evolution of detection methods from manual interpretation to advanced deep learning techniques. We highlighted the benefits and challenges of using synthetic data, underscoring its potential to enhance seismic fault detection while noting the limitations of current approaches.

The subsequent chapter will introduce the concept of generative artificial intelligence, elucidating its foundational principles and exploring its applications in image synthesis.

Chapter 3

Generative Artificial Intelligence

Introduction

In recent years, GAI has emerged as a transformative force across various domains, from computer vision to natural language processing. This chapter explores the evolution of generative AI techniques. We begin by tracing the development of generative models, starting with VAEs (Variational Autoencoders) and GANs, before delving into the more recent and powerful Diffusion Models highlighting the resason for choosing it as the backbone for our method. The chapter then shifts focus to conditional synthesis methods, such as Pix2Pix and ControlNet. These techniques allow for more precise control over the generated outputs, making them especially relevant for our proposed approach.

3.1 GAI: From VAE to Diffusion

In the realm of machine learning and artificial intelligence, two fundamental paradigms have emerged as cornerstones of data modeling: discriminative and generative approaches. These distinct methodologies, while both aimed at extracting meaningful patterns from data, diverge significantly in their underlying principles and applications.

Discriminative models, characterized by their focus on delineating decision boundaries between classes, excel in tasks that require mapping input data to specific output labels. These models operate by learning the conditional probability distribution $P(Y|X)$, where Y represents the output and X the input. This approach has proven particularly effective in classification and regression tasks across various domains.

In contrast, generative models adopt a more holistic perspective, attempting to capture the joint probability distribution $P(X, Y)$ of both inputs and outputs. This fundamental difference endows generative models with the remarkable ability not only to classify existing data but also to synthesize new, realistic samples.

Generative models aim to learn and represent the underlying probability distribution $P(X)$ of observed data X. Once trained, generative models can synthesize new data instances X' that exhibit similar characteristics to the original data.

Generative models are broadly divided into two categories: likelihood-based models and implicit models. Though both aim to generate realistic data, they differ significantly in their underlying principles and approaches. Likelihood-based models focus on directly estimating the probability distribution governing the data. These models work by maximizing the likelihood of observed data, given the parameters of the model, requiring a defined mathematical structure for the probability distribution. The parameters are then optimized to best fit the data.

On the other hand, implicit generative models take a different approach. Rather than explicitly defining the probability distribution, they focus on the data generation process itself. The term "implicit" refers to the fact that these models do not directly model the probability distribution but instead learn a transformation from a simple, known distribution to the complex one underlying the data.

In this context, we turn our attention to three prominent generative modeling approaches: VAE, GAN, and Diffusion Models. Each of these methods embodies unique strategies for data generation and presents distinct advantages and challenges in modeling data.

3.1.1 VAE

VAE [23], introduced in 2014, are likelihood-based generative models that learn complex data distributions by encoding data into a latent space and reconstructing it. The latent space is a lower-dimensional, abstract representation of the data, where each point corresponds to a latent variable z. These latent variables are variables that we do not observe directly and are not part of the training dataset, although they are part of the model. They capture key features of the data in a compressed form. For instance, in the case of images, the latent space might encode attributes such as shape, color, or orientation. The VAE learns two probability distributions: the encoder distribution $q_\phi(z|x)$, which approximates the true posterior $p(z|x)$, and the decoder distribution $p_\theta(x|z)$, which generates data from the latent variables as shown in Figure 3.1.

The training goal of a VAE is to maximize the evidence lower bound (ELBO):

$$\text{ELBO} = \mathrm{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p(z)), \tag{3.1}$$

where:

- $\mathrm{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ represents the reconstruction loss or negative expected log-likelihood (NLL).
- $D_{\mathrm{KL}}(q_\phi(z|x)\|p(z))$ is the Kullback-Leibler divergence.

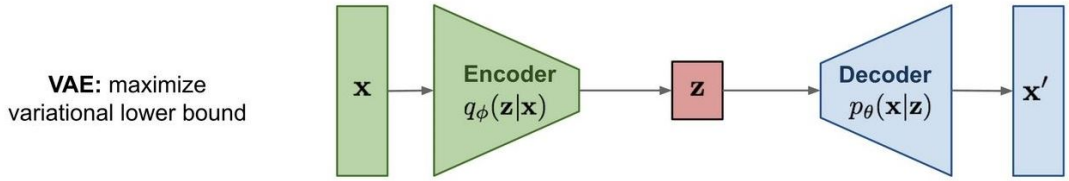

Figure 3.1: VAE architecture

The negative log-likelihood (NLL) measures how well the generative model can recreate or explain the observed data x, conditioned on the latent variable z. This can be expressed mathematically as:

$$\mathrm{E}_{q_\phi(z|x)}[-\log p_\theta(x|z)] = -\frac{1}{N}\sum_{i=1}^{N}\log p_\theta(x_i|z_i) \tag{3.2}$$

where:

- $p_\theta(x|z)$ is the likelihood of the data x given the latent variable z.
- $\mathrm{E}_{q_\phi(z|x)}[\cdot]$ represents the expectation over the approximate posterior distribution $q_\phi(z|x)$.

The reconstruction loss is minimized when the model assigns high probability to the observed data x, meaning the model can accurately reconstruct the data from the latent space. In the VAE, this term encourages the decoder to generate data that closely resembles the input. Minimizing this loss improves the quality of the generated samples.

The second term, $D_{\mathrm{KL}}(q_\phi(z|x)\|p(z))$, is the Kullback-Leibler divergence, which acts as a regularization term. It measures the divergence or difference between two probability distributions—in this case, the learned approximate posterior distribution $q_\phi(z|x)$ and the prior distribution $p(z)$.

Mathematically, the KL divergence is defined as:

$$D_{\mathrm{KL}}(q_\phi(z|x)\|p(z)) = \mathrm{E}_{q_\phi(z|x)}\left[\log\frac{q_\phi(z|x)}{p(z)}\right] \tag{3.3}$$

where:

- $q_\phi(z|x)$ is the approximate posterior, representing the distribution of latent variables z given the input data x.
- $p(z)$ is the prior distribution over the latent space, typically a standard normal distribution $\mathcal{N}(0, I)$.

The KL divergence penalizes deviations of the approximate posterior from the prior, enforcing structure on the latent space. By minimizing the KL divergence, the VAE pushes the approximate posterior closer to the prior, encouraging a smooth and continuous latent space that facilitates interpolation and generation of new data points.

Therefore, training a VAE is equivalent to solving an optimization problem: The autoencoder's loss function aims to minimize both the reconstruction loss (how closely the output matches the input) and the latent loss (how closely the latent space aligns with a normal distribution). Reducing the latent loss limits the amount of information that can be encoded, increasing the reconstruction loss. This creates a trade-off: if the latent loss is small, generated images resemble training data but may look poor in quality. If the reconstruction loss is small, the reconstructed images will look good, but newly generated images will differ significantly from the originals. This trade-off often results in VAE generating blurry images, which is a known limitation.

3.1.2 GAN

GAN, introduced later in 2014 by Goodfellow et al. [14], brought a novel implicit approach to generative modeling and quickly became influential in the field. GANs consist of two neural networks: a generator and a discriminator as shown in Figure 3.2.

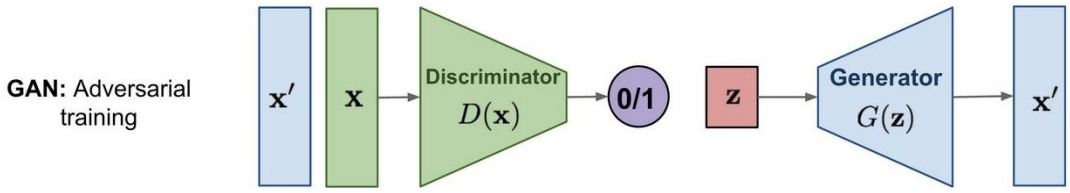

Figure 3.2: GAN architecture

The generator produces synthetic samples from random noise, aiming to capture the underlying distribution of real data. Its goal is to create outputs that are as realistic as possible, thereby fooling the discriminator into classifying these generated samples as real. Conversely, the discriminator acts as a critic, assessing whether a given sample originates from the real dataset or is a product of the generator. It is trained to distinguish between genuine and synthetic data with high accuracy.

The performance of both networks is evaluated using a loss function. The loss function for GANs is formulated as follows:

$$L(x) = E_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathrm{E}_{z\sim p(z)}[\log(1 - D(G(z)))] \tag{3.4}$$

where:

- $E_{x\sim p_{\text{data}}(x)}[\log D(x)]$: This term represents the expected value of the log probability

that the discriminator D classifies real data x as real. It encourages the discriminator to correctly identify real samples.

- $E_{z\sim p(z)}[\log(1-D(G(z)))]$: This term represents the expected value of the log probability that the discriminator D classifies synthetic data $G(z)$, generated from latent vectors z, as fake. It motivates the discriminator to correctly classify generated samples as fake.

The generator's objective is to minimize this loss, which corresponds to maximizing the discriminator's error, while the discriminator's goal is to maximize this loss to accurately differentiate between real and fake data. Thus, the training of GANs involves a two-player minimax game where the generator and discriminator are in a constant adversarial process, each improving their strategies in response to the other.

Despite their significant achievements in generating high-quality images and other types of data, GAN face challenges of slow and unstable training. One major difficulty is achieving a Nash equilibrium, a concept from game theory where each player adopts a strategy that is optimal given the strategies of others. In other words, no player can improve their outcome by changing their strategy while others maintain theirs.

Salimans et al. [31] have examined this challenge in the context of GANs, noting the limitations of the gradient-descent-based training procedure. In GANs, two models—the generator and the discriminator—are trained simultaneously to achieve a Nash equilibrium in a two-player non-cooperative game. However, the training process is flawed because each model independently updates its cost function without considering the adjustments made by the other. This independent gradient updating can lead to instability, as the models' simultaneous adjustments often prevent convergence to a stable equilibrium.

Another significant issue arises when the discriminator achieves perfect accuracy. In this case, the loss function drops to zero, resulting in a lack of gradients for further updates. This situation creates a dilemma: if the discriminator performs poorly, the generator does not receive accurate feedback, leading to a loss function that fails to represent the true quality of the generated data. Conversely, if the discriminator performs exceptionally well, the gradient becomes very small, causing extremely slow learning or even halting the training process altogether.

Lastly, GANs are also known to suffer from mode collapse, where they produce a limited variety of outputs despite the diversity of inputs [34]. This issue further complicates the training and effectiveness of GANs.

3.2 Diffusion

The next major development came with the introduction of Diffusion Models [17]. These models also fall under the likelihood-based category but use a distinct approach to data generation. Diffusion models define a corruption process that incrementally destroys

the structure in the data, and then train a model to reverse this process, progressively denoising the data step-by-step as shown in Figure 3.3.

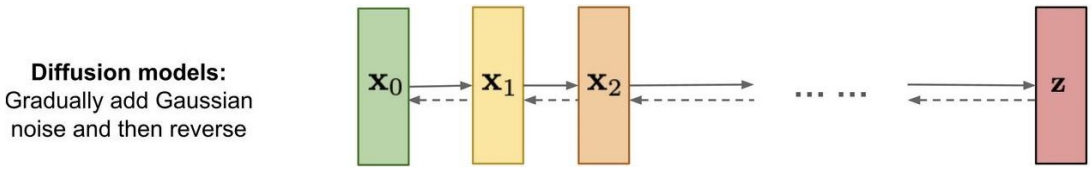

Figure 3.3: Diffusion Mechanism

Building on this foundational approach, DPPM (Denoising Diffusion Probabilistic Model) emerged [17] in 2020. DDPM learn to generate images by iteratively denoising them. This process is framed as two Markov chain processes: a forward diffusion process and a reverse process. In a Markov chain, the probability of $\mathbf{x}_t$ depends only on $\mathbf{x}_{t-1}$ and not on previous states $\mathbf{x}_{t-2}$, and so on. The advantage of a Markovian structure is that it is memoryless, meaning that if we know the state $\mathbf{x}_{t-1}$, we can determine $\mathbf{x}_t$ directly without needing information about earlier states. However, a Markov chain may require many steps to achieve convergence.

Forward Process In the forward process, random noise is incrementally added to the original image $\mathbf{x}_0$ over T steps. At each step, a small amount of noise is added according to a fixed schedule β, gradually transforming the image into pure noise $\mathbf{x}_T$. This process is defined by:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}), \tag{3.5}$$

where $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ represents the Gaussian distribution with mean $\sqrt{1-\beta_t}\,\mathbf{x}_{t-1}$ and variance $\beta_t \mathbf{I}$.

This process can be reformulated for direct sampling of $\mathbf{x}_t$ from $\mathbf{x}_0$ using a single draw of Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, \tag{3.6}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$.

Reverse Process In the reverse process, a neural network is trained to gradually denoise an image starting from pure noise. The equation for the reverse process is:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right), \sigma_t^2 \mathbf{I}\right), \tag{3.7}$$

where

- $\epsilon_\theta(\mathbf{x}_t, t)$ is the estimated noise function.
- σ_t^2 is the variance term.

The loss function used for training the neural network is:

$$L = E_{x_0,t,\epsilon} \left\| \epsilon - \epsilon_\theta(x_t, t) \right\|^2 . \tag{3.8}$$

The training algorithm is as follows:

Algorithm 1: Diffusion training

1: **Repeat**
2: $x_0 \sim q(x_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2$$

6: **Until** converged

For sampling, the reverse process being a markov chain lead to slow generation. Ideally, the number of timesteps used during sampling should be fewer than those used during training to balance between computational efficiency and output quality. Thus Song et al. [32] proposed DDIM (Denoising Diffusion Implicit Models) defining the diffusion process as a non-Markovian approach while maintaining the same forward marginals as DDPM. This modification allows for quicker sampling, with the sampling steps described by:

$$x_{t-1} = \sqrt{\alpha_{t-1}} \hat{x}_0(x_t) + \sqrt{(1 - \alpha_{t-1} - \sigma_t^2)} \epsilon_\theta(x_t, t) + \sigma_t \epsilon \tag{3.9}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, σ_t represents the variance of the noise during sampling, and $\hat{x}_0(x_t)$ is the estimated x_0 from x_t, given by:

$$\hat{x}_0(x_t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \sqrt{1 - \alpha_t} \epsilon_\theta(x_t, t) \right) + \frac{1 - \alpha_t}{\alpha_t} \nabla_{x_t} \log p(x_t). \tag{3.10}$$

When σ_t is set to 0, the sampling process becomes deterministic, which facilitates the inversion of samples from p_{data} to their corresponding latent representations. Compared

to DDPM sampling , DDIM generates higher quality samples using a much fewer number of steps and is more consistent.

Algorithm 2: Diffusion sampling

1: $x_T \sim \mathcal{N}(0, \mathbf{I})$
2: **for** t = T, . . . , 1 **do**
3: $z \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $z = 0$
4:
$$x_{t-1} = \sqrt{\alpha_{t-1}}\hat{x}_0(x_t) + \sqrt{(1 - \alpha_{t-1} - \sigma_t^2)}\epsilon_\theta(x_t, t) + \sigma_t\epsilon$$
5: **end for**
6: **Return** x_0

In summary , the iterative refinement of diffusion models effectively address some of the limitations associated with VAE and GAN, such as diminished sample quality and training instability [9]. This makes diffusion models a compelling choice for our model, as it combines robust performance with stable training dynamics. The following table 3.1 summarizes the characteristics of each generative model:

Table 3.1: Comparison of training and output characteristics of generative models

Model	**Year**	**Training**	**Output**
VAE	2014	Moderate	Unrealistic and Blurry
GAN	2014	Slow and Unstable	Good Realism but Limited Data Distribution
Diffusion	2020	Relatively straightforward	State of the Art

3.3 Conditional Synthesis

Conditional image synthesis has emerged as a powerful technique in computer vision, enabling the generation of images based on specific constraints. Unlike unconditional image synthesis, which generates images from random noise that resembles its training data distribution, conditional methods allow for greater control over the output, producing images that adhere to given specifications such as text prompts [22], segmentation maps [13] or image references [30].

Among these conditioning approaches, using image references as input has given rise to a particularly important sub-field known as image-to-image translation. This sub-field has evolved alongside general generative AI, progressing from early approaches using conditional GANs, with Pix2Pix [19] being one of the most influential frameworks, to more recent advancements based on diffusion like ControlNet [44].

3.3.1 Pix2pix

Pix2Pix, introduced in 2016, is a notable advancement in the field of image-to-image translation based on GAN. As detailed in our previous section 3.1, these models generate new data from latent space representations. However, Pix2Pix employs conditional GAN, which incorporate additional input conditions to control and refine the generation process, setting it apart from traditional, unconditional GAN.

The primary difference between unconditional and conditional GANs lies in their input mechanisms. Unconditional GANs generate data purely from random noise, without any contextual guidance. In contrast, conditional GANs take additional input data into account, which allows them to produce outputs that are tailored to specific conditions. This capability is particularly useful for tasks where the output needs to correspond to particular input characteristics, such as translating a sketch into a realistic image or converting a grayscale image into color as shown in Figure 3.4

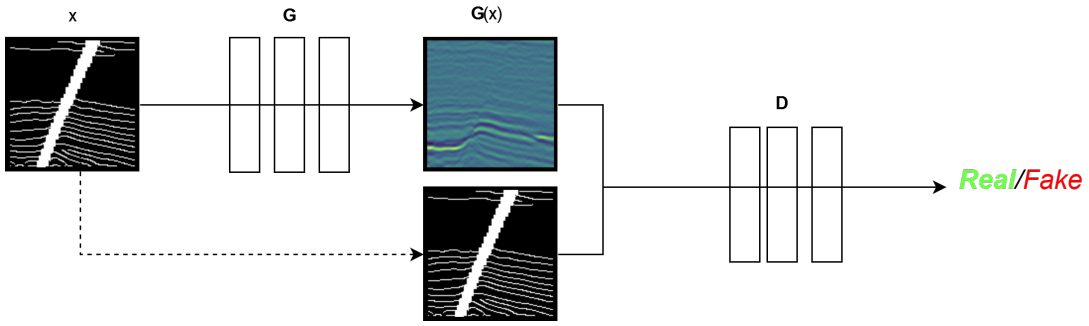

Figure 3.4: Pix2pix architecture

Although Pix2Pix was not the first to apply GANs in a conditional framework, its contributions lie in the meticulous specification of the generator model, discriminator model, and optimization procedures. One of the significant innovations introduced by Pix2Pix at the time was the use of the famous U-Net architecture for the generator. Unlike earlier models that used simple encoder-decoder structures without skip connections, Pix2Pix's U-Net architecture includes these connections to preserve spatial information and improve the quality of generated images.

In Pix2Pix, the generator model accepts an image as input rather than a point from the latent space. To introduce randomness and variability into the generation process, dropout layers are employed both during training and at prediction time. This approach helps to create diverse outputs while maintaining the model's robustness. The discriminator employs a PatchGAN strategy. Rather than assessing the entire image, the PatchGAN discriminator focuses on small, overlapping patches, determining whether each patch is real or fake. This approach provides more granular feedback on the generated images, enhancing their quality and realism.

The discriminator model is trained similarly to traditional GANs, aiming to minimize the negative log-likelihood of distinguishing between real and generated images, with conditioning on a source image. To balance the training speeds of the discriminator and

the generator, the discriminator's loss is halved to slow down its training process:

$$\text{Discriminator Loss} = 0.5 \times \text{Discriminator Loss} \tag{3.11}$$

The generator model is trained using both the adversarial loss, which aims to deceive the discriminator, and the L1 loss, which measures the mean absolute pixel difference between the generated image and the target image.

The combined loss function for the generator incorporates both adversarial and L1 losses. Although L2 loss was also tested, it led to blurrier images compared to L1 loss. Consequently, L1 loss is preferred for its ability to reduce blurring:

$$\text{Generator Loss} = \text{Adversarial Loss} + \lambda \times \text{L1 Loss} \tag{3.12}$$

where λ is a hyperparameter set to 10, meaning the L1 loss is weighted ten times more heavily than the adversarial loss during the generator's training.

3.3.2 ControlNet

ControlNet was developed as a plug-and-play method to enhance the functionality of existing diffusion models. It primarily serves to control established diffusion models, such as Stable Diffusion. To fully comprehend ControlNet's application, it is essential to first understand the architecture of the underlying diffusion model.

The SD (Stbale Diffusion) model represents a sophisticated approach to text-to-image synthesis, building upon the foundational principles of latent diffusion models (LDM) [29]. This model integrates a VAE framework with diffusion processes to produce high-quality images from textual descriptions.Figure 3.5 shows the framerwork in detail.

In the SD framework, the process begins with an encoder E that transforms a given image $a \in \mathrm{R}^{H \times W \times 3}$ into a latent representation a_l. This latent representation undergoes a gradual introduction of Gaussian noise, akin to the approach in DPPM. Specifically, noise is added to a_l over time, resulting in a progressively noisy image a_{lt} at each time step t.

A U-Net architecture is employed to perform denoising. The denoising process is managed by the function $\epsilon_\theta(a_{lt}, t)$, which serves as a sequence of denoising autoencoders aimed at predicting the clean version of the image. The final clean representation a'_l is then reconstructed from the latent space using a decoder D.

The objective function for training this model is given by:

$$L_{\text{LDM}} := \mathrm{E}_{a,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(a_{lt}, t)\|_2^2\right] \tag{3.13}$$

where ϵ is sampled from a standard normal distribution $\mathcal{N}(0,1)$. Further enhancing its

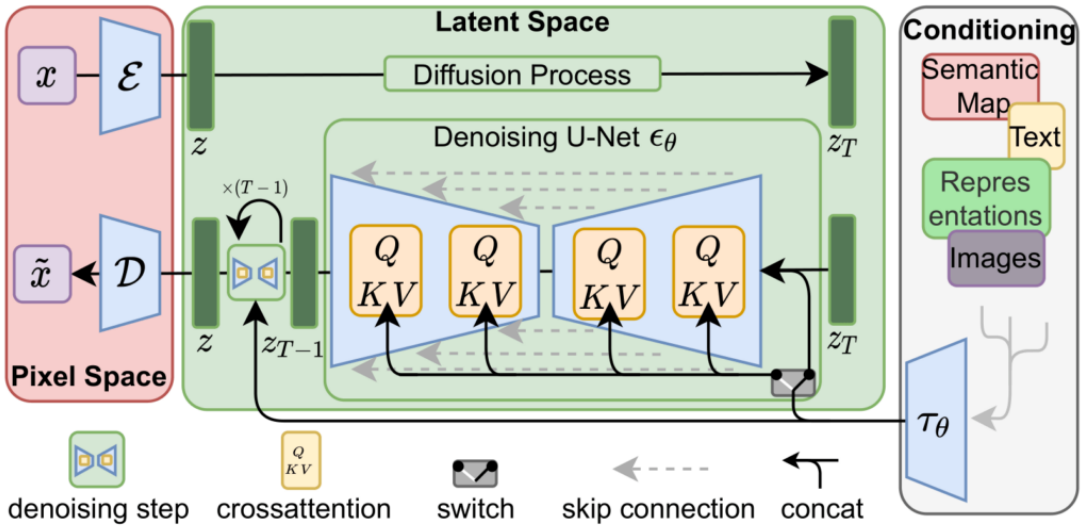

Figure 3.5: Stable Diffusion architecture

functionality, SD incorporates a text encoder, specifically a pre-trained CLIP model [27]. This text encoder translates text prompts into embeddings, which are subsequently integrated with the U-Net's encoder and decoder through cross-attention layers. This cross-attention mechanism facilitates conditioning the model on textual prompts b by processing these prompts through an encoder Z. The refined objective function, incorporating the text conditioning, is:

$$L_{\text{LDM}^b} := \mathrm{E}_{a,b,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(a_{lt}, t, Z_\theta(b))\|_2^2\right] \tag{3.14}$$

This formulation allows SD to leverage both image and text information, enabling it to generate high-quality images that align with textual descriptions.

ControlNet is designed to guide diffusion models to perform specific downstream tasks by utilizing an input control map, which allows for manipulation of the generated output. In its standard configuration, ControlNet supports various control maps based on different conditions, such as edge maps, scribbles, segmentation maps, and pose information. It maintains the stability of the pre-trained SD model by creating a locked copy, while simultaneously employing a trainable copy that incorporates task-specific conditional control for the downstream task as shown in Figure 3.6. These two copies are linked through 1×1 zero convolution layers, with both the bias and weights initialized to zero. During training, the weights of these layers gradually optimize, resulting in faster training without introducing additional noise.

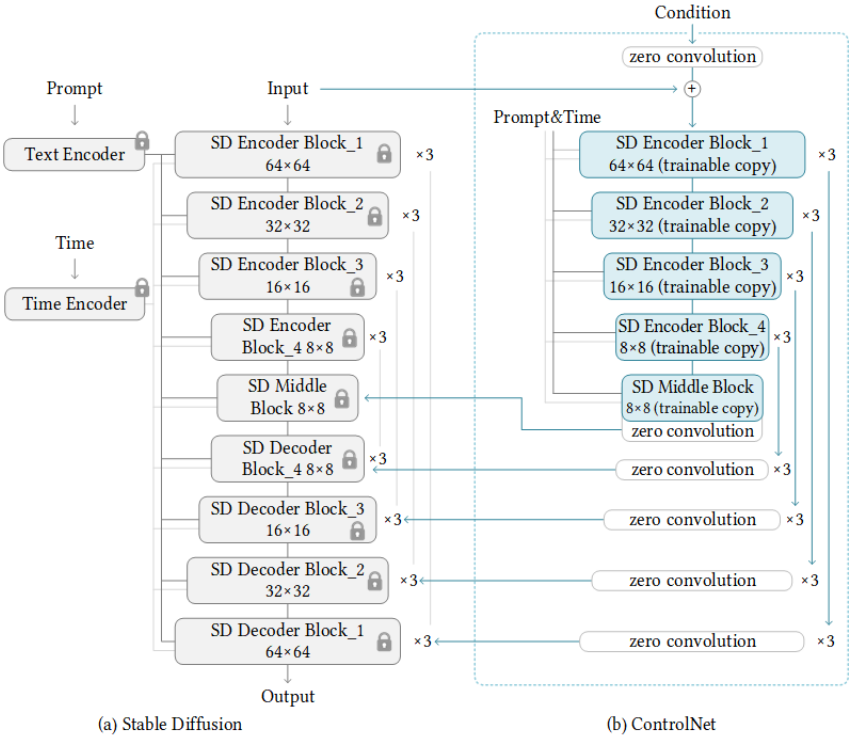

Figure 3.6: ControlNet method

Conclusion

In conclusion, this chapter has explored the evolution of generative AI, tracing the development from foundational models such as VAEs and GANs to the more recent and robust Diffusion Models, which we have chosen as the backbone for our approach. We also examined conditional synthesis methods, including Pix2Pix and ControlNet, which offer enhanced control over generated outputs and serve as the benchmark for comparison in our work. However, these methods primarily focus on natural images and are rarely adapted to the noisier and more complex conditions of seismic data. Our work aims to bridge this gap by introducing a condtional diffusion model specifically tailored for seismic imaging, addressing the unique challenges of this domain. In the next chapter, we will introduce our proposed method and present the experiments conducted to validate its effectiveness. This will include a detailed explanation of our model architecture, the training process, and a thorough evaluation of its performance

Chapter 4

Proposed Method and Experiments

Introduction

This chapter presents FaultDiff, our conditional diffusion model. We delve into the model's design details, architecture, and implementation, providing a thorough overview of how FaultDiff operates and its innovations in the context of seismic imaging.

Following the introduction of the model, we present a comprehensive evaluation framework used to assess its effectiveness. This includes outlining the evaluation strategy, discussing the metrics employed, and reviewing the results obtained. This structured approach allows us to critically analyze FaultDiff's performance and its practical utility in generating and using synthetic seismic data for fault detection.

4.1 FaultDiff

4.1.1 Architecture

Building upon the DDPM framework discussed in previous chapter 3.2, we enhance seismic data synthesis by refining the generation process to better control the output. Our approach leverages diffusion models while introducing conditioning mechanisms to produce more realistic synthetic data.

To achieve this, we incorporate conditions into the neural network using a dual-path encoder architecture. Specifically, our method involves two separate encoder paths: one for processing the original input data and another for handling the conditioning information. These encoders process the input and conditional data independently, and their encoded features are subsequently fused as shown in Figure 4.1.

This strategy allows us to retain the original diffusion model's forward process while

modifying the estimate function to integrate the conditioning information. The updated estimate function is expressed as:

$$\epsilon_\theta(x_t, c_t, t) = D\left((E_c(c_t) + E_x(x_t))_t\right) \tag{4.1}$$

where:

- $E_c(c_t)$ represents the embedding of the conditional input c_t,
- $E_x(x_t)$ denotes the embedding of the original image x_t,
- D is the decoder that generates the predicted noise ϵ_θ based on both the image and the condition.

Our model is an adaptation of the UNet architecture called ResUNet, which incorporates ResNet blocks instead of traditional convolutional layers, enhancing the model's capability with residual learning. Additionally, we use two parallel ResNet encoders, a Transformer block in the middle path, and a ResNet decoder. The encoder consists of four downsampling stages, each with two ResNet blocks followed by one linear attention block. After summing the encoded features from both paths, they are processed by the ViT then fed into the mirrored decoder.

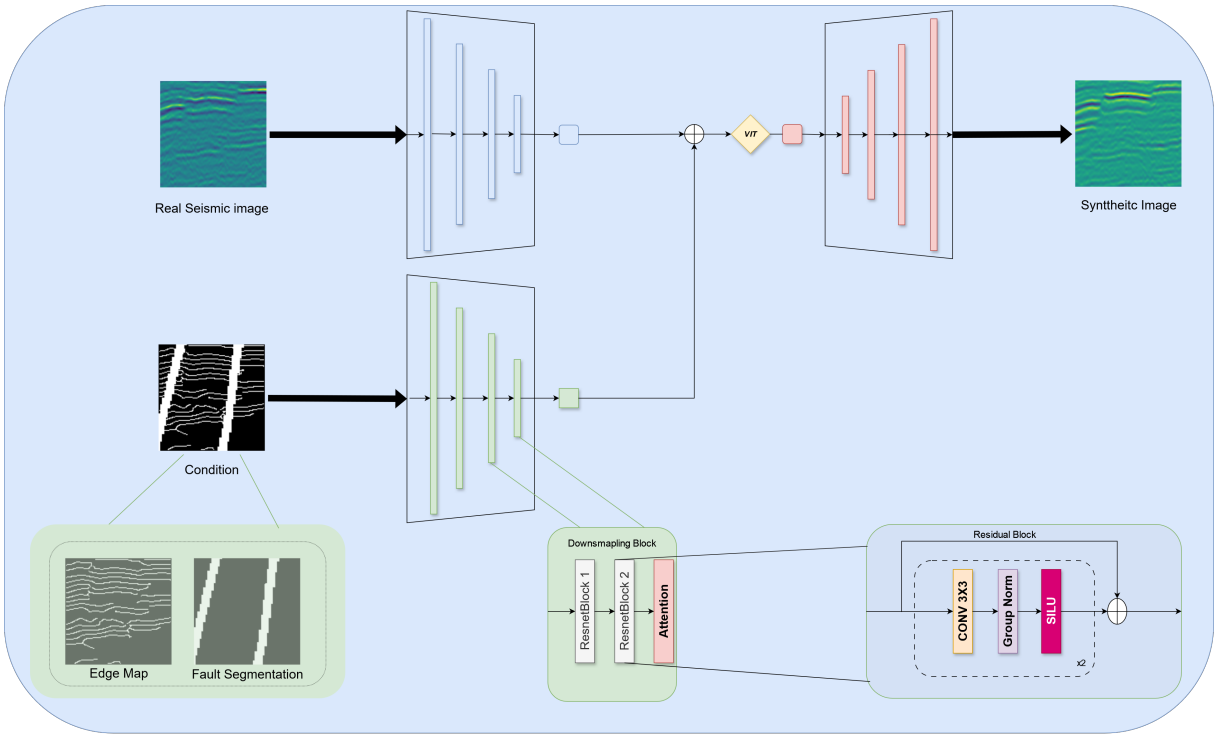

Figure 4.1: FaultDiff architecture

- ResNet Block

  The ResNet block is a fundamental component of our encoder's architecture. ResNet(Residual Networks), use residual connections to address the vanishing gradient problem in deep networks. These connections bypass the non-linear transformations of intermediate layers, facilitating the flow of gradients during training and allowing the network to be deeper without training difficulties.

  In our architecture, each ResNet block contains:

  - 3x3 Convolutional Layer.
  - Group Normalization: Instead of batch normalization, we use group normalization, which normalizes features within groups rather than across the entire batch. This approach provides stability with smaller batch sizes.
  - SiLU Activation Function: We use the SiLU (Sigmoid Linear Unit) activation function, defined as $\text{SiLU}(x) = x \cdot \text{sigmoid}(x)$. This function combines linear and non-linear properties, improving gradient flow and convergence.

- Linear Attention Block

  Following the ResNet blocks, we incorporate a linear attention block as in the original DDPM implementation [17]. Attention mechanisms enable the network to focus on important regions in the feature maps, they can be formulated as kernel smoother to learn similarity between different pixels. Linear attention [21] approximates the attention matrix more efficiently, reducing the computational complexity from quadratic $\mathcal{O}(n^2)$ to linear $\mathcal{O}(n)$.

- Downsampling Operation

  Each stage of the encoder includes a downsampling operation by preforming 2x2 pooling then applying 1x1 convolution layer. This operation reduces the spatial dimensions of the feature maps, allowing the network to focus on abstract features and reducing computational complexity.

- Transformer Block

  In the middle of the network, we use a Transformer block as done in Simple Diffusion[18] for faster training. Transformers, originally designed for NLP tasks, are adapted for vision tasks due to their ability to model global context through self-attention mechanisms.The input consists of a C×H×W feature map, where C, H and W denote the channel, height and width of the feature map, respectively. The output is a feature map of the same dimensions. The transformer block as shown in Figure 4.2 consists of two residual blocks. The first block includes LayerNorm followed by a multi-head attention mechanism that computes attention scores across the spatial dimensions. The second block contains LayerNorm and a feed-forward network composed of two 1x1 convolutional layers with a GELU (Gaussian Error Linear Unit) activation in between.

- ResNet Decoder

  The feature maps, after passing through the transformer block, are processed by the ResNet decoder. The decoder mirrors the encoder's architecture but performs

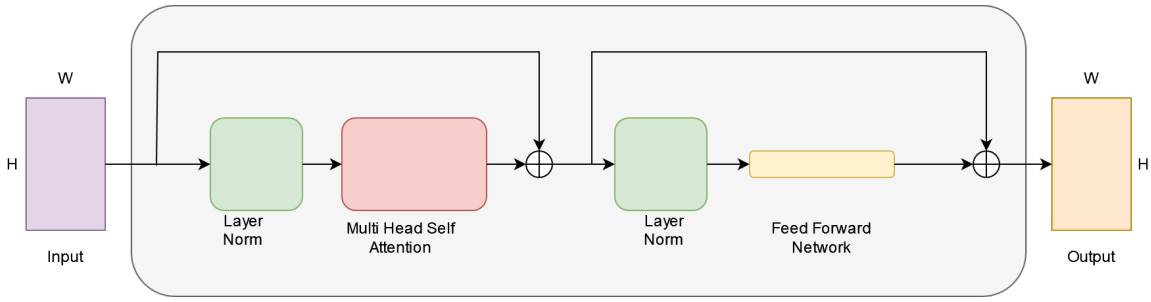

Figure 4.2: Transformer block

upsampling to restore the original spatial resolution. Skip connections between the encoder and decoder transfer high-resolution features, which helps maintain fine details in the final output.

- Time Embedding

  In our model, we use time embedding to incorporate temporal information. This is done by encoding the time step t into a fixed-length vector using sinusoidal embeddings. These embeddings are added to the feature maps, allowing the model to adjust its processing based on the current time step.

4.1.2 Condition Design

In order to make a more realistic synthetic dataset for seismic fault detection, the core idea is to use existing fault segmentation from public datasets such as Thebe [4] to control the generation. To enhance the conditioning process, we use both fault and edge maps as input conditions, rather than fault maps alone. Fault maps capture critical information about seismic discontinuities, but they lack additional context provided by geological boundaries and fine-scale details that are crucial for generating realistic seismic images. Edge maps, on the other hand, capture these fine-grained details, such as layer boundaries and subtle changes within the seismic data. By conditioning the model on both fault and edge maps, we provide richer contextual information that improves the model's understanding of spatial relationships and patterns; this approach enhances the model's ability to generalize. In fact, when we trained a model using only fault maps as a condition (see Figure 4.3), it exhibited poor generalization leading to sub-optimal results. The inclusion of edge maps addresses this limitation, allowing the model to synthesize more realistic images and accurately represent diverse geological features.

To create the condition for our diffusion model, we apply the Canny edge detector to real seismic images. Canny edge detection is a widely used technique in image processing that excels at identifying edges with precision while minimizing noise. The process begins with Gaussian smoothing, where a Gaussian filter is applied to the image to reduce noise and prevent the detection of false edges; in our case, we use $\sigma = 3$ for this smoothing. This is followed by calculating the gradient magnitude and direction at each pixel, which helps identify areas of significant intensity change—typically where edges are located.

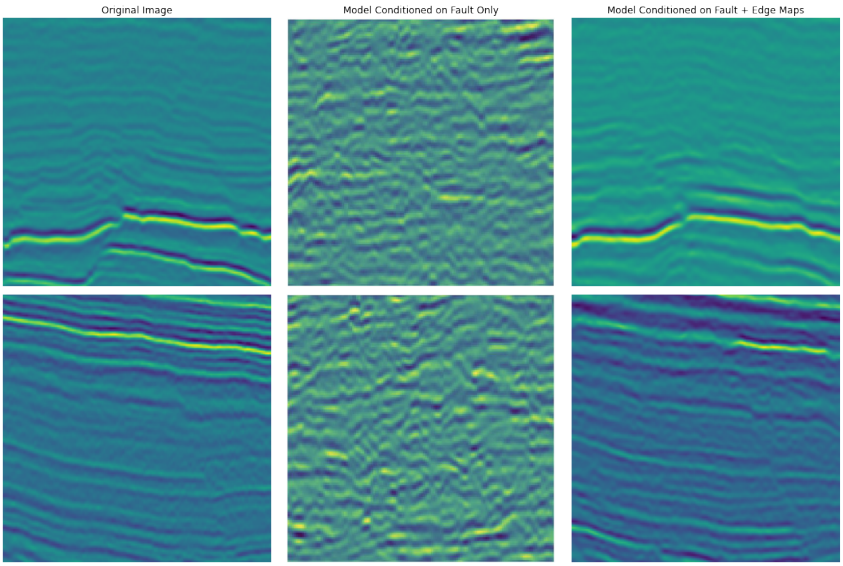

Figure 4.3: Comparison between models using fault segmentation as condition only and using both fault and edge map

The algorithm then performs non-maximum suppression to thin the detected edges to a single pixel width, retaining only the local maxima in the gradient direction to ensure clean and precise edge maps. To further refine the edge detection, We employ double thresholding, where a low threshold of 0.25 and a high threshold of 0.75 are used to classify edge pixels into strong, weak, and non-relevant categories. Strong edges are those with gradient magnitudes above the high threshold, while weak edges are included if they are connected to strong edges. Finally, edge tracking by hysteresis connects weak edges to strong ones if they are contiguous, resulting in a coherent and continuous edge map. The edge map obtained from this process highlights prominent boundaries and fine details in the seismic image. This edge map is then combined with the fault mask to create a comprehensive conditioning input for our diffusion model, enhancing its ability to generate realistic synthetic seismic images by capturing both major fault structures and intricate geological features.

4.2 Implementation

4.2.1 Dataset

The Thebe Dataset [4] is a comprehensive, large-scale seismic dataset meticulously labeled by experts to facilitate automatic fault recognition through machine learning techniques. Originating from the Thebe Gas Field in the Exmouth Plateau of the Carnarvon Basin on the NW shelf of Australia, this dataset includes processed seismic images paired with detailed fault annotations. Designed to address the scarcity of publicly available seismic datasets, Expert intepreters annotated faults with vertical displacements greater than 20 meters within a depth range of 2-4 km using Petrel software. then the SEGY data is

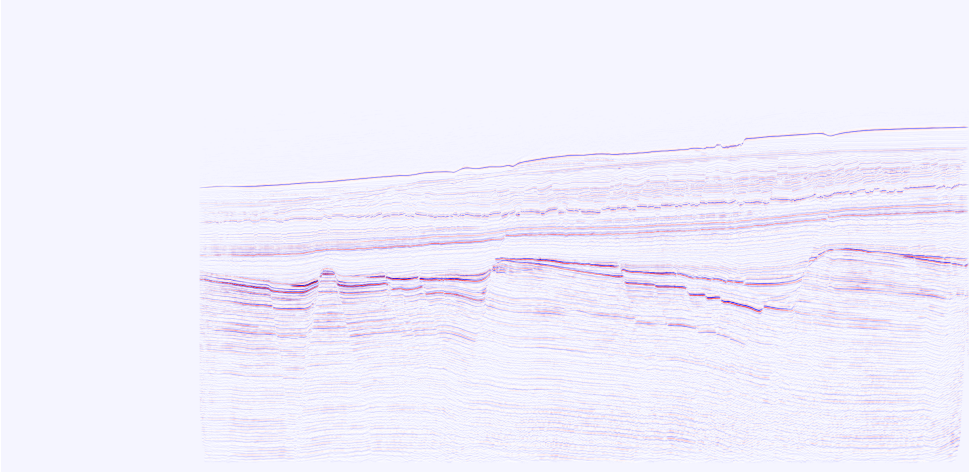

Figure 4.4: A crossline from Thebe dataset

processed into a 3D rectangular NumPy volume format, covering a substantial 3D volume of 1804 crossline × 3174 inline × 1537 sample. The data is segmented into training, validation, and test sets.The first 900 crosslines were used for training, the next 200 for validation, and the remaining 703 for testing. Figure 4.4 demonstrates a crossline from Thebe dataset.

We adopt a slicing window approach to generate image patches from our seismic dataset and corresponding fault masks. Before extracting patches, the seismic data is normalized on a per-slice basis to ensure that each slice's pixel values are scaled between 0 and 1.

Padding is applied to the edges of each slice to ensure that the dimensions are compatible with the chosen patch and step sizes. Specifically, the images and masks are padded symmetrically along both the horizontal and vertical axes to avoid missing regions at the edges of the data. This padding ensures that the extracted patches cover the entire image without losing any information.

For each image slice, we use a sliding window technique that extracts overlapping patches of size 128 x 128, moving the window by 64 pixels at each step, resulting in 50% overlap between adjacent patches. This overlap helps capture contextual information between patches, improving the model's ability to learn spatial patterns in the data.

To focus on patches containing relevant information, we exclude any patches from the fault mask that contain less than 3% fault pixels, as these patches are likely to provide minimal value for fault detection. This threshold-based filtering significantly reduces the number of irrelevant patches, ensuring that the dataset consists of high-quality samples with sufficient fault representation and combat the unbalance problem.

Specifically, we extract 128 x 128 patches from every 10th crossline of the training set, which helps to reduce the size of the dataset while still capturing sufficient geological variation across the seismic data. This results in a total of 12,133 images and 12,133 corresponding condition created as detailed in subsection 4.1.2. By leveraging only around 10% of the entire training set, we significantly reduce the computational load and training

time. This approach demonstrates that a smaller, well-selected subset of the training data is sufficient to generate high-quality synthetic seismic data effectively using our conditional diffusion model. Figure 4.5 shows some of the images and their corresponding conditions used for training.

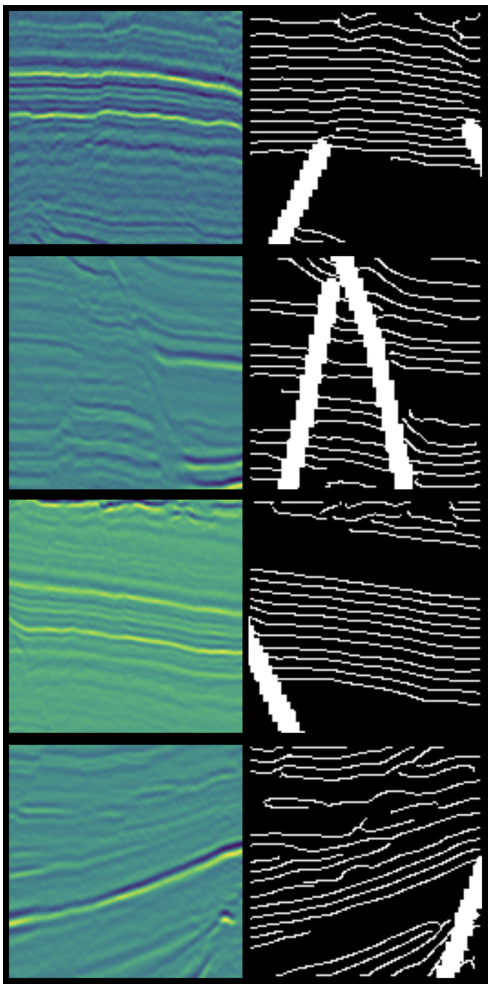

Figure 4.5: 128x128 images and their corresponding conditions used for training the diffusion model

4.2.2 Training

All experiments are implemented using PyTorch, a popular deep learning framework known for its flexibility and efficiency in handling large datasets and complex models. Training and testing are performed on an NVIDIA A6000 GPU with 16 GB of RAM, providing the necessary computational power to efficiently handle the large datasets and the intensive operations required for training diffusion models.

For training deep learning models , several hyperparameters need to be carefully chosen to optimize both performance and computational efficiency. Hyperparameters are variables that are set prior to the training process and govern the overall behavior of the

learning algorithm. These parameters are not learned from the data but instead guide the optimization process, influencing the model's performance, convergence speed, and generalization ability. Choosing appropriate hyperparameters is crucial, as they directly affect the model's ability to learn from data efficiently. Some common hyperparameters include the learning rate, batch size, and the specific settings of the optimizer.

In our case, the hyperparameters are defined as follows:

- **Learning Rate ($\alpha = 0.0005$):** The learning rate controls how much to adjust the model weights with respect to the loss gradient during each iteration. A smaller learning rate helps in fine-tuning the weights, ensuring that the model learns gradually and avoids overshooting minima in the loss landscape. The chosen value of 0.0005 strikes a balance between convergence speed and stability during training.
- **Adam Optimizer:** Adam (Adaptive Moment Estimation) is selected as the optimizer due to its ability to handle sparse gradients and its efficient parameter updates. Adam combines the advantages of RMSProp (Root Mean Square Propagation) and SGD (Stochastic Gradient Descent) by using moving averages of both the gradient and the squared gradient to compute adaptive learning rates for each parameter.
  - $\beta_1 = 0.95$: Controls the exponential decay rate for the moving average of the first moment (mean of the gradients). A value of 0.95 ensures smooth momentum-based updates, providing stability while adapting to gradient changes effectively.
  - $\beta_2 = 0.999$: Controls the exponential decay rate for the second moment (variance of the gradients). A value close to 1 allows precise updates of the adaptive learning rates, particularly in cases of noisy or sparse gradients.
  - **Weight Decay (1×10^{-6}):** Weight decay serves as regularization to prevent overfitting by penalizing large weights. A small value of 1×10^{-6} ensures model generalization while avoiding underfitting.
  - $\epsilon = 1 \times 10^{-8}$: A small constant added to prevent division by zero in the Adam update rule. The value ensures numerical stability, particularly with very small gradients.
- **Mixed Precision with BF16:** Mixed precision training is employed to reduce memory usage and speed up computations. BF16 (Brain Floating Point) offers similar accuracy to FP32 while using less memory and faster arithmetic operations on modern GPUs like the A6000. This technique significantly accelerates training while maintaining precision for high accuracy.
- **Batch Size (64):** Defines the number of training examples processed together in one forward/backward pass. A batch size of 64 allows for efficient memory utilization while stabilizing gradient updates. This size strikes a balance between memory consumption and convergence.
- 1,000 **Diffusion Timesteps:** The model uses 1,000 diffusion timesteps during training. Each timestep in a diffusion model represents a stage in the forward process,

where noise is gradually added, followed by a denoising step. Using 1,000 timesteps ensures a smooth denoising process and high-quality output while maintaining computational efficiency.

- **DDIM Sampling with** 100 **Timesteps:** DDIM algorithm is employed for sampling. By using only 100 timesteps during the sampling phase, DDIM significantly reduces the time needed to generate synthetic data while maintaining high image quality. This method provides an efficient trade-off between computational resources and the fidelity of the generated data.
- **Model Convergence after** 200 **Epochs:** The model is trained for 200 epochs, where each epoch represents a complete pass through the training data. This was found to be sufficient for the model to converge, meaning further training does not significantly improve performance.

4.3 Evaluation

In evaluating FaultDiff, we utilize a two-part strategy to thoroughly assess its performance. The first part involves generative evaluation, where we measure the model's capability to produce high-quality synthetic seismic data. This includes comparing FaultDiff against state-of-the-art generative models then assessing its generative fidelity using metrics such as FID (Fréchet Inception Distance). The second part of our evaluation strategy focuses on segmentation performance, specifically examining how well the synthetic data supports fault detection tasks. We compare the effectiveness of FaultDiff-generated data in improving fault detection accuracy against both real seismic data and other synthetic datasets. This comprehensive evaluation approach provides insights into FaultDiff's practical utility in real-world fault detection scenarios and demonstrates its potential to enhance seismic imaging and analysis.

4.3.1 Comparing to State of the Art

To evaluate the performance of our model, We compare to both Pix2pix and ControlNet which were explained in detail in section 3.3.

We train Pix2Pix using the default settings outlined by the authors [19], employing a batch size of 64 for 200 epochs. The learning rate is set to 10^{-5}, consistent with the recommended parameters in their implementation.

For ControlNet, Given that seismic data exhibits significant differences from the large-scale natural image datasets used in the pre-training, we followed the approach proposed by the authors for adapting models to out-of-distribution data.

Initially, we fine-tuned the VAE for 200 epochs to tailor it specifically to the unique characteristics of seismic data. This fine-tuning process ensured that the VAE's feature

extraction and latent space representation were effectively adapted to the distinctive patterns and structures found in seismic observations. Following the VAE fine-tuning, we continued by fine-tuning the SD model for an additional 400 epochs. This step further customized the SD model to accommodate the specific attributes of seismic data, improving its ability to generate high-quality synthetic images that reflect the unique features of seismic datasets. Subsequently, we fine-tuned ControlNet for another 400 epochs, utilizing both the adapted VAE and SD models. During this stage, we trained ControlNet on seismic images along with their corresponding conditions, using empty prompts for text inputs.

Throughout all stages of training, we employed Stable Diffusion 2 as our diffusion model, capitalizing on its advanced capabilities for high-quality image generation. We adhered to the recommended training parameters, using a batch size of 64 and a learning rate of 1×10^{-5}.

To evaluate the performance of FaultDiff, we employ several metrics to assess the quality of the synthetic seismic data generated. These metrics are crucial for comparing the results obtained from different models. We use the following metrics:

MSE (Mean Squared Error) quantifies the average squared differences between the pixel values of the generated images and the ground truth images. It is calculated using the formula:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{4.2}$$

where x_i represents the pixel value of the ground truth image, $\hat{x}_i$ denotes the pixel value of the generated image, and N is the total number of pixels. Lower MSE values indicate better performance with minimal deviation between the generated and ground truth images.

PSNR (Peak Signal-to-Noise Ratio) measures the ratio between the maximum possible pixel value and the noise introduced by errors. It provides a quantitative measure of image quality, with higher values reflecting better quality. The formula for PSNR is:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{R^2}{\text{MSE}}\right) \tag{4.3}$$

where R is the maximum possible pixel value of the image (e.g., $R = 255$ for 8-bit images), and MSE is the Mean Squared Error. Higher PSNR values indicate a higher signal-to-noise ratio and better image quality.

DSSIM (Difference of Structural Similarity Index) evaluates the structural similarity between images, focusing on dynamic or time-dependent changes. It is defined as:

$$\text{DSSIM} = 1 - \text{SSIM} \tag{4.4}$$

where the Structural Similarity Index (SSIM) is computed as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.5}$$

Here, μ_x and μ_y are the mean values of images x and y, σ_x^2 and σ_y^2 are the variances, σ_{xy} is the covariance, and c_1 and c_2 are constants used for stabilization. DSSIM values range from 0 to 1, with lower values indicating higher structural similarity between the generated and ground truth images.

Figure 4.6 presents a visual comparison of the generated images from FaultDiff, Pix2pix, and ControlNet, showcasing the results of each model under the same condition. From the generated seismic images, we observe that FaultDiff is most comparable to the Original image, with fewer discrepancies in the seismic reflections and fault structures. Additionally, the generated image by FaultDiff shows a smoother texture and more accurate replication of the seismic patterns, effectively preserving the original image's structure. In contrast, Pix2Pix, while competent in following the condition, introduces minor distortions in the reflections, and ControlNet struggles more with maintaining the geological continuity, displaying more artifacts and noise. This poor performance can be attributed to its limitations, including the need for a larger dataset, the lack of suitable text prompts, and its untested nature in adapting to the specific challenges of seismic data.

Table 4.1 illustrates quantitative results in terms of generation metrics. FaultDiff outperforms the other models by up to 25% in MSE, achieving the lowest error rate 3.72 db in PSNR and 2%in less in DSSIM.

Table 4.1: Generation results for FaultDiff, Pix2pix, and ControlNet

Model	**MSE**	**PSNR**	**DSSIM**
FaultDiff	**0.0028**	**26.74**	**0.1905**
Pix2pix	<u>0.0053</u>	23.02	<u>0.2143</u>
ControlNet	0.0066	<u>23.54</u>	0.2218

4.3.2 Generative Fidelity

To evaluate the generative fidelity of our synthetic seismic images, we assess how closely these images match the distribution of field seismic data. One of the key metrics used for this evaluation is FID, which quantifies the similarity between the distributions of real and generated images.

FID is a metric that measures the distance between two distributions of feature vectors extracted from a pre-trained Inception-v3 model. The feature vectors are obtained from

the last average pooling layer of the Inception-v3 network, which captures high-level features of the images.

A lower FID score indicates that the synthetic images are closer to the real images in terms of feature distribution, reflecting higher generative fidelity.

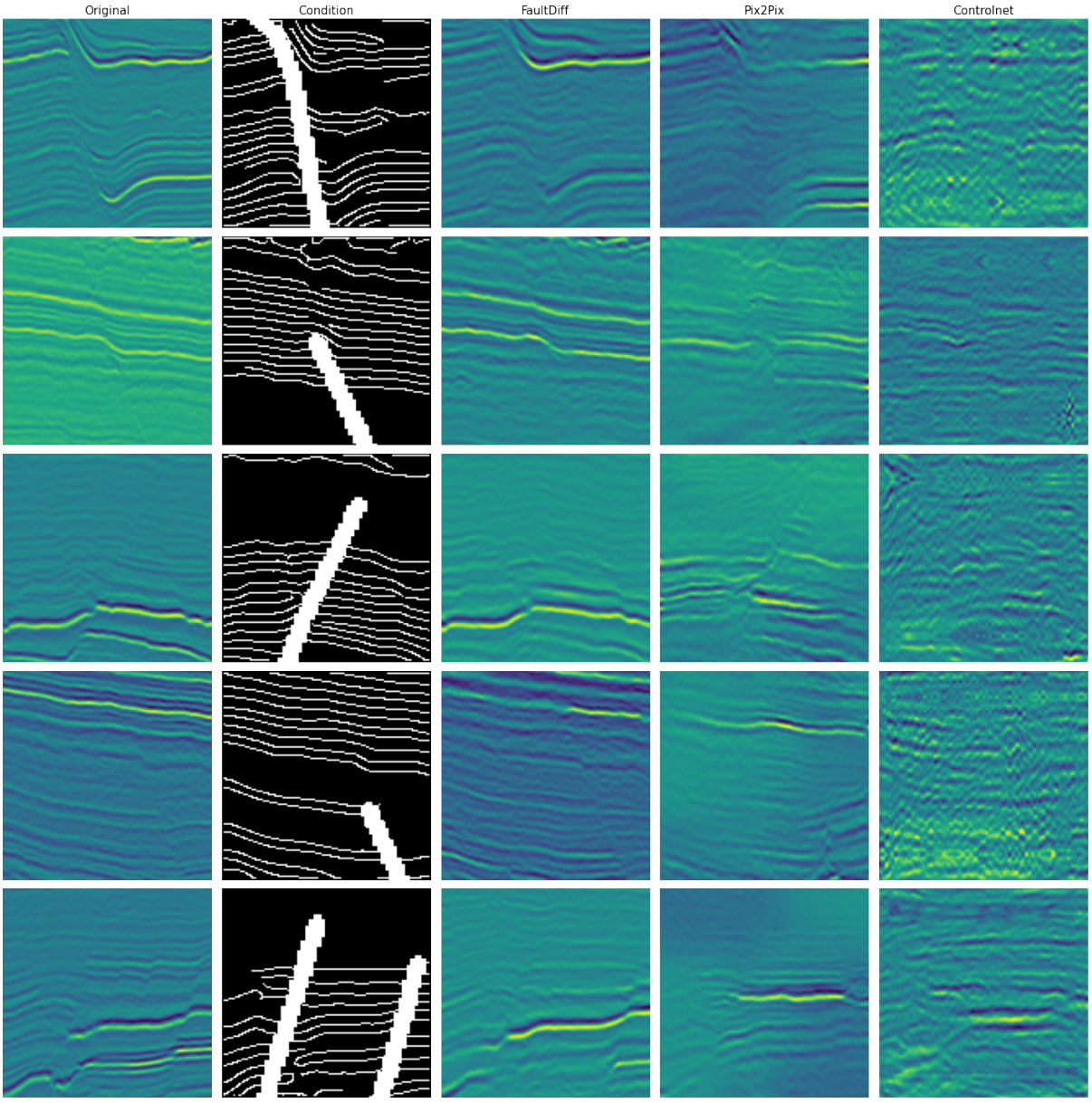

Figure 4.6: Visual comparison of seismic image synthesis across different models

In our evaluation, we utilize the Thebe dataset [4] and the F3 dataset [1] as our sources of real data. The F3 dataset is particularly valuable as it serves as an unbiased benchmark, being independent of the training process for our diffusion model. This independence allows us to assess the quality of the synthetic images impartially. We compare our synthetic data with those generated by the method introduced by Wu et al. [39] to benchmark the performance of our approach against existing methods.

Table 4.2: Comparison of FID scores between synthetic and real datasets

Synthetic Dataset	Real Dataset	FID Score
FaultDiff	Thebe	6.47562
	F3	6.47568
FaultSeg3d	Thebe	130.12766
	F3	130.12789

Table 4.2 presents the FID scores. Our approach, FaultDiff, achieves notably low FID scores for both the Thebe and F3 datasets, with values of 6.47562 and 6.47568, respectively. These results indicate that FaultDiff-generated images closely resemble real field data in terms of feature distribution, reflecting its high generative fidelity. In contrast, the FaultSeg3D method [39] yields significantly higher FID scores, indicating that its synthetic images diverge considerably from the real data distributions. This high FID score is likely due to its simulation-based approach, explaining the poor generalization in models trained on such data.

4.3.3 From the Perspective of Segmentation

To evaluate the effectiveness of synthetic data in fault detection, we measure the ability of segmentation models trained solely on synthetic data to generalize to real field data. We compare the performance of segmentation models, based on a 2D U-Net architecture, trained on real data (Thebe) and synthetic data (FaultDiff and FaultSeg3D). For the Thebe dataset, we use the slicing window approach to extract 120,976 images of size 128x128 from every crossline in the training set. For FaultDiff, we generate an equivalent number of 128x128 images from the respective conditions. Additionally, we experiment with training on a reduced dataset of synthetic images by generating images conditioned on patches from every 5th crossline, resulting in 24338 images less than 5x the original trainset size.

For FaultSeg3D , we convert every crossline and inline profile into 2D images, resulting in 51,200 images of size 128x128. The models are tested on the TheBe test set consisting of 143,560 images. We trained each model for 100 epochs with a batch size of 16 and a learning rate of 10^{-3}.

Evaluation is performed using standard metrics, including Accuracy, Precision, Recall, AP (Average Precision), and AUC (Area Under Curve).

The results presented in Table 4.3 demonstrate the efficacy of the FaultDiff-generated synthetic data in training segmentation models. The model trained on FaultDiff-synthesized images exhibits performance metrics that are not only comparable but marginally superior to those of models trained on real data. Notably, the FaultDiff-trained model achieved an accuracy of 0.9322 and precision of 0.6879, slightly outperforming the model trained on original data (accuracy: 0.9316, precision: 0.6699). Besides, the segmentation model trained on a reduced subset of FaultDiff synthetic data, utilizing fewer crosslines, main-

Table 4.3: segmentation performance metrics for real and synthetic data

Training Set	**Strategy**	**Accuracy**	**Precision**	**Recall**	**AUC**	**AP**
Thebe	Real	<u>0.9316</u>	<u>0.6699</u>	**0.5438**	**0.9291**	**0.6640**
FaultDiff	Synthetic	**0.9320**	**0.6873**	<u>0.5142</u>	<u>0.9232</u>	<u>0.6430</u>
FaultDiff	Synthetic (reduced set)	0.9315	0.6689	0.5122	0.9212	0.6330
FaultSeg3D	Synthetic	0.9043	0.4115	0.0322	0.6593	0.1933

tained comparable performance metrics. This reduced dataset model achieved an accuracy of 0.9315 and precision of 0.6689, closely mirroring the performance of models trained on both the full original and synthetic datasets. The consistency in recall measurements (0.5142 for the reduced set vs. 0.5244 for the full set) is particularly noteworthy. These results suggest that a small set of our synthetic data can yield performance comparable to larger datasets, while simultaneously offering advantages in computational efficiency during the training phase.

Conclusion

Chapter 4 provided an in-depth examination of the FaultDiff model, which is designed for synthesizing seismic data for fault detection. We thoroughly detailed the model's design, including the modifications to the DDPM framework and the dual-path encoder architecture that enables effective conditioning of the generated seismic data. Then we provided the implementation details from dataset prepossessing to hyper-parameters, to ensure reproducibility.

Our evaluation strategy was twofold. Firstly, we assessed the generative fidelity of FaultDiff by comparing the synthetic seismic data against real seismic data using metrics such as FID. This evaluation demonstrated how closely the synthetic data produced by FaultDiff matches the distribution of real seismic data, highlighting its ability to generate high-quality images.

Secondly, we conducted a segmentation evaluation to determine how well the synthetic data supports fault detection tasks. By comparing the impact of FaultDiff-generated data on fault detection accuracy with real seismic data and other synthetic datasets, segmentation models trained on FaultDiff data demonstrate competitive performance in fault detection.

The results underscore FaultDiff's effectiveness in synthesizing realistic seismic data that enhances fault detection. The evaluation shows that FaultDiff produces high-quality synthetic seismic images faithful to their fault maps.

Conclusion

In this thesis, we introduced and explored the application of FaultDiff, a novel conditional denoising diffusion model designed for generating high-fidelity synthetic seismic data. By conditioning on fault segmentation maps and edge features, FaultDiff demonstrated enhanced realism in synthetic data, outperforming state-of-the-art models in key quality metrics. Our research revealed that models trained on FaultDiff-generated data perform comparably to those trained on real data, effectively bridging the gap between synthetic and real datasets.

The primary contribution of this research is the introduction of FaultDiff, marking the first application of diffusion models for seismic data synthesis. This model not only improves the quality of synthetic seismic images but also provides a more reliable basis for training fault detection algorithms, contributing significantly to the fields of geophysics and deep learning.

However, this study has limitations. The current model operates on 2D seismic data, which lacks the spatial correlations present in 3D data. This limitation may affect the model's ability to fully capture the complex geological structures found in real-world seismic surveys. Additionally, FaultDiff currently generates only single-attribute seismic images, which do not encompass the range of multi-attribute data used in advanced seismic interpretation tasks. These constraints could limit the model's generalizability and effectiveness in more comprehensive geological analysis.

Building upon the success of FaultDiff, future research should leverage recent advancements in video diffusion models to enhance our approach's capabilities. These techniques offer promising avenues for extending dimensionality to 3D. Furthermore, evaluating the model's performance across various seismic tasks will help assess its robustness and generalizability. Overall, this thesis represents a significant step forward in generating realistic synthetic seismic data. By leveraging advanced generative models, this work contributes to improving fault detection accuracy and reliability, thereby enhancing seismic interpretation capabilities.

Bibliography

[1] Yazeed Alaudah, Patrycja Michałowicz, Motaz Alfarraj, and Ghassan AlRegib. A machine-learning benchmark for facies classification. *Interpretation*, 7(3):SE175–SE187, 2019.

[2] Nasher M. AlBinHassan and Kurt Marfurt. *Fault detection using Hough transforms*, pages 1719–1721. 2005.

[3] Yu An, Haiwen Du, Siteng Ma, Yingjie Niu, Dairui Liu, Jing Wang, Yuhan Du, Conrad Childs, John Walsh, and Ruihai Dong. Current state and future directions for deep learning based automatic seismic fault interpretation: A systematic review. *Earth-Science Reviews*, 243:104509, 2023.

[4] Yu An, Jiulin Guo, Qing Ye, Conrad Childs, John Walsh, and Ruihai Dong. Deep convolutional neural network for automatic fault recognition from 3d seismic datasets. *Computers & Geosciences*, 153:104776, 2021.

[5] Mike Bahorich and Steve Farmer. 3-d seismic discontinuity for faults and stratigraphic features: The coherence cube. *The Leading Edge*, 14(10):1053–1058, 1995.

[6] Arthur E. Barnes. Weighted average seismic attributes. *GEOPHYSICS*, 65(1):275–285, 2000.

[7] Ran Chen, Zeren Zhang, and Jinwen Ma. Seismic fault sam: Adapting sam with lightweight modules and 2.5d strategy for fault detection, 2024.

[8] Jennifer Elizabeth Cunningham. *The Seismic Imaging and Interpretation of Faults: A Case Study from the Snøhvit Field, Barents Sea.* Phd thesis, University of Stavanger, Stavanger, 2021.

[9] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *ArXiv*, abs/2105.05233, 2021.

[10] Haibin Di, Amir Shafiq, and Ghassan Alregib. Seismic-fault detection based on multiattribute support vector machine analysis. pages 2039–2044, 08 2017.

[11] Yimin Dou, Kewen Li, Jianbing Zhu, Xiao Li, and Yingjie Xi. Attention-based 3-d seismic fault segmentation training by a few 2-d slice labels. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2022.

[12] Rodrigo S. Ferreira, Julia Noce, Dario A. B. Oliveira, and Emilio Vital Brazil. Generating sketch-based synthetic seismic images with generative adversarial networks. *IEEE Geoscience and Remote Sensing Letters*, 17(8):1460–1464, 2020.

[13] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors, 2022.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[15] Paul Goyes-Peñafiel, León Suárez-Rodríguez, Claudia V. Correa, and Henry Arguello. Gan supervised seismic data reconstruction: An enhanced learning for improved generalization. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–10, 2024.

[16] Bowen Guo, Lu Liu, and Yi Luo. *Automatic seismic fault detection with convolutional neural network*, pages 1786–1789. 2018.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[18] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images, 2023.

[19] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[20] Jiankun Jing, Zhe Yan, Zheng Zhang, Hanming Gu, and Bingkai Han. Fault detection using a convolutional neural network trained with point-spread function-convolution-based samples. *GEOPHYSICS*, 88(1):IM1–IM14, 2023.

[21] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.

[22] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models, 2023.

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[24] Lei Lin, Zhi Zhong, Chenglong Li, Andrew Gorman, Hao Wei, Yanbin Kuang, Shiqi Wen, Zhongxian Cai, and Fang Hao. Machine learning for subsurface geological feature identification from seismic data: Methods, datasets, challenges, and opportunities. *Earth-Science Reviews*, 257:104887, 2024.

[25] Dario A. B. Oliveira, Rodrigo S. Ferreira, Reinaldo Silva, and Emilio Vital Brazil. Improving seismic data resolution with deep generative networks. *IEEE Geoscience and Remote Sensing Letters*, 16(12):1929–1933, 2019.

[26] Stein Inge Pedersen, Trygve Randen, Lars Sonneland, and Øyvind Steen. *Automatic fault extraction using artificial ants*, pages 512–515. 2005.

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[28] Andy Roberts. Curvature attributes and their application to 3d interpreted horizons. *First Break*, 19:85–100, 02 2001.

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[30] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2023.

[31] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[33] Zhanxin Tang, Bangyu Wu, Weihua Wu, and Debo Ma. Fault detection via 2.5d transformer u-net with seismic data pre-processing. *Remote Sensing*, 15(4), 2023.

[34] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10, 2020.

[35] Kristofer Tingdahl and Matthijs Rooij. Semi-automatic detection of faults in 3d seismic data. *Geophysical Prospecting*, 53:533 – 542, 06 2005.

[36] Shenghou Wang, Xu Si, Zhongxian Cai, and Yatong Cui. Structural augmentation in seismic data for fault prediction. *Applied Sciences*, 12(19), 2022.

[37] Z. Wang, J. You, W. Liu, and X. Wang. Transformer assisted dual u-net for seismic fault detection. *Frontiers in Earth Science*, 11:1047626, 2023.

[38] Xinming Wu, Zhicheng Geng, Yunzhi Shi, Nam Pham, Sergey Fomel, and Guillaume Caumon. Building realistic structure models to train convolutional neural networks for seismic structural interpretation. *GEOPHYSICS*, 85(4):WA27–WA39, 2020.

[39] Xinming Wu, Luming Liang, Yunzhi Shi, and Sergey Fomel. Faultseg3d: Using synthetic data sets to train an end-to-end convolutional neural network for 3d seismic fault segmentation. *GEOPHYSICS*, 84(3):IM35–IM45, 2019.

[40] Xinming Wu, Yunzhi Shi, Sergey Fomel, and Luming Liang. Convolutional neural networks for fault interpretation in seismic images. 08 2018.

[41] Wei Xiong, Xu Ji, Yue Ma, Yuxiang Wang, Nasher M. AlBinHassan, Mustafa N. Ali, and Yi Luo. Seismic fault detection with convolutional neural network. *GEOPHYSICS*, 83(5):O97–O103, 2018.

[42] Haihang Zhang, Guangzhi Zhang, Jianhu Gao, Shengjun Li, Jinmiao Zhang, and Zhenyu Zhu. Seismic impedance inversion based on geophysical-guided cycle-consistent generative adversarial networks. *Journal of Petroleum Science and Engineering*, 218:111003, 2022.

[43] Hao-Ran Zhang, Yang Liu, Yu-Hang Sun, and Gui Chen. Seisresodiff: Seismic resolution enhancement based on a diffusion model. *Petroleum Science*, 2024.

[44] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[45] Zeren Zhang, Ran Chen, and Jinwen Ma. Improving seismic fault recognition with self-supervised pre-training: A study of 3d transformer-based with multi-scale decoding and fusion. *Remote Sensing*, 16:922, 03 2024.

[46] Tao Zhao and Pradip Mukhopadhyay. *A fault-detection workflow using deep learning and image processing*, pages 1966–1970. 2018.

[47] ZhenWang* and GhassanAlRegib. *Automatic fault surface detection by using 3D Hough transform*, pages 1439–1444. 2014.