Pontifícia Universidade Católica
do Rio de Janeiro

Antonio José Grandson Busson

A Self-supervised Method for Blind Denoising of Seismic Shot Gathers

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática.

Advisor: Prof. Sérgio Colcher

Rio de Janeiro
March 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Antonio José Grandson Busson

A Self-supervised Method for Blind Denoising of Seismic Shot Gathers

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática. Approved by the Examination Committee.

Prof. Sérgio Colcher
Advisor
Departamento de Informática – PUC-Rio

Prof. Alberto Barbosa Raposo
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio

Prof. Marcelo Gattass
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio

Prof. Jônatas Wehrmann
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio

Prof. André Bulcão
Petróleo Brasileiro – Petrobras

Prof. Julio Cesar Duarte
Instituto Militar de Engenharia – IME

Rio de Janeiro, March 24th, 2022

Antonio José Grandson Busson

Antonio José Grandson Busson is a researcher and software developer at TelemidiaLab/PUC-Rio. His research interests are Artificial Intelligence and Multimedia Systems, working mainly on the following topics: Analysis, Coding Processing and Streaming of Multimedia Data, Multimedia Big Data, Social Network Analysis, Hypermedia Document Models, and iDTV. Antonio Busson received his B.S. degree (2013) and M.S. degree (2015) in Computer Science from the Federal University of Maranhão (UFMA).

Bibliographic data

Busson, Antonio José Grandson

A Self-supervised Method for Blind Denoising of Seismic Shot Gathers / Antonio José Grandson Busson; advisor: Sérgio Colcher. – Rio de Janeiro: PUC-Rio, Departamento de Informática, 2022.

v., 85 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Seismogram Denoising. 3. Deep Learning. 4. Blind-denoising. 5. Geophysical data. 6. Seismic Data Generation.. I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To everyone who encourages me in this journey: my family, my advisor, and my dear friends.

Acknowledgments

First and foremost, I dedicate this work to my sons, Dante and Antoni. I also want to thank my wife, Camila, who gave me support and help during much of my doctorate studies.

I want to express my sincere gratitude to my advisor Sérgio Colcher, for his encouragement, immense knowledge, and advice. His guidance over the past four years was fundamental to the conclusion of this thesis. Without a doubt, he contributed a lot to shape the professional that I am today.

Besides my advisor, I would like to thank Alberto Raposo, Marcelo Gattass, Jônatas Wehrmann, André Bulcão, and Julio Duarte for participating in the thesis committee. Their encouragement and insightful comments helped me improve the thesis's presentation quality.

I would like to thank my fellow labmates in Telemidia: Álvaro Veiga, Alan Guedes, Pedro Almeida, Paulo Mendes, Gabriel Noronha, Daniel Moraes, Polyana Costa, Arthur Serra, Eduardo Silva, Ivan Pereira, Guilherme Marques, Bruno Rocha, Matheus Adler, João Vale, Pedro Cutrim, e José Boaro. They have built a welcoming environment where everyone can help each other, ideal for teamwork.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Abstract

Busson, Antonio José Grandson; Colcher, Sérgio (Advisor). **A Self-supervised Method for Blind Denoising of Seismic Shot Gathers**. Rio de Janeiro, 2022. 85p. Tese de doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In the last years, the geophysics community has been devoted to seismic data quality enhancement by noise attenuation and seismogram interpolation using CNN-based methods. Discriminative CNN-based methods can achieve state-of-the-art denoising results. However, they do not apply to scenarios without paired training data (i.e., noisy seismic data and corresponding ground-truth noise-free seismic data). In this work, we treat seismic data denoising as a blind denoising problem to remove unknown noise from noisy shot gathers without ground truth training data. The basis used by the denoiser model is learned from the noisy samples themselves during training. Motivated by this context, the main goal of this work is to propose a self-supervised method for blind denoising of seismic data, which requires no prior seismic signal analysis, no estimate of the noise, and no paired training data. Our proposed self-supervised method assumes two given datasets: one containing noisy shot gathers and the other noise-free shot gathers. From this data, we train two models: (1) Seismic Noise Transfer (SNT), which learns to produce synthetic-noisy shot gathers containing the noise from noisy shot gathers and the signal from noise-free shot gathers; And (2) Seismic Neural Denoiser (SND), which learns to map the synthetic-noisy shot gather back to original noise-free shot gather. After training, SND alone is used to remove the noise from the original noisy shot gathers. Our SNT model adapts the Neural Style Transfer (NST) algorithm to the seismic domain. In addition, our SND model consists of a novel multi-scale feature-fusion-based CNN architecture for seismic shot gather denoising. Our method produced promising results in a holdout experiment, achieving a PSNR gain of 0.9 compared to other state-of-the-art models.

Keywords

Seismogram Denoising; Deep Learning; Blind-denoising; Geophysical data; Seismic Data Generation.

Resumo

Busson, Antonio José Grandson; Colcher, Sérgio. **Um Método Autosupervisionado para Atenuação Cega de Ruídos de Sismogramas**. Rio de Janeiro, 2022. 85p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Nos últimos anos, a geofísicos tem se dedicado ao aprimoramento da qualidade dos dados sísmicos por meio da atenuação de ruído e interpolação de sismogramas usando métodos puramente baseados em CNN. Métodos baseados em CNN podem alcançar resultados estado-da-arte para remoção de ruídos. No entanto, eles não se aplicam a cenários sem dados de treinamento emparelhados (ou seja, dados sísmicos ruidosos e dados sísmicos sem ruído correspondentes). Neste trabalho, tratamos a atenuação de ruídos de dados sísmicos como um problema de atenuação de ruído cega, que consiste em remover ruídos desconhecidos sem dados pareados. Em outras palavras, a base usada pelo modelo de denoiser é aprendida a partir das próprias amostras ruidosas durante o treinamento. Motivado por este contexto, o principal objetivo deste trabalho é propor um método autosupervisionado para atenuação cega de dados sísmicos, que não requer análise prévia do sinal sísmico, nenhuma estimativa do ruído e nenhum dado de treinamento pareado. O método proposto assume dois conjuntos de dados: um contendo *shot gathers* com ruídos e o outro com *shot gathers* sem ruídos. A partir desses dados, treinamos dois modelos: (1) Seismic Noise Transfer (SNT), que aprende a produzir *shot gathers* com ruído sintético contendo o ruído dos *shot gathers* com ruído e o sinal dos *shot gathers* sem ruído; E (2) Sismic Neural Denoiser (SND), que aprende a mapear os *shot gathers* com ruído sintético de volta aos *shot gathers* sem ruído original. Após o treinamento, o SND sozinho é usado para remover o ruído das capturas ruidosas originais. Nosso modelo SNT adapta o algoritmo Neural Style Transfer (NST) ao domínio sísmico. Além disso, nosso modelo SND consiste em uma nova arquitetura CNN baseada em fusão de atributos em várias escalas para eliminação de ruído em *shot gathers*. Nosso método produziu resultados promissores em experimentos, alcançando um ganho de PSNR de 0,9 em comparação com outros modelos de última geração.

Palavras-chave

Atenuação de ruídos; Aprendizagem Profunda; Atenuação cega de ruídos; Geração de dados geofísicos.

Table of contents

1 Introduction **14**
1.1 Objectives 16
1.2 Contributions 18
1.3 Outline 18

2 Related Work **19**
2.1 Deep Learning Models for Seismic Denoising 19
2.2 Deep Learning Models for Seismic Data Generation 22

3 Method for shot gather blind denoising **25**
3.1 Shot Gathers Quality Classification 26
3.2 Data Preparation 27
3.3 Seismic Noise Transfer (SNT) 29
3.3.1 Theory 29
3.3.2 Backbone tuning for seismic domain 31
3.3.3 Generation 32
3.4 Seismic Neural Denoiser 34
3.4.1 Theory 34
3.4.2 Proposed CNN-based Architectures for Seismic Shot gather Denoising 35
3.4.2.1 Feature Pyramid Network (FPN) for Seismogram Denoising 35
3.4.2.2 Res-U-net 37

4 Experimentation **39**
4.1 Dataset 39
4.2 Seismic Noise Transfer (SNT) 40
4.2.1 Backbone tuning 40
4.2.2 SNT'S hyperparameters tuning 41
4.2.3 Generation of synthetic patches 42
4.3 Seismic Neural Denoiser (SND) 43
4.3.1 Metrics 43
4.3.2 Setup 44
4.3.3 Results 44
4.3.3.1 Den_Valid_01 set 44
4.3.3.2 Den_Valid_02 set and Den_Valid_03 set 46
4.4 Concluding Remarks 55

5 Conclusion **56**
5.1 Summary of the Contributions 56
5.2 Future Research 57

Bibliography **57**

A Fundamentals of Deep Convolutional Neural Networks **66**
A.1 Artificial Neural Networks 66
A.2 Backpropagation Algorithm 67

A.3 Convolutional Neural Networks 69
A.3.1 Convolution Layer 70
A.3.2 Pooling Layer 71
A.3.3 Inception 72
A.3.4 Residual Connections 74

B Seismic Shot Gather Noise Localization Using a Multi-Scale Feature-Fusion-Based Neural Network **77**
B.1 MobileNet+FPN backbone 77
B.2 Single Shot Multibox Detector 78
B.3 Focal Loss 79
B.4 Experimentation 80
B.4.1 Seismic shot-gather dataset for noise localization 81
B.4.2 Training configuration 81
B.4.3 Results 81
B.4.4 Concluding Remarks 83

List of figures

Figure 1.1 Examples of shot gathers images classified by a geophysicist as (A) “Good” and (B) “Bad”, according to their noise intensity. 14
Figure 1.2 The theoretical model of the proposed framework. 17

Figure 3.1 Overview of the proporsed method for blind denoising. 25
Figure 3.2 Examples of annotation with bounding boxes to specify noisy regions. 27
Figure 3.3 Shot gathers partition scheme. 28
Figure 3.4 (A) All shot gather patches are valid (blue); (B) Patches larger than the shot gather dimensions are considered invalid (red); (C) patches that cross the shot gather dimensions are invalid; (D) *invalid* patches become *restricted* after the zero-padding operation. 29
Figure 3.5 Comparison between NST and SNT. 30
Figure 3.6 NST architecture for seismogram generation. 31
Figure 3.7 Seismic feature representations learned by the VGG-16 network. 32
Figure 3.8 Examples of synthetic patches generated by NST Model. 33
Figure 3.9 The problem with using restricted noisy patches in the noise transfer process. 34
Figure 3.10 Patches of the $S_{denoised}$ set are arranged and merged to compose the denoised shot gather. 35
Figure 3.11 FPN architecture adapted for the seismic denoising task. 36
Figure 3.12 Res-U-Net architecture. 38

Figure 4.1 Three different types of *shot gathers* used to build the dataset 40
Figure 4.2 Iterative workflow of the SNT’S hyperparameters tuning. 41
Figure 4.3 Generated patches with their corresponding hyperparameters configuration. 42
Figure 4.4 Convergence curve of FPN RFB-TDM, Res-U-Net, and FPN RFB-RON on the “Den_Valid_01” set. 45
Figure 4.5 An example of a shot gather from “Den_Valid_01” set denoised by the FPM RBF-TDM network. 47
Figure 4.6 Examples of patches from “Den_Valid_01” set denoised by the FPM RBF-TDM network. 48
Figure 4.7 Example of a seismic trace (with normalized signal value) from a shot gather of the “Den_Valid_01” set denoised by the FPM RBF-TDM network. 48
Figure 4.8 Convergence curve of Res-U-Net, U-Net, and Res-U-Net v2 on the “Den_Valid_02” set. 49
Figure 4.9 Convergence curve of Res-U-Net, U-Net, and Res-U-Net v2 on the “Den_Valid_03” set. 50

Figure 4.10 An example of a shot gather from “Den_Valid_02” set denoised by the Res-U-Net network. 51
Figure 4.11 An example of a shot gather from “Den_Valid_03” set denoised by the Res-U-Net network. 52
Figure 4.12 Examples of patches from “Den_Valid_02” set denoised by the Res-U-Net model. 53
Figure 4.13 Examples of patches from “Den_Valid_03” set denoised by the Res-U-Net model. 53
Figure 4.14 Example of a seismic trace (with normalized signal value) from a shot gather of the “Den_Valid_02” set denoised by the Res-U-Net network. 54
Figure 4.15 Example of a seismic trace (with normalized signal value) from a shot gather of the “Den_Valid_03” set denoised by the Res-U-Net network. 54

Figure A.1 Perceptron’s structure. 66
Figure A.2 (A) Multiclass Perceptron; (B) Multilayer Perceptron. 67
Figure A.3 Activation functions. 68
Figure A.4 Example of a convolutional process. 71
Figure A.5 Example of max-pooling process with *kernel* 2x2 and *stride* 2. 72
Figure A.6 Example of a CNN architecture. 72
Figure A.7 Inception module. 73
Figure A.8 Three main inception modules of the InceptionNet v2 architecture. 74
Figure A.9 IR-A, IR-B and IR-C: Three types of Inception module introduced by InceptionNet v4. 74
Figure A.10 Residual building block used by ResNet. 75
Figure A.11 IR-A, IR-B, and iR-C: Three types of Inception-Resnet modules used by Inception-Resnet v1 and v2 architecture. 75
Figure A.12 Global residual learning. 76

Figure B.1 Overview of proposed network for noise localization in seismic shot-gather images. 77
Figure B.2 MobileNet+FPN Network. 84
Figure B.3 Prediction examples of the MobileNet+FPN+FocalLoss on test set. 85

List of tables

Table 2.1 Comparative table of deep learning-based models for seismic denoising task. 22
Table 2.2 Comparative table of deep learning-based models for seismic data generation task. 24

Table 4.1 Description of the dataset used for experimentation. 39
Table 4.2 Results comparison on set “Den_Valid_01” 45
Table 4.3 Results comparison on set “Den_Valid_02” 49
Table 4.4 Results comparison on set “Den_Valid_03” 50

Table B.1 Results of basic backbone models on validation set. 81
Table B.2 Results of scenario with MobileNet supplemented by FPN and Focal Loss on validation set. 82
Table B.3 Results of the best model on test set 82

A man's dream will never die!

Eiichiro Oda, *One Piece.*

1
Introduction

A classical challenge in geophysics consists in correctly estimating Earth's subsurface characteristics based on measurements acquired by sensors on the surface. One of the most widely used methods for this estimation involves seismic reflection, in which seismic waves are produced using controlled active sources on the surface (e.g., dynamite explosions in land acquisition or air guns in marine acquisition), followed by the collection of the reflected data with sensors located above the area (e.g., geophones or hydrophones). Each survey of these seismic sources is known as a *shot*. The recorded wave at a receiver on the surface is called a *trace*. A proper arrangement of these shots produces a *shot gather*, which is an image that can be used as a visual representation of the characteristics of the actual Earth's subsurface (Duarte et al., 2014). Figure 1.1 shows examples of shot gathers, in which each column corresponds to a seismic trace recorded during the same seismic shot. The abscissa here stands for the position of the sensor relatively to the shot position (offset).

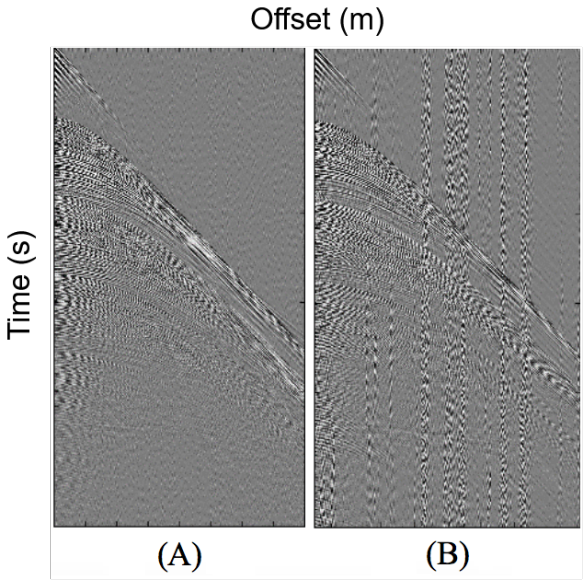

Figure 1.1: Examples of shot gathers images classified by a geophysicist as (A) "Good" and (B) "Bad", according to their noise intensity.

Recorded seismic signals are inevitably contaminated by noise and non-seismic signals from various sources including ocean waves, wind, traffic, instrumental noise, electrical noise etc. The diversity of noise types often makes

separation of signal and noise a challenging process. Spectral filtering, usually based on the Fourier transform (Naghizadeh, 2012), the time-frequency peak filtering (Wu et al., 2011; Liu et al., 2014), different curvelet-like transforms (Herrmann and Hennenfent, 2008; Gan et al., 2015; Zhang and Ulrych, 2003) are frequently used to suppress noise in routine seismic data processing; however, these approaches are not very effective when noise and seismic signal occupy the same frequency range (Zhu et al., 2019).

Convolutional Neural Networks (CNNs) techniques can be valuable tools in accomplishing many tasks related to image processing. Rather than using various inefficient visual quality control techniques, CNN-based methods can be trained to learn how to remove noise without degrading the essential seismic signal data. Shot-gather data is usually acquired by a sequence of spaced sensors on the surface; in such a setting, columns in a shot gather image are often organized following the same order of the positioning of sensors, resulting in images where the abscissa represents the position of the corresponding sensor relative to the shot position (Duarte et al., 2014). As a consequence, a strong spatial correlation between columns in those images can be observed, which in turn motivates the investigation about whether machine learning techniques commonly applied to plain 2D images (such as those based on CNNs) could be readily applied in accomplishing the denoising task. Recently, CNNs have also been applied to other problems pertaining to seismic imaging, such as seismic texture classification (Chevitarese et al., 2018), seismic facies classification (Zhao, 2018), seismic fault detection (Pochet et al., 2019), and salt segmentation (Shi et al., 2019).

In the last three years, the geophysics community has been devoted to the problem of seismic data quality enhancement by noise attenuation and seismogram interpolation using CNN-based methods. Wang et al. (2018a,b) introduced an eight-layer residual learning network based on ResNet (He et al., 2016) to interpolate seismic data without aliasing. Mandelli et al. (2019) investigated a CNN architecture called U-Net (Ronneberger et al., 2015a) for noise attenuation and reconstruction of missing traces in pre-stack seismic gathers. Richardson and Feller (2019) investigated an U-Net architecture, based on a pretrained ResNet, and used information from multiple adjacent gathers to produce an efective tool for denoising seismic data. Sun et al. (2020) proposed the use of a customized U-Net design with element-wise summation as part of the skip-connection to create an end-to-end network for seismic denoising. Zhao et al. (2018) used the DnCNN (Zhang et al., 2017) network to suppress low-frequency noise in desert seismic data.

Discriminative CNN-based methods, such as those previously cited, can

achieve state-of-the-art denoising results. However, they do not apply to scenarios without paired training data (i.e., noisy seismic data and corresponding ground-truth noise-free seismic data). In this work, we treat seismic data denoising as a *blind denoising problem* in which we aim to remove unknown noise from noisy shot gathers without any ground truth training data. The basis used by our denoiser model is learned from the noisy samples themselves during training.

1.1 Objectives

The overall aim of this work is *to investigate how to perform denoising of seismic shot gathers in scenarios with lack of paired training data*. More specifically, the main goal is to propose a self-supervised method for blind denoising of seismic data which requires no prior seismic signal analysis, no estimate of the noise, and no paired training data.

Following the degradation model $y = x + v$, denoising of shot gathers targets at recovering a noise-free data x from its noisy observation y by removing the noise v. As illustrated in Figure 1.2, the proposed self-supervised method is based on the following steps: given a *noise-free shot gather* ($y_1 = x_1$) and a *noisy shot gather* ($y_2 = x_2 + v_2$), in the *training step*, a function F can learn to generate a *synthetic shot gather* ($y_3 = x_1 + v_2$) that simultaneously matches the signal data of the *noise-free shot gather* and noise data of the *noisy shot gather*; next, given the *synthetic shot gather*, a function G can learn to restore the *noise-free shot gather* by removing the transferred noise v_2; after training, function G alone can be used to remove the noise v_2 from an original *noisy shot gather* y_2.

The proposed self-supervised method is composed of two modules, which are based on the functions F and G mentioned beforehand. This proposal brings the following research question:

RQ1: *Is it possible to remove noise from noisy shot gathers in scenarios with a lack of paired training data using modules based on the F and G functions?*

The module F is intended to synthesize a seismic shot gather that contains the combination of a *noise-free shot gather* signal and the noise of a *noisy shot gather*. This module address the second research question:

RQ2: *How is it possible to generate synthetic shot gathers by transferring noise from noisy shot gathers to clean shot gathers?*

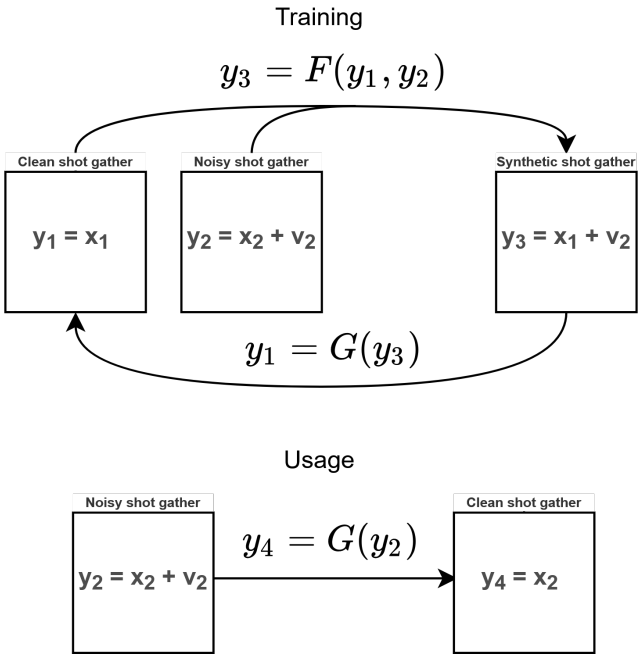

Figure 1.2: The theoretical model of the proposed framework.

Given the *synthetic shot gather* generated by module F, the module G aims to restore it to the *noise-free shot gather*. This module address the third research question:

RQ3: *How is it possible to restore the original signal of the clean shot gathers from the respective synthetic shot gathers generated by module F?*

1.2 Contributions

In summary, the thesis brings the following contributions:

1. A novel self-supervised method for blind denoising of seismic shot gathers in scenarios with a lack of training paired data which requires no prior signal analysis and no estimate of the noise;

2. A novel model to generate synthetic seismic data from real samples. Our proposed model called Seismic Noise Transfer is a custom implementation of the Neural Style Transfer algorithm adapted to the seismic domain;

3. Two novel architectures of CNN-based denoisers for seismic *shot gathers* with state-of-the-art performance.

1.3 Outline

The remainder of this thesis is organized as follows. **Chapter 2** discusses some related works and compares them with the proposals herein. Then, **Chapter 3** outlines the proposed method for blind denoising of seismic shot gathers. **Chapter 4** describes the experiments to evaluate the effectiveness of the proposal. And finally, **Chapter 5** concludes this work and presents some future work possibilities.

Additionally, Appendix A introduces the fundamentals of deep neural networks and CNNs, such as mathematical background, algorithms, and architectures. Next, Appendix B presents a model for noisy-region localization in shot gathers. It is a complementary model developed in this thesis and can be used to generate pseudo-labels for the proposed method.

2
Related Work

The sections that follow present two fields of work related to the scope of this thesis. First, Section 2.1 presents works that use deep models for seismic denoising. Next, Section 2.2 presents works that use deep models for seismic data generation.

2.1
Deep Learning Models for Seismic Denoising

Recent works in many different areas have been devoted to reduce signal artifacts or noise from different kinds os information or media by using Deep Learning-based methods. Many popular methods were originally designed for raster images and later adapted to the geophysical domain. The DnCNN (Zhang et al., 2017), for example, is one of the most prominent architectures that operate in the spatial domain. It is based on a residual learning strategy commonly applied to denoising, deblocking, and super-resolution tasks. Zheng et al. (2020) proposed a variation of the DnCNN to attenuate linear noise in seismograms. Because DnCNN was initially designed to attenuate the Gaussian noise of images, the results of suppressing linear noise in seismic data with the direct application of DnCNN were not satisfactory. So, they changed some network hyper-parameters like patch size and convolutional kernel number to fit the properties of linear noise. Regarding desert seismic data, Zhao et al. (2018) and Dong et al. (2019) also modified the DnCNN according to the characteristics of desert noise. They changed the convolution kernel size and network depth to make it suitable for low-frequency and non-Gaussian noise suppression, typically found in desert noise.

Dong et al. (2015b) followed a similar path with a model that aims at learning an end-to-end mapping between low and high-resolution images called SRCNN (Super-Resolution Convolutional Neural Networks). It is composed of three convolutional layers, each responsible for a specific task. The first layer extracts overlapping patches from the input and represents each of them as a high-dimensional vector; the second layer introduces a non-linear mapping between each high-dimensional vector to another high-dimensional one. Lastly, the third layer aggregates these patch-wise representations to generate the final

output. Related to SRCNN's applications in geophysics, Jun et al. (2020), for example, proposed a seismogram reconstruction method that uses the SRCNN network to predict the missing content of an incomplete seismogram. Their methods achieved competitive performance in comparison with the traditional f-x interpolation and POCS interpolation methods.

As an improvement to the SRCNN, the same authors proposed the AR-CNN (Artifacts Reduction Convolutional Neural Networks) (Dong et al., 2015a) aiming at reducing compression artifacts with the addition of one or more layers to clean noisy features. It also reuses features learned in a shallow network, transferring them to a deeper architecture with specific fine-tuning techniques. Results show the effectiveness of the approach in suppressing blocking artifacts and in maintaining edge patterns and sharp details. Additionally, Yu et al. (2016) modified the original model to speed up the inference process by inserting a "shrinking" layer between the first two layers, and by using large-stride convolution filters in the first layer (and the corresponding deconvolution filters in the last layer). According to the authors, this model, named Fast AR-CNN, can be 7.5 times faster than the original AR-CNN with almost no loss of quality.

Mandelli et al. (2019) proposed a U-Net-based method for reconstruction of corrupted seismic data, focusing on noise attenuation and interpolation of missing traces in the shot-gather domain. Their approach is capable of effectively and efficiently restore seismic corrupted data, and it is also able to deal with the task of spatially upsampling the shot gathers; Richardson and Feller (2019) used the U-Net combined with the ResNet pretrained on ImageNet for denoising and deblending of seismic data; Sun et al. (2020) proposed a variation of U-Net for marine seismic interference denoising. They propose a slight modification of the standard U-Net by replacing concatenation with element-wise summation at every skip–connection point. This avoids doubling the number of feature maps as well as the issue of vanishing gradients.

Busson et al. (2020b,a) proposed a U-Net model improved with DnCNN's global residual learn mechanism for predicting frequencies in media degraded by DCT-based compression. Their method can double the quality of degraded media by converting a media with a quality factor (QF) of 10 to slightly higher than 20. Zhang et al. (2020) applied the same strategy to eliminate low-frequency swell noise in marine seismic data. Their results show that the U-Net combined with DnCNN can efficiently learn and high-precise noise removal and avoid the overfitting problem, which is very common in CNN-based methods; Li and Ma (2021) proposed the DUDnCNN, a fusion of the DnCNN and U-Net with the addition of a dilation convolution operator. Their proposal has

presented near-optimal noise reduction in experimentation.

Fang et al. (2021) developed a seismic interpolator based on the U-Net combined with a texture loss, rather than only optimizing for reconstruction loss. The proposed texture loss ensures the accuracy of local structural information, which is calculated by a pre-trained texture extraction neural network. As a result, their proposed interpolator performed better than common DNN-based approaches that only use reconstruction loss. Li et al. (2021b) used the U-Net to multiple seismic removals in the context of seismic oil exploration. The inclusion of prediction and subtraction in the multiple removal methods requires adaptive subtraction to remove the complex differences between the actual and modeled multiples. Compared with traditional adaptive subtraction methods, such as linear regression (LR), their method has improved in the signal-to-noise ratio. Li and Ma (2021) proposed a U-Net-based model for simultaneous seismic image super-resolution and denoising. Their model uses a loss function that combines the l_1 loss and MS-SSIM loss to improve the perception of quality and alleviate overly smoothed geological edges. Their proposed method performs well on both synthetic and field seismic data.

Recent work takes advantage of multiscale feature fusion strategies. Zhao et al. (2019) proposed a method for low-frequency desert noise suppression based on a multiscale geometric analysis Convolutional Neural Network. Their method combines the Nonsubsampled Contourlet Transform (NSCT) and the DnCNN to compose the NC-CNN network. In experiments, their method achieved good results of noise suppression and signal recovery for low SNR seismic data; Yu et al. (2021) propose a noise attenuation method using the MSFN (Deep Multiscale Fusion Network). Their network uses Multiscale Fusion (MSF) block to adaptively exploit local signal features at different scales from seismic data. And then, a series of stacked MSF blocks are formed into MSFN, which can restore the noisy seismic data effectively and preserve more useful signal information.

Complementary to the proposed method of this thesis, we also proposed a method for seismic noise localization using a multiscale feature-fusion-based network inspired by the FPN (Feature Pyramid Network) (Lin et al., 2017a) (Appendix B). This model for noise localization can be used to generate valuable pseudo-labels for the proposed method, as explained in the next chapter. FPN shows significant improvement as a generic feature extractor in several applications, including object detection and semantic segmentation. We also developed adapted versions of the FPN for the seismic denoising task, obtaining state-of-the-art competitive results. Table 2.1 compares the deep learning-based models for seismic denoising tasks.

Table 2.1: Comparative table of deep learning-based models for seismic denoising task.

#	Author	Base	Improvements
01	Zhao et al. (2018)	DnCNN	- Customized hyperparameters for the seismic context
02	Dong. et al. (2019)	DnCNN	- Customized hyperparameters for the seismic context
03	Zhao et al. (2019)	DnCNN	- Nonsubsampled Contourlet Transform (NSCT)
04	Mandelli et al. (2019)	U-Net	- Customized hyperparameters for the seismic context
05	Richardson and Feller (2019)	U-Net	- Residual blocks (ResNet)
06	Zheng et al. (2020)	DnCNN	- Customized hyperparameters for the seismic context
07	Jun et al. (2020)	SRCNN	- Vanilla version
08	Sun et. al (2020)	U-Net	- Element-wise summation at skip-connections
09	Busson et al. (2020c,a) (Our)	U-Net	- Global residual learning - Fully convolutional
10	Zhang et al. (2020)	U-Net	- Global residual learning
11	Lia and Ma (2021)	U-Net	- Global residual learning - Dilated convolution
12	Fang et al. (2021)	U-Net	- Texture loss
13	Li et al. (2021)	U-Net	- Ensembles - Adptative subtraction
14	Yu et al. (2021)	MSFN	- Vanilla version
15	Busson et al. (Our)	FPN	- Multiscale denoising head - Reverse Fusion Block - TDM - Reverse Fusion Block - RON

2.2 Deep Learning Models for Seismic Data Generation

The works presented in last section are considered state-of-the-art in the context of quality enhancement of seismic data. However, such works are generally limited to supervised denoising scenarios. Oliveira et al. (2020) proposed a self-supervised method to attenuate ground-roll noise in seismic prestack images. In their approach, they use a CNN to detect ground-roll-affected areas, and then they filter ground-roll noise in the detected areas using cGAN (Isola et al., 2017). To train their cGAN-based denoiser, they

built a paired noisy/noise-free dataset using a frequency cut operation based on a pre-computed power spectrum analysis to synthesize synthetic patches from noisy and signal patches.

GAN-based methods have been widely used to generate missing seismic traces and improve the quality of the seismic datasets. Chang et al. (2018) proposed the SIGAN, a generative adversarial network for seismic data interpolation. Their discriminator model is a generic CNN-based classifier, while their generator model uses a ResNet-based network to reconstruct missing seismic traces. Going further, Chang et al. (2020) proposed the DD-CGAN, a dual-domain conditional generative adversarial network for seismic data interpolation. Their generator and discriminator networks use seismic data, and discrete Fourier transformed data in the frequency domain as input vectors. Recently, Wei et al. (2021) proposed the cWGAN, which consists of the cGAN combined with the Wasserstein distance. Their model can avoid gradient vanishing, and mode collapse is used in model training to improve the quality of the interpolated data. Their results on different seismic datasets show that the cWGAN with Wasserstein distance is effective for seismic data interpolation. Li and Wang (2021) proposed a cGAN-based denoising framework based on the data augmentation strategy. They introduced the RCGAN, which couples residual learning into the cycle-GAN to improve the training efficiency of the network. In experiments, they augment training samples by adding Gaussian noise with different levels of variance to noise-free data. Their results prove that RCGAN produces a good random noise suppression ability and a minimally damaging effect on proper seismic signals.

Besides the GAN-based methods, the Neural Style Transfer (NST) technique (Gatys et al., 2015) has also been adapted for the domain of geophysics. Ovcharenko et al. (2019) used the NST to generate realistically textured subsurface models based on synthetic prior models. They demonstrate examples where realistically random models are stylized to mimic texture patterns from Marmousi II (Martin et al., 2006) and a section from the BP 2004 benchmark velocity models (Billette and Brandsberg-Dahl, 2005). Takemoto et al. (2019) used the NST to produce synthetic data with noise characteristics extracted from a real dataset. Their results show that the stylized synthetic seismic data preserves the modeled content while incorporating characteristics of some real data chosen as style, creating synthetic data with realistic characteristics.

In this work, we opted to use the NST technique instead of the GAN to synthesize *shot gathers*. NST is a computationally cheaper alternative, and in addition, it generates appropriate content using fewer data. Notably, the NST

architecture is similar to the theoretical model presented in Section 1.2, which naturally facilitates its adaptation to our proposed method.

Unlike the works cited, our adaptation of the NST goes beyond the application of its vanilla version. We changed the loss function and added weights to the NST style transfer layers to allow for more customization of the synthetic *shot gathers*. In addition, we also propose an NST hyper-tuning method to find appropriate weight settings so that it is possible to generate synthetic images more similar to real ones.

Finally, we developed two new CNN-based denoiser architectures that learn the inverse function of SNT. In other words, they learn to map the synthetic images back to the original, thereby learning to remove transferred noise. The following chapter describes the proposed method in detail. Table 2.2 compares the deep learning-based models for seismic data generation tasks.

Table 2.2: Comparative table of deep learning-based models for seismic data generation task.

#	**Author**	**Base**	**Improvements**
01	Chang et al. (2018)	SIGAN	- Vanilla version
02	Ovcharenko et al. (2019)	NST	- Vanilla version
03	Takemoto et al. (2019)	NST	- Vanilla version
04	Chang et tal. (2020)	DD-CGAN	- Vanilla version
05	Oliveira et al. (2020)	cGAN	- Pseudo-labels - Frequency cut operation based on a prior analysis
06	Wei et al. (2021)	cGAN	- Wasserstein distance
07	Li and Wang (2021)	cycle-GAN	- Data augmentation (gausian noise with different levels of variance)
08	Busson et al. (Our)	NST	- Pseudo-labels - Backbone tuning - Customized noise transfer loss - Seismic embeddings-based hypertuning

3
Method for shot gather blind denoising

Based on the scheme presented in Chapter 1, the proposed self-supervised method for blind denoising of seismograms was designed based on two operations: (1) Seismic Noise Transfer (SNT), which learns how to produce *synthetic noisy shot gathers* containing the noise from *noisy shot gathers* and the signal from *noise-free shot gathers*; and (2) Seismic Neural Denoiser (SND), which learns how to map *synthetic-noisy shot gathers* back to original *noise-free shot gathers*.

The overview of our proposed method is depicted in Fig. 3.1. Following the degradation model $y = x + v$, shot gather denoising targets at recovering a noise-free data x from its noisy observation y by reducing the noise v. During the training step, given patch pairs from a *noise-free shot gather* ($y_1 = x_1$) and a *noisy shot gather* ($y_2 = x_2 + v_2$), the SNT learns how to generate a *synthetic-noisy shot gather* ($y_3 = x_1 + v_2$) that simultaneously matches the signal data of the *noise-free shot gather* and the noise data of the *noisy shot gather*. Next, given the *synthetic-noisy shot gather*, the SND learns how to restore the *noise-free shot gather* by removing the transferred noise v_2. After training, SND alone is used to remove the noise v_2 from original *noisy shot gather* y_2, producing the denoised shot gather y_4.

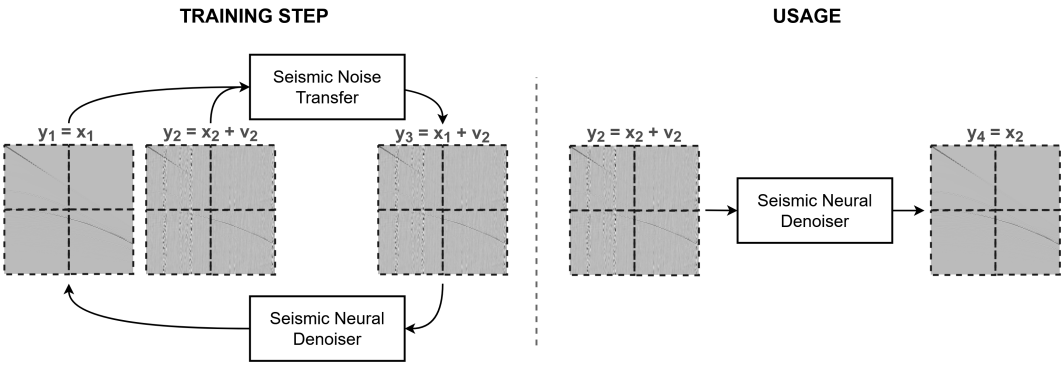

Figure 3.1: Overview of the proporsed method for blind denoising.

The complete pipeline consists of four stages, as follows:

1. **Shot gathers quality classification:** Given a collection of shot gathers, each shot gather is classified into two distinct categories according to its noise intensity: *clean*, or *noisy*;

2. **Data preparation:** Each shot gather is partitioned into non-overlapping patches of common size. Next, each patch is labeled according to source shot gather quality category;

3. **Seismic Noise Transfer (SNT)**

   3.1. **Backbone tuning:** The CNN used as the SNT's backbone is trained to learn relevant features from the seismic domain. It is trained in a discriminative way using paired patches from *clean* and *noisy* categories;

   3.2. **Synthetic Noisy Shot Gather Generation:** Given a list of paired *clean/noisy* patches, the SNT generates a list of *synthetic noisy* patches;

4. **Seismic Neural Denoiser (SND)**

   4.1. **Training:** The denoiser is trained to restore the *synthetic noisy* patches back to the original *clean* patches;

   4.2. **Denoising:** The trained denoiser is used to attenuate the noise from original *noisy* patches;

The four stages of the proposed method are detailed in the remainder of this chapter.

3.1 Shot Gathers Quality Classification

Given a collection of shot gathers (S), each shot gather $s \in S$ is defined as a 2-tuple $s \equiv< c, B >$, where: $c \in \mathbb{R}^n$ is the shot gather content, and B is a list of bounding boxes, in which each bounding box $b \in B$ represents noisy regions in the shot gather content. A bounding box b is simply defined as a 4-tuple $b \equiv< x, y, w, h >$, where the pair $< x, y >$ corresponds to the coordinate of the left lower corner of the bounding box, while w and h are the width and the height, respectively.

Figure 3.2 illustrates three different bounding box annotation examples. In the first one, a single bounding box is used to specify a *shot gather-level* annotation for the presence of noise in the entire shot gather. In this case, the list of bounding boxes B contains just a single box $b =< 0, 0, W, H >$, where W and H denotes the shot gather width and height; The second example is a noise-free shot gather, then the list of bounding boxes is empty; Finally, in the last example, there are four bounding boxes around specific noisy regions.

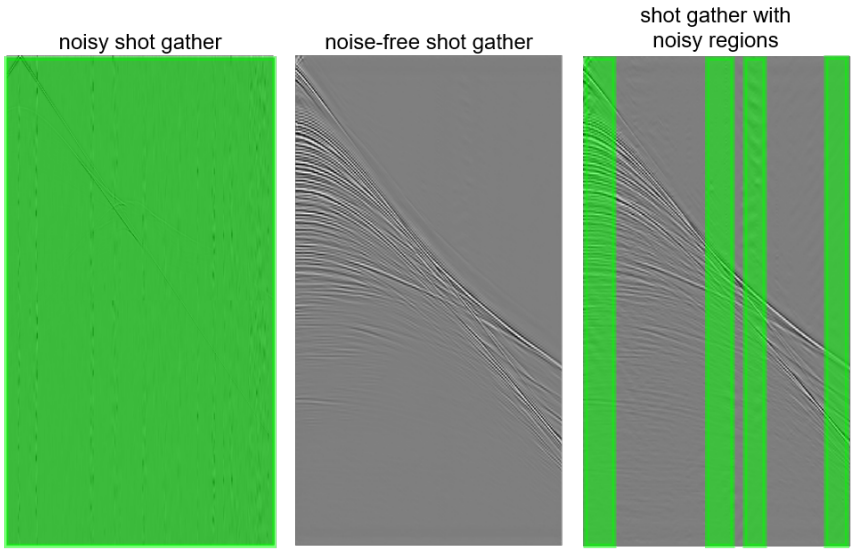

Figure 3.2: Examples of annotation with bounding boxes to specify noisy regions.

In this method, the annotated noisy regions (bounding boxes) are the minimum information necessary to perform denoising. Such annotation could be done by two main sources: (1) human experts – geophysicists – that analyze the shot gathers, recognize and mark the noisy regions manually; and (2) *pseudo labelling process*, in which a pre-trained ML model analyze the shot gathers and perform the annotation automatically.

Regarding the pseudo labeling strategy, some works that focused on analyzing the quality of seismic data can be used to classify the shot gathers. The network introduced by Betine Bucker et al. (2019), for example, is useful to perform *shot gather-level* quality classification. Additionally, another contribution derived from that work is a model for detecting noisy regions in *shot gathers* (Appendix B).

3.2 Data Preparation

Each shot gather s is partitioned into P non-overlapping patches of size $N \times N$. Each patch $p \in P$ is defined as a 3-tuple $< s.c_{(x,y,w,h)}, class, validity >$, where $s.c_{(x,y,w,h)}$ is the region content of the source shot gather s that is delimited by four spatial attributes: x (horizontal origin), y (vertical origin), w (patch width), and h (patch height); $class$ and $valid$ attributes define the signal quality and patch validity (auxiliary attribute used in the following stages).

Given a shot gather patch p, its $class \in \{clean, noisy\}$ attribute is calculated as follows:

$$class = \begin{cases} clean, & \text{if } \sum_{b \in s.B} b \cap s.con_{(x,y,w,h)} = 0 \\ noisy, & \text{otherwise} \end{cases}$$

where $s.B$ denotes the list of bounding boxes of the source shot gather s.

Figure 3.3 shows examples of a partition using the same example shown in Figure 3.2. Note that all patches that have intersection with any bounding box are classified as *noisy*, while patches that do not intersect with noisy bounding boxes are classified as *clean*.

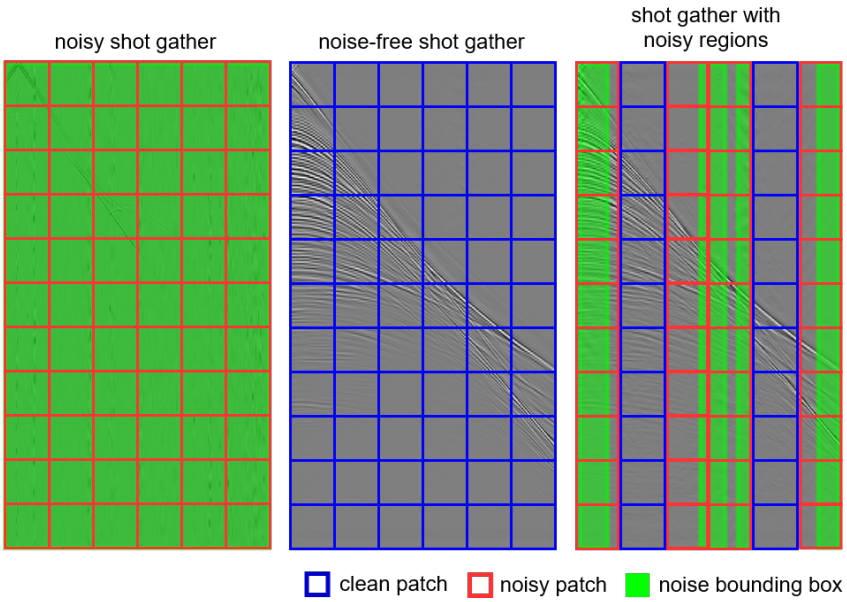

Figure 3.3: Shot gathers partition scheme.

The $validity \in \{valid, invalid, restricted\}$ attribute is obtained in two steps. First, patches that fall within the dimensional limits of the source shot gather are considered *valid*, while patches that cross these limits are considered *invalid*. More precisely, the validity is established as follows:

$$validity = \begin{cases} valid, & \text{if } (x + w) \leq s.w \wedge (y + h) \leq s.h \\ invalid, & \text{otherwise} \end{cases}$$

where $s.w$ and $s.h$ denote the width and height of the source shot gather s; x, y, w and h are the patch spatial attributes.

In the second step, the content of each *invalid* patch is extended with the zero-padding operation. Next, each *invalid* patch is reclassified as *restricted*. Figure 3.4 illustrates the examples of patch validity classification. In shot gather A, all patches are *valid* since all of them are within the shot gather dimension limits. Next, in shot gather B, all patches are classified as *invalid*, since its width are larger than the shot gather width. In shot gather C, only patches from the third column are considered invalid, as they cross the shot

gather limits. Finally, shot gather D exemplifies the zero-padding operation that transforms *invalid* patches of shot gather C into *restricted* patches.

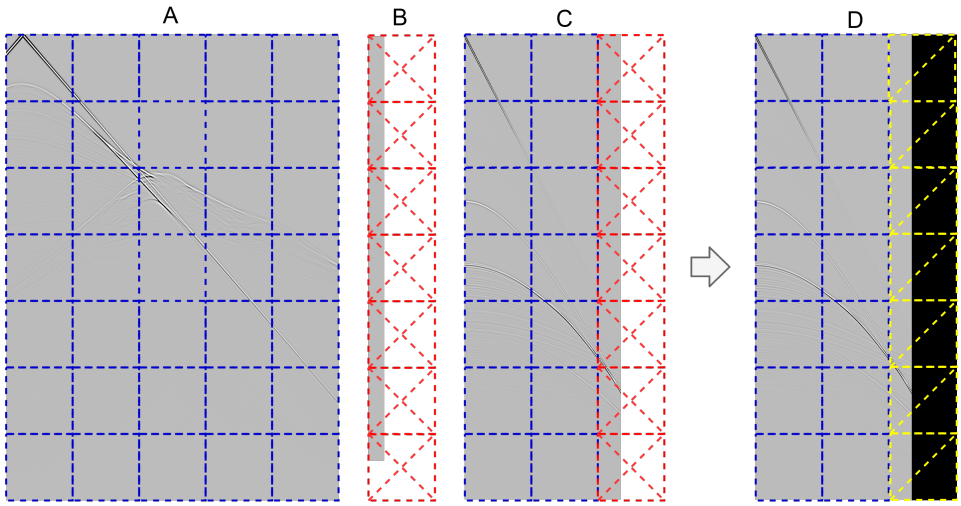

Figure 3.4: (A) All shot gather patches are valid (blue); (B) Patches larger than the shot gather dimensions are considered invalid (red); (C) patches that cross the shot gather dimensions are invalid; (D) *invalid* patches become *restricted* after the zero-padding operation.

3.3 Seismic Noise Transfer (SNT)

Given pairs of *clean* and *noisy* shot gather patches, the objective of the SNT stage is to generate a collection of *synthetic-noisy* patches. As Illustrated in Fig. 3.5, the proposed SNT model is inspired on the Neural Style Transfer (NST) algorithm (Gatys et al., 2015). The NST is a popular method for image manipulation to adopt another image's appearance or visual style. In the course of this section, we describe how the same style transfer principle for conventional images can be adapted to transfer noise between seismic shot gathers, despite the significant differences between both data types.

The remainder of this section is structured as follows. First, Subsection 3.3.1 presents the SNT theoretical model. Next, Subsection 3.3.2 presents the SNT's backbone tuning process. Finally, Subsection 3.3.3 presents the process of *synthetic-noisy* patch generation.

3.3.1 Theory

The NST process is defined by two distance functions L_{signal} and L_{noise} that describes how different the signal and noise of synthesized patch are from the original *clean* and *noisy* patches. Let C be a pre-trained CNN, then p be the *clean* patch's content, and x be the content of the patch that is synthesized.

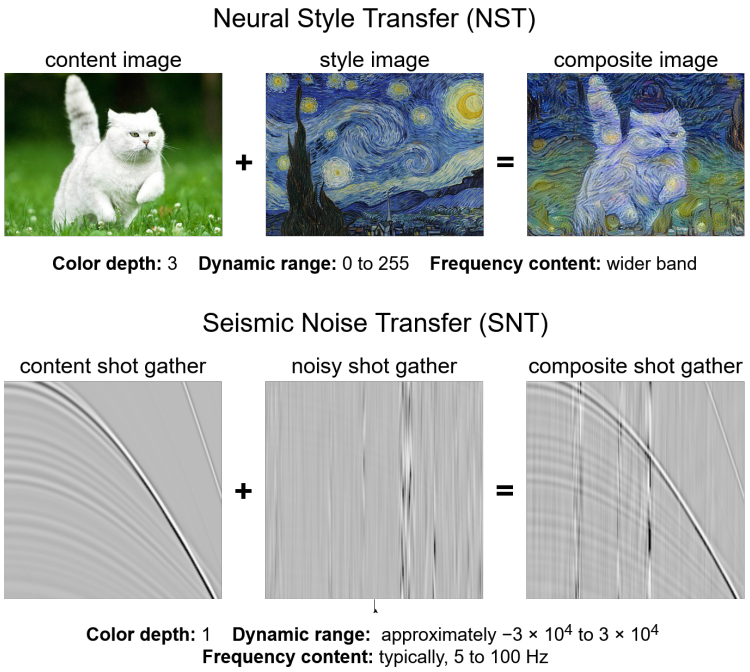

Figure 3.5: Comparison between NST and SNT.

Let $F^l_{ij} \in C(x)$ and $P^l_{ij} \in C(p)$ their respective feature representation in i^{th} filter at position j in layer l. The signal loss is described as the squared-error loss between the two feature representations:

$$L_{signal}(p, x, l) = \frac{1}{2}\sum_{i,j}(F^l_{ij} - P^l_{ij})^2 \tag{3-1}$$

The noise representation in layer l is given by the Gram matrix G^l, where $G^l_{ij} = \sum_k F_{ik}F_{jk}$ is the inner product between the vectorized feature map i and j in layer l. So let then a be the *noisy* patch's content, and x be the content of the patch that is synthesized. Let $A^l \in C(a)$ and $G^l \in C(x)$ their respective Gram matrix in layer l. The noise loss is described by mean-squared error of the Gram matrix of *clean* patch and the Gram matrix of the patch that is synthesized. The contribution of that layer to the total noise loss is then:

$$E_l = \frac{1}{4N_l^2M_l^2}\sum_{i,j}(G^l_{ij} - A^l_{ij})^2 \tag{3-2}$$

and the total noise loss is:

$$L_{noise}(a, x) = \sum_l^L w_l E_l \tag{3-3}$$

where N_l, M_l and w_l are respectively: the number of feature maps, the feature maps size (width × height) and the weighting factor of layer l.

The total loss function we minimise is:

$$L_{total}(p, a, x) = L_{signal}(p, x) + L_{noise}(a, x) \tag{3-4}$$

Figure 3.6 illustrates the SNT model. Using a pre-trained N-layered CNN with frozen weights. In the propagation stage, feature representations are generated in each layer for the *clean* (P), *noisy* (B), and *synthetic-noisy* (F) patches. By minimizing the distance between the feature representations of the clean and synthetic patches (in a specific layer), the synthetic patch, which starts with random values, begins to have a signal close to the clean patch signal after several backpropagation iterations. In parallel, by minimizing the distance between the Gram matrix of the noisy patch and the Gram matrix of the synthetic patch (in each layer), the synthetic patch begins to have similar noises to the noises in the noisy patch.

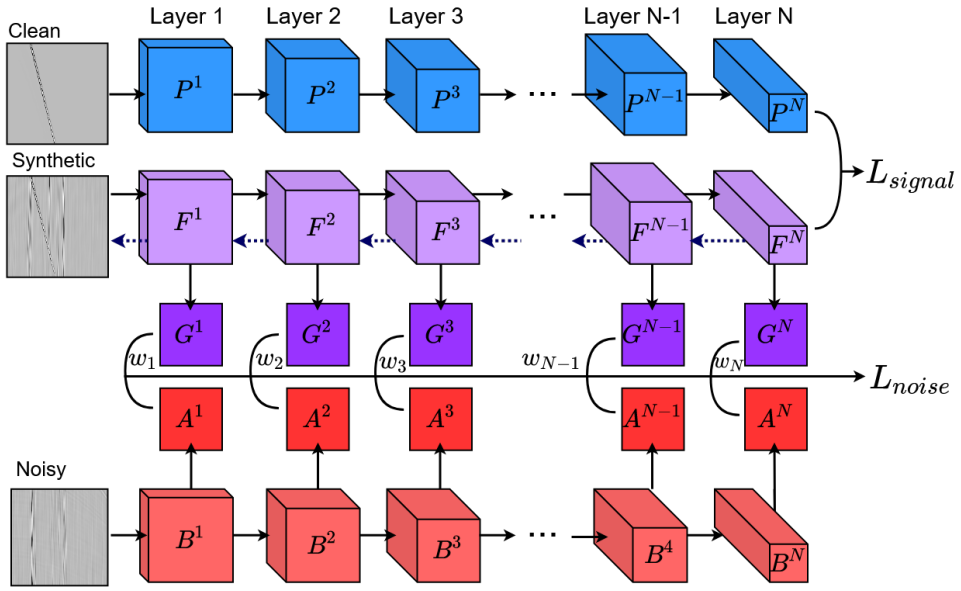

Figure 3.6: NST architecture for seismogram generation.

3.3.2 Backbone tuning for seismic domain

In this step, the CNN used as the SNT's backbone is trained to learn the seismic domain's relevant features. More precisely, the SNT's backbone must learn to generate features representations related to seismic signals and noise present in data. For this, *clean* and *noisy* patches are used to train the SNT's backbone as a binary classifier using the cross-entropy loss function:

$$L_{cross-entropy}(p, y) = -\frac{1}{N}\sum_{i}^{N} CE(p_i, y_i)$$
$$CE(p, y) = \begin{cases} log(p) & if \quad y = clean \\ log(1-p) & otherwise. \end{cases} \tag{3-5}$$

In the above $y \in \{clean, noisy\}$ specifies the ground truth class, while $p \in [0, 1]$ is the model's estimated probability for the class with label $y = clean$.

The SNT's backbone is trained repeatedly until it reaches the loss plateau. Figure 3.7 exemplifies seismic feature representations learned by the VGG-16 network. Given a patch that contains seismic signal and noise, it is possible to analyze which features properties are extracted according to the depth order of the VGG-16's layers. In the shallower layers (*conv_1_2* and *conv_2_2*), it is possible to observe low-level features. Next, in the deeper layers (*conv_3_4* and *conv_4_4*), these low-level features hierarchically compose higher-level features, which are used to discriminate the input patch into different classes. These deeper layers can distinguish the seismic signal from the noise, separating them into different higher-level feature maps.

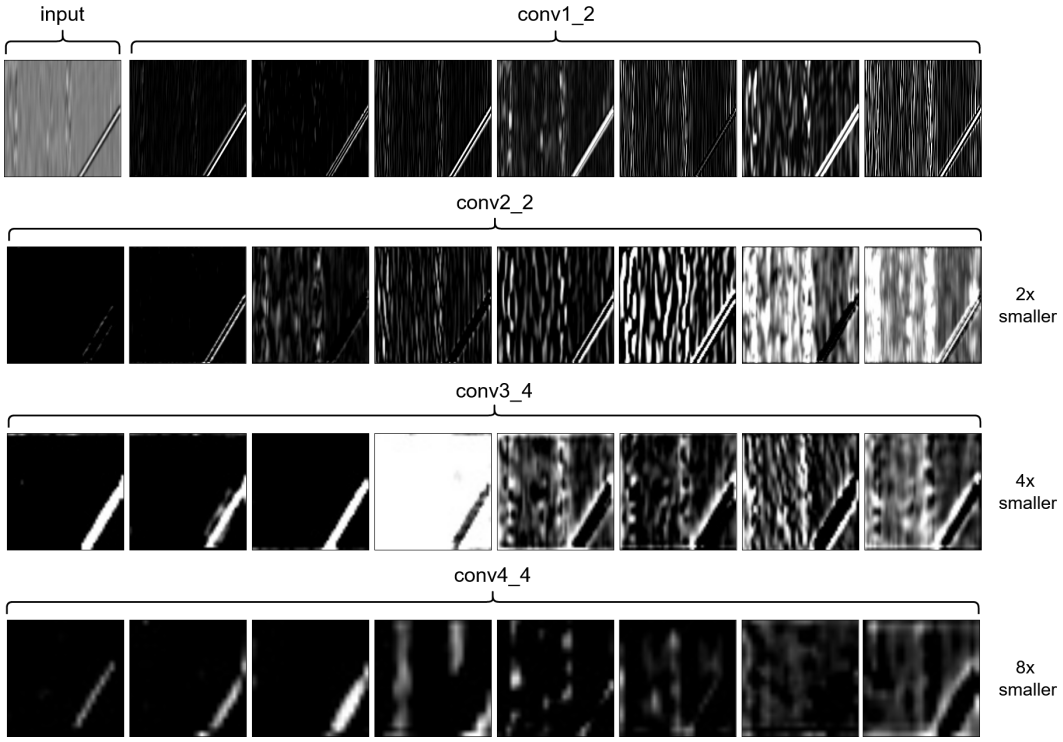

Figure 3.7: Seismic feature representations learned by the VGG-16 network.

3.3.3 Generation

At this stage, synthetic patches are generated. For this, the *clean* and *noisy* patches are separated into two distinct sets. The first one, called signal set (S_{signal}), contains all clean patches whose signal will be transferred to the synthetic patches:

$$S_{signal} = \{p \in P_i \mid p.class = clean\}\ i = 1, ..., N$$

where P_i denotes the partition set of the i-th shot gather, and N is the number of shot gathers in the collection.

The second set, called the noise set (S_{noise}), contains the *noisy* patches whose noise will be transferred to the synthetic patches. In this case, only valid patches are used for noise transfer, as restricted patches generate poor results due to zero-padding operation. The noise set is defined as follows:

$$S_{noise} = \{p \in P_i \mid p.class = noisy \wedge p.validity = valid\}\ i = 1, ..., N$$

The synthetic images are performed with the SNT model by combining a pair of patches, one from the S_{signal} set and another from the S_{noise} set. The set of synthetic patches is composed of 3-tuple elements, i.e.:

$$S_{synthetic} = \{< c = SNT(p, a), l_{signal} > \mid p \in S_{signal}, a \in S_{noise}\}$$

where: SNT denotes the function that generates the synthetic content c by combining a *clean* patch p and a *noisy* patch a; $l_{signal} \in S_{synthetic} \times S_{signal}$ is an association that reference the *clean* patch p.

Figure 3.8 shows examples generated by the NST model. Examples (A) and (B) show the result of seismic patch generation using a clean patch with a high-frequency signal combined with a noisy image with a low-frequency signal. The high-frequency signals from the clean patch have been preserved in the synthetic patch, and the transferred noise is similar to the original noise from the noisy patch. In example (C), the synthetic patch also preserves the low-frequency signals present in the clean patch. Finally, example (D) shows a scenario where there is a high-frequency signal in both clean and noisy patches. Note that in this case, the synthetic patch preserved only the signal of the clean patch.

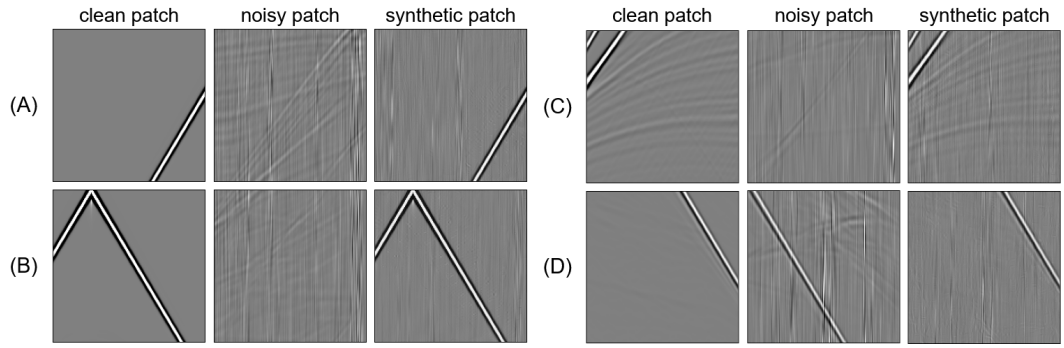

Figure 3.8: Examples of synthetic patches generated by NST Model.

Figure 3.9 shows the reason why restricted noisy patches cannot be used for noise transfer. In example (A), the clean patch is restricted, and the noisy

patch is valid. In this case, the SNT model generates a satisfactory synthetic patch. However, in example (B), when the clean patch is valid, and the noisy patch is restricted, the generation's result is poor. This problem occurs because the SNT model confuses the patch's noise with content filled by the zero-padding operation, producing anomalies that do not look like the original geophysical data.

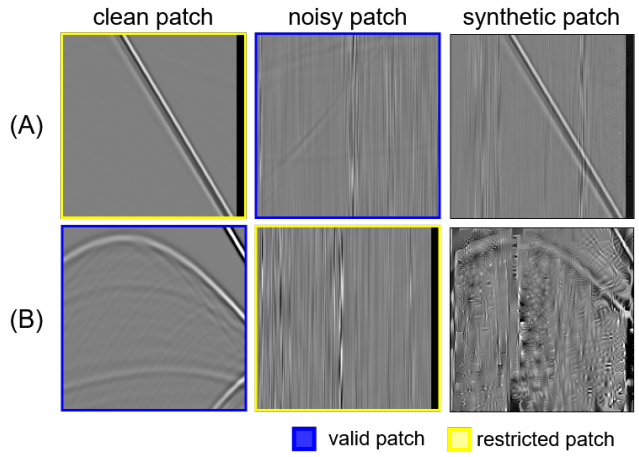

Figure 3.9: The problem with using restricted noisy patches in the noise transfer process.

Details about the CNN architecture used as the SNT's backbone and its hyper-parameters are presented and discussed in Chapter 4.

3.4 Seismic Neural Denoiser

At this stage, a CNN-based denoiser is trained to reverse the process done by the SNT, i.e., given a collection of *synthetic-noisy* patches; the model must learn to restore the original signal by removing the transferred noise in the SNT process. After learning to attenuate the synthetic noise, the model is used to attenuate the real noises present in the original *noisy* patches.

3.4.1 Theory

Let D be a CNN, the input of D is a synthetic patch $y = x + v$, where x and v represent the signal of the clean patch and the noise of the noisy patch used to create the synthetic patch. The Denoiser model aim to learn a mapping function $D(y) = x$ to predict the latent clean patch. For training, the averaged mean squared error between the original *clean* path and *synthetic-noisy* input is used:

$$L_{denoising}(x, y) = \frac{1}{N}\sum_{i}^{N}(D(x_i) - y_i)^2$$

where $x = s.c \mid s \in S_{synthetic}$ is the content of the *synthetic-noisy* patch s, $y = a(s.l_{signal})$ is the *clean* patch returned by function a (given the correspondent l_{signal} association), which was used to generate the content of the *synthetic-noisy* patch s.

After training, all patches contained in the noise set are processed by the denoiser D. Then, the resulting patches are placed in the denoised set:

$$S_{denoised} = \{D(s) \mid s \in S_{noise}\}$$

Finally, as shown in Figure 3, after the denoising stage, all patches of the $S_{denoised}$ set are merged to compose the denoised shot gathers.

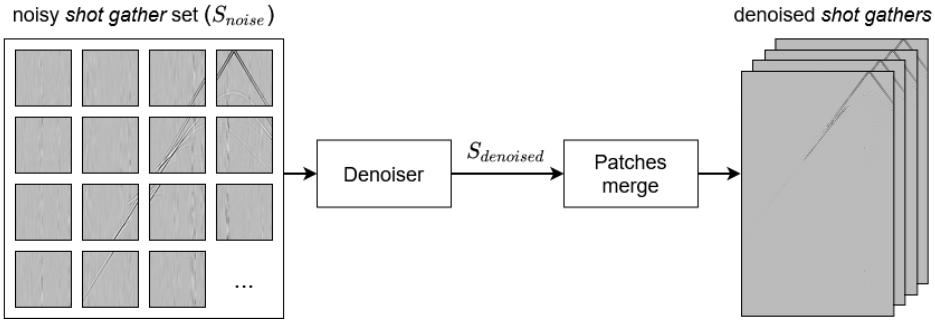

Figure 3.10: Patches of the $S_{denoised}$ set are arranged and merged to compose the denoised shot gather.

3.4.2
Proposed CNN-based Architectures for Seismic Shot gather Denoising

This section presents the models designed for the seismic denoising task. Section 3.4.2.1 presents the adapted FPN model and its variants. Next, Section 3.4.2.2 presents the Res-U-Net model.

3.4.2.1
Feature Pyramid Network (FPN) for Seismogram Denoising

Figure 3.11 illustrates the architecture of the adapted FPN (Lin et al., 2017a). It starts with a sequence of 5 convolutional layers, each with kernel size 3x3, batch normalization, ReLU activation, and is followed by max pooling. Next, in reverse order, a sequence of 3 RFBs (Reverse Fusion Blocks) performs the multiscale feature fusion of the hidden feature maps. Each RFB takes the feature map of a deeper layer (smaller dimensions) and merges them with the feature maps of a shallower layer (larger dimensions). Four branches receive

feature maps at different scales from the 3 RFBs and the deepest convolutional layer. Each branch is composed of 2 convolutional layers with kernel size 3x3, batch normalization, and ReLU activation. Finally, the four branches are scaled up to the input scale and then concatenated together. The last convolution has linear activation and produces the denoised seismic patch.

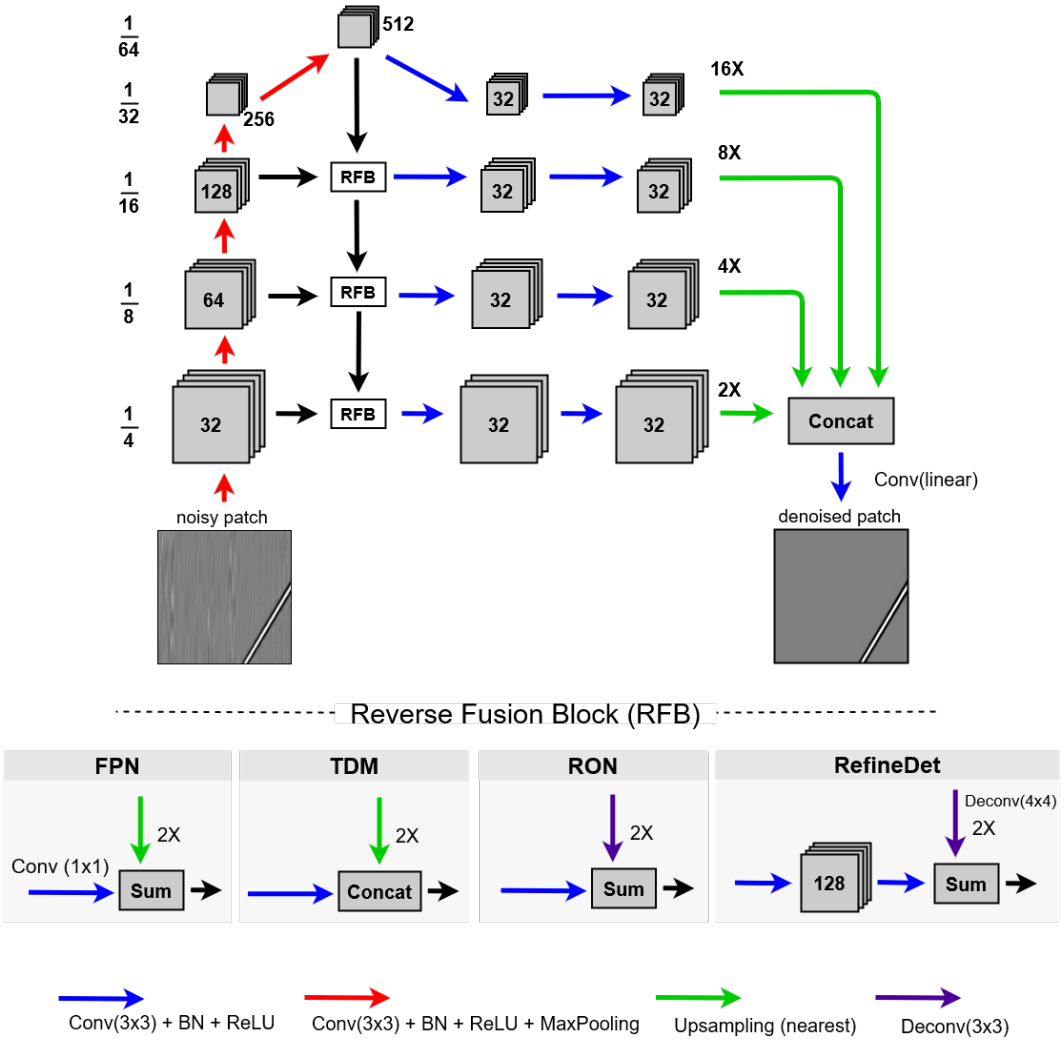

Figure 3.11: FPN architecture adapted for the seismic denoising task.

In this work, four variants of the FPN were developed based on different strategies for feature merging:

- Vanilla FPN: The nearest-neighbor interpolation operation scales up the feature map with the smallest dimension. At the same time, the lateral feature map goes through convolution with kernel size 1x1. Then, both feature maps are merged by element-wise sum operation.
- TDM (Shrivastava et al., 2016): The nearest-neighbor interpolation operation scales up the feature map with the smallest dimension. At the same time, the lateral feature map goes through convolution with

kernel size 3x3. Then, both feature maps are merged by concatenation operation.

– RON (Kong et al., 2017): The deconvolution operation scales up the feature map with the smallest dimension. At the same time, the lateral feature map goes through convolution with kernel size 3x3. Then, both feature maps are merged by element-wise sum operation.

– RefineDet (Zhang et al., 2018): The deconvolution operation scales up the feature map with the smallest dimension. At the same time, the lateral feature map goes through two convolution layers with kernel size 3x3. Then, both feature maps are merged by element-wise sum operation.

3.4.2.2 Res-U-net

Figure 3.12 illustrates the Res-U-Net architecture (Busson et al., 2020b,a), in which the authors have extended the plain U-Net (Ronneberger et al., 2015b) with the addition of a Global Residual Learning (GRL) mechanism.

The Res-U-Net uses convolutional layers with batch normalization, kernel size 3×3, and ReLU activation. Our implementation uses 11 convolutional layers. The first five are downsampled by a max pooling with a kernel size of 3×3 and stride 2. Starting at the seventh convolution layer, before each convolutional layer, an upsampling is applied by a transpose-convolution with a kernel of size 3×3, stride 2, and concatenated with the output of the correspondent convolutional layer of the first half of U-Net.

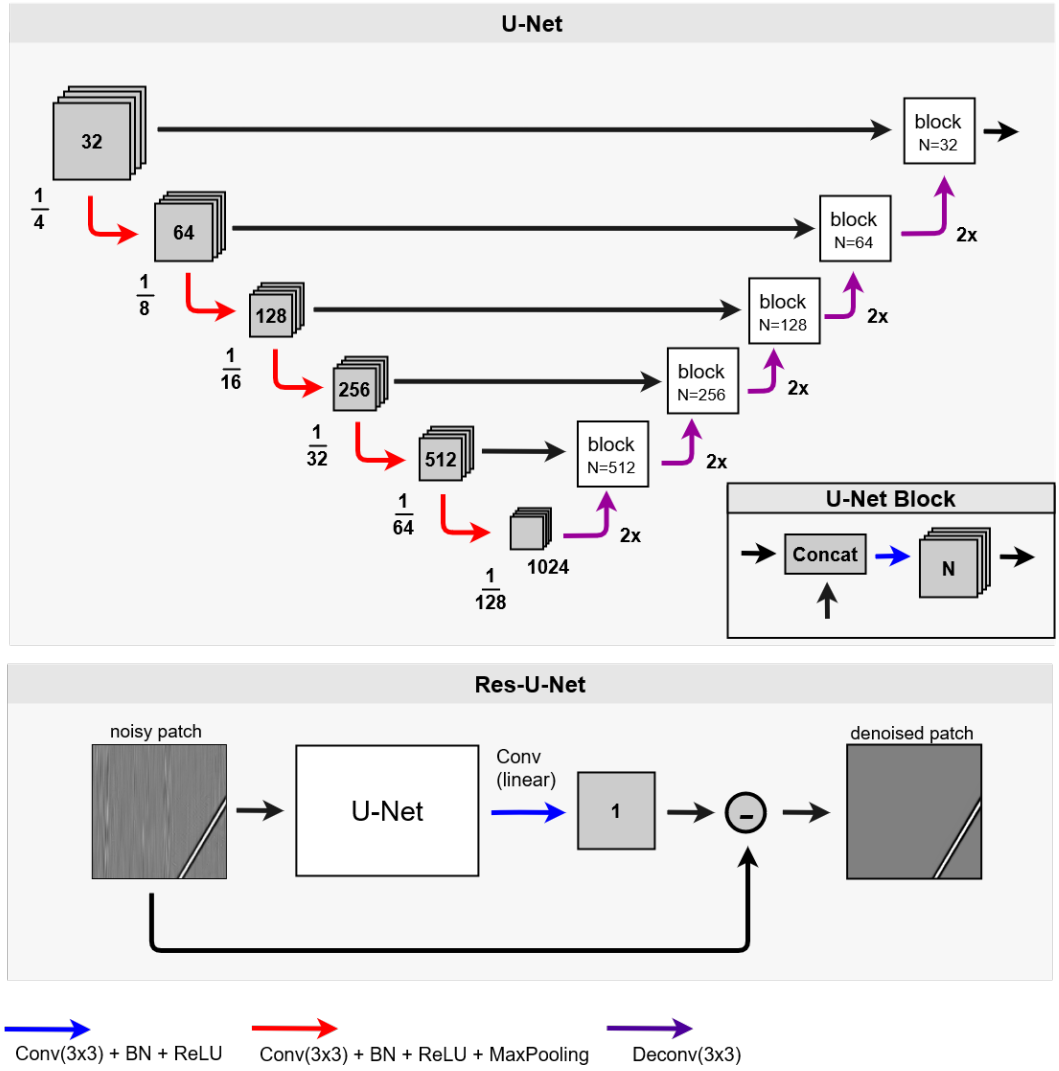

Figure 3.12: Res-U-Net architecture.

4
Experimentation

In this chapter, we evaluate the effectiveness of the self-proposed method for seismogram blind denoising. All source code of the models and experiments presented in this section are available in a git repository[1].

This section is structured as follows. First, Section 4.1 describes the dataset used in the experimentation. Next, Section 4.2 presents the scheme for generating shot gathers with SNT. Section 4.3 presents our empirical findings. Finally, Section 4.4 presents presents the experiment's final remarks.

4.1
Dataset

Table 4.1 details the sets that compose the dataset. The "SNT_Gen" set is used in SNT stage. And the other three sets, "Den_Valid_01", "Den_Valid_02", and "Den_Valid_03" are used as validation sets for the holdout cross-validation in the denoising stage. All these sets were produced by geophysicists from CENPES-Petrobras.

Table 4.1: Description of the dataset used for experimentation.

#	Set Name	Shot gathers Qtd.	Patches Qtd.	Type	Paired
01	SNT_Gen	400	10400	(A)	No
02	Den_Valid_01	100	2600	(A)	Yes
03	Den_Valid_02	40	2200	(B)	Yes
04	Den_Valid_03	100	2200	(C)	Yes

Each *shot gather* was partitioned into non-overlapping patches of common size of 200×200. In each set, half of all patches are classified as *clean*, and the others are *noisy*. The "SNT_Gen" set is the only one that does not have paired data, as SNT uses it to synthesize the patches used to train the denoiser. The other three sets are used to evaluate the denoiser during training. In addition, each of these sets is composed of a different type of *shot gather*, as illustrated in Figure 1. The "Den_Valid_01" set is composed of type *A*

[1] https://github.com/TeleMidia/Self_Supervised_Seismogram_Blind_Denoising

shot gather, the same type used in training. While the "Den_Valid_02" and "Den_Valid_03" sets are composed of types B and C, respectively. The validation sets were built with three different types of *shot gather* to estimate the generalization power of the proposed method.

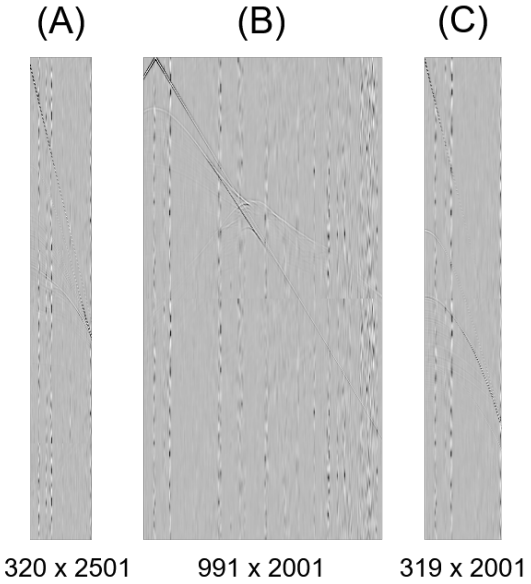

Figure 4.1: Three different types of *shot gathers* used to build the dataset

4.2 Seismic Noise Transfer (SNT)

The subsections that follow detail the experimentation with the SNT model. More specifically, Subsection 4.2.1 details the SNT's backbone tuning. Next, Subsection 4.2.2 presents the SNT hyperparameter search process. Finally, Subsection 4.2.3 describes the generation of the synthetic *shot gathers*.

4.2.1 Backbone tuning

The VGG-16 (Simonyan and Zisserman, 2014) network was used as the SNT's backbone. Average-pooling layers have replaced all VGG-16's max-pooling layers. Then, the VGG-16 network was trained to classify *clean* and *noisy* patches using the "SNT_Gen" set samples. The training was based on the Adam (Kingma and Ba, 2014) optimization with a momentum of 0.999, an exponential decay of 0.9 and epsilon of 1e-07, a batch normalization with a decay of 0.9997 and an epsilon of 0.001 with fixed learning rate of 0.001.

4.2.2 SNT'S hyperparameters tuning

In the SNT model, each CNN layer used for noise transfer has an associated hyperparameter. The selection of these hyperparameters is a challenge for the generation of appropriate seismic patches. Figure 4.2 illustrates the process of tuning these hyperparameters. After the synthetic patch generation, the seismic embeddings of the clean, noisy, and synthetic patches are extracted with the SNT's backbone. Then, the L2 distance measures the embeddings similarity between the *synthetic*, *clean*, and *noisy* patches. If the result is not satisfactory, a new hyperparameters configuration is inserted in the SNT model, and another iteration is started.

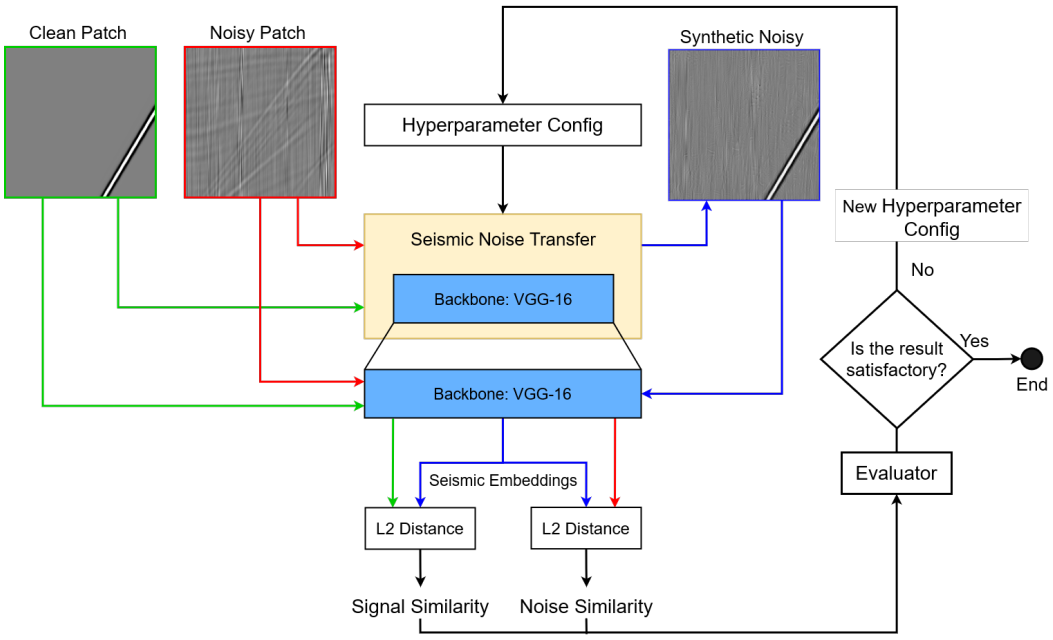

Figure 4.2: Iterative workflow of the SNT'S hyperparameters tuning.

Figure 4.3 illustrates generated examples with their corresponding hyperparameters configuration. The five numerical values (above the patches) correspond to the weights associated with the following VGG-16 layers: conv1_1, conv2_1, conv3_1, conv4_1, conv5_1. Below the patches are the embeddings distance between the *synthetic*, *clean*, and *noisy* patches. A trade-off analysis was carried out between these embedding distances to find out how each VGG-16's layers contribute to the generation of synthetic patches.

First, to understand what types of noise each layer can generate, each layer was activated at a time, as can be seen in examples (A), (B), (C), (D), and (E). Note that examples (A) and (B) use the shallower layers to synthesize patches. In this case, synthetic patches' noise does not look like real noise, as these layers produce low-level seismic noise features. In examples (C), (D), and

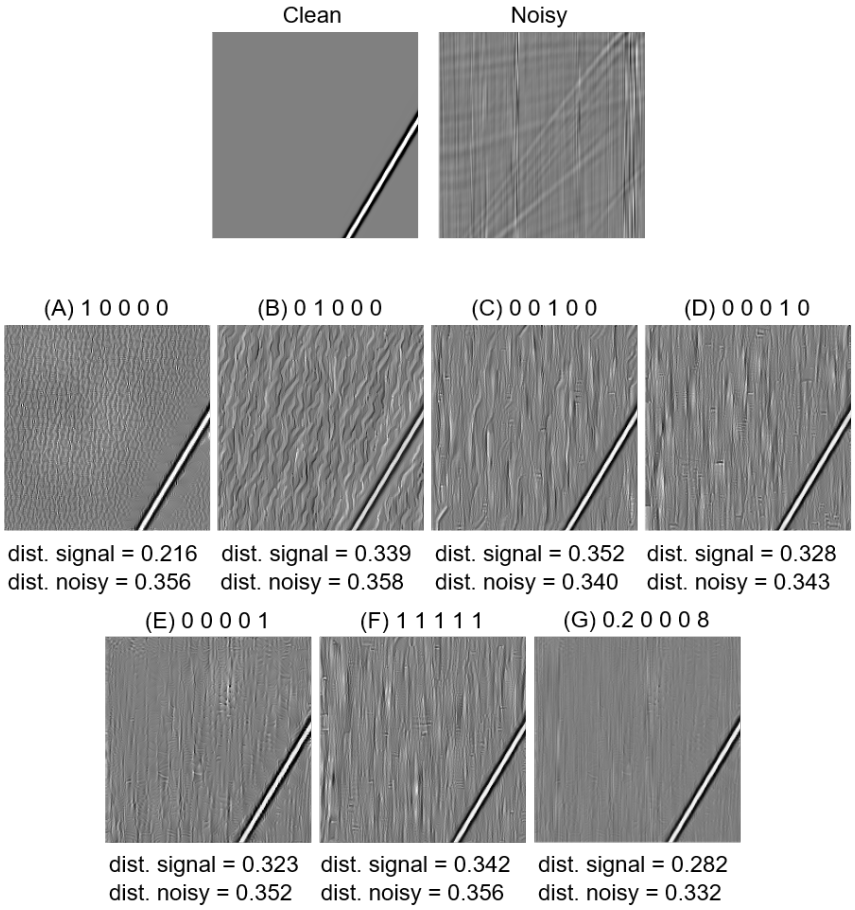

Figure 4.3: Generated patches with their corresponding hyperparameters configuration.

(E), it can be seen that when using the deeper layers, the synthesized noise has characteristics more similar to the real ones. However, still, the generated synthetic noise is unsatisfactory. Example (F) shows the synthesis result when using all layers, showing that all layers together do not produce a noise similar to the real one. However, using the workflow described above, it is possible to search for configurations of proper weights for the generation of appropriate shot gathers. Example (G) shows the weight configuration used to generate the synthetic patches in the experiment.

4.2.3 Generation of synthetic patches

The "SNT_Gen" set was used by the SNT model to generate 5200 synthetic patches. Also, for each *synthetic* patch, the reference to the corresponding *clean* patch used for its generation was stored. This reference is useful in denoiser training stage, which is described in the following section.

4.3 Seismic Neural Denoiser (SND)

This section describes the experiments with the SND module. The blind-denoising method proposed in this work is based on a cyclic flow, where a model learns to transfer noise from noisy samples to noise-free samples, and then a denoiser model learns to remove these same noises to restore the noise-free samples. Therefore, the denoising model is limited to the noises that the first model transferred. However, this experiment evaluates the denoiser model with samples from the same source used in training ("Den_Valid_01" set) and samples from different sources ("Den_Valid_02" and "Den_Valid_03" sets). The second aim of this experiment is to understand how the denoiser performs with unknown noise and signals.

Among the denoising architectures evaluated, those proposed in this work are described in Section 3.4.2. In addition, the proposed denoisers are compared with the other methods presented in Chapter 2 (Related Work).

The remainder of this section is detailed in subsections that follow. Subsection 4.3.1 discusses the selected metrics denoising evaluation. Next, Section 4.3.2 details the experiment setup. Finally, empirical findings and results are registered in Section 4.3.3.

4.3.1 Metrics

The PSNR (Peak Signal-to-Noise Ratio) and NRMSE (Normalized Root Mean Square Error) metrics are used to evaluate the denoising models. PSNR is a logarithmic scale (in decibels) of the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Given two samples x and y of size $m \times n$, the PSNR is calculated by

$$PSNR(x, y) = 10 \log_{10}(\frac{MAX^2}{MSE(x, y)}) \tag{4-1}$$

where MAX is the maximum possible singal value, and

$$MSE(x, y) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [x(i, j) - y(i, j)]^2 \tag{4-2}$$

The NRMSE consists of the square root of MSE normalized by the range (defined as the maximum value minus the minimum value) of the measured data. Given two samples x and y of a common size, the NRMSE is defined as

$$NRMSE(x, y) = \frac{\sqrt{MSE(x, y)}}{y_{max} - y_{min}} \tag{4-3}$$

with the MSE defined by (4-2).

4.3.2 Setup

The networks were trained using an octa-core i7 3.40 GHz CPU with a GTx-1070Ti GPU. The training was based on the Adam (Kingma and Ba, 2014) optimization with a momentum of 0.999, an exponential decay of 0.9 and epsilon of 1e-07, a batch normalization with a decay of 0.9997 and an epsilon of 0.001 with fixed learning rate of 0.001, and MSE (Mean Square Error) as the loss function.

4.3.3 Results

The subsections that follow present the experiment results with the three different validation sets. First, Subsection 4.3.3.1 presents results with the "Den_Valid_01" set. Next, Subsection 4.3.3.2 presents the results with the "Den_Valid_02" and "Den_Valid_03" sets.

4.3.3.1 Den_Valid_01 set

As illustrated in Table 4.2, with the "Den_Valid_01" set, the best result was achieved by FPN RFB-TDM, which produced a PSNR of 45.6545 and NRMSE of 0.0302, followed by Res-U-Net, FPN RFB-RON, Res-U-Net v2, FPN, U-Net v2, and other models. All networks based on multi-scale feature-fusion mechanisms (FPN and U-Net) achieved positions above the top-10. Highlighting that the top-4 models are novel denoiser models proposed in this work. The FPN variants that used the RBFs (Reverse Fusion Blocks) TDM and ROM achieve improvements over the vanilla FPN. In the scope of this experiment, the TOM and ROM strategy of using a lateral $3x3$ convolution before merging features works better than the traditional $1x1$ convolution. The global residual learning mechanism used in networks Res-U-Net and Res-U-Net v2 proved helpful. Their performance achieved considerable improvements over its vanilla versions (U-Net and U-Net v2).

Figure 4.4 shows the convergence curve of the top-3 networks. The Res-U-Net converged faster than the other networks. However, the FPN RFB-TDM network achieved its best result in epoch 76. Although, the convergence curve of Res-U-Net's has less variation than FPN RFB-TDM and FPN RFB-ROM networks.

Figure 4.5 illustrates an example of a *shot gather* that was denoised by FPN RFB-TDM network. In order, the first shot gather is the input, the second is the predicted shot gather, and the last is the reference shot gather. Note that

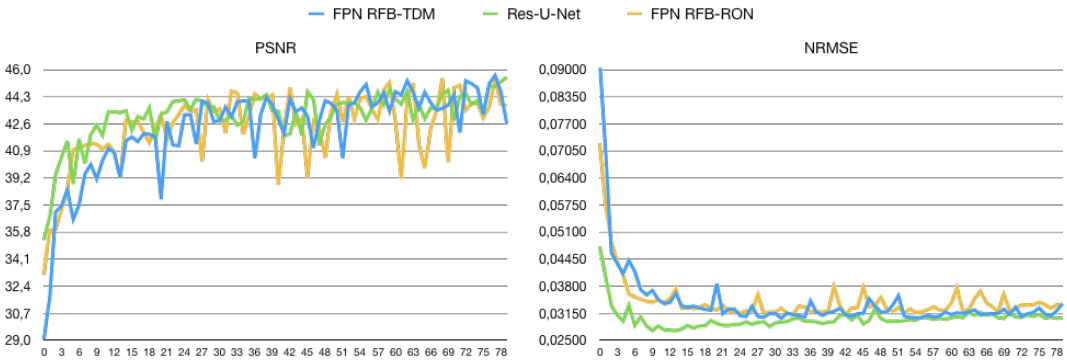

Figure 4.4: Convergence curve of FPN RFB-TDM, Res-U-Net, and FPN RFB-RON on the "Den_Valid_01" set.

Table 4.2: Results comparison on set "Den_Valid_01"

#	**Network**	**PSNR**	**NRMSE**
01	FPN RFB-TDM (Our)	45.6545	0.0302
02	Res-U-Net (Our)	45.5363	0.0273
03	FPN RFB-RON (Our)	45.4638	0.0313
04	Res-U-Net FullyConv (Our)	44.8942	0.0254
05	FPN (Lin et al., 2017a)	44.8364	0.0299
06	U-Net FullyConv (Ronneberger et al., 2015a)	44.1769	0.0248
07	U-Net (Ronneberger et al., 2015a)	43.4106	0.0270
08	FPN RFB-RefineDet (Our)	43.3361	0.0324
09	U-Net Mandelli (Mandelli et al., 2019)	37.9488	0.0450
10	SE-ResNet (Hu et al., 2018)	34.5819	0.0517
11	CANDI (Lee and Cho, 2020)	34.3299	0.0529
12	DnCNN (Zhang et al., 2017)	33.8975	0.0556
13	SR-ResNet (Ledig et al., 2017)	32.6526	0.0648
14	EDSR-ResNet (Lim et al., 2017)	32.4127	0.0725
15	ResNet (He et al., 2016)	29.7010	0.0868
16	AR-CNN (Dong et al., 2015a)	29.3711	0.0950
17	Deeper SR-CNN (Dong et al., 2015a)	29.2277	0.0930
18	Fast AR-CNN (Yu et al., 2016)	27.6337	0.1321
19	U-Net Sun (Sun et al., 2020)	26.2417	0.1981

practically all the noise has been attenuated, and most of the signal of interest has been preserved. However, some of the low amplitude signals have also been attenuated. In more detail, Figure 4.6 illustrates examples of denoised patches.

There is a high amplitude signal in the patch on the first row. In this case, the model works well. Note that the difference between the ground truth and the predicted signal is slight. There is a predominance of low amplitude signals in the patches of the second, third and fourth rows. Note that part of these signals was attenuated along with the noise, demonstrating limitations of the current model. Figure 4.7 shows the result of the denoiser in a single trace of a shot gather of the "Den_Valid_01" set to demonstrate this issue better. Note that the model can reconstruct the signal at the beginning of the trace, where there is a high amplitude signal. However, the model fails to reconstruct the signal with low amplitude, especially those below the 0.2 value.

4.3.3.2 Den_Valid_02 set and Den_Valid_03 set

Comparing with the previous experiment, all models had performance loss in the scenario with unknown seismic data. As illustrated in Table 4.3 and Table 4.4, with the validation sets "Den_Valid_02" and "Den_Valid_03", the best result in both sets was achieved by Res-U-Net, which produced a PSNR of 37.8728 and NRMSE of 0.0545 on "Den_Valid_02" set, and a PSNR of 37.6354 and NRMSE of 0.0510 on "Den_Valid_03" set. As shown in the convergence curve illustrated in Figures 4.8 and 4.9, the denoising model has been overfitted by the signal and noise characteristics of the training set. As discussed in the previous experiment, U-Net-based networks converged faster than FPN-based ones. That is why Res-U-Net had the best performance in the first few seasons, but its performance dropped throughout training.

Figures 4.10 and 4.11 show examples of *shot gathers* from "Den_Valid_02" and "Den_Valid_03" sets denoised by the Res-U-Net network. In more detail, Figures 4.12 and 4.13 show examples of denoised patches. And finally, Figures 4.14 and 4.15 show the result from the perspective of a single seismic trace.

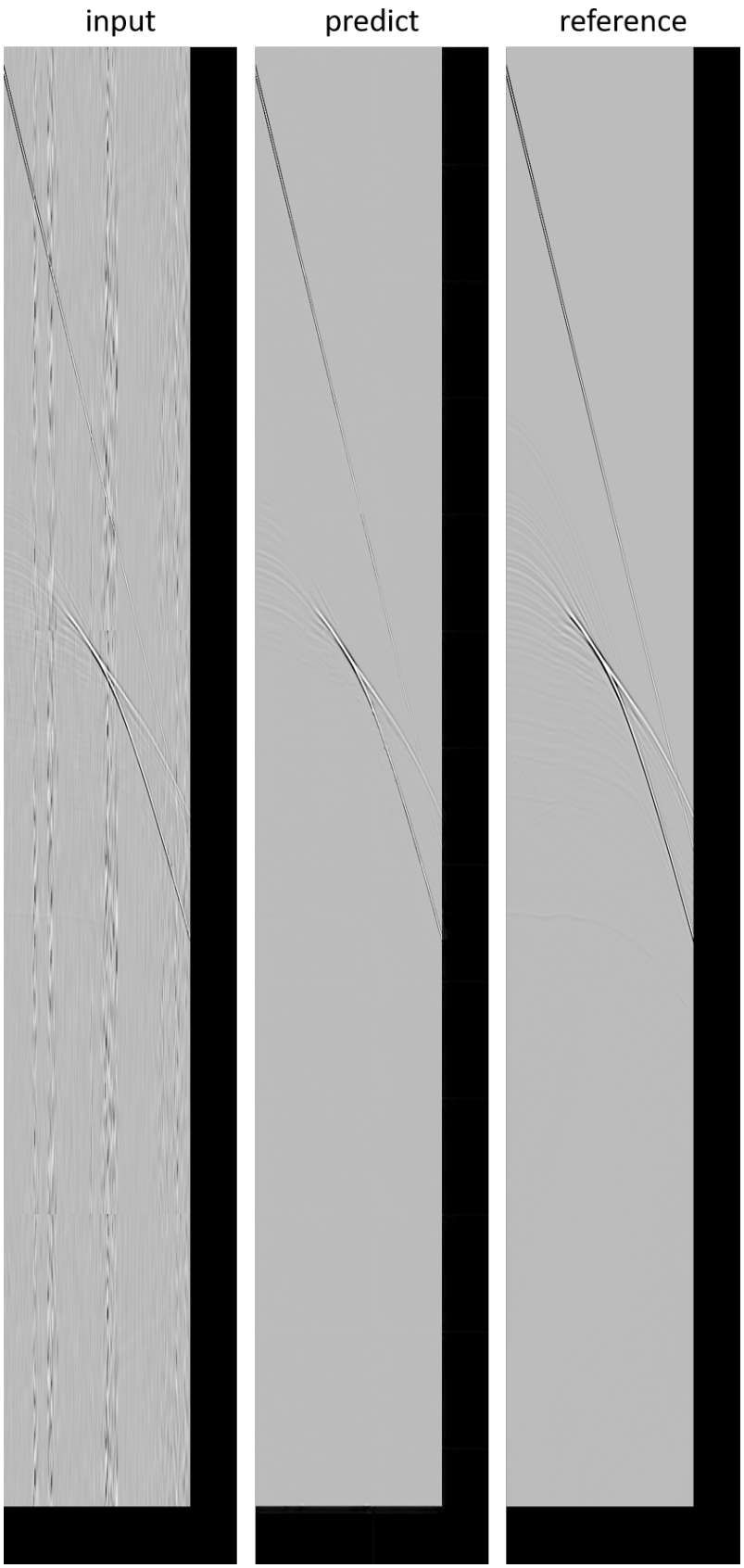

Figure 4.5: An example of a shot gather from "Den_Valid_01" set denoised by the FPM RBF-TDM network.

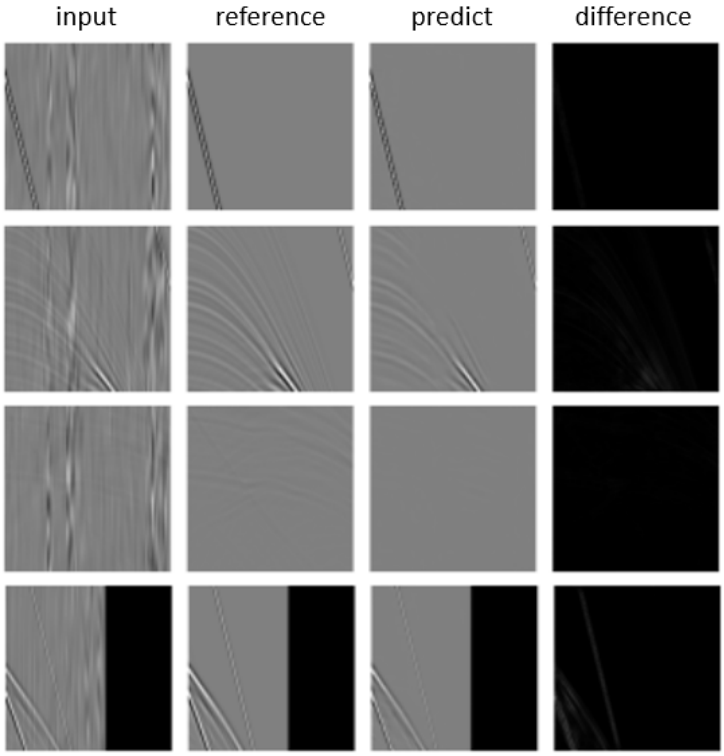

Figure 4.6: Examples of patches from "Den_Valid_01" set denoised by the FPM RBF-TDM network.

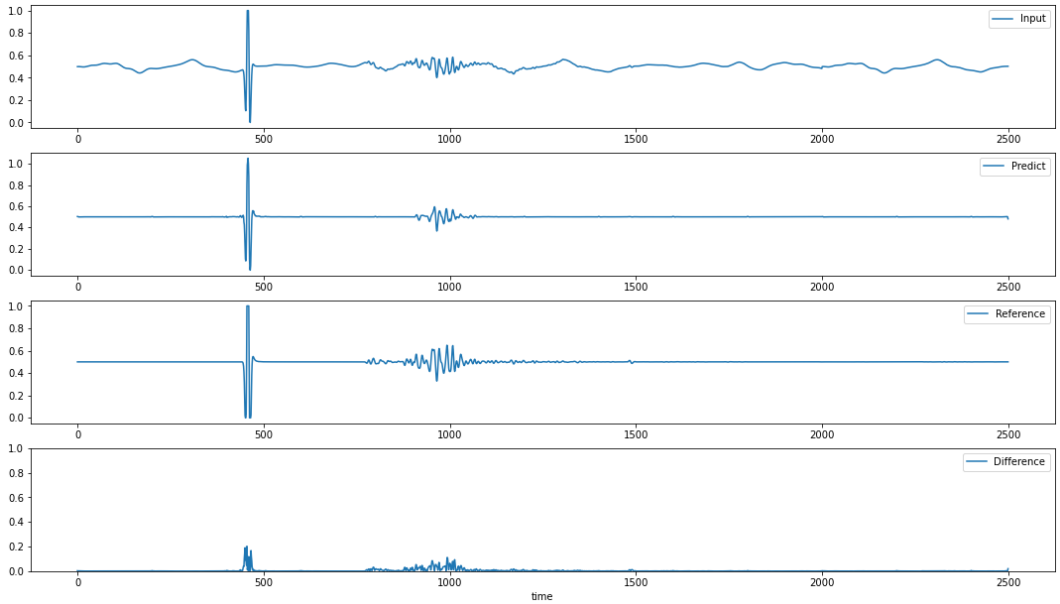

Figure 4.7: Example of a seismic trace (with normalized signal value) from a shot gather of the "Den_Valid_01" set denoised by the FPM RBF-TDM network.

Table 4.3: Results comparison on set “Den_Valid_02”

#	**Network**	**PSNR**	**NRMSE**
01	Res-U-Net (Our)	37.8728	0.0545
02	U-Net	37.3029	0.0746
03	Res-U-Net FullyConv (Our)	37.0662	0.0594
04	U-Net FullyConv	36.8526	0.0793
05	FPN	34.5806	0.1951
06	FPN RFB-RON (Our)	34.3868	0.1922
07	FPN RFB-TDM (Our)	34.3559	0.1917
08	U-Net Mandelli	34.5543	0.1737
09	CANDI	34.3042	0.1583
10	FPN RFB-RefineDet (Our)	34.1579	0.1964
11	DnCNN	34.0286	0.0943
12	SE-ResNet	33.8249	0.1622
13	EDSR-ResNet	33.1689	0.2474
14	SR-ResNet	32.3873	0.1839
15	Deeper SR-CNN	30.5630	0.1318
16	ResNet	29.9083	0.1425
17	AR-CNN	29.9351	0.2959
18	Fast AR-CNN	29.1852	0.5027
19	U-Net Sun	28.5039	0.7822

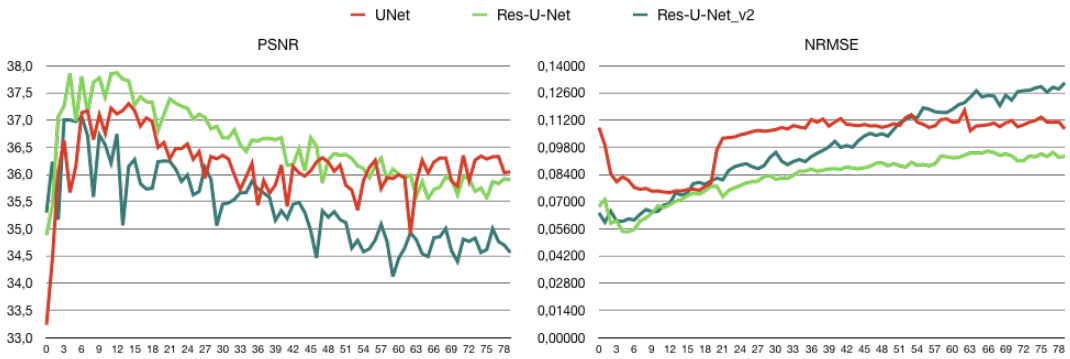

Figure 4.8: Convergence curve of Res-U-Net, U-Net, and Res-U-Net v2 on the “Den_Valid_02” set.

Table 4.4: Results comparison on set "Den_Valid_03"

#	**Network**	**PSNR**	**NRMSE**
01	Res-U-Net (Our)	37.6354	0.0510
02	U-Net	37.1853	0.0624
03	U-Net FullyConv	36.5611	0.0672
04	Res-U-Net FullyConv (Our)	36.4501	0.0563
05	FPN RFB-RefineDet (Our)	34.6355	0.1500
06	FPN RFB-RON (Our)	34.5483	0.1475
07	FPN	34.5079	0.1477
08	FPN RFB-TDM (Our)	34.2827	0.1525
09	SE-ResNet	34.2976	0.1233
10	U-Net Mandelli	33.9452	0.1366
11	DnCNN	33.9225	0.0852
12	CANDI	33.8985	0.1249
13	SR-ResNet	32.1686	0.1486
14	EDSR-ResNet	31.9018	0.2024
15	ResNet	30.1608	0.1238
16	Deeper SR-CNN	29.1559	0.1293
17	AR-CNN	28.7171	0.2402
18	Fast AR-CNN	27.0445	0.4123
19	U-Net Sun	25.4683	0.6513

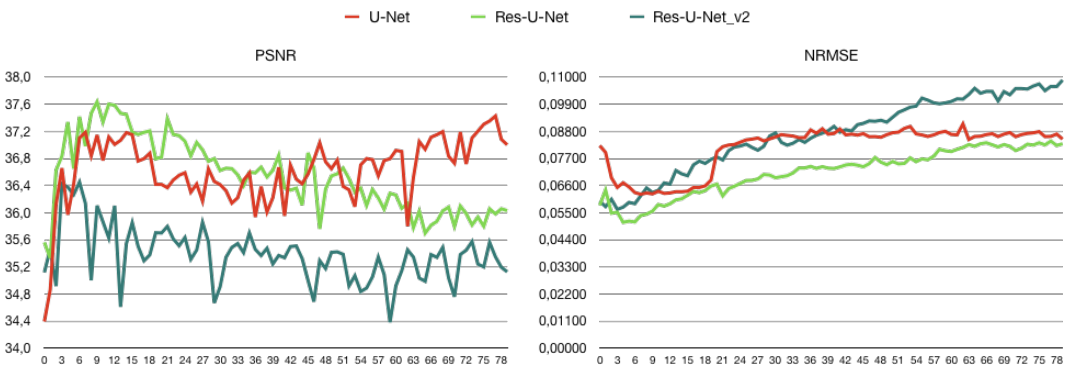

Figure 4.9: Convergence curve of Res-U-Net, U-Net, and Res-U-Net v2 on the "Den_Valid_03" set.

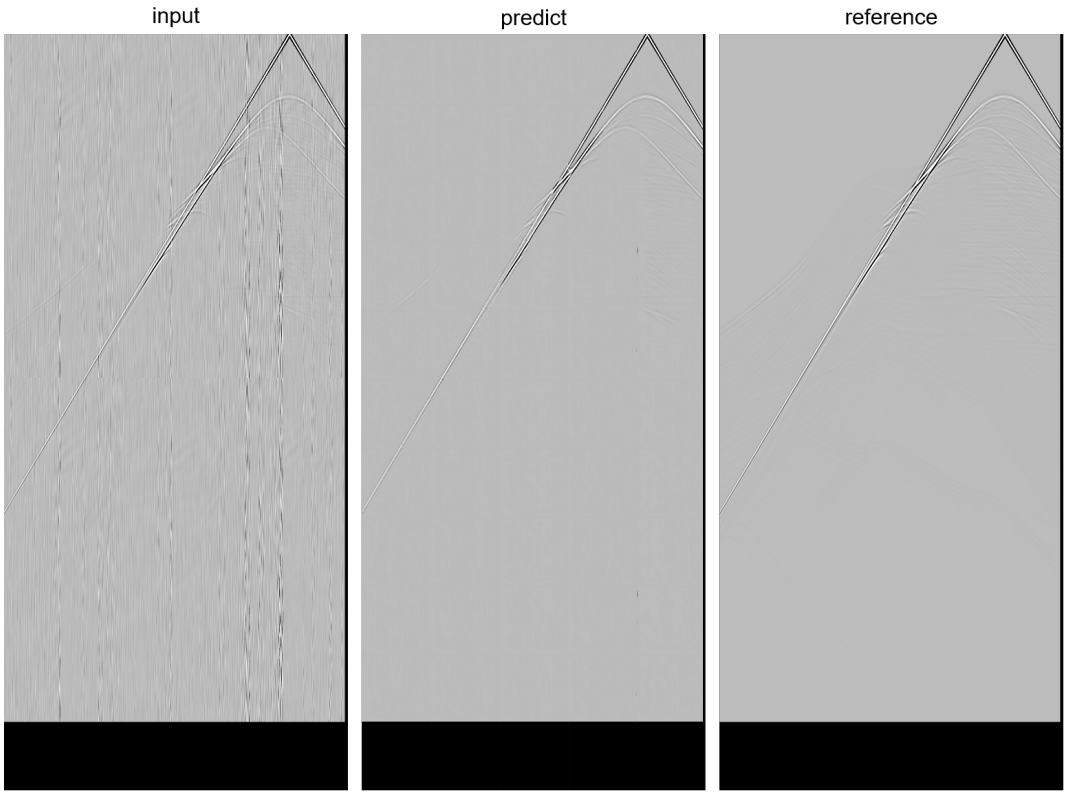

Figure 4.10: An example of a shot gather from "Den_Valid_02" set denoised by the Res-U-Net network.

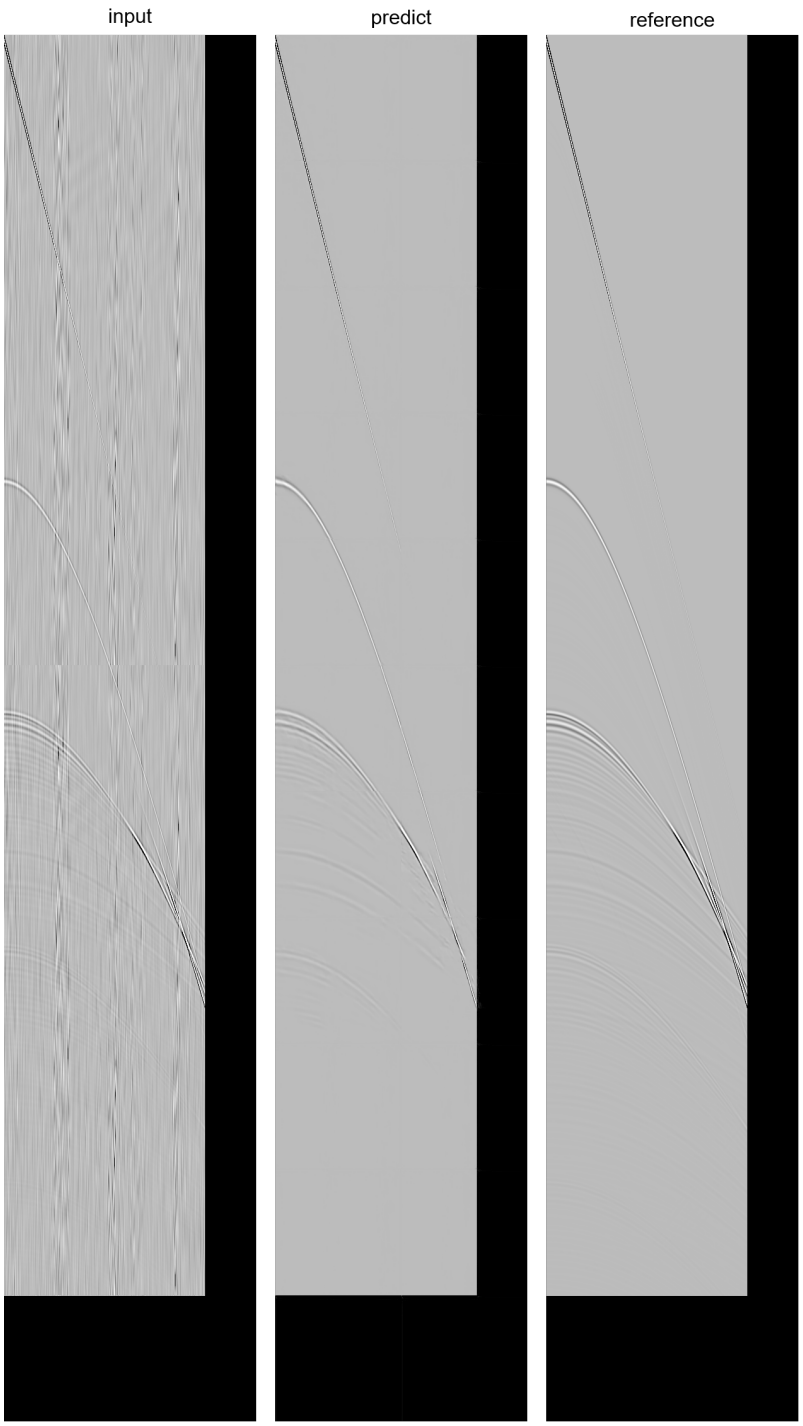

Figure 4.11: An example of a shot gather from "Den_Valid_03" set denoised by the Res-U-Net network.

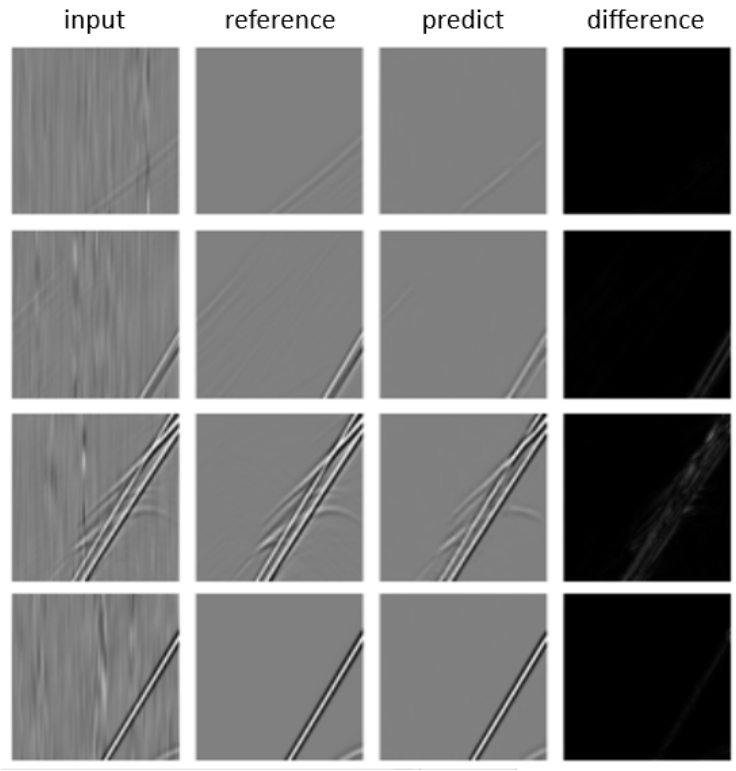

Figure 4.12: Examples of patches from "Den_Valid_02" set denoised by the Res-U-Net model.

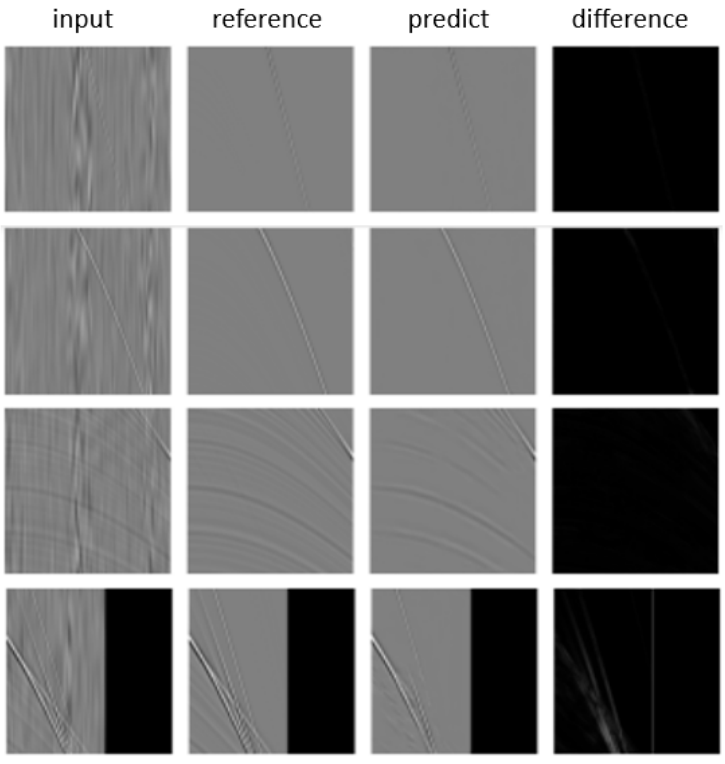

Figure 4.13: Examples of patches from "Den_Valid_03" set denoised by the Res-U-Net model.

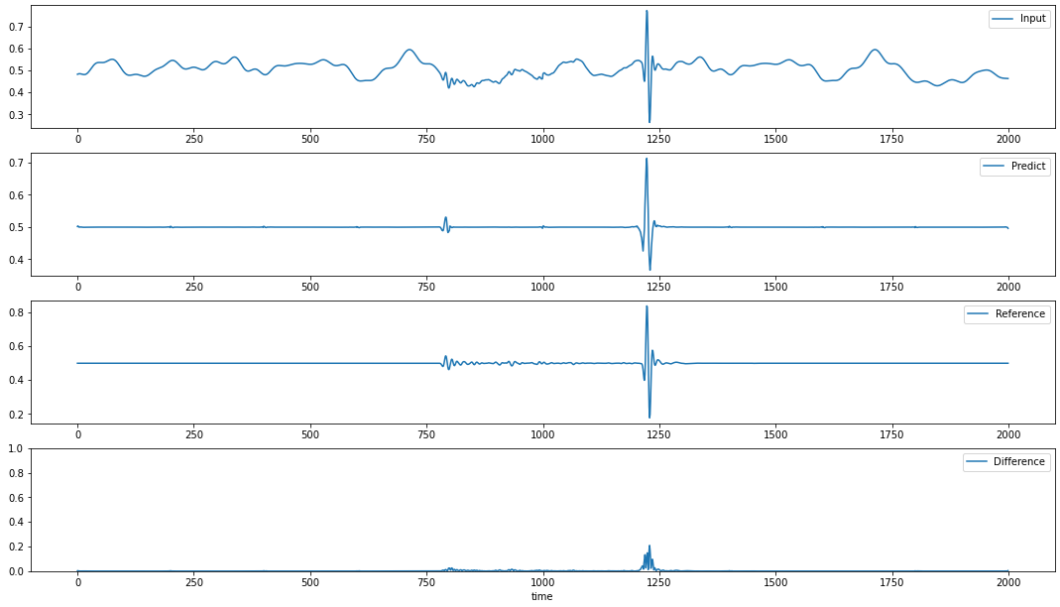

Figure 4.14: Example of a seismic trace (with normalized signal value) from a shot gather of the "Den_Valid_02" set denoised by the Res-U-Net network.

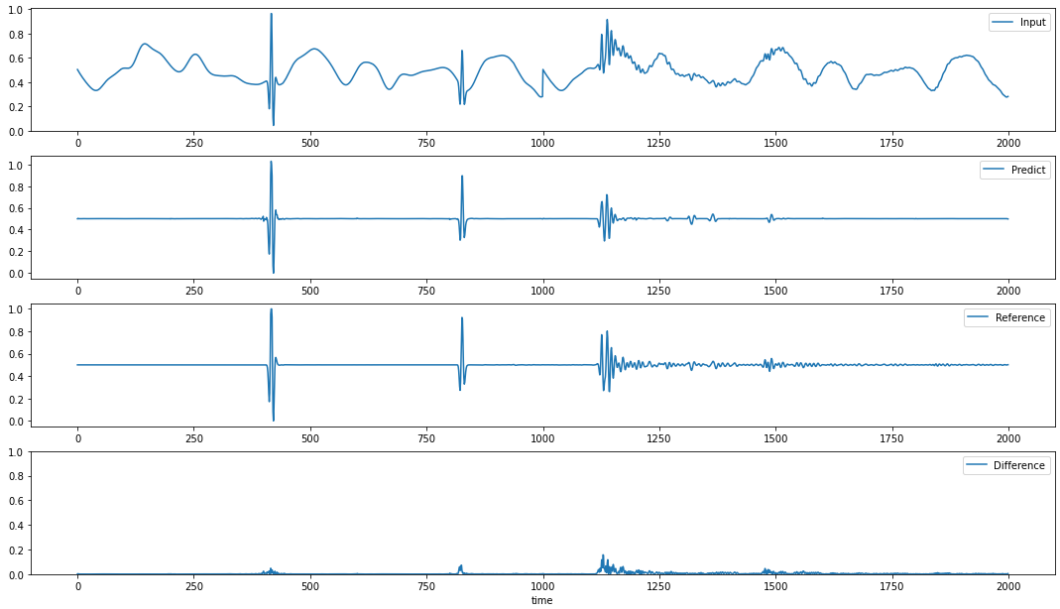

Figure 4.15: Example of a seismic trace (with normalized signal value) from a shot gather of the "Den_Valid_03" set denoised by the Res-U-Net network.

4.4 Concluding Remarks

This chapter presented experiments with the proposed method, whose objective is to perform blind-denoising in scenarios with unpaired seismic data. However, in order to be able to evaluate the proposed method, we prepared sets with paired data to test the method, even though its training ran with unpaired data. Moreover, we also evaluated the model with three different *shot gathers* types, two of which are different from the type of *shot gather* used in training. The model produced by the FPN RFB-TDM network obtained the best result in the scenario where the SND is evaluated with data from the same source used in the SNT. However, the model generated by the U-Net network generalizes better when the data used by the SNT is different from those used by the SND. Besides, both models are still susceptible to the attenuation of low seismic amplitudes, which will be investigated in future research.

Our changes to the original Style Transfer algorithm made it possible to generate synthetic samples similar to the real ones. Mainly because of the SNT hyper-tuning method, which allowed finding appropriate hyperparameters. The application of the original Style Transfer algorithm made by Takemoto et al. (2019) is equivalent to example F of Figure 4.3, which produced texture with seismic noise characteristics drastically different from the real ones. While the example G, which was produced by our method, presents signal and noise with characteristics closer to the real ones.

The proposed method achieved the objective, but there are some limitations:

1. The performance of blind-denoising is limited by the quality of the *shot gathers* used as a "good" reference (noise-free in the ideal case). Therefore, if there is noise in these *shot gathers*, then the model cannot remove such noise. In other words, it is impossible to apply the method to sets without good quality samples;

2. It is impossible to apply the method on sets whose dimensions of all *shot gathers* are smaller than those dimensions defined for patches. In this case, all patches produced for these *shot gathers* are considered restricted, as shown in example B in Figure 3.4, and therefore could not be used by SNT to generate synthetic samples, as shown in Figure 3.9.

5
Conclusion

Section 1.2 of the introduction defines the main questions we aim to answer with this research project. Section 5.1 presents how the contributions generated by our proposed method answer these research questions. Finally, Section 5.2 presents future research.

5.1
Summary of the Contributions

The main objective of this thesis is to investigate how to perform denoising of seismic shot gathers in scenarios with a lack of paired training data. For this, we proposed a self-supervised method for blind denoising of seismic data, which requires no prior seismic signal analysis, no estimate of the noise, and no paired training data.

The proposed method (Chapter 3) provides a solution that answers RQ1. Our proposed self-supervised method is based on two modules: (1) Seismic Noise Transfer (SNT), which learns to produce synthetic-noisy shot gathers containing the noise from noisy shot gathers, and the signal from noise-free shot gathers; And (2) Seismic Neural Denoiser (SND), which learns to map the synthetic-noisy shot gather back to the original noise-free version.

The SNT module provides a solution that answers RQ2 and is one of the contributions of this thesis. We have adapted the Neural Style Transfer algorithm to synthesize seismic *shot gathers* from authentic *shot gathers*. It is noteworthy that our adaptation of the style transfer goes beyond the application of its vanilla version. We have added weights to the noise transfer layers, allowing more customization power over the generated *shot gathers*.

In addition, we also developed an SNT hypertuning method to find appropriate noise transfer weight settings. Our proposal takes advantage of the same backbone used by NST to extract seismic embeddings from real and synthetic *shot gathers* and then calculate the similarity to assess whether the generated content has similar characteristics to the real ones.

The SND module provides a solution that answers RQ3. We propose two new CNN-based denoiser architectures to compose the SND module. Our proposed networks obtained the best performance in two scenarios. In the first

scenario, where the validation set has *shot gathers* of the same type as those used in training, the FPN RFB-TDM network produced the best model. Next, in the second scenario, where the validation set *shot gathers* are of different types, the best model was produced by Res-U-Net. Both models performed better than the other models presented in related work chapter.

5.2 Future Research

Although all research questions have been answered and many significant advances have been achieved, there are still open limitations that must be met for the proposed method to achieve more outstanding quality. As discussed in Section 4.4, the method is still susceptible to attenuation of low seismic frequencies.

When discussing this limitation with experts, two possibilities for future research arose:

1. Investigate how it is possible to use the Hilbert transform combined with the model proposed in this thesis. The hypothesis is that the signal envelope extracted by the Hilbert transform in the input domain can help restore the attenuated low-frequency signal at the model output. Recent works from other contexts already combine Hilbert's transform with deep learning models, such as Hilbert Transform Long Short-Term Memory (Heo et al., 2021), and MSHilbNet (Tsinganos et al., 2021). However, the challenge of this future work is to find a way to incorporate the Hilbert transform into the current proposed model without losing its self-supervised nature.

2. Another possibility is to investigate whether the spatial attention mechanism can solve the low-frequency signal attenuation problem. The hypothesis is that by using the attention mechanism, the model will learn to excite regions with low-frequency signals, thus avoiding their attenuation. The deep learning attention technique has had promising results when applied in geophysics, either supervised (Yang et al., 2021; Saad et al., 2021) or semi-supervised methods (Li et al., 2021a).

Bibliography

Betine Bucker, E., José Grandson Busson, A., Milidiú, R. L., Colcher, S., Pereira Dias, B., and Bulcão, A. (2019). A deep convolutional network for seismic shot-gather image quality classification. *arXiv*, pages arXiv–1912.

Billette, F. and Brandsberg-Dahl, S. (2005). The 2004 bp velocity benchmark. In *67th EAGE Conference & Exhibition*, pages cp–1. European Association of Geoscientists & Engineers.

Busson, A. J., Mendes, P. R., Moraes, D. d. S., da Veiga, A. M., Guedes, A. L., and Colcher, S. (2020a). Video quality enhancement using deep learning-based prediction models for quantized dct coefficients in mpeg i-frames. In *2020 IEEE International Symposium on Multimedia (ISM)*, pages 29–32. IEEE.

Busson, A. J. G., Figueiredo, L. C., dos Santos, G. P., Damasceno, A. L. d. B., Colcher, S., and Milidiú, R. L. (2018). Desenvolvendo modelos de deep learning para aplicações multimídia no tensorflow. *Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web*.

Busson, A. J. G., Mendes, P. R. C., de S. Moraes, D., da Veiga, Á. M. G., Colcher, S., and Guedes, Á. L. V. (2020b). Decoder-side quality enhancement of jpeg images using deep learning-based prediction models for quantized dct coefficients. In *Proceedings of the Brazilian Symposium on Multimedia and the Web*, pages 129–136.

Chang, D., Yang, W., Yong, X., and Li, H. (2018). Generative adversarial networks for seismic data interpolation. In *SEG 2018 Workshop: SEG Maximizing Asset Value Through Artificial Intelligence and Machine Learning, Beijing, China, 17-19 September 2018*, pages 40–43. Society of Exploration Geophysicists and the Chinese Geophysical Society.

Chang, D., Yang, W., Yong, X., Zhang, G., Wang, W., Li, H., and Wang, Y. (2020). Seismic data interpolation using dual-domain conditional generative adversarial networks. *IEEE Geoscience and Remote Sensing Letters*.

Chevitarese, D. S., Szwarcman, D., Brazil, E. V., and Zadrozny, B. (2018). Efficient classification of seismic textures. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Dong, C., Deng, Y., Change Loy, C., and Tang, X. (2015a). Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584.

Dong, C., Loy, C. C., He, K., and Tang, X. (2015b). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307.

Dong, X., Li, Y., and Yang, B. (2019). Desert low-frequency noise suppression by using adaptive dncnns based on the determination of high-order statistic. *Geophysical Journal International*, 219(2):1281–1299.

dos Santos, G. N., de Freitas, P. V., Busson, A. J. G., Guedes, Á. L., Colcher, S., and Milidiú, R. L. (2019). Métodos baseados em deep learning para análise de vídeo. *Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web*.

Duarte, L. T., Donno, D., Lopes, R. R., and Romano, J. M. T. (2014). Seismic signal processing: Some recent advances. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2362–2366. IEEE.

Fang, W., Fu, L., Zhang, M., and Li, Z. (2021). Seismic data interpolation based on u-net with texture loss. *Geophysics*, 86(1):V41–V54.

Gan, S., Wang, S., Chen, Y., Zhang, Y., and Jin, Z. (2015). Dealiased seismic data interpolation using seislet transform with low-frequency constraint. *IEEE Geoscience and remote sensing letters*, 12(10):2150–2154.

Gatys, L., Ecker, A., and Bethge, M. (2015). A neural algorithm of artistic style. *Nature Communications*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heo, S., Kim, H., et al. (2021). Toward load identification based on the hilbert transform and sequence to sequence long short-term memory. *IEEE Transactions on Smart Grid*.

Herrmann, F. J. and Hennenfent, G. (2008). Non-parametric seismic data recovery with curvelet frames. *Geophysical Journal International*, 173(1):233–248.

Hou, J., Si, Y., and Li, L. (2019). Image super-resolution reconstruction method based on global and local residual learning. In *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*, pages 341–348. IEEE.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Jun, W., Xin, Z., Houde, Z., and Yinghui, W. (2020). Seismic data reconstruction based on super resolution convolutional neural network. In *SEG 2020 Workshop: Broadband and Wide-azimuth Deepwater Seismic Technology*, pages 38–42. Society of Exploration Geophysicists.

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

Kong, T., Sun, F., Yao, A., Liu, H., Lu, M., and Chen, Y. (2017). Ron: Reverse connection with objectness prior networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5936–5944.

LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

Lee, H. and Cho, S. (2020). Locally adaptive channel attention-based network for denoising images. *IEEE Access*, 8:34686–34695.

Li, W. and Wang, J. (2021). Residual learning of cycle-gan for seismic data denoising. *IEEE Access*, 9:11585–11597.

Li, Y., Luo, X., Wu, N., and Dong, X. (2021a). The application of semisupervised attentional generative adversarial networks in desert seismic data denoising. *IEEE Geoscience and Remote Sensing Letters*.

Li, Y. and Ma, Z. (2021). Deep learning-based noise reduction for seismic data. *Journal of Physics: Conference Series*, 1861(1):012011.

Li, Z., Sun, N., Gao, H., Qin, N., and Li, Z. (2021b). Adaptive subtraction based on u-net for removing seismic multiples. *IEEE Transactions on Geoscience and Remote Sensing*.

Lim, B., Son, S., Kim, H., Nah, S., and Mu Lee, K. (2017). Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Liu, Y., Dang, B., Li, Y., and Lin, H. (2014). Local spatiotemporal time–frequency peak filtering method for seismic random noise reduction. *Journal of Applied Geophysics*, 111:76–85.

Mandelli, S., Lipari, V., Bestagini, P., and Tubaro, S. (2019). Interpolation and denoising of seismic data using convolutional neural networks. *arXiv preprint arXiv:1901.07927*.

Martin, G. S., Wiley, R., and Marfurt, K. J. (2006). Marmousi2: An elastic upgrade for marmousi. *The leading edge*, 25(2):156–166.

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

Oliveira, D. A., Semin, D. G., and Zaytsev, S. (2020). Self-supervised ground-roll noise attenuation using self-labeling and paired data synthesis. *IEEE Transactions on Geoscience and Remote Sensing*.

Ovcharenko, O., Kazei, V., Peter, D., and Alkhalifah, T. (2019). Style transfer for generation of realistically textured subsurface models. In *SEG International Exposition and Annual Meeting*. OnePetro.

Park, B., Yu, S., and Jeong, J. (2019). Densely connected hierarchical network for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Pochet, A., Diniz, P. H., Lopes, H., and Gattass, M. (2019). Seismic fault detection using convolutional neural networks trained on synthetic post-stacked amplitude maps. *IEEE Geoscience and Remote Sensing Letters*, 16(3):352–356.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Richardson, A. and Feller, C. (2019). Seismic data denoising and deblending using deep learning. *arXiv preprint arXiv:1907.01497*.

Ronneberger, O., Fischer, P., and Brox, T. (2015a). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Ronneberger, O., Fischer, P., and Brox, T. (2015b). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

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

Saad, O. M., Oboué, Y. A. S. I., Bai, M., Samy, L., Yang, L., and Chen, Y. (2021). Self-attention deep image prior network for unsupervised 3-d seismic data enhancement. *IEEE Transactions on Geoscience and Remote Sensing*.

Shi, Y., Wu, X., and Fomel, S. (2019). Saltseg: Automatic 3d salt segmentation using a deep convolutional neural network. *Interpretation*, 7(3):1–36.

Shrivastava, A., Sukthankar, R., Malik, J., and Gupta, A. (2016). Beyond skip connections: Top-down modulation for object detection.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, J., Slang, S., Elboth, T., Larsen Greiner, T., McDonald, S., and Gelius, L.-J. (2020). Attenuation of marine seismic interference noise employing a customized u-net. *Geophysical Prospecting*, 68(3):845–871.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Takemoto, N., Araújo, L. d. M., Coimbra, T. A., Tygel, M., Avila, S., and Borin, E. (2019). Enriching synthetic data with real noise using neural style transfer. In *Int. Congress of the Brazilian Geophysical Society*, volume 78.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Tsinganos, P., Cornelis, B., Cornelis, J., Jansen, B., and Skodras, A. (2021). Hilbert semg data scanning for hand gesture recognition based on deep learning. *Neural Computing and Applications*, 33(7):2645–2666.

Wang, B., Zhang, N., Lu, W., Zhang, P., and Geng, J. (2018a). Seismic data interpolation using deep learning based residual networks. In *80th EAGE Conference and Exhibition 2018*, volume 2018, pages 1–5. European Association of Geoscientists & Engineers.

Wang, H., Wu, Y., Xie, Q., Zhao, Q., Liang, Y., Zhang, S., and Meng, D. (2021). Structural residual learning for single image rain removal. *Knowledge-Based Systems*, 213:106595.

Wang, W., Yang, F., and Ma, J. (2018b). Automatic salt detection with machine learning. In *80th EAGE Conference and Exhibition 2018*.

Wei, Q., Li, X., and Song, M. (2021). De-aliased seismic data interpolation using conditional wasserstein generative adversarial networks. *Computers & Geosciences*, 154:104801.

Wu, N., Li, Y., and Yang, B. (2011). Noise attenuation for 2-d seismic data by radial-trace time-frequency peak filtering. *IEEE geoscience and remote sensing letters*, 8(5):874–878.

Yang, C., Zhou, Y., He, H., He, J., and Chi, Y. (2021). Pyramid residual neural network with attention for seismic data denoising. In *2021 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI)*, pages 387–390. IEEE.

Yu, K., Dong, C., Loy, C. C., and Tang, X. (2016). Deep convolution networks for compression artifacts reduction. *arXiv preprint arXiv:1608.02778*.

Yu, S., Sun, J., Gao, D., and Wu, H. (2021). Noise attenuation of seismic data via deep multiscale fusion network. *Wireless Communications & Mobile Computing (Online)*, 2021.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017). Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155.

Zhang, R. and Ulrych, T. J. (2003). Physical wavelet frame denoising. *Geophysics*, 68(1):225–231.

Zhang, R.-q., Song, P., Liu, B.-h., Zhang, X.-b., Tan, J., Zou, Z.-h., Xie, C., and Wang, S.-w. (2020). Low-frequency swell noise suppression based on u-net. *Applied Geophysics*, 17(3):419–431.

Zhang, S., Wen, L., Bian, X., Lei, Z., and Li, S. Z. (2018). Single-shot refinement neural network for object detection. In *CVPR*.

Zhao, T. (2018). Seismic facies classification using different deep convolutional neural networks. In *SEG Technical Program Expanded Abstracts 2018*, pages 2046–2050. Society of Exploration Geophysicists.

Zhao, Y., Li, Y., Dong, X., and Yang, B. (2018). Low-frequency noise suppression method based on improved dncnn in desert seismic data. *IEEE Geoscience and Remote Sensing Letters*, 16(5):811–815.

Zhao, Y., Li, Y., and Yang, B. (2019). Low-frequency desert noise intelligent suppression in seismic data based on multiscale geometric analysis convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing*, 58(1):650–665.

Zheng, Y., Yuan, Y., and Si, X. (2020). The improved dncnn for linear noise attenuation. In *SEG 2019 Workshop: Mathematical Geophysics: Traditional vs Learning, Beijing, China, 5-7 November 2019*, pages 56–59. Society of Exploration Geophysicists.

Zhu, W., Mousavi, S. M., and Beroza, G. C. (2019). Seismic signal denoising and decomposition using deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 57(11):9476–9488.

A
Fundamentals of Deep Convolutional Neural Networks

This appendix reviews the fundamental concepts of Artificial Neural Networks (Section A.1), Backpropagation Algorithm (Section A.2), and Convolutional Neural Networks (Section A.3). It is noteworthy that this appendix aims to summarize the main concepts related to Deep Learning (DL). For a deeper discussion about the concepts of Deep Learning and its applications, other publications can be consulted (Goodfellow et al., 2016; Busson et al., 2018; dos Santos et al., 2019).

A.1
Artificial Neural Networks

The Perceptron, introduced by Rosenblatt (1957), is considered the most basic structure of a Artificial Neural Networks (ANN). Figure A.1 illustrates the Perceptron's structure, each x entry has an associated weight w. Then the dot product between the input data and their weights is calculated ($z = w_1x_1 + w_2x_2 + ... + w_nx_n = w^t.x$). Next, an activation function is applied on the dot product, resulting in the output of the perceptron: $a_w(x) = ativ(z) = ativ(w^t.x)$. In literature, some source uses an input with a constant value of 1 to represent the bias b, resulting in the equation: $z = w^t.x + b$.

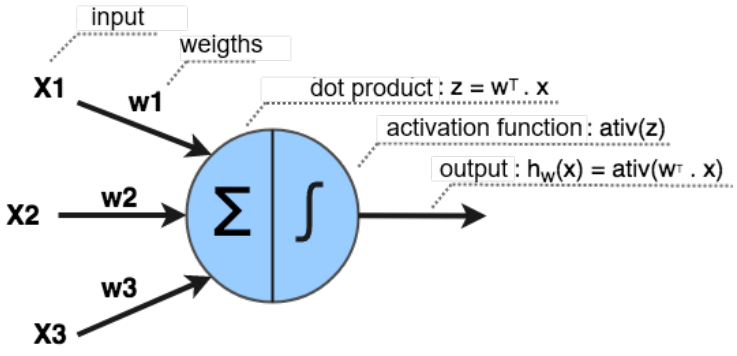

Figure A.1: Perceptron's structure.

Perceptrons can be arranged to perform multiclass classification. As exemplified in Figure A.2, each neuron activates for a specific class. Then the predicted class is selected by using the *argmax* function to obtain the highest activation among all neuron outputs. This model is known as a Multiclass

Perceptron. Additionally, as shown in network B in Figure A.2, neurons can also be structured in multiple layers, where each neuron in the intermediate (or hidden) layers is connected with all neurons in the previous layer. The input data is considered the input layer, while the network's last layer is called the output layer. In this model, the network learns to apply a hierarchy of linear or non-linear transformations (through activations) to generate new representations of the input data. This model is known as the Multilayer Perceptron or MLP. A neural network is considered deep if it has more than two hidden layers.

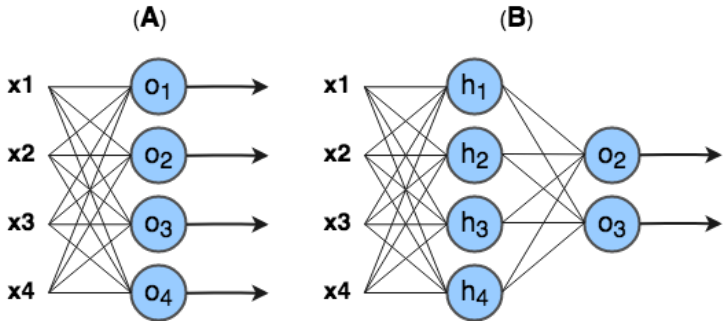

Figure A.2: (A) Multiclass Perceptron; (B) Multilayer Perceptron.

Figure A.3 shows examples of activation functions[1]. The step activation function (*step*) was used in the first Perceptron versions. However, it does not provide a useful derivative that can be used to train multilayer models. Logistic activation (*sigmoid*) and hyperbolic tangent (*tanh*) functions became popular in the 1980s as smoother approximations of the step function and allowed the application of the backpropagation algorithm. Modern activation functions such as linear rectified ($ReLU$) and *maxout* are piecewise linear, computationally cheap and work well in practice.

A.2 Backpropagation Algorithm

The algorithm that allows learning the neural network is called backpropagation (Rumelhart et al., 1986). What is called "learning" in neural networks is the adjustment in the weights (w) and *biases* (b) of the neurons in order to reduce the cost function error. More specifically, this algorithm computes the δ_j^l (error of the j-th neuron of the l-th layer) and then relates it to the partial derivatives of the weights and bias ($\frac{\partial C}{\partial w_{jk}^l}$ e $\frac{\partial C}{\partial b_j^l}$).

[1]https://denizyuret.github.io/Knet.jl/latest/mlp.html

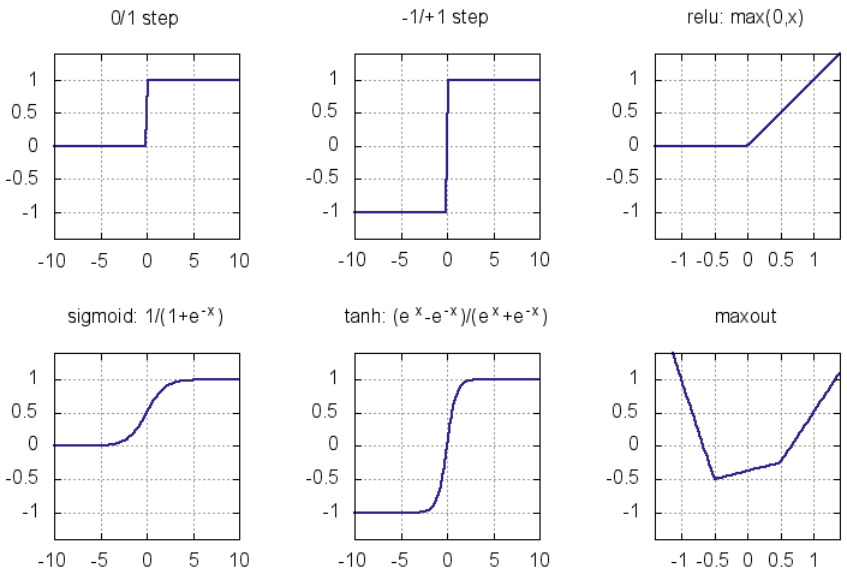

Figure A.3: Activation functions.

The equation of the error δ^L in the output layer is given by:

$$\delta_j^L = \frac{\partial C}{\delta a_j^l}\sigma'(z_j^L) \tag{A-1}$$

The first term $\frac{\partial C}{\delta a_j^l}$ measures how important the activation output of the j-th neuron is for the cost function C. If, for example, the output of a neuron does not contribute to the cost function, then δ_j^L will be small. Similarly, the second term, $\sigma'(z_j^L)$, measures how important the dot product of the j-th neuron is to its activation output.

By rewriting the previous formula to a matrix-based version:

$$\delta^L = \bigtriangledown_a C \odot \sigma'(z^L) \tag{A-2}$$

Where, $\bigtriangledown_a C$ is a vector of the partial derivatives $\frac{\partial C}{\delta a_j^l}$ and the symbol $\odot$ denotes elementary multiplication between vectors.

The equation of the error δ^l with respect to the error in the next layer (δ^{l+1}) is given by:

$$\delta^l = ((w^{l+1})^T\delta^{l+1}) \odot \sigma'(z^l) \tag{A-3}$$

Where, $((w^{l+1})^T$ is the transpose of the matrix of weights and the δ^{l+1} is the error of the (l+1)-th layer. allows the error to be back-propagated through the network. By using equation (A-1) to compute the δ^L, and then using equation (A-2) in sequence to compute $\delta^{L-1}, \delta^{L-2}, \delta^{L-3}, ..., \delta^1$.

The equation for changing the cost in relation to any bias and weight are given respectively by:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{A-4}$$

That is, error δ_j^l is equal to change $\frac{\partial C}{\partial w_{jk}^l}$.

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{A-5}$$

The term $\frac{\partial C}{\partial w_{jk}^l}$ is calculated in relation to the error δ_j^l and activation of the previous layer a_k^{l-1}. Thus, when the activation of the previous layer is small, it is expected that the gradient term $\frac{\partial C}{\partial w_{jk}^l}$ will tend to be small.

Based on the 4 equations described above, the backpropagation algorithm is summarized in the following five steps:

1. **Input**: Insertion of the X (input) in the network and calculation of activation of the input layer;

2. **Propagation**: For each layer l = 2, 3, ..., L, calculate the activation of the neurons receiving the activation of the previous layer neurons as input ($z^l = w^l a^{l-1}$ e $a^l = \sigma(z^l)$);

3. **Output error**: Calculate the error vector $\delta^L = \bigtriangledown_a C \odot \sigma'(z^L)$;

4. **Error backpropagation**: For each layer l = L-1, L-2, ..., 2, calculate the layer error $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$;

5. **Weights and bias updating**: Update the weights and bias with the respective gradients: $\frac{\partial C}{\partial b_j^l} = \delta_j^l$ and $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l$.

A.3 Convolutional Neural Networks

Convolutional Neural Networks, CNNs, or ConvNets (LeCun et al., 1999) are networks specialized in processing data commonly organized in a grid topology. This model gets its name because it uses a mathematical operation called convolution. Convolutional layers allow CNN to find local features, which compose more complex features as it approaches deeper layers. The ability to find this hierarchical structure of *features* is the main reason why CNNs work so well for pattern recognition in spatial data (in the most common case, images).

The remainder of this section presents elementary CNN components related to this thesis's scope. Section A.3.1 covers convolutional layers. Next, Section A.3.2 describes the pooling layers. Section A.3.3 presents the Inception technique. And finally, section A.3.4 introduces the concept of residual learning.

A.3.1 Convolution Layer

Convolution consists of a linear operator that uses two functions to generate a third. It is the sum of the product of these functions along the region implied by their superposition related to the displacement between them. For continuous functions, convolution is defined as the integral of the product of the two functions after one is reversed and shifted:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da$$

For discrete domain functions, the convolution is given by:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

In CNNs, the convolution operation is done in more than one dimension at a time. The neurons weights are represented by a tensor called *kernel* (or *filter*). The convolution process between neurons and *kernels* produces outputs called *features* maps. Specifically, based on the discrete convolution equation, the output of a neuron located in row i, column j of the map of *features* k in a given convolution layer l is given by the equation:

$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f'_n} x_{i',j',k'}.w_{u,v,k',k}$$

where:

- $z_{i,j,k}$ is the output of the neuron located in row i, column j and in the map of *features* k of convolutional layer l;
- $x_{i',j',k'}$ is the output of the neuron located in row i', column j' and in the *features* k' map of the previous layer (1 -1);
- $w_{u,v,k',k}$ is the connection weight between any neuron of the map of *features* k of layer l and its input located in row u, column v and map of *features* k;
- b_k is the bias for the map of *features* k in layer l;
- The parameters s_h and s_w represent the vertical and horizontal *strides* (offsets), f_h f_w is the height and width of the *kernel*, and $f_{n'}$ is the map number of *features* in the previous layer.

Figure A.4 shows an example of a convolution between two 2D tensors. The *kernel* (in blue) has dimensions (2.2), the input tensor (i) has dimensions (3.3) and the *stride* is equal to 1. In the first iteration, the output (o) described

by the calculation: $(1 \times\ 3) + (-1 \times\ 7) + (-1 \times\ 10) + (1 \times\ 8)$ = -6; In the second iteration, $(1 \times\ 7) + (-1 \times\ 4) + (-1 \times\ 8) + (1 \times\ 11) = 6$; In the third iteration, $(1 \times\ 10) + (-1 \times\ 8) + (-1 \times\ 12) + (1 \times\ 1)$ = -9. And finally, in the fourth iteration, $(1 \times\ 8) + (-1 \times\ 11) + (-1 \times\ 1) + (1 \times\ 2)$ = -2 .

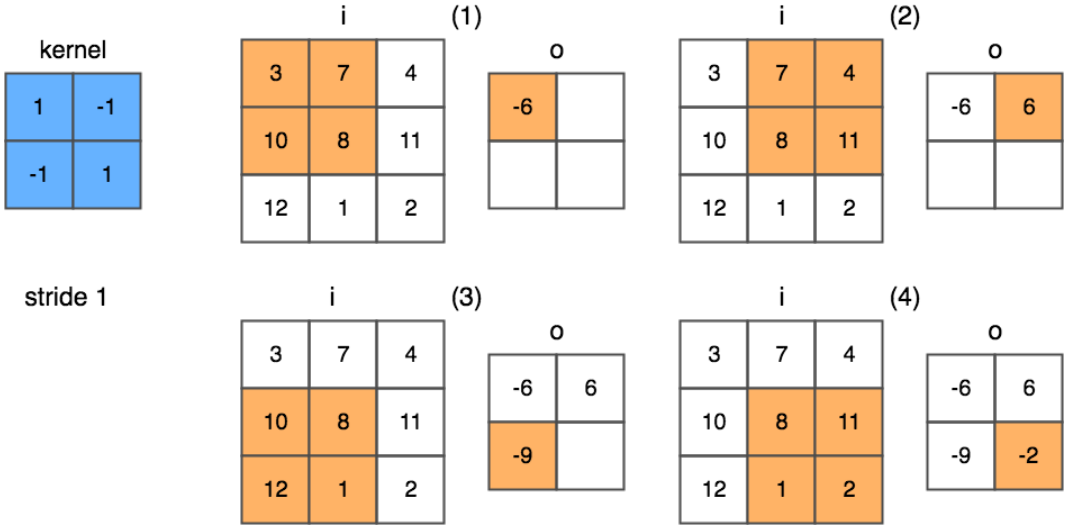

Figure A.4: Example of a convolutional process.

A.3.2 Pooling Layer

In CNNs the *pooling* layers have the function of reducing the dimensionality of the feature maps to reduce the computational load, memory usage, and the number of parameters (thus reducing the risk of *overfitting*). Furthermore, the dimensionality reduction allows the network to tolerate small changes in *features* maps (location invariance).

The *pooling* layers operate similarly to the convolution layers, with the difference that the *pooling kernels* have no weights. The *pooling kernels* aggregate data through aggregation functions such as *max* or *mean*. The *max pooling* function, for example, returns the largest value within an area. Other functions of *pooling* include, for example, the mean or distance L^2 between the elements of a tensor area.

Figure A.5 illustrates an example of the *max pooling* process. Each colored area represents an operation step that uses a *pooling kernel* with dimensions 2x2 and stride 2. In the orange-colored area, the highest value is 28; Then, in the green color area, 21; In the blue area, 27; And finally, in the purple area, 17.

Figure A.6 illustrates the architecture of a CNN that uses convolution layers and *pooling*. The network input consists of a 16x16x3 tensor, thats corresponds to an image with 16 height, 16 width and 3 channels (RGB). The

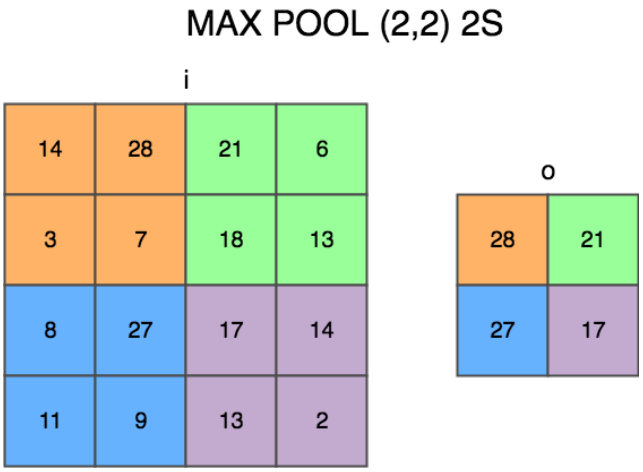

Figure A.5: Example of max-pooling process with *kernel* 2x2 and *stride* 2.

first convolution uses 8 kernels with dimensions (4.4) and *stride* 1 followed by a ReLU activation function, resulting in 8 feature maps with dimensions 16x16. Then, a *pooling* layer is applied that uses a kernel with dimensions (4.4) and *stride* 4, resulting in *features* maps with reduced dimensions (4.4). The next convolution layer uses 4 *kernels* (2,2) followed by a ReLU, resulting in 4 feature maps with 4x4 dimensions. Finally, a last layer of *pooling* uses a kernel(4,4) and *stride* 1, resulting in 4 maps of 1x1 *features*. The last layer is then connected to an output layer *Fully Connected* (FC).

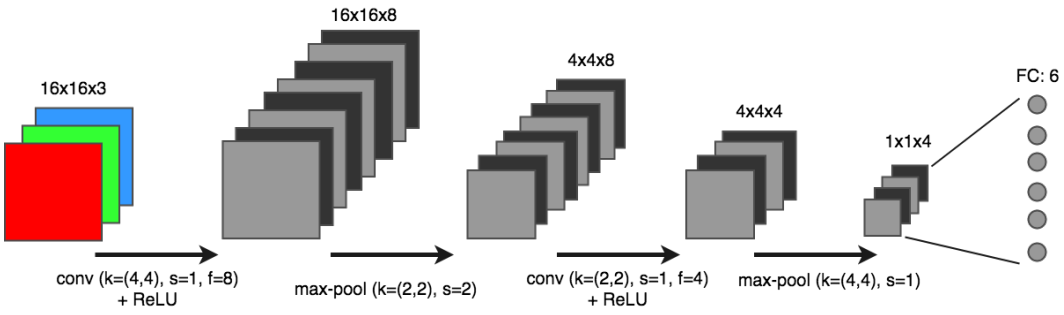

Figure A.6: Example of a CNN architecture.

A.3.3 Inception

The first version of the InceptionNet (or GoogleNet) network introduced by Szegedy et al. (2015) was the winning model of ILSVRC 2014. The Inception technique is essential for the Deep Learning area, as it solves the problem of locating information when there are many variations in feature sizes. Because of this variety, choosing an appropriate *kernel* size becomes difficult. A wide *kernel* is suitable when the information is distributed over large regions, and a short *kernel* when the information is distributed over small regions.

The solution proposed by the Inception (Figura A.7) is to use *kernels* of different sizes in parallel on the same layer, leaving the network a little wider than it is deep. Three different sizes of *kernel*, 1x1, 3x3 and 5x5 are used in the convolution operations. Additionally, is used a *maxpooling* with a 3x3 *kernel*. A 1x1 convolution limits the number of input channels before the 3x3 and 5x5 convolutions to not make processing too cumbersome.

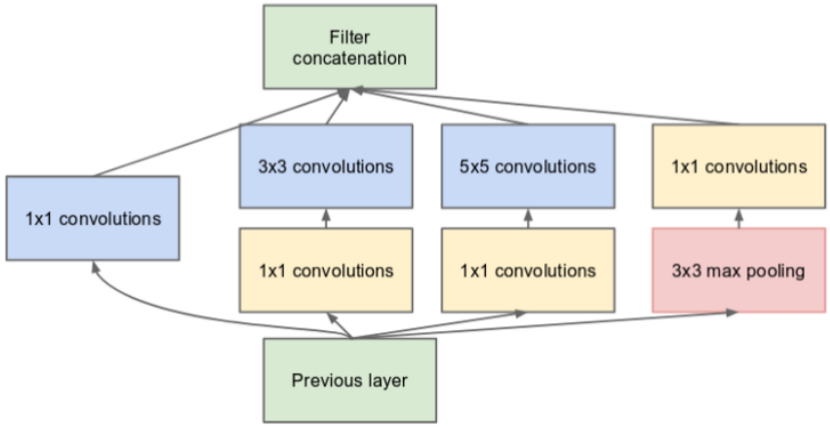

Figure A.7: Inception module.

Later, new versions of InceptionNet introduced improvements to the model. The InceptionNet v2 (Szegedy et al., 2016) architecture attempts to reduce the impact of an issue known as a "representational bottleneck". CNN's work best when convolutions do not drastically change the input size, as this reduction can cause information loss. For this, the authors created three new versions of the Inception module, which refactored the 5x5 convolutions into two smaller convolutions.

Figure A.8, shows the Inception modules used in InceptionNet v2. In module A, the 5x5 convolutions were replaced by a sequence of two 3x3 convolutions, which implies a performance improvement, since a 5x5 convolution is 2.78 times more computationally expensive than a 3x3 convolution. In module B, the authors replaced each 3x3 convolution with a 1xN convolution sequence followed by an Nx1. Finally, in module C the position of the convolution layers was changed so that they were more sparse than deep. This design tries to alleviate the representational bottleneck because if the module were deep, there would be an excessive dimension reduction and, consequently, loss of information.

The third version, called InceptionNet v3 (Szegedy et al., 2016) adapted the RMSProp optimizer, refactored the 7x7 convolutions and applied *Batch Normalization*. Finally, the fourth version, called InceptionNet v4 (Szegedy et al., 2017) reformulated some modules of the architecture. Figure A.9 details

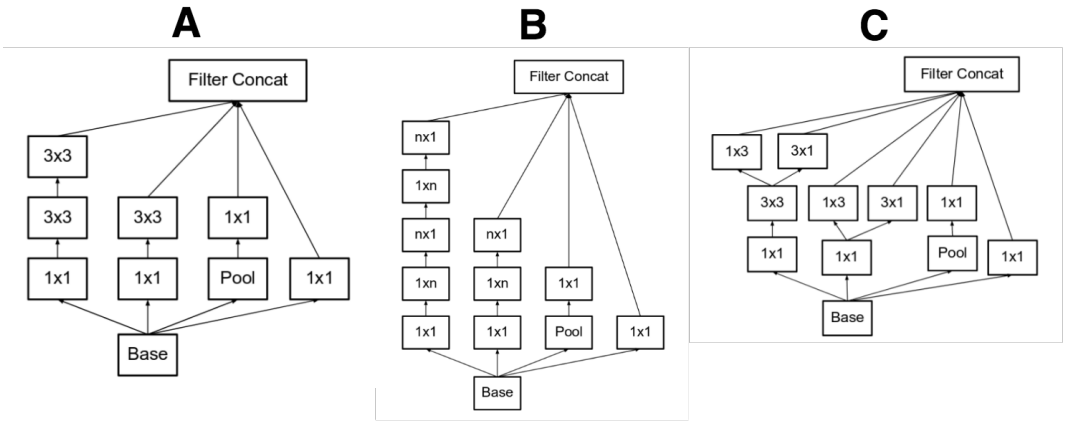

Figure A.8: Three main inception modules of the InceptionNet v2 architecture.

each block in the network. Modules A, B, and C are similar to previous versions. A novelty proposed by InceptionNet v4 is the definition of reduction modules, which are used to reduce the dimensionality of *features* maps.

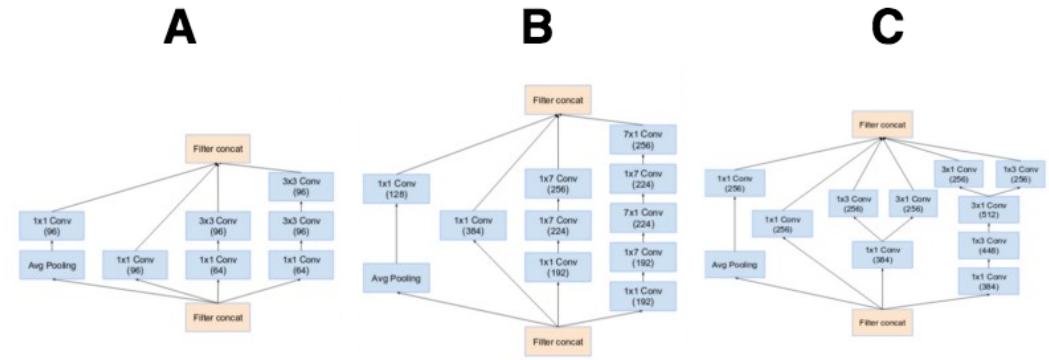

Figure A.9: IR-A, IR-B and IR-C: Three types of Inception module introduced by InceptionNet v4.

A.3.4 Residual Connections

Residual connections were introduced by ResNet He et al. (2016). Layers that use this mechanism are called residual blocks (Figure A.10). Skip-connections ($F(x)$) learn only the residual functions referring to the block input (x). The background intuition of the residual block is that it is easier to optimize the residual mapping than to optimize the original input mapping. For example, in the scenario where identity mapping is ideal, it would be easier to push the residual mapping closer to zero than to fit an identity mapping through a stack of convolutional layers. Additionally, skip-connection allows the network to learn identity mappings more easily and mitigates the vanishing gradients problem.

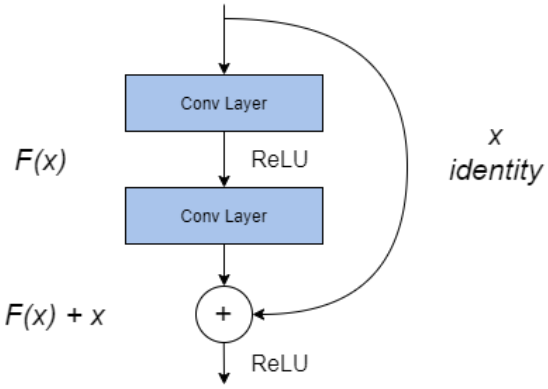

Figure A.10: Residual building block used by ResNet.

Inspired by ResNet, the authors of InceptionNet created two hybrid networks called Inception-Resnet v1 and v2 (Szegedy et al., 2017) that use the residual connection mechanism. Figure A.11 illustrates the three modules that compose both networks. Inception-Resnet (IR) modules combine residual connections and inception mechanisms. The input and output tensors must have the same dimension to make their sum possible. For that, a 1x1 convolution was added after the traditional convolutions of the Inception module to standardize the tensor sizes. Inception's *pooling* operation was removed in favor of the residual connection. The authors found that the network tends to instability when the network exceeds a thousand filters. To stabilize the network, they scaled the residual activations by values between 0.1 and 0.3.

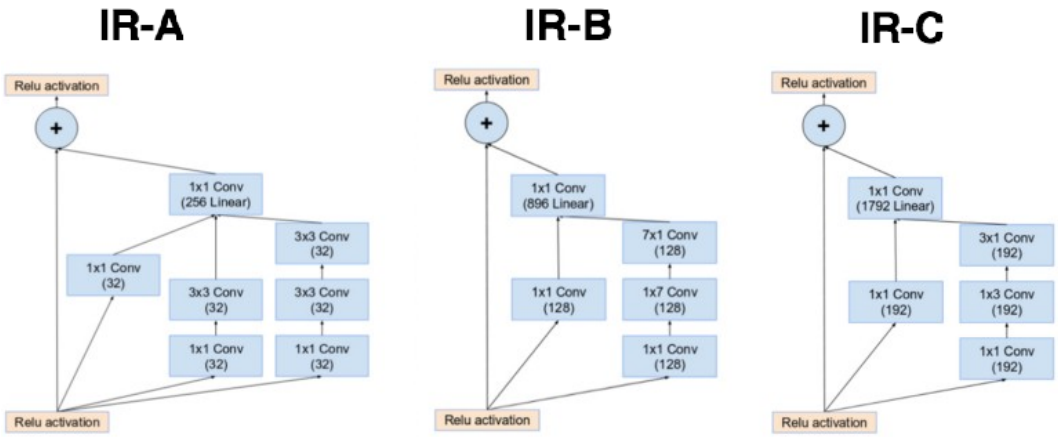

Figure A.11: IR-A, IR-B, and iR-C: Three types of Inception-Resnet modules used by Inception-Resnet v1 and v2 architecture.

The Global Residual Learning (GRL), proposed by Zhang et al. (2017), is a variation of residual connections. However, differently, the connection is made between the input and output of a model, as shown in Figure A.12. With the model, the GRL strategy implicitly removes the desired content from the latent space in its hidden layers. It just learns the difference between the input and desired output: $y = x - R(x)$, where $R(x)$ is learnt.

GRL is a trend in the deep learning field and is applied on models for other several image-to-image tasks, such as super-resolution, denoising, and artifacts removal (Park et al., 2019; Hou et al., 2019; Wang et al., 2021).

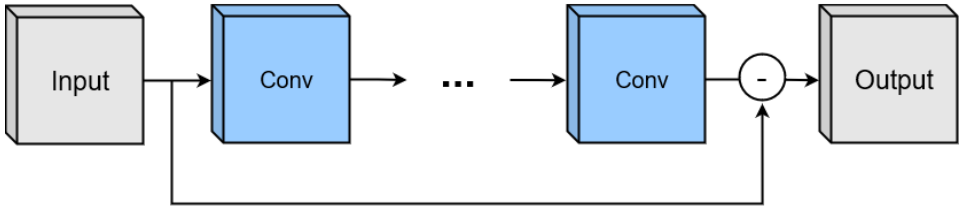

Figure A.12: Global residual learning.

B
Seismic Shot Gather Noise Localization Using a Multi-Scale Feature-Fusion-Based Neural Network

A CNN-based detector is generally composed of two modules. The first is referred to by researchers as the *backbone*, which acts as the feature extractor. The second module, the *detector meta-architecture*, operates on the extracted features from the backbone to generate detection boxes.

Figure B.1 illustrates our detector based on this structure. For the backbone, we use a feature fusion architecture that results from combining MobileNet (Howard et al., 2017) with the FPN to generate convolutional features with rich semantic information on different scale levels. Our meta-architecture is based on the SSD structure (Liu et al., 2016), which performs the regression and classification of box coordinates over the features generated by the backbone. In addition, we use the focal loss rather than the classical cross-entropy-based loss function to improve the prediction accuracy of the SSD.

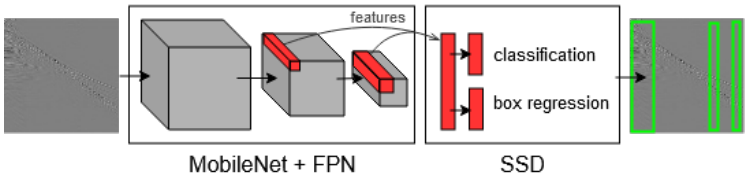

Figure B.1: Overview of proposed network for noise localization in seismic shot-gather images.

B.1
MobileNet+FPN backbone

Our backbone is based on a feature fusion model, combining MobileNet and the FPN into a single network that projects rich feature maps on three different scales. MobileNet is the core of the backbone. It is built on a depth-wise separable convolution (3x3 depth-wise convolution, followed by 1x1 convolution) that requires 8 to 9 times less computation than traditional convolution. The FPN augments the MobileNet with lateral 1x1 convolutions

and fuses feature maps of different scales by nearest-neighbor up-sampling and element-wise sum operation. The final output of the network consists of feature maps (called *projections*) on three different scales; these are used by the SSD for noise detection.

Figure B.2 depicts the MobileNet+FPN architecture. The notation "Conv 32 3x3 S2" denotes a convolutional layer with 32 filters, a 3x3 kernel and stride 2. The MobileNet consists of a Conv layer with stride 2 followed by 13 depth-wise separable (DWS) blocks. Internally, each DWS block has a 3x3 depth-wise convolution followed by a 1x1 convolution (also called point-wise convolution). Then, the FPN processes the lateral features maps from 8, 16 and 32 strides with a 1x1 convolution and combines them by element-wise summation after nearest-neighbor up-sampling. The final three projections used for noise detection are 32, 16 and 8 times smaller than the input image. All convolutional layers use batch normalization and ReLU activation.

B.2 Single Shot Multibox Detector

The SSD is a single stage framework for object detection with an accuracy similar to other state-of-the-art detectors such as YOLO (Redmon and Farhadi, 2018) and Faster R-CNN (Ren et al., 2015). Owing to the variance in noise size, we selected the SSD to take advantage of its multi-scale box matching strategy. The SSD operates by creating thousands of default boxes corresponding to different regions on three feature maps generated by the MobileNet+FPN backnone. The SSD determines which default boxes correspond to ground truth detection and trains the network accordingly. For each ground truth box, the SSD selects the most appropriate box from the default boxes by matching it to the default box with the best intersection over union (IoU) coefficient (higher than a threshold of 0.5). During training, the SSD learns to predict class scores (in our case, classes 0 and 1 for background and noise, respectively) and box offsets from the selected default boxes.

The SSD achieves its objective with the help of a multitask loss function, which is the weighted sum of the confidence loss (conf) and localization loss (loc) as follows:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{B-1}$$

where N is the number of matched default boxes. Let $x_{ij}^{p} = \{1, 0\}$ be an indicator for matching the i-th default box to the j-th ground truth box of category p. The confidence loss is the *softmax* loss over the confidence of multiple classes (c):

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{e\in Neg} log(\hat{c}_i^0)$$
$$where \quad \hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{B-2}$$

The localization loss is a Smooth L1 loss between the predicted box l and the offset box $\hat{g}$. The offset box is calculated from ground truth box g and default box d. The parameters cx, cy, w and h denotes the center, width and height of the box, respectively.

$$L_{loc}(x,l,g) \sum_{i\in Pos}^{N} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^{p} smooth_{L1}(l_i^m - \hat{g}_j^m)$$
$$where \quad smooth_{L1}(z) = \begin{cases} 0.5z^2 & if|z| < 1 \\ |z| - 0.5 & otherwise, \end{cases} \tag{B-3}$$
$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$
$$\hat{g}_j^w = log(g_j^w/d_i^w) \quad \hat{g}_j^h = log(g_j^h/d_i^h)$$

By combining predictions for all default boxes with different scales and aspect ratios from all positions of features maps generated by the backbone network, the SSD obtains a diverse set of predictions, covering various input noise sizes and shapes. Because many boxes attempt to localize objects during the inference step, a post-processing step called greedy non-maximum suppression (NMS) is applied to suppress duplicate detection.

B.3 Focal Loss

The focal loss function was introduced by RetinaNet (Lin et al., 2017b) and solves the foreground-background class imbalance problem in one-stage detectors. As described in Subsection B.2, the SSD evaluates thousands of default boxes; however, most of these boxes do not contain noise (negative examples). The principle of the focal loss function is to reduce the load of these simple negative boxes in order for the loss to focus on boxes with useful content, which can improve the prediction accuracy.

We first introduce the cross-entropy loss (CE) for binary classification:

$$CE(p,y) = \begin{cases} -log(p) & if \quad y = 1 \\ -log(1-p) & otherwise. \end{cases} \tag{B-4}$$

In the above $y \in \{\pm 1\}$ specifies the ground truth class, while $p \in [0,1]$ is the model's estimated probability for the class with label $y = 1$. For notational convenience, p_t is defined as:

$$p_t = \begin{cases} p & if \quad y = 1 \\ 1 - p & otherwise, \end{cases} \tag{B-5}$$

and then $CE(p, y) = CE(p_t) = -log(p_t)$.

The focal Loss, adds a modulating factor $(1 - p_t)^\gamma$ to the cross entropy function. The tunable focusing parameter $\gamma \geq 0$ reduces the relative loss for the simple examples. In this work, we use the α-balanced variant of the focal loss, where weighting factor $\alpha \in [0, 1]$ is used to balance the importance of negative/positive examples. For notational convenience, α_t is defined as:

$$\alpha_t = \begin{cases} \alpha & if \quad y = 1 \\ 1 - \alpha & otherwise, \end{cases} \tag{B-6}$$

The α-balanced focal loss is defined as:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma log(p_t) \tag{B-7}$$

B.4 Experimentation

In this section, we evaluate the effectiveness of the proposed approach for seismic shot-gather noise localization. First, to attest the choice of our basic backbone, we compare the results of MobileNet with the other two popular CNNs, VGG16 (Simonyan and Zisserman, 2014), and InceptionV3 (Szegedy et al., 2016). Next, we supplement the MobileNet with FPN and Focal Loss and evaluate each component to determine the contribution to the final architecture. The dataset and source code of the models developed in this experimnt are available on our git repository.[1]

We decided to use the average precision (AP)[2] metric, as it is a popular metric for measuring detector performance. In problems related to localization, the AP is calculated over an IoU threshold. We used two AP metrics: the traditional AP@0.5 and the AP@[0.5:0.05:0.95]. The latter corresponds to the average of 10 IoU thresholds from 0.5 to 0.95 with a step size of 0.05.

The remainder of this section is structured as follows. In Subsection B.4.1 we present the dataset used in experimentation. In Subsection B.4.2 we describe the training configuration. Next, in Subsection B.4.3 we present our empirical findings. And finally, Subsection B.4.4 presents the concluding remarks.

[1] `http://bit.ly/3oDPZJ6`

[2] http://cocodataset.org/#detection-eval

B.4.1
Seismic shot-gather dataset for noise localization

Our dataset is derived from an offshore towed in a targeted region with 7,993 shot gathers from eight cables each for a total of 63,944 *shot gathers*. Of the total generated *shot gathers*, 6,500 were randomly selected and manually classified by a geophysicist with "Good" and "Bad" labels based on the visual inspection of artifacts related to swell noise and anomalous recorded amplitude. This resulted in two sets with 1,579 and 4,921 *shot gathers* for the "Good" and "Bad" *shot gathers*, respectively. Then, geophysicists used the VoTT[3] tool to manually annotate bounding boxes around the noise regions for each image in the "Bad" set, resulting in 14,101 annotations (bounding boxes). Finally, we split our dataset into 80%, 10%, and 10% for the training, validation, and testing sets, respectively. Our final dataset had 5,200 *shot gathers* for training, 650 *shot gathers* for validation, and 650 *shot gathers* for testing.

B.4.2
Training configuration

The training used RMSprop Tieleman and Hinton (2012) optimization with a momentum of 0.9, a decay of 0.9 and epsilon of 0.1; batch normalization with a decay of 0.9997 and epsilon of 0.001; fixed learning rate of 0.004; Focal loss with alpha of 0.7 and gamma of 2.0; Batch size of 32 images and 200 epochs for training.

B.4.3
Results

As illustrated in Table B.1, the best result was achieved by the MobileNet, which produced an AP@0.5 of 72.11% and AP@[0.5:0.05:0.95] of 32.80% followed by the InceptionV3, which produced an AP@0.5 of 70.94% and AP@[0.5:0.05:0.95] of 32.71%. The worse result was produced by the VGG16. with an AP@0.5 of 66.04% and AP@[0.5:0.05:0.95] of 29.89%.

#	**Backbone**	**AP@0.5 (%)**	**AP@[0.5:0.05:0.95] (%)**
1	VGG16	66.04	29.89
2	IncepionV3	70.94	32.71
3	**MobileNet**	**72.11**	**32.80**

Table B.1: Results of basic backbone models on validation set.

[3]https://github.com/Microsoft/VoTT

Considering MobileNet supplemented by FPN and Focal Loss scenario, architecture #2 produced the best AP@0.5 of 78.90%, while architecture #3 produced the best AP@[0.5: 0.05:0.95] of 45.62%, as illustrated in Table B.2. Because the latter metric is stricter than the former, we consider the architecture #3 to be the winner. Not only the two architectures that used the focal loss were the ones that produced the best performances but also the focal loss performed better in combination with the FPN. In fact, using the FPN without the focal loss was less effective approach.

#	**Backbone**	**AP@0.5 (%)**	**AP@[0.5:0.05:0.95] (%)**
1	MobileNet + FPN	74.71	41.12
2	MobileNet + Focal Loss	**78.90**	40.08
3	**MobileNet + FPN + Focal Loss**	78.37	**45.62**

Table B.2: Results of scenario with MobileNet supplemented by FPN and Focal Loss on validation set.

The results of the best model for the test set are presented in Table B.3. The MobileNet+FPN+FocalLoss network produced an AP@0.5 of 73.13% and AP@[0.5:0.05:0.95] of 38.14%. This shows that the test set is a little more complicated than the validation set, since there is a performance loss of 5.24% in AP@0.5 and 7.48% in AP@[0.5:0.05:0.95]. Figure B.3 presents examples of MobileNet+FPN+FocalLoss prediction on test set.

Backbone	**AP@0.5(%)**	**AP@[0.5:0.05:0.95](%)**
MobileNet + FPN + Focal Loss	**73.13**	**38.14**

Table B.3: Results of the best model on test set

B.4.4 Concluding Remarks

In this experiment, we investigated a multi-scale feature-fusion based neural network for noise localization in seismic *shot gathers*. We built a real-world dataset containing 6,500 seismic *shot gathers* and 14,101 bounding boxes of regions with noise. Our proposed detector model used MobileNet in combination with FPN as the backbone, an SSD as the detector meta-architecture, and focal loss. Our experiments revealed the contribution of each component of the proposed network. In the validation step, the proposed model achieved an AP@0.5 of 78.37% and AP@[0.5:0.05:0.95] of 45.62%. in the test step, it produced an AP@0.5 of 73.13% and AP@[0.5:0.05:0.95] of 38.14%.

To achieve higher performance, we plan to investigate the effectiveness of others mechanisms of state-of-the-art networks for object detection. Specifically, in future work, we plan to extend our actual network with the ARM (Anchors Refinament Module) and ODM (Object Detection Module) of the RefineDet network (Zhang et al., 2018) aiming to further improve noise localization. Further future work involves the construction of a multitask network that both localizes and clears region with noise in seismic *shot gathers*.

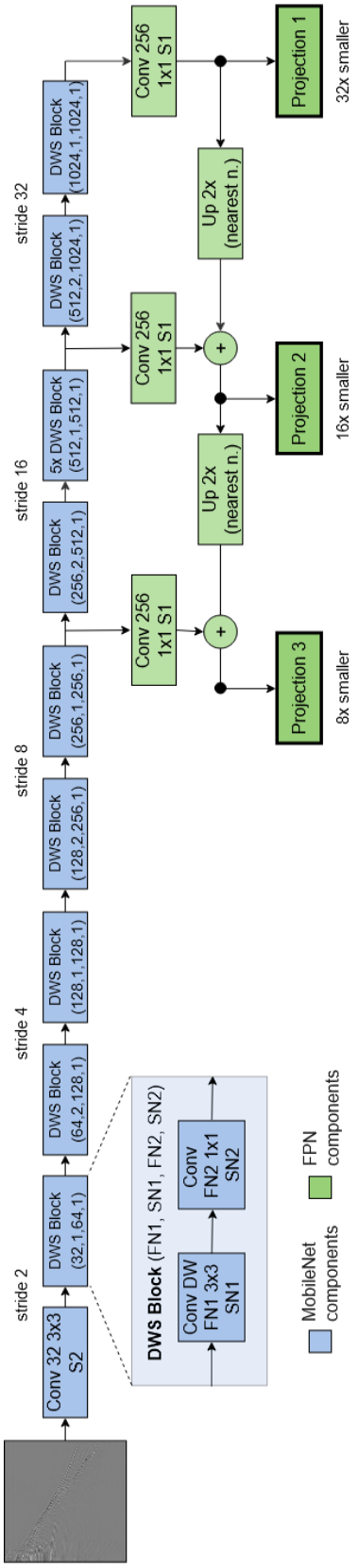

Figure B.2: MobileNet+FPN Network.

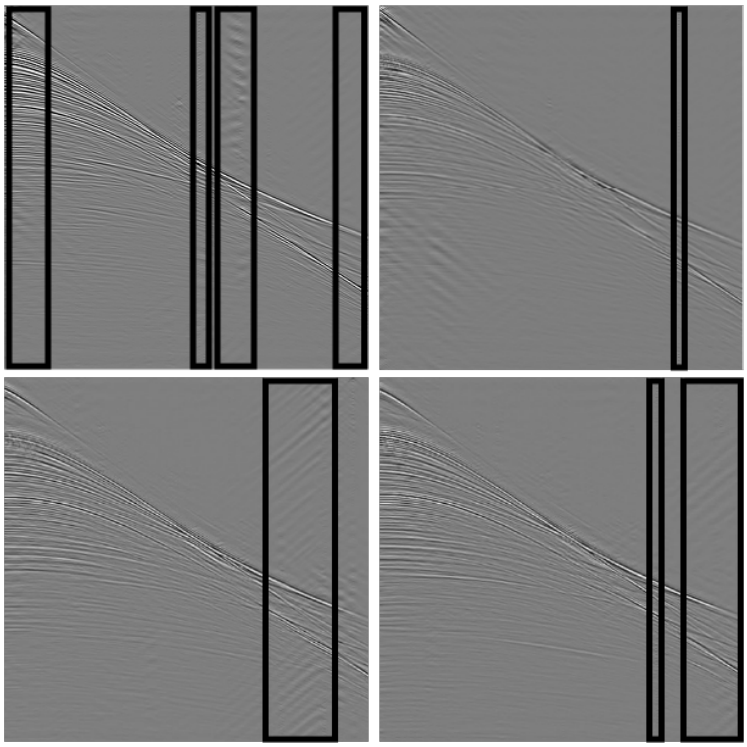

Figure B.3: Prediction examples of the MobileNet+FPN+FocalLoss on test set.