IDEA League

Master of Science in Applied Geophysics

Research Thesis

Real-time Denoising of Distributed Acoustic Sensing (DAS) Data with Unsupervised Deep Learning

In collaboration with TU Delft and TNO

LongSang Ma

August 1, 2024

Real-time Denoising of Distributed Acoustic Sensing (DAS) Data with Unsupervised Deep Learning

In collaboration with TU Delft and TNO

Master of Science Thesis

for the degree of Master of Science in Applied Geophysics

by

LongSang Ma

August 1, 2024

IDEA LEAGUE
JOINT MASTER'S IN APPLIED GEOPHYSICS

Delft University of Technology, The Netherlands
ETH Zürich, Switzerland
RWTH Aachen, Germany

Dated: *August 1, 2024*

Supervisor(s):

Dr. Ir. D.S. (Deyan) Draganov

Prof. Dr. sc. Florian M. Wagner

Committee Members:

Dr. Ir. D.S. (Deyan) Draganov

Prof. Dr. sc. Florian M. Wagner

Dr. Boris Boullenger & Drs. V.P. (Vincent) Vandeweijer MSc.

Abstract

Fiber-optic distributed acoustic sensing (DAS) excels in high-quality seismic signal acquisition and detection but is often hindered by noise, significantly reducing signal-to-noise ratio (SNR) and impeding microseismic event detection. Moreover, continuous seismic monitoring campaigns generates huge data volumes. While numerous denoising approaches exist, they often demand substantial computational resources, hindering real-time implementation. We propose an unsupervised neural network to suppress random noise without requiring noise-free ground truth or prior noise characteristics. The network learns to extract features by masking random input traces and reconstructing the target using long-term coherence from neighboring traces. We explore hyperparameter optimization by varying input sample generation, activation functions, scaling methods, and the number of input traces. We evaluate the model by running a detection algorithm on FORGE data and achieve a 43% increase in event detection. We further exploit our algorithm in real-time experiments and achieved within a 90% processing time compared to the data acquisition rate with denoising implemented . Our approaches can be incorporated real-time data acquisition, effectively facilitate the screening and storing the data timeframe with useful information.

August 1, 2024

Acknowledgements

I would like to express my sincere gratitude to the following individuals for their invaluable contributions to this research:

Dr. Boris Boullenger, my supervisor at TNO, for his patient guidance and insightful contributions throughout the project. His expertise in DAS and machine learning was instrumental to the success of this project;
Drs. V.P. (Vincent) Vandeweijer MSc, my second supervisor at TNO, for his expertise in real-time acquisition and experimental setup, and his continuous encouragement throughout the project.;
Dr. Ir. D.S. (Deyan) Draganov, my academic supervisor at TU Delft, for providing in-depth theoretical guidance and support, and his unwavering enthusiasm for the project's progress, coupled with his valuable feedback on thesis drafts;
The Geophysics Team at TNO, the Geological Survey of the Netherlands for their generous provision of resources and and their insightful guidance in integrating the results with state-of-the-art methodologies;

Special thanks to my friends, Rhea, Ryan, Sargun, Karla, Kilian and Yuan for the peer mental and technical support throughout the challenging phases of this thesis..

Delft University of Technology LongSang Ma
August 1, 2024

Table of Contents

List of Figures

List of Tables

Acronyms

CCS	Carbon Capture Storage
CNN	Convolutional neural network
DAS	Distributed Acoustic Sensing
HECTOR	Coherence-based Earthquake Detector
FORGE	the Utah Frontier Observatory for Research in Geothermal Energy
ReLu	rectified linear unit
RMSE	Root Mean Square Error
SNR	Signal-to-Noise Ratio
STA/LTA	Short-Term Average/Long-Term Average
TanH	Hyperbolic Tangen

Chapter 1

Introduction

Initially designed for signal recording, Distributed Optical Fiber Acoustic Sensing (DAS) has emerged as a powerful tool for seismic signal detection (In Section 2.3.1, we provide a detailed explanation of the characteristics and concepts underlying DAS). Compared with conventional seismic geophones, DAS provide an advantage in having high spatial resolution, covering large distance, strong resistance to harsh environments, and most importantly, provides the possibility for continuous monitoring along the entire length of the fiber optic cable [Yang et al., 2022]. Therefore, DAS has been widely adopted in the geophysical sector, with the ability of doing hydraulic-fracture monitoring, microseismicity detection, downhole surveillance and flow monitoring, fault characterisation and other near surface seismological applications [Li et al., 2021].

Due to the inherent high resolution of fiber optic data, seismic data acquired by DAS typically contains a complex mixture of signals. Often, our target signals would be obscured by various types of noise. Ambient noise such as wind, ocean waves that are heterogeneous in space and non stationary in time will contribute to the seismic signals coherently [Correa et al., 2017] while random noise or coda waves can be appearing in a total trace with unpredictable format [Ma, 2024]. The noise would create a low signal-to-noise ratio (SNR) for the DAS seismic data, leading to challenges in doing imaging, inversion and interpretation, or even a false detection of microseismic events.

To tackle the noise problem, many denoising methods have been developed and used to suppress noise in seismic data and yet there is still limitations for DAS data. For example, bandpass filtering can remove stationary noise at a specific frequency [Hudson et al., 2021], but it cannot tackle coherent noise with overlapping frequency in the signal [van den Ende et al., 2023a]. Deconvolution is aimed at making data sparser but it fails when the reverberations have comparable velocity to the first arrivals and are spatially not separated from the direct arrival [Zhu et al., 2023a]. Median filtering and dip filtering can tackle optical abnormal noise and horizontal noise respectively, but achieving unsatisfactory results in a mixed scenario [Binder and Tura, 2020]. Therefore, methods for suppressing complex noise and improving SNR in geophysical data obtained from DAS are desired.

Machine learning, especially in deep learning where the algorithm has the ability to execute feature engineering on its own, has been showing great potential in building a flexible modular architecture which can learn complex relationships and predict outputs fast with the high inference speed. For example, convolutional neural network (CNN) which learns features by filters optimisation has proved to be applicable in various geophysical fields, including seismic event detection, first-arrival picking, seismic inversion, and earthquake early warning [Wang et al., 2023]. Studies also show that CNN are capable of suppressing noise blindly and revealing the hidden signals. [Yang et al., 2022]

With all the findings, there is always an interesting gap to look for the use of deep learning approaches together for DAS data, particularly in microseimsic monitoring when we have data with high resolution. The approach and applications are generally similar to earthquake monitoring, but microseismic monitoring mainly deals with weak seismic signal with the absence of visible signals in individual receivers [Foulger et al., 2018]. Despite extra steps in data processing, clear microseismic events can allow scientists to monitor the progress and safety of subsurface projects like oil and gas production, carbon capture storage projects and geothermal energy, or the storage of water in dams [Anikiev et al., 2023]. A denoised seismic profile would provide a more precise detection of microseismic events. Moreover, the fast usage of the denoised model drives the possibility of applying as a screening of incoming real-time data. The successful implementation can highly reduce the manual effort in incorrect detections and inaccurate locations of the weak events as mentioned by [Iqbal et al., 2018] during data processing.

This study focuses on implementing the U-Net architecture (proposed by [Ronneberger et al., 2015]) for the blind denoising of DAS data acquired from the FORGE geothermal site at the University of Utah. For the FORGE Project, 1.3 TB of DAS data is collected per day which highlights the problem of data storage and the huge manual effort for data interpretation in months long seismic minoring campaigns [Porras et al., 2024]. The outline of the thesis as follow:

1. A review of the structure of the selected U-Net algorithm and the mathematical concepts behind blind denoising;
2. A methodology focusing on data processing, including data augmentation, scaling methods, and the detection algorithm for microseismic events;
3. A detailed explanation of the training phases, with a structured massing strategy, input samples generation and a list of tuning parameters;
4. A real-time DAS monitoring experiment with the implementation of the denoising algorithm in TNO Utrecht;
5. A quantitative result analysis with regards to the output models such as SNR comparison, training, validation loss, detection accuracy;

In the end, this study evaluates a denoising algorithm's performance on strong and weak microseismic events, and explores its potential for real-time implementation.

Chapter 2

Methodology

2-1 Blind Denoising Machine Learning Model

2-1-1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a feed-forward neural network that learns the feature engineering (process of designing informative attributes from raw data for an algorithm to learn from) by the kernel optimisation. CNNs are widely adopted due to their ability in learning critical features directly from data without any human supervision. Convolution, a core concept in image processing, involves sliding a small kernel over an image. By calculating the weighted sum of overlapping pixels, it captures how image features are influenced by their neighbours, enabling sharpening effects which is particularly crucial in denoising.

Figure 2-1 shows a general examples of CNNs Architecture. The first layer would be convolutional layer, where it applies multiple filters slides over the images to extract the features. In the following activation layer, the moving filters would use an activation map which distinguishes the discriminative region of the image and influence the model to make the decision. To reduce the spatial size of the representation and the required amount of computation and weighs, a pooling layer would be adopted after each convolutional layer to summarise the statistics locally with nearby outputs.

Multiple functions can be performed by CNNs, including clustering, classification, object detection. In Figure 2-1 it demonstrates the classification ability in CNN. In classification phase, algorithm would connect the weights, build a model and adopt backpropagation to compute the gradient of a loss function with respect to weights to look for the optimal output of the model. As a result, with different figures, the algorithm could be able to classify the input figures into different categories, which would be ”fox” in Figure 2-1.

CNNs can be utilised in both supervised and unsupervised learning contexts. In supervised learning, the model is trained with labelled data, providing a ’ground truth’ for the desired output. In contrast, unsupervised learning does not use labelled data; the model identifies patterns and structures without prior knowledge. The idea of unsupervised learning will be discussed in further detail.

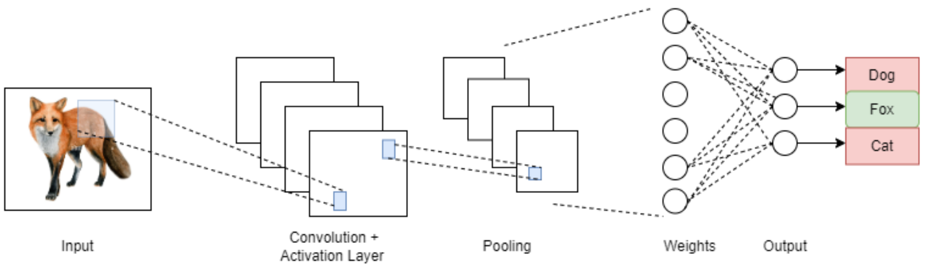


Figure 2-1: CNN Architecture: By given an input of a fox image, the algorithm would analyse the features in the hidden layers by doing convolutional layers and pooling to summarise the local features. In classification phase, the model will study the weights and perform a decision making with the results.

Activation Function

Activation functions play an important role in finding relationships in datasets by determining a neuron should be activated based on the input of the network. The functions will activate the neuron when the input is important for prediction and it is commonly defined by binary steps, linear and non-linear mathematical formulas. In practice, non-linear activations are used to add additional complexity to the problem as real-world data mostly cannot be addressed in linear operations [Szandala, 2021]

Figure 2-2 shows three non-linear activations used in this project, namely rectified lin-

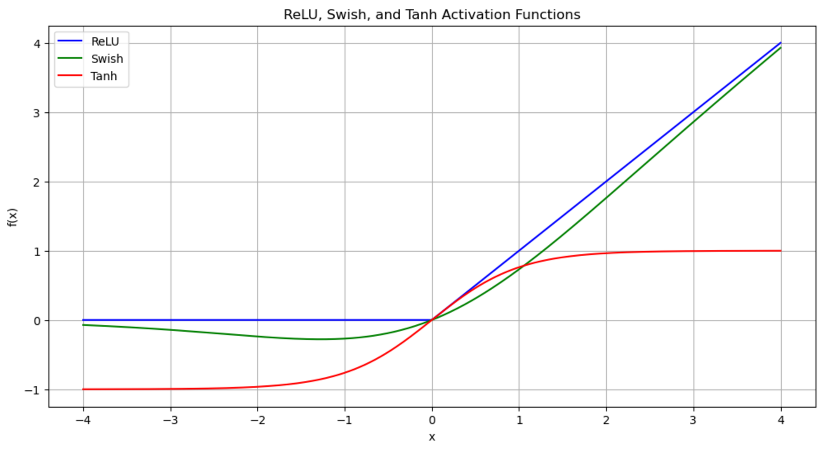


Figure 2-2: Graphical Illustration of ReLU (blue line), Swish (green line), and Tanh (red line). ReLu will turn all the negative value into zero, Swish performs as an exponential function with limited tolerance to negative value. TanH shows a curve similar to a Sigmoid function but shifted to result in a zero centered tendency in loss from both positive and negative value.

ear unit (ReLU), Swish, and Hyperbolic Tangent (TanH). ReLU turns all the negative values to 0 while the positive values remain unchanged so that it avoids gradient issues during backpropagation and speeds up the computation. Swish is a smooth, non-monotonic curve that could learn complex models. It addresses the undefined derivative if the maximum positive value is also 0 and avoids the occurrence of dead neurons in the training when they face negative values. It provides more complexity of the models with negative inputs. TanH tends to turn input function to become very small (close to -1) or very large (close to +1), leading to a certain desire of output. Note that Swish and TanH are more computationally expensive than ReLU. TanH would also suffer from vanishing gradient problems, which the weights in the earlier layer struggle to update and learn representative features from input data.

Pooling Layers

Pooling layers or downsampling layers are essential for reducing the spatial dimensions of the input data, while retaining the most important information for the programme to learn. Figure 2-3 shows two common pooling techniques, namely max pooling and average pooling. Max pooling selects the maximum value within a region as the output, while average pooling calculates the mean of a region. Both techniques could reduce overfitting, but max pooling may neglect some small-valued features and thus result in information loss while average pooling cannot preserve fine details and results in a resolution reduction and image blurring [Gholamalinezhad and Khosravi, 2020].

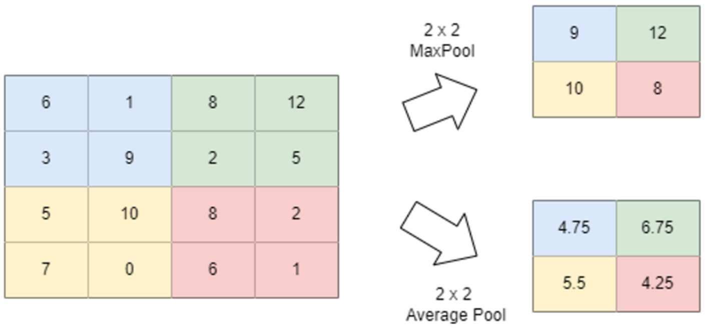


Figure 2-3: Mathematical illustration of a 2x2 max pooling (selecting the maximum value in a block) and average pooling (averaging the value in the region)

Loss Function

The loss function quantifies the difference between the actual target values with the predicted outputs in the deep learning algorithm. In a regression problem, we aim to minimise the loss with the gradient of the loss function to effectively improve the model's performance on the input data. In general, we would use the root mean square error (RMSE) and Mean Absolute

Error (MAE), as defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{2-1}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2-2}$$

where:

- n is the number of data points,
- y_i is the actual value,
- $\hat{y}_i$ is the predicted value.

In this study, we adopted RMSE for the loss calculation. RMSE would be more sensitive to outliers when comparing with MAE [Li et al., 2022]. Therefore, in our data pre-processing, outlier removal would be performed by percentile clipping.

Backpropagation

In a training, forward propagation would compute the loss from the predicted model, while backpropagation will use the loss function and feed the neural network layers to fine-tune the weights. Mathematically,

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{2-3}$$

where:

- Δw_{ij} is the change in weight from neuron j to neuron i,
- η is the learning rate,
- E is the error function or loss function,
- $\frac{\partial E}{\partial w_{ij}}$ is the partial derivative of E with respect to the weight.

As stated in the formula, Weight updates in backpropagation rely on both the learning rate and the loss function. During each iteration, the error between the output layer's predictions and the ground truth labels is calculated. Additionally, gradients of this error with respect to each weight in the network are determined. The learning rate, typically a value between 1 to 10^{-6}, controls the magnitude of these updates (how rapidly or slowly the model learns). By subtracting the product of the learning rate and the computed gradient from each weight, the network iteratively refines its internal parameters to minimise the loss function. Figure 2-4 shows a graphical demonstration on how backpropagation works in reconstruction problems.

In general, backpropagation is a supervised learning method that updates the weights

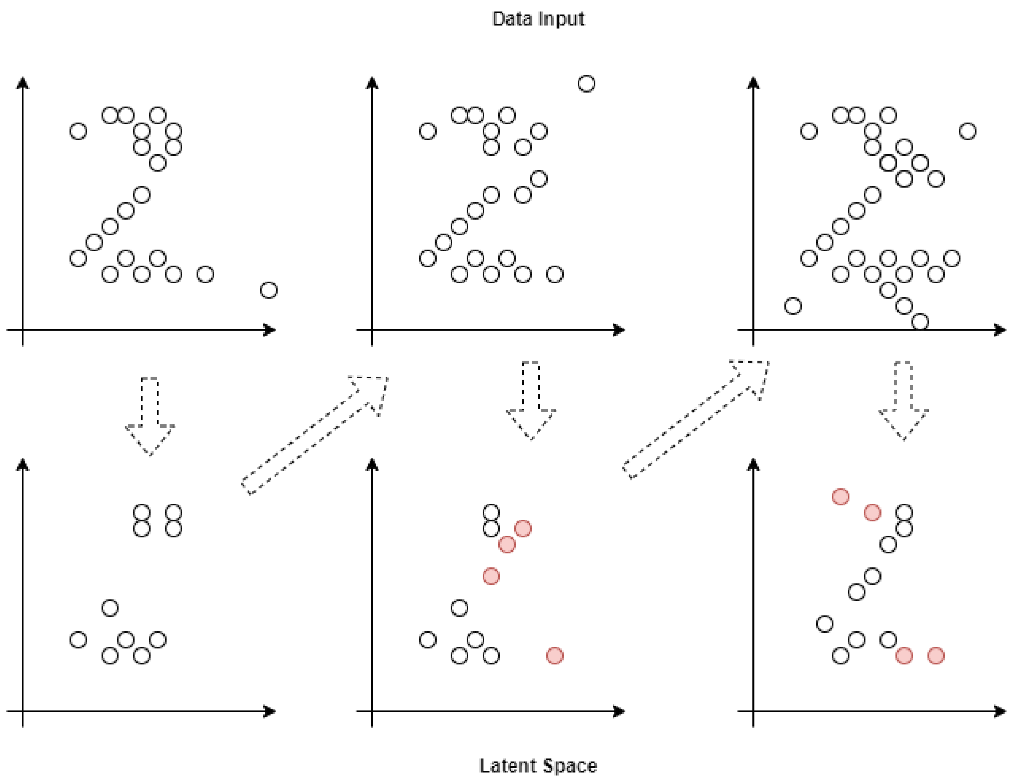


Figure 2-4: Illustration of backpropagation: The above figures show the sequence of an algorithm to learn the features of "2" with propagation. For each data, the computer will create a latent space where it compress the data and extract the key features to cluster the hidden patterns. With a new data input, the algorithm will recalculate the weight and update the latent space so that it represents the data pattern better, which the red points represent the update from data input. In the end, the latent space will have the skeleton shape "2" and remove all the incoherent data in reconstruction problems.

of neural networks to reduce the discrepancy between predicted and observed results. In unsupervised learning, input data would be compressed by an auto encoder neural network which has a smaller latent space, and the reconstruct the full latent space by a decoder network which aims at minimising the reconstruction error between the input and output data.

Optimisation Algorithm

Optimisation algorithms are crucial in keeping the neural networks to learn efficiently and converge to an optimal results. In our cases, Adaptive Moment Estimation (Adam) Optimiser proposed by Kingma and Ba, (2017) were used and it was proven to achieve good results for image denoising [Tian et al., 2019] [Ilesanmi, 2021]. Mathematically, the parameter θ will be updated as follows:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{2-4}$$

where

1. $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$,
2. $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$,
3. α denotes as the step-size and ϵ denotes as the numerical stability.

In the above equations, m_t represent the moving average of the gradient from previous update with am exponential decay rates β_1 between 0 and 1, and v_t represent the moving average of the squares of the recent gradients with another rate β_2 with same properties as β_1. In practice, we get the gradients g_t with respect to the objective at timestep t, and update the estimation $\hat{m}_t$ and $\hat{v}_t$ with bias and our parameter θ in the end.

With Adam, we would have adaptive learning rates such that each parameter would have an individual learning rate, leading to a convergence speed up and a better quality of the final results. Adam can stabilise the training process by applying bias correction during the early iterations of training, and it is memory-efficient with only 2 additional variables for each parameters.

Unsupervised Learning

In unsupervised learning, CNNs based approaches do not depend on human knowledge of image priors. One typical usage of the unsupervised learning algorithm would be organising the large datasets into clusters, which identify the undetected patterns in data and study the features shown in figure 2-5. It is an example of exclusive clustering method called K-means clustering, which a data points closet to a given centroid will be identified as the same category and K represent the distance from each group centroid. To achieve a better

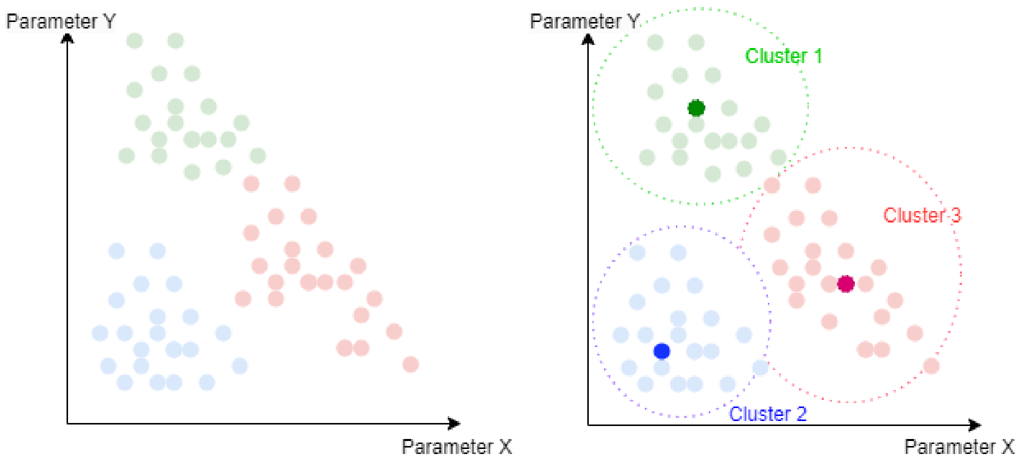


Figure 2-5: Clustering in unsupervised learning: the algorithm goes through the data set and categorising pattern with similar features. In the figure, three clusters are identified with demarcations trends based on the given centroid of the subsets

performance by reducing the complexity of a model, dimensionality reduction would be used to reduce the number of features while retaining as much information as possible.

With similar approach, unsupervised learning can be applied to blind denoising by assuming the underlying clean signal and different noise components form distinct clusters. By strategically selecting the desired cluster(s), the denoised signal can potentially be recovered. In practice, the algorithm designs by projecting the noisy input onto a low-dimensional sub-place which concentrates with plausible signal content and eliminating the orthogonal complement of the subspace which mostly identified as noise [Mohan et al., 2020]. For known noise like Gaussian noise, the algorithm can fully exploit the capability of network architecture to learn from the data and results in a huge improvements of performance when we have limitations in understanding the source and pattern of noise [Chen et al., 2018].

2-1-2 U-Net Architecture

U-Net, a CNN architecture which was first proposed in the use of biomedical image segmentation [Ronneberger et al., 2015], and afterwards developed for a wide range of uses in the geophysical field, including but not limited to multiple removal in seismic data [Durall et al., 2024], resistivity data feature engineering and inversion [Zhang et al., 2022], seismic data reconstruction [Zhu et al., 2023b]. U-Net shows a great potential in identifying noise and signal and provides promising results in signal reconstruction.

U-Net architecture consists of two main parts: the contracting path (encoder) and the expansive path (decoder). The contracting path aims at capturing the main features of the input image, reduce spatial dimensions, and increases the depth of the feature maps, while the decoder path is used to recover the spatial resolution of the input image. One important feature of U-Net are the skipping connections, which directly connect the corresponding layers in the encoder and decoder for preservation of spatial dimensionality. This leads to more accurate segmentation results by leveraging both global context and local details [Xu et al., 2024].

Figure 2-6 shows the structure U-Net Algorithm in our applications. It consists of four pooling blocks that computes 3 x 3 convolutions in both contracting and expansive paths. In the contracting path, several concepts are used including ReLu, instance Normalisation, and max pooling. An example is demonstrated as figure 2-6b. ReLu drops out the negative value, introduce non-linearity to the network. Instance Normalisation would be applied to transform features into a similar scale for the sake of training stability. Afterwards, a max pooling is applied so that the local maxima are kept and assembled to form the input for the next later. In the first block, 64 features will be extracted and max pooling applied to extract the maximum of 2 x 2 neighbouring pixels across the spatial dimension. As a results, the data input size to the second layer is halved and we double the features parameters for learning. The above steps would go on until we reach the layer in which 1024 features are extracted.

Afterwards, we reconstruct the output (i.e, X) by expansion from the bottom convolutional layer. We also use an overlap-tile strategy with which we bypass part of the data from the layer above (i.e, the grey area) with a smaller amount of features extraction and

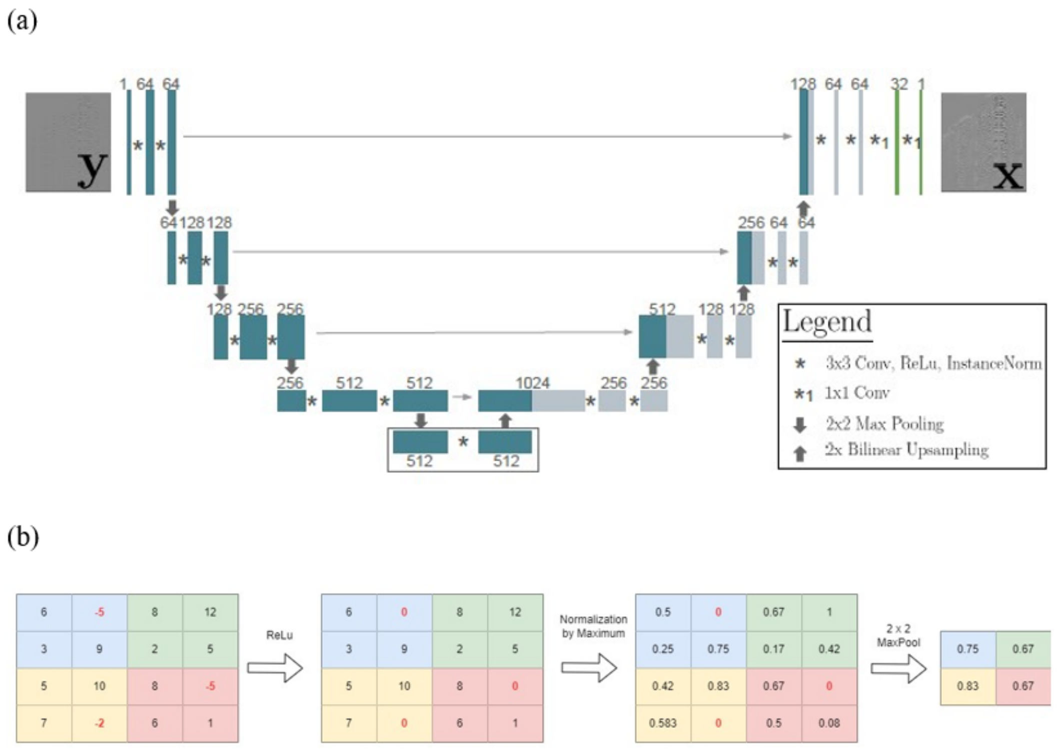


Figure 2-6: (a) An overview of U-Net Structure figure from [Kuijpers, 2020]. The blue represents the data used in downsampling, the grey represents the data used for upsampling and the green represent the fully constructed data. The number on top of each layer states the number of features represented and the more towards the bottom, the heavier the weights of data. (b) A synthetic illustration of the contracting path with ReLU, Normalisation and Maxpooling. A detailed explanation would be found in the main text.

extrapolate the missing input. The above processes continue until the full data is being constructed as X which has the same size as the input Y. In total, the network has 23 convolutional layers.

2-1-3 Mathematical Concepts of Blind Denoising

In self-supervised blind denoising approaches [van den Ende et al., 2023b] [Batson and Royer, 2019], J-invariance functions are calculated to identify the long-range coherence and the results would be independent with the proportion of incoherent data. We start with an arbitrary feature measurements x as shown in figure 2-7, where we have two subsets of elements, J characterised by long-range coherence and thus allowing interpolation even with we are missing the data, and Jc characterised by incoherent data and thus it is not reproducible when data are missing. The loss in the clustering can be calculated and should be minimised using:

$$\mathcal{L}(f) = \mathbb{E}\|f(x) - x\|^2, \tag{2-5}$$

where E[·] denotes the expectation operator and f is the J-invariant functions. It is important to note that the J-invariant function cannot be an identity matrix as it has nothing to predict

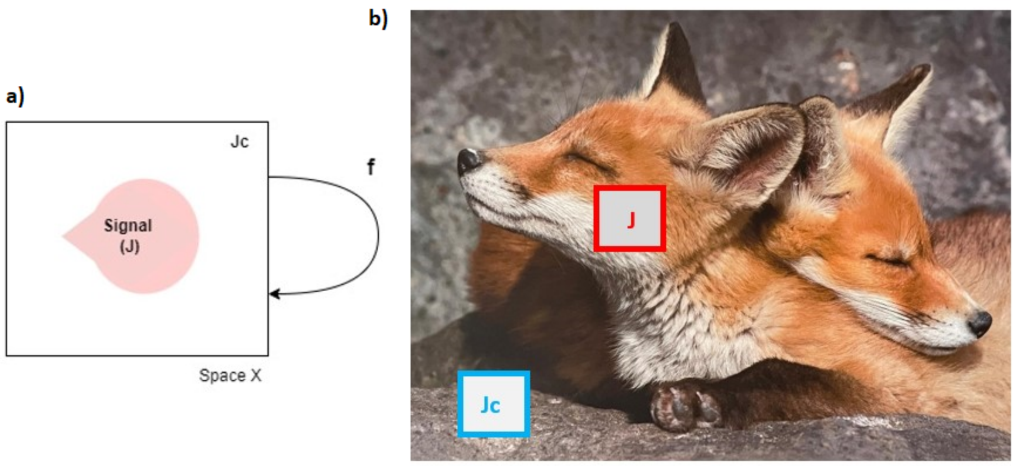


Figure 2-7: Illustration of the blind denoising. (a) The illustration of a conceptual profile. Signal J represents a subject within this profile. Function f, a J-invariant function, maps inputs to outputs which if we calculate the results of it (i.e, f(x)) would be restricted with the results with f(J), and they would not produce the same results and reconstruct things from the results of f(Jc). (b) An example of reconstruction. There would be two independent variable, J which has a high coherent value and Jc which is incoherent signal. With long-range coherent contents as shown in the red blurred block in the right picture, the algorithm should able to interpolate accurately with the missing data in the surroundings of J. However, for the blue blurred block, the details cannot be predicted but only the average value (light grey) could be predicted in the region. Therefore, the output of the function f to reformulate the red block region would not depend on the contents in the Jc, which is the J-invariance.

while blinding the information.

Assume x is an unbiased operator of y (i.e, the average of the values of the estimates determined from all possible random samples equals the parameter we are trying to estimate):

$$\mathbb{E}[x \mid y] = y, \tag{2-6}$$

In other words, for an image (x) as input and our posterior distribution (y) (i,e, the denoised output), we can build the J-invariant function which identify the long-term coherence as:

$$f_J^*(x)_J = \mathbb{E}[y_J \mid x_{J^c}], \tag{2-7}$$

and the self-supervised losses can be expressed as the the sum of the ordinary supervised loss and the variance of the noise:

$$\mathbb{E}\|f(x) - x\|^2 = \mathbb{E}\|f(x) - y\|^2 + \mathbb{E}\|x - y\|^2, \tag{2-8}$$

Loss can be reduced when there is more features in J can be correlated. The closer the $f_J^*(x)_J$ and E[y|x], the better estimation of y. In the end, it can be optimised as:

$$\mathbb{E}\|y - f_J^*(x)\|^2 \leq \mathbb{E}\|y - f_J^{G*}(x)\|^2, \tag{2-9}$$

where we would stop updating when we estimate that our current covariance matrix is performing better than the current covariance matrix with Gaussian random variables x^G.

In machine learning, the set of J-invariant functions g is explored with a neural network f_θ with θ denoting a parameter, which is defined as:

$$g(\cdot) = \sum_{J \in \mathcal{J}} \Pi_J(f_\theta(\Pi_{J^c}(\cdot))), \tag{2-10}$$

where Π_J is the project operator that keeps the elements representing J (coherent signal) and set elements representing Jc (incoherent signal) into zero. In the end, we minimise $\|g(y) - y\|^2$ with respect to θ to achieve an efficient performance by training with limited amount of parameters.

2-1-4 Related Study - jDAS

Van der Dende et al., (2023) introduced jDAS, a self-supervised deep learning method utilising U-Net and J-invariance for waveform coherence enhancement and blind denoising in DAS data. By distinguishing between incoherent (noise) and coherent (earthquake) signals, jDAS recovers signals at low SNR. Trained on synthetic data for 300 epochs and further fine-tuned on two earthquake datasets, the model demonstrated the ability to separate signals within a common frequency band.

While effectively demonstrating the theoretical concepts applicable to blind denoising, the pre-trained jDAS model produced undesirable artifacts preceding the first arrival when applied to microseismic data, as illustrated in Figure 2-8. jDAS was specifically trained on strong seismic data characterised by significantly higher magnitude and distinct noise profiles compared to microseismic data. This discrepancy between the two datasets highlights the need for a model retraining to effectively address the characteristics of microseismic data.

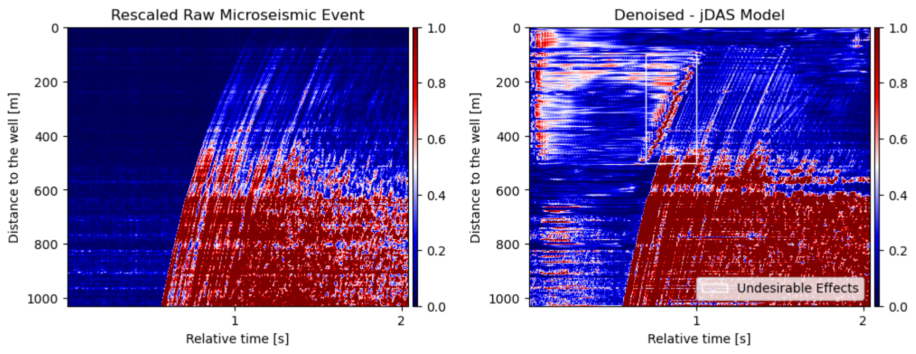


Figure 2-8: Impact of jDAS on Raw Microseismic Data. (a) Rescaled raw microseismic event. The detail description of the data would be provided in section 3.2. (b) Denoised event using jDAS. The white block highlights an undesirable artifact preceding the first arrival, which hinder subsequent data processing.

2-2 Training Strategy

To adopt the J-invariance filtering in machine learning, we aims to train a neural network that can reconstruct the coherent signal from a noisy input. The following subsections would reveal the training strategy in this project.

2-2-1 Masking

Masking is a technique that informs sequence processing layers that certain input elements are absent, and thus would be reproduced when the computer process the data. As illustrated in Figure 2-9, a blank trace and its adjacent traces are fed into a U-Net encoder to extract relevant features. The reconstructed trace is then compared to the original blank trace to compute a loss function. In synthetic clean DAS data, the original trace serves as the ground truth. The U-Net's contraction path effectively preserves strong coherent signals while suppressing random noise, resulting in a reconstructed trace devoid of noise components.

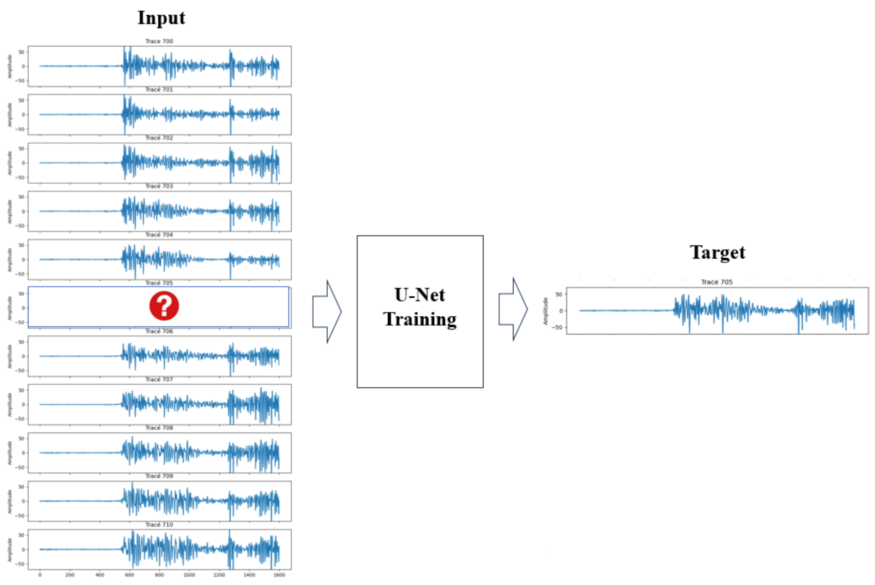


Figure 2-9: Illustration of the Masking: For each learning, we provide an input y with n neighbouring traces (where n = 11 in this cases), and we randomly blank one of the input traces and try to reconstruct the blanked one from the rest neighbouring traces.

2-2-2 Machine Learning Glossary and Values

Table 2-1 summarise the use of parameters in the training processes.

Parameter	**Description**	**Value**
Batch Size	the number of traces used for updating one step during network training. The number is determined by the balance between available computational resources and the dataset size .	32
Dropout Rate	A fraction in a layer that randomly sets some nodes to zero during the update step in training, which avoid overfitting. The chosen value is a standard practice	0.1
Epochs	The number of cycles for the model to learn all the training data. The chosen value is a standard practice	50
Learning Rate	The rate at which the model learns - controls the step size in the gradient descent update. The chosen value is a standard practice	0.0005
Loss Function	The function to penalise the model for being overly confident and the model should minimise it during training. Binary cross-entropy are used to measure the dissimilarity between the predicted probability distribution and the true binary labels of a dataset. It is defined as $-\log(\text{likelihood})$	Binary cross-entropy
Optimiser	The algorithm used to update weights during training. Details in section 2.1.1	Adam
Training Split	Percentage of input data used in training. The chosen value is a standard practice	80 %
Validation Split	Percentage of input data used in validation. The chosen value is a standard practice	20 %

Table 2-1: Fixed parameters used in the training process

2-3 Data Preparation

2-3-1 Distributed Acoustic Sensing Data (DAS)

DAS is a technology built on fiber optic sensing, which treats the glass fibers as the sensors and based on the photons interactions with the fluctuations of refractive index in the glass [Zhan, 2019]. As shown in figure 2-10, DAS uses Rayleigh backscattering to infer the longitudinal strain (i.e., ϵ_{xx} with x along the cable) or strain change over time ($\dot{\epsilon}_{xx}$) every few meters along the fiber [Li et al., 2021]. When there is a disturbance from seismic waves or other vibrations, the strain in each fiber section would change accordingly and carry the signals in return. With numerous scattering points along the fiber, it creates a high spatial resolution with the ability to measure any tiny extension or compression as the fiber alters the change in distances. As shown in figure 2-11, it is highly sensitive when P-wave travelling along the fiber and S-wave coming perpendicular.

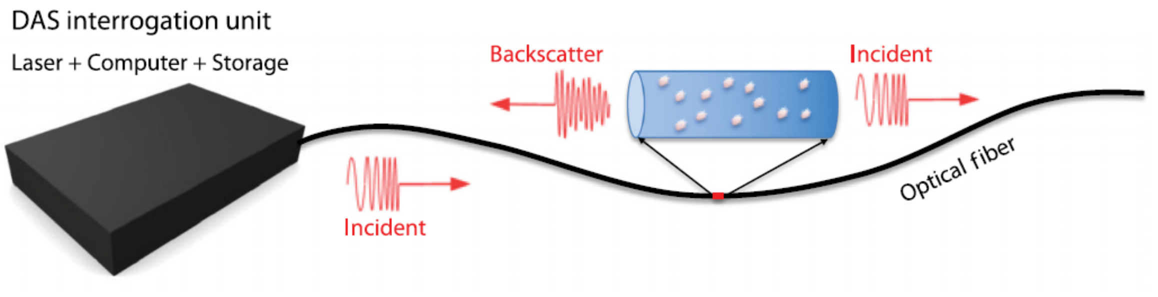


Figure 2-10: Principles of DAS and a synthetic demonstration. (a) A DAS unit is connected to one end of a long optical fiber cable. It sends laser pulses (either harmonic or chirp) into the fiber and analyses the Rayleigh backscattered light from the fiber's inherent defects. Data processing and storage are performed in real time within the DAS unit. Figure from [Zhan, 2019]

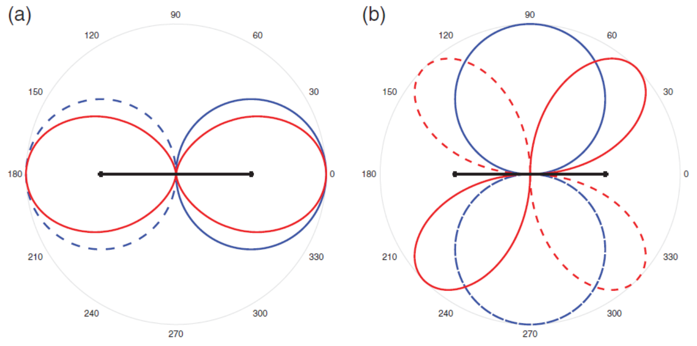


Figure 2-11: Illustrations of the directional sensitivity of DAS in a synthetic example along the horizontal axis (the black line). Blue line represent the incoming wave directions and red line represents the reaction from DAS. The solid line and dashed line represent the positive value and negative value respectively as a scalar. (a) is the situation when receiving signal horizontally. It reproduces the same waveform. (b) is the situation when receiving signal in a perpendicular directions and it distorts significantly. Figure from [Zhan, 2019]

Since 2010s, DAS has been used widely adopted in the field of in geophysics and seismology including in the multichannel channel analysis of shear-wave velocity in real-time [Dou S, 2017], industrial monitoring of microseismicity and hydraulic fracturing [Li et al., 2021] and surface imaging [Lapins et al., 2023], yet showing a high sensitivity to ambient noise, local and internal disturbances from the interrogators [Lindsey et al., 2020].

2-3-2 FORGE Data

In 2018, Frontier Observatory for Research in Geothermal Energy (FORGE) was established by the US Department of Energy (DOE) in south-central Utah to study the potential development of reservoir and geothermal energy. The Utah FORGE Site covers the surrounding 5 km^2 with a wind farm, solar field, and 38 MWe Blundell geothermal plant at Roosevelt Hot Springs. As shown in figure 2-12, the DAS was installed in monitoring well 78-32 from the surface down to around 1000 m depth. The main objective of the DAS data is to monitor the occurrence of injection-related microseimsic events which provides information about the fracture growth [Moore et al., 2023].

Acquisition has been quasi-continuously active from April 23rd, 2019, to May 3rd, 2019, and a single day acquisition on April 22, 2022 [Martin, 2022]. In this project we only focus on the data collected on April 22, 2022. DAS data have been recorded with a 1-m channel spacing, 10-m gauge length at a sampling frequency of 2000 Hz (2000 data points per second) after a 16-fold internal stacking of the laser sampling rate before writing to the disk [Lellouch et al., 2021].

Figure 2-13 will show some of the examples of the FORGE Data, which contains some strong microseismic events with SNR larger than 10 and weak microseimsic events with SNR smaller than 3. A dataset of 45 strong microseismic events, each consisting of 1029 channels and 4096 time samples, is used for both training and validation. The weak microseismic events would be used as evaluating the performance of the denoised algorithm, as they are hard to detect and reveal without proper data processing.

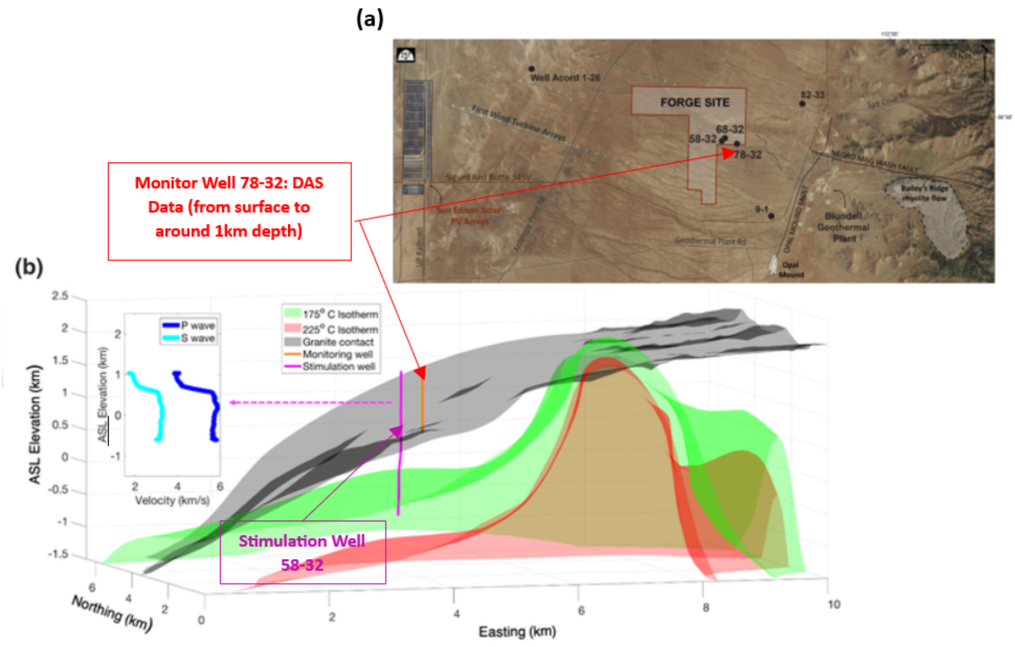


Figure 2-12: FORGE Site Plan. (a): The aerial photo of the site. 3 wells are drilled: 58-32 is the stimulation well, 78-32 is the monitoring well with installed DAS and 68-32 is instrumented with traditional geophones. (b) The cross-section of the subsurface area in the Utah. Two isotherm surfaces (green – 175°C, red – 225°C) and the granite contact (black) are plotted. Figure adapted from [Moore et al., 2023] and [Lellouch et al., 2021]

2-3-3 Data Pre-processing

Before the data are put in training process, the raw data undergoes several preprocessing steps to be transformed into a structured format suitable for machine learning.

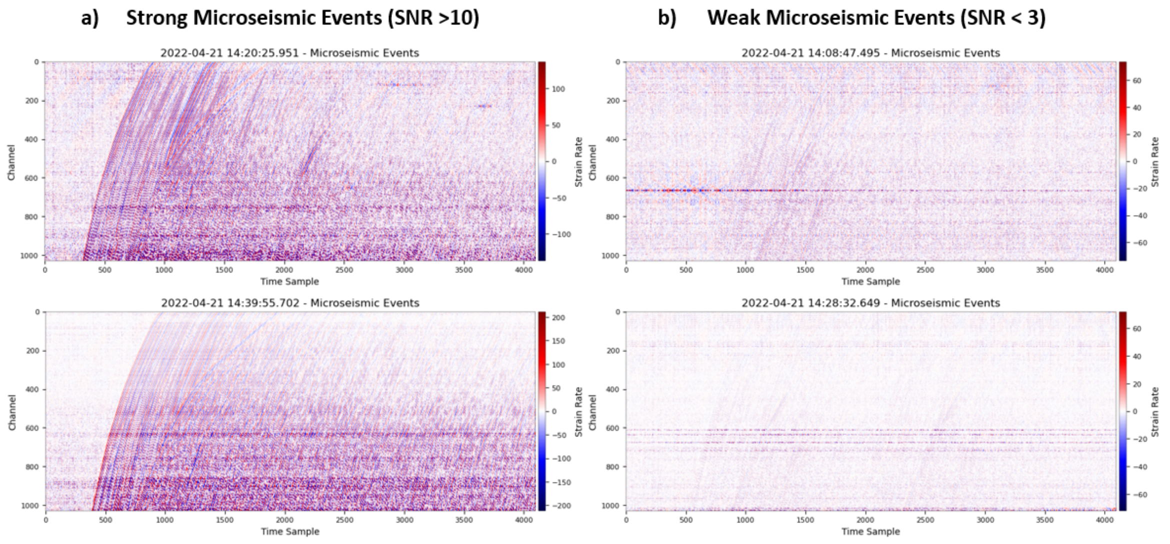


Figure 2-13: (a) Examples of strong microseismic events with SNR > 10 recorded on April 22, 2022. Such events are are used in training. (b) Examples of the weak microseismic events SNR < 3. Such events are are hard to detect without proper denoising.

Percentile Clipping

As stated in the chapter "Activation Layer" under subsection 2-1-1, the outlier (i.e, extreme value in strain rate amplitude) should be removed by percentile clipping for better training when using activation layers like Swish or Relu. We use the 99.7th percentile and 92610 data points which containing the extreme value will be set to zero for each profile (one profile contains 1029 channels, with 30000 points collected per channels per 15 second). Strong microseismic events are trimmed within the data, and their corresponding 4096-sample segments extracted for each channel which mostly include a full period of the events (with primary-waves,secondary waves and coda waves).

Scaling Methods

Feature scaling is a crucial step to build accurate and effective machine learning models as it makes easier for algorithms to find the optimal solution, and results in a faster convergence speed. We investigate three types of scaling methods would be investigated, namely min-max normalisation, max-absolute normalisation and standardisation.

Figure 2-14 shows the concepts of the three scaling methods and effects on the data. Min-max normalisation would take the maximum value as 1 and minimum value as 0. With generally symmetric amplitude distribution of seismic waves, background noise tends to cluster around zero would be normalised with value close to 0.5. Max-absolute takes the absolute value of the amplitude, which all data would also fall between 0 and 1. Background noise, initially centered around zero, remains concentrated near this lower bound within the transformed data. For Standardisation, The standardised value is given by the formula:

$$\text{Standardised value} = \frac{X - \mu}{\sigma},$$

where X is the data, μ represents the mean and σ represents the standard deviation. The value would not have a fixed boundary of maximum and minimum, and the directional property of seismic waves are maintained with positive and negative value.

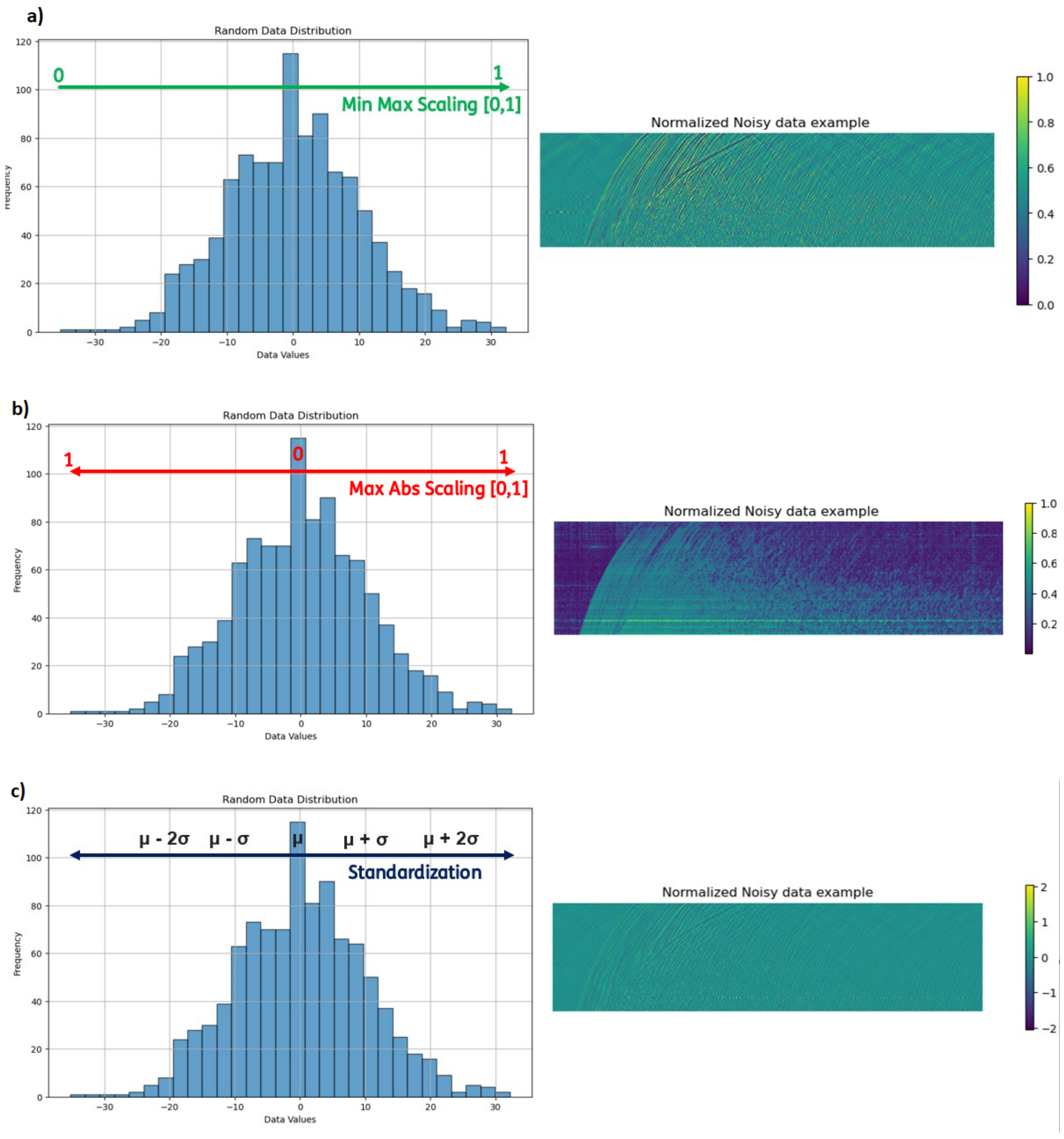


Figure 2-14: Using various scaling methods on the same microseismic data. (a) is min-max normalisation. Values that are more negative will be scaled towards 0, while values that are more positive will be scaled towards 1. (b) is max absolute normalisation. Smaller values will be scaled towards 0, while larger values will be scaled towards 1. (c) is standardisation. Data are scaled with mean μ and standard deviation σ.

2-3-4 Input Samples Generation

Time Slice Windows

To create a more diverse dataset, we extract different time-slice windows raw data. As shown in figure 2-15, for the creation of each input data, we create a random starting point between time sample point 500 and 2000 and extract a window slice from that point with a length of 2048 points to prevent profiles with only noise or only signals. As a result, for each training data, we would have a first arrival with different start time for each input to generate a comprehensive training dataset from only one event. In the figure 2-15 to 2-17, for the sake of maintaining the continuity of the explanation, we demonstrated the process using input traces that had undergone min-max scaling with 11 traces as a patch which is .

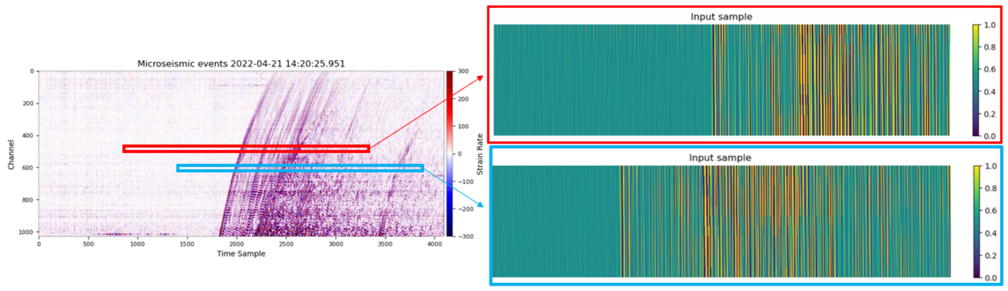


Figure 2-15: Illustration of the time slice windows. The left panel is the raw data while the right panels show a patch of input traces (11 traces) after doing a min-max scaling per trace. The red window demonstrates an earlier start of the sampling windows while the blue window demonstrates a later start.

Time Flipping

A model trained on both original and time-flipped data can learn to recognise patterns more robustly, regardless of their temporal orientation. Therefore, we create an extra condition for the algorithm to decide whether to flip a dataset randomly as shown in figure 2-16.

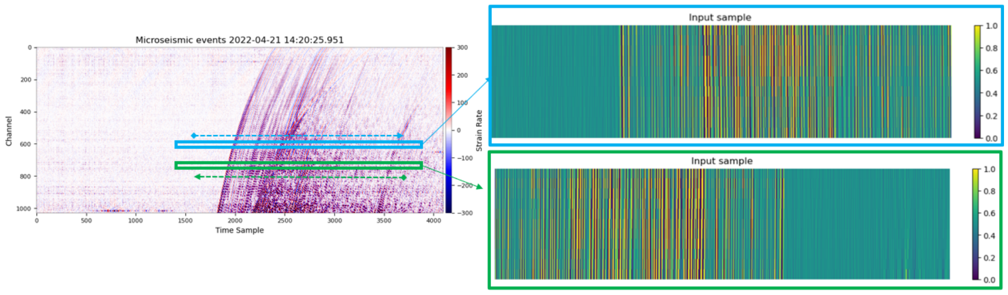


Figure 2-16: Illustration of the time flipping, with the same dataset as in figure 2-15 The blue window demonstrates an example along the time sequence while the green window demonstrates a time-flipped case.

Data Expansion

To expand the dataset, we generate additional masks for each data input. This involves creating variations by blanking out traces randomly from the same input, effectively enlarging the dataset. Figure 2-17 shows how we create three pairs of input samples and targets based on one patch of input data.

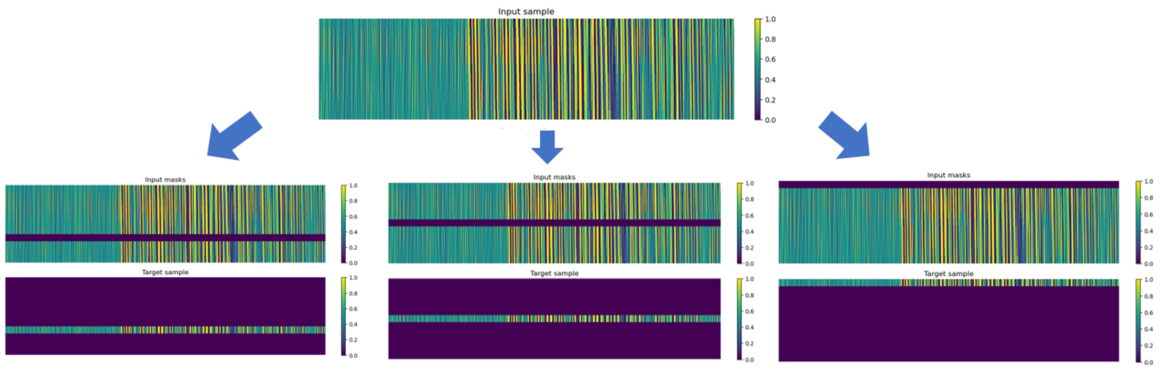


Figure 2-17: Different masking applied on one input data (top). Data enlarged by a factor of 3 while we select randomly a different trace as a missing trace (middle) and thus as the target to be reconstructed (bottom).

2-3-5 Hyperparameter Testing

Table 2-1 reveals the fixed parameter used in the model training, while the table 2-2 details the hyperparameters subjected to optimisation. To enhance model performance, a series

Hyperparameter	Description	Variables
Activation Function	The function applied to the output of a layer to introduce non-linearity.	*ReLu, Swish, Tanh
Number of Input Traces	The quantity of seismic traces processed simultaneously in a single masking operation..	*11, 31, 51
Mask Generation Rate	The number of masks generated per input data which determines the the overall size of the augmented dataset.	1, 2, *3
Scaling Methods	Features are scaled to be governed in a particular range and perform faster convergence.	*Min-max normalisation, Max-abs normalisation, Standardisation
Time Flipping Activation	Whether time flipping is applied during input data generation.	Yes or *No

Table 2-2: Varying hyperparameters in the training process, (* represents the chosen value for baseline model)

of retraining experiments would be conducted. Hyperparameters including input sample generation techniques (e.g., additional mask generation, time flipping), activation functions, scaling methods, and the number of input traces would systematically varied.

2-4 Microseismic-event Detection

Microseismic event detection is a key step in the analysis of monitoring data. Two primary methods for this task are the Short-Term Average/Long-Term Average (STA/LTA) and the Coherence-based Earthquake Detector (HECTOR). STA/LTA offers rapid computation but is susceptible to noise interference. Conversely, HECTOR exhibits superior noise resilience and accuracy but demands significantly higher computational resources, rendering real-time implementation impractical. This study will evaluate the performance of STA/LTA on denoised data and employ HECTOR as a benchmark for validating detection results, distinguishing between true and false positives.

2-4-1 STA/LTA Method

The STA/LTA method is often considered as the most popular method for (micro)seismic-event picking and detection. It is applied by continuously calculating the ratio between two running windows with different lengths [Vaezi and Van der Baan, 2015]. Mathematically, the ratio R is defined as:

$$R = \frac{\text{STA}}{\text{LTA}}, \tag{2-11}$$

where

$$\text{STA} = \frac{1}{N_S}\sum_{n=1}^{N_S} y_{k,n}, \tag{2-12}$$

and

$$\text{LTA} = \frac{1}{N_L}\sum_{n=-N_L}^{0} y_{k,n}. \tag{2-13}$$

The ratio is the amplitude y calculated continuously at each time moment corresponding to time sample n=0 for every kth data channel, and N_S and N_L is the number of time samples used for the average in a short and long window. In general, long-term average windows would be representing the ambient noise and an event should have a sufficiently higher amplitude than the background noise. Combining the studies of [LLC, 2022] and [Trnkoczy, 2009], 1309 microseismic events were detected on the 22nd April 2022, with N_S defined as 0.5 second and N_L defined as 10 seconds so that the STA can capture rapid amplitude variations while LTA tcan represent a window longer than several seismic wave periods. Only SNR which is larger than 1.9 after a simple bandpass filtering 10Hz to 150Hz was able to detected in their studies.

2-4-2 Coherence-based Earthquake Detector (HECTOR)

HECTOR exploits the dense spatial sampling of DAS for real-time microseismic monitoring by evaluating the coherence of seismic waveforms along hyperbolic trajectories on a linear segment of the optical fiber using the Semblance function [Porras et al., 2024]. Begin with the equation of the coherence hyperbola:

$$t_i(T, X, C) = \sqrt{T^2 + \frac{(x_i - X)^2}{C^2}}, \tag{2-14}$$

where t_i is the time of the trace recorded at position xi, X is the spatial offset with respect to the vertex of the hyperbola, T is the time offset, and C is the velocity. We can then formulate the Semblance function used for the coherence analysis:

$$S(T, X, C) = \frac{\sum_{j=1}^{N} \left(\sum_{i=1}^{M} A(t_{ij}) \right)^2}{M \sum_{j=1}^{N} \sum_{i=1}^{M} A(t_{ij})^2} \quad \text{with} \quad t_{ij} = t_i(T, X, C) + j\, dt, \tag{2-15}$$

where

- M is the total number of seismic traces
- N is the length of the sample window
- A is the amplitude of the $i$traces at time ij.

Figure 2-18 shows the sketch of the HECTOR Method. In short, seismic noise is less likely to follow a hyperbolic trend like the seismic signals and will have a low coherence value, which is a similar concept to the denoising approach. With the HECTOR Method we find 1018 extra detections on the same day with an even lower percentage of ”false detection” (i.e, 5.6 % for HECTOR while 6.0 % for STA/LTA). While covariance matrix creation is computationally demanding, requiring 250% of acquisition time with with an Intel quad- core i7 processor and 16 GB of Random Access Memory (RAM) as reported by Porreas et al., (2024), it remains a valuable tool for validating denoised algorithms in conjunction with STA/LTA analysis.

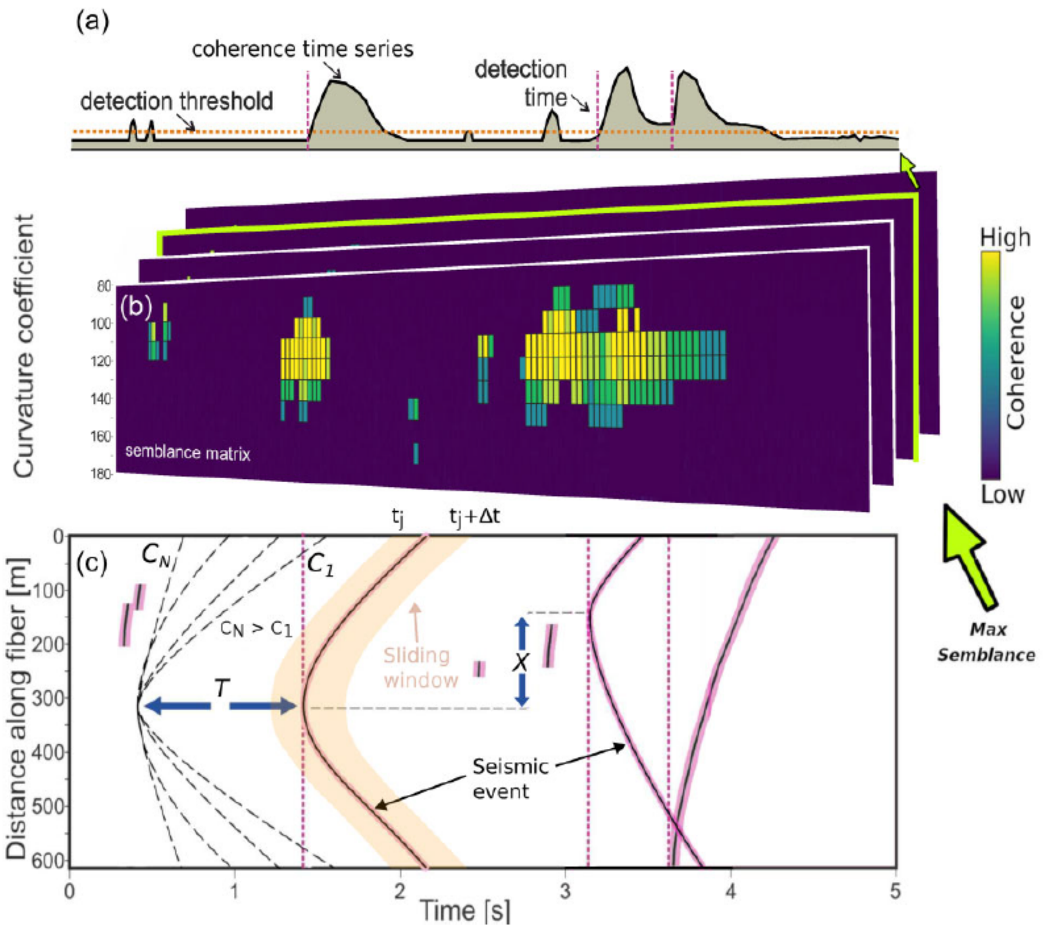


Figure 2-18: Illustration of the semblance-based detection method, figure from [Porras et al., 2024]. (a) Coherence time series obtained from the waveform data allowing detection of long-term and strong coherence. The detection time would always be assigned at the first arrival. (b) Example 2-D semblance matrices based on equation 2-15 in time domain and the vertical axis is the relative distance. (c) The geometrical hyperbola (coefficient C_1 to C_N) at each X; the hyperbolic trajectories that do not follow the seismic wavefield will have a low semblance value (close to 0) while the presence of a hyperbolic event along a finite-width data window would imply a seismic event.

Chapter 3

Results

3-1 Machine Learning Analysis

Based on the discussed training strategy, we perform a series of retraining experiments to optimise model performance. Model performances are evaluated using RMSE calculated for both training and validation datasets, which is called training loss and validation loss. The training loss can indicate how well the model is fitting with the training data, while the validation loss can indicate how well the model fits unseen data.

Figure 3-1 to 3-5 shows the comparison of training loss and validation loss among different hyperparameter testing. As mentioned in Table 2-2, a baseline model is established using 11 input traces, min-max scaling applied independently to each trace, 3-mask implementation, time-slice windows, and the Swish activation function. Hyperparameter tuning is conducted by systematically varying one parameter at a time. The baseline model generates 12555 samples, divided into 313 training mini-batches and 104 validation mini-batches (80% to 20% training and validation data split), each mini-batch containing 32 samples. To prevent overfitting, an early stopping criterion is implemented. Training is terminated after 10 consecutive epochs without a significant reduction in loss. The model parameters corresponding to the epoch with the lowest validation loss is retained.

The detailed model architecture for the baseline model is provided in the appendix. The U-Net comprises a total of nine layers, evenly divided into four downsampling and four upsampling stages. The model contains 204,729 trainable parameters. For an input of size 11 x 2048, the network undergoes four levels of downsampling and upsampling, each reducing or expanding spatial dimensions by a factor of four. Concurrently, the number of parameters per layer doubles and halves, respectively, across these stages. The bottleneck layer reduces the data to a shape of 11 x 8 with 61504 parameters tunning based on these 88 data samples.

Influence of Data Augmentation

We investigate the efficiency of masking within the data augmentation pipeline for model training. To assess the impact of the masking extent, we apply varying numbers of masks (1, 2, or 3) to each data point. This results in the creation of mini-batches with sizes corresponding to the masking scheme (99, 209, and 313 for training and 26, 52, and 104 for validation). As anticipated, computational cost per epoch increases with the number of applied masks (53, 100, and 152 seconds for 1, 2, and 3 masks, respectively).

As shown in Figure 3-1, both training and validation performance show significant improvement with a general trend of decreasing loss when employing 3 masks compared to a single mask. when using 3 masks, the model achieves a minimum loss of around 0.0030 with higher stability and reduced fluctuations compared to other masking approaches. This observation holds true with the adopted evaluation metric, Root Mean Squared Error (RMSE), emphasising average error and not explicitly accounting for data size. Based on these findings, we select the 3-masking strategy to optimise the training process.

We further investigate the effect of introducing random flips (reverse flipping) during data augmentation. Flipping is applied randomly to approximately half the data (49.6% with

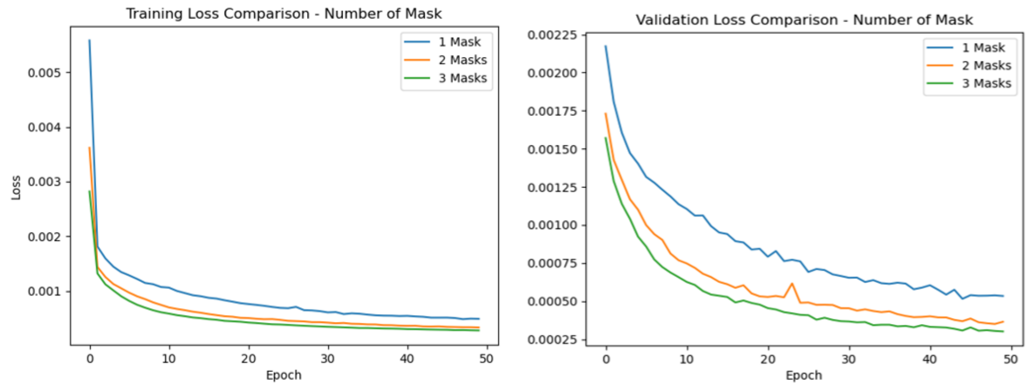


Figure 3-1: The training and validation loss with 1,2, and 3 masking applied in retraining.

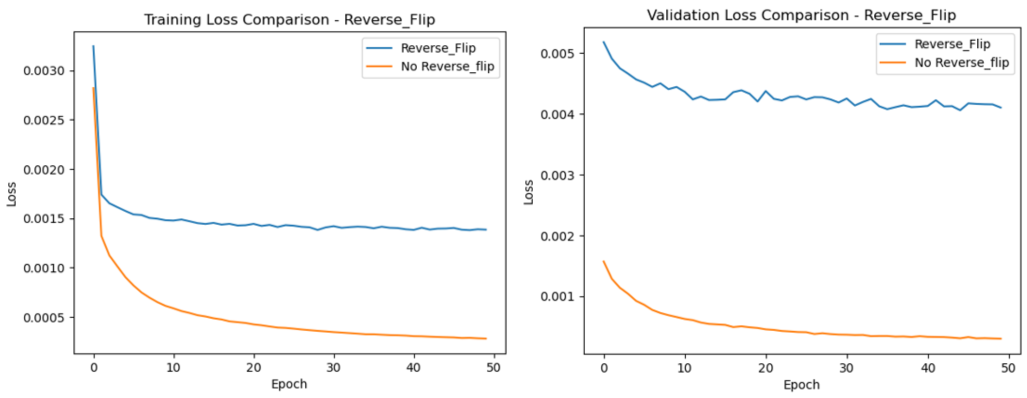


Figure 3-2: The training and validation loss with or without random flipping.

4968 out of 10016 data patches for the 3-masks strategy). As shown in figure 3-2, the flipped datasets introduce greater data diversity, leading to a significant increase in both training and validation loss, with factors of 7.5 and 20.5, respectively, and reduce stability. Despite the higher loss, we see that the training and validation curves demonstrate overall improved learning in some cases. This suggests the model learns long-range coherence independent of data orientation, enhancing its applicability to general blind denoising.

Influence of Scaling Methods

In Figure 3-3, we find that both scaling methods that map data to the range of 0 to 1 (MinMaxScaler, MaxAbsScaler), and exhibit smooth convergence and achieve a loss magnitude of 10^{-4}. Both MinMaxScaler and MaxAbsScaler demonstrate high learning stability, with MinMaxScaler attaining the minimum loss for both training and validation at $2.87*10^{-4}$ and $2.94*10^{-4}$ respectively. Conversely, training with StandardScaler, which typically scales data to a range of -2 to 2, results in oscillatory convergence and low stability, characterised by the wriggles along the learning curve. Both training and validation loss reach a minimum around 0.114, triggering early stopping at approximately 20 epochs.

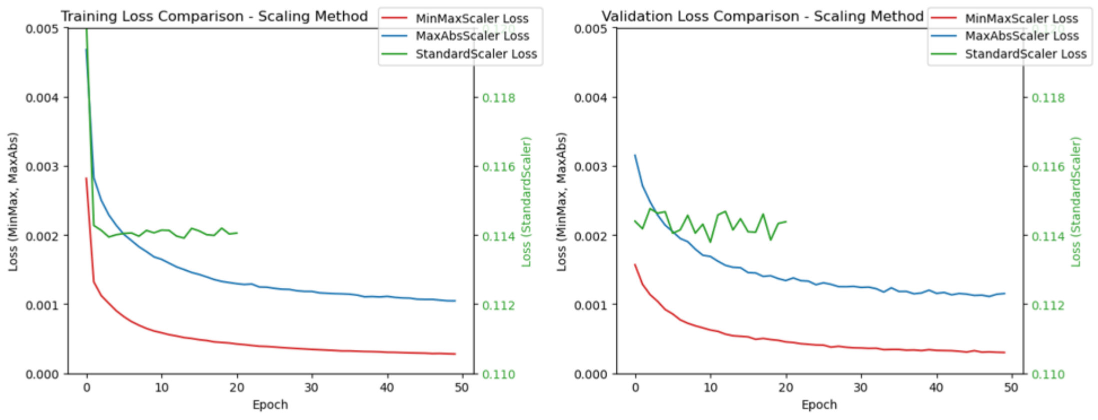


Figure 3-3: The training and validation loss with 3 different scaler: MinMax normalisation, MaxAbs normalisation, and StandardScaler. The left y-axis (black colour) indicates the loss with regards to MinMax and MaxAbs, while the right y-axis (green colour) indicates the loss with regards to StandardScaler

Influence of Activation Layer

As shown in Figure 3-4, the choice of activation layers (ReLU, Swish, and Tanh) minimally impacts overall convergence and learning process. With MinMax scaling, all activation functions achieve convergence. Standardising the data (StandardScaler) leads to oscillatory convergence and early stopping activation in all cases. An interesting phenomenon observed with StandardScaler combining with Tanh activation is a longer learning period and delayed early stopping, suggesting a higher initial convergence level in the first few epochs before oscillations commence.

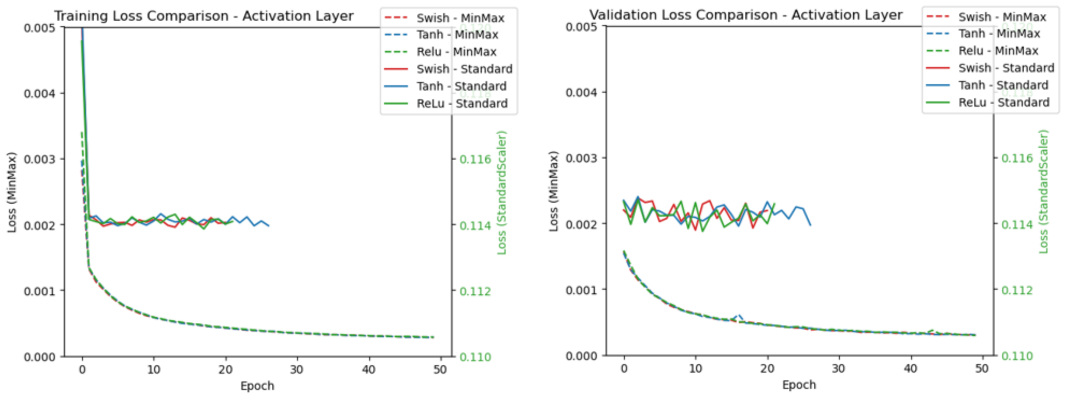


Figure 3-4: The training and validation loss with 3 different activation layer on both minmax scaler and standardscaler, which are ReLu, Swish, and Tanh. The left y-axis (black colour) indicates the loss with regards to MinMax, while the right y-axis (green colour) indicates the loss with regards to StandardScaler

Influence of Input Traces

As shown in Figure 3-5, We investigate the influence of different numbers of input traces (11, 31, 51 traces) on the number of samples considered (10, 30, and 50 neighbouring traces) for reconstructing the blanked target. In all cases (11, 31, and 51 input traces), the model achieves promising convergence during training. Employing more input traces (31 and 51) leads to a greater decrease in both training and validation loss compared to using only 11 traces and results in a lower minimum loss. Interestingly, while both 31 and 51 traces achieve similar loss, the model with 31 traces demonstrates higher stability in convergence during both training and validation with less oscillatory behaviour, indicating a good balance between accuracy and stability.

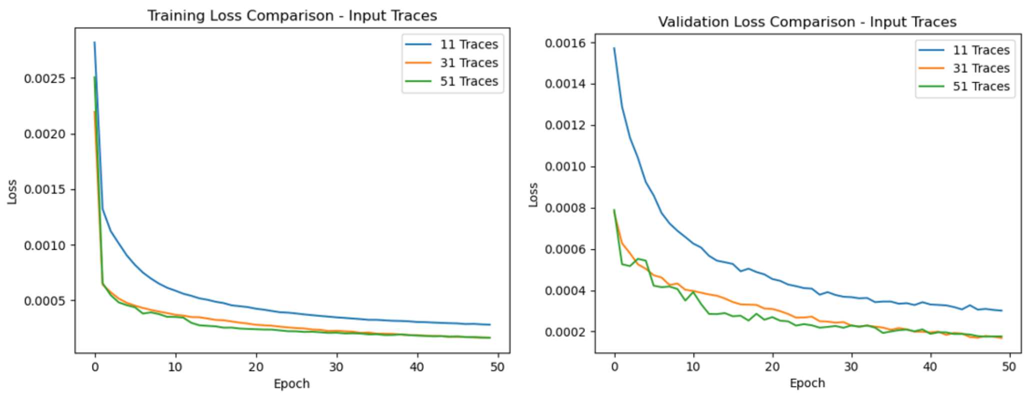


Figure 3-5: The training and validation loss with 11, 31, and 51 input traces

3-2 Denoising Test Results

3-2-1 Performance Metrics

To begin with, we define our relative SNR with the following steps.

1. Extract the top 10% of the signal data.
2. Calculate the signal power:

$$P_{\text{signal}} = \frac{1}{N} \sum_{i=1}^{N} (x_i)^2 , \tag{3-1}$$

where x_i represents the top 10% of the signal data, and N is the number of data points in the top 10%.

3. Calculate the noise power:

$$P_{\text{noise}} = \frac{1}{M} \sum_{j=1}^{M} (n_j)^2 , \tag{3-2}$$

where n_j represents the noise data, and M is the number of noise data points.

4. Calculate the relative SNR:

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} \tag{3-3}$$

3-2-2 Experimental Design

Table 3-1 outlines the retraining process employed in this study. To evaluate the denoising performance, the listed models are applied to both strong and weak microseismic events.

Retraining Strategy	**Mask Generation**	**Input Trace Number**	**Scaling Methods**	**Time Flipping**
A (Baseline)	3 masks	11 traces	Min-max	No
B	3 masks	11 traces	Min-max	Yes
C	3 masks	11 traces	Max-abs	No
D	3 masks	11 traces	Standardisation	No
E	3 masks	31 traces	Min-max	No
F	3 masks	51 traces	Min-max	No

Table 3-1: The varying variables in different retraining strategy

3-2-3 Strong Microseismic Events Analysis

Figure 3-6 presents an example of a recorded testing event with a high SNR of 9.77. To differentiate information carried by different seismic wave types, the analysis defines two SNR categories: P-wave SNR and S-wave SNR. P-wave SNR reflects signal strength relative to the first arrival (P-wave), crucial for microseismic detection. Conversely, S-wave SNR

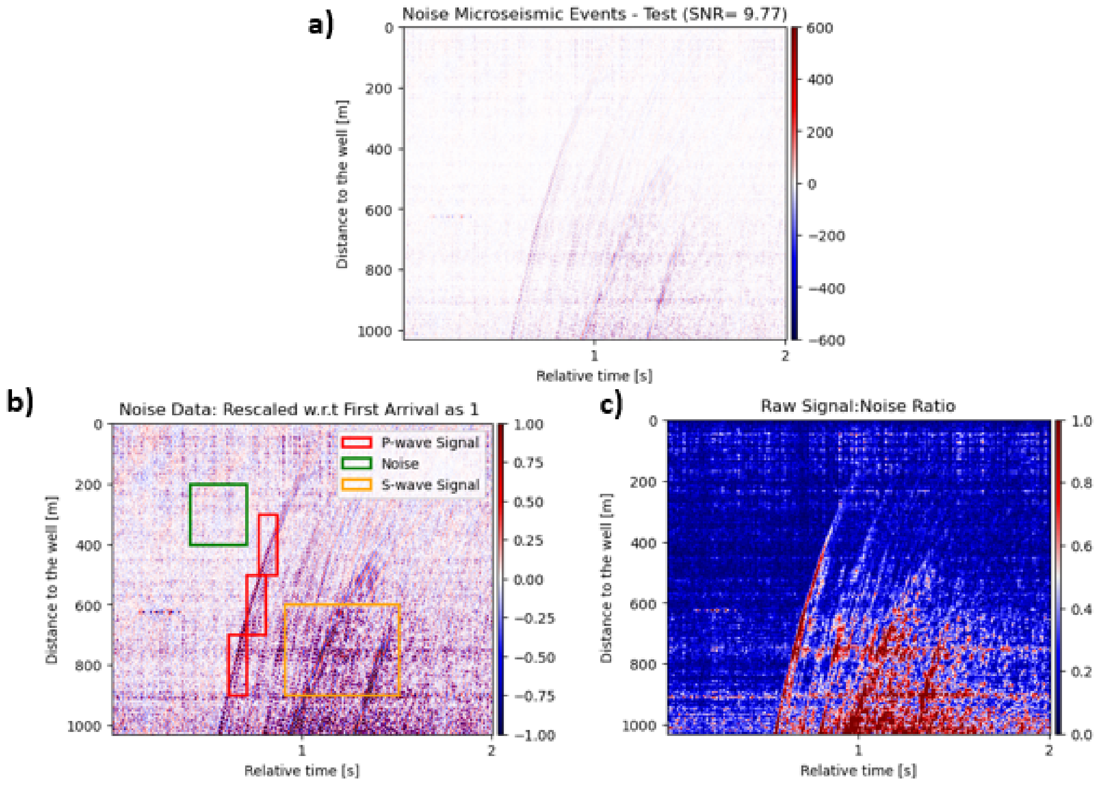


Figure 3-6: (a)Example of a raw FORGE microseimsic event with SNR ratio = 9.77. (b) Rescaled version of (a) with normalisation by the amplitude of the first P-wave arrivals. The red block, green block and yellow block represent the part extracting the P-wave signal, Noise and S-wave Signal respectively. (c) Absolute value of the data in (b) used to facilitate qualitative assessment

indicates potential for retrieving information for full-waveform inversion based on shear wave strength. For comparison, Figure 3-6b and 3-6c display the same data rescaled and with absolute amplitude calculated, both referenced to the first arrival.

The listed six denoising methods in Table 3-1 are evaluated (Figures 3-7 to 3-10). The corresponding P-wave and S-wave SNR values for each method are presented in Table 3-2. Combining results, the Strategy B and Strategy F demonstrate the most promising performance, achieving P-wave and S-wave SNR values exceeding 10 in both cases. Figures 3-8b and 3-12b reveal effective preservation of high amplitudes, particularly at greater depths (beyond channel 500), with clear identification of first arrival and shear waves. Interestingly, Strategy B demonstrates superior denoising performance before the arrival of seismic events, while Strategy F excels better in post-shear wave denoising.

Strategy A shows a good performance especially with locating the first arrival of events as evident in Figure 3-7. A clear first arrival and shear wave are identified, supported by a fivefold increase in P-wave SNR compared to raw data. A common observation among all strategies using 11 input traces (Strategy A to D) is their effectiveness in retrieving the first arrival. However, these methods also exhibit some signal loss within the middle portion of the signal, which is further proven by the insignificant improvement or even a decrease in SNR in S-Wave SNR calculation. Additionally, Strategy C and Strategy D struggle with parallel noise, as shown in Figures 3-9b and 3-10b. These methods tended to retain a significant amount of noise, particularly when encountering parallel noise patterns.

Strategy E demonstrates poor performance, amplifying noise instead of the desired signal (Figure 3-11). Corresponding SNR values in Table 3-2 confirm this, with P-wave SNR comparable to raw data and decreased S-wave SNR. Consequently, Strategy E is excluded from further analysis.

In conclusion, all denoising methods achieve improvement in retrieving the first arrival of seismic events. However, only Strategy B and Strategy F demonstrated a significant ability to preserve the original signal integrity.

Strategy	**P-wave SNR**	**S-wave SNR**
Raw Data	9.776	16.501
A (Baseline)	32.511	17.253
B	93.280	203.815
C	15.275	7.699
D	14.715	17.107
E	10.265	8.741
F	133.348	264.136

Table 3-2: Comparison of P-wave and S-wave SNR values for different processing strategies.

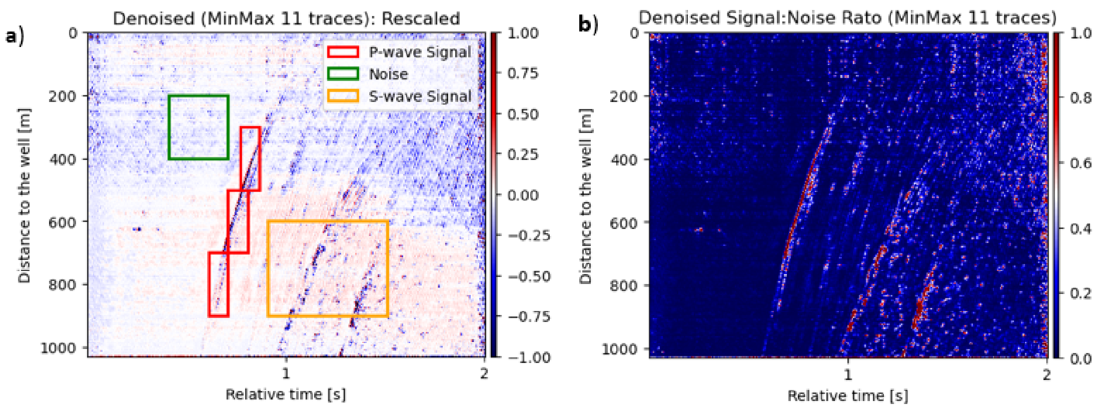


Figure 3-7: Test case with Strategy A: (a) The denoised Profile with respect to 3-6b and (b) the denoised absolute amplitude profile with respect to 3-6c. A darker background in the noise distribution plot indicates more effective noise removal.

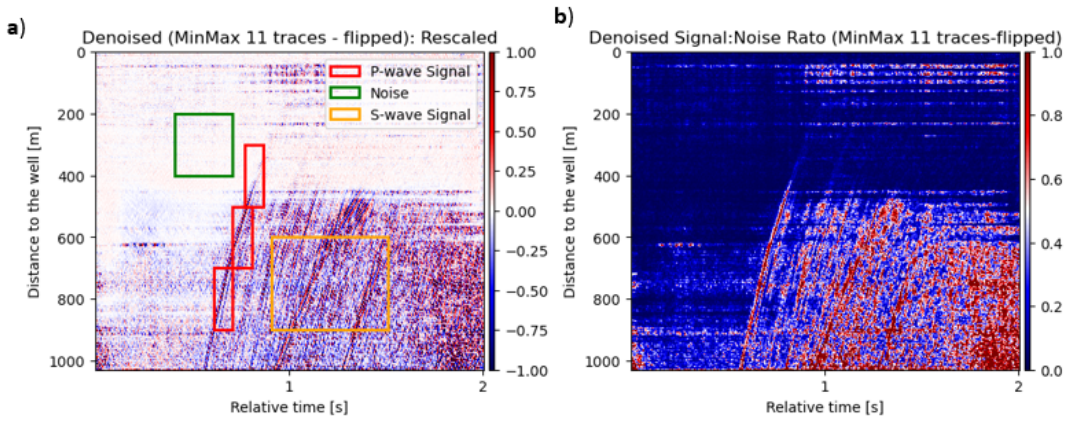


Figure 3-8: Test case with Strategy B: (a) The denoised Profile with respect to 3-6b and (b) the denoised absolute amplitude profile with respect to 3-6c. A darker background in the noise distribution plot indicates more effective noise removal.

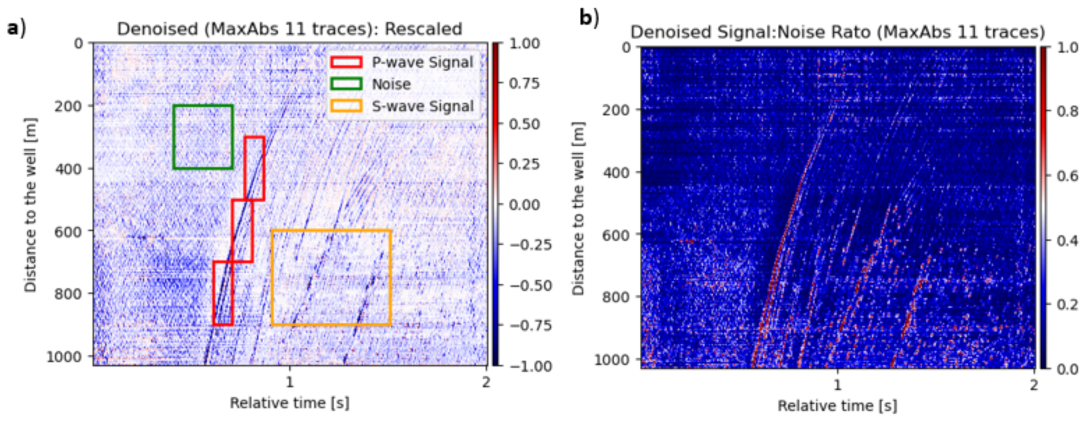


Figure 3-9: Test case with Strategy C: (a) The denoised Profile with respect to 3-6b and (b) the denoised absolute amplitude profile with respect to 3-6c. A darker background in the noise distribution plot indicates more effective noise removal.

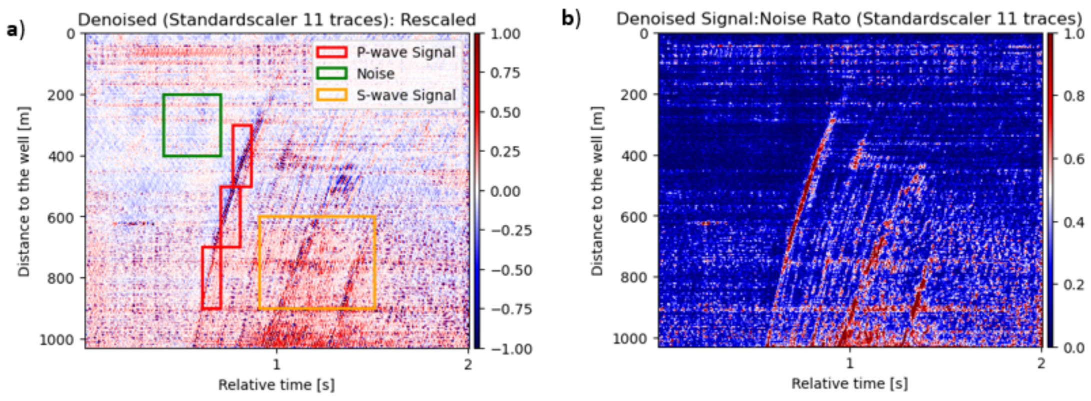


Figure 3-10: Test case with Strategy D: (a) The denoised Profile with respect to 3-6b and (b) the denoised absolute amplitude profile with respect to 3-6c. A darker background in the noise distribution plot indicates more effective noise removal.

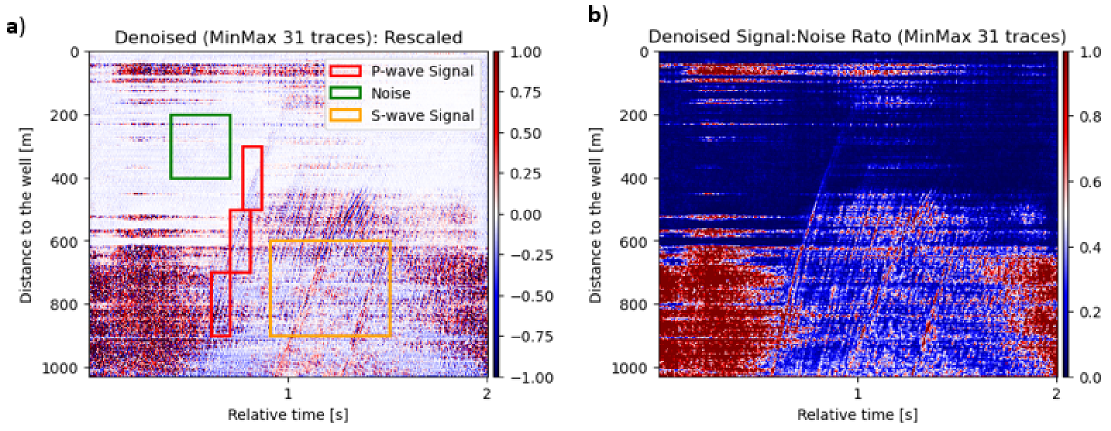


Figure 3-11: Test case with Strategy E: (a) The denoised Profile with respect to 3-6b and (b) the denoised absolute amplitude profile with respect to 3-6c. A darker background in the noise distribution plot indicates more effective noise removal.

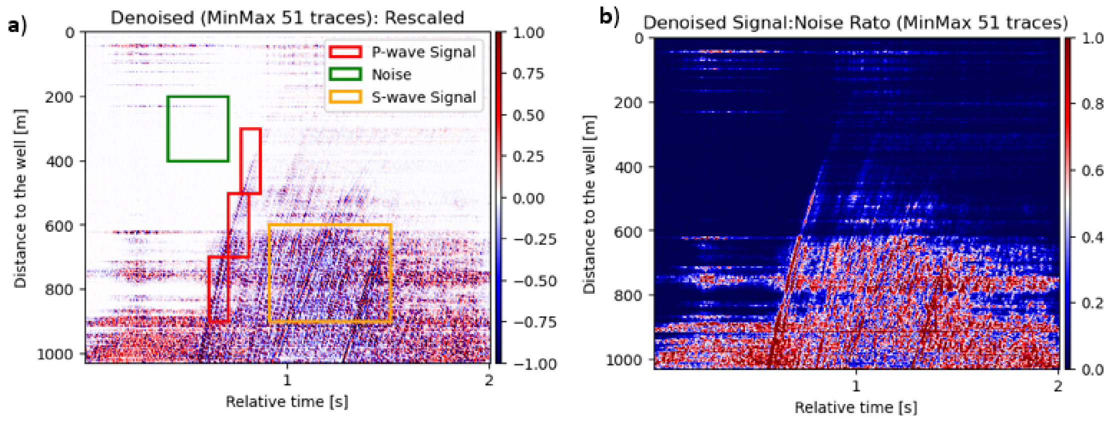


Figure 3-12: Test case with Strategy F: (a) The denoised Profile with respect to 3-6b and (b) the denoised absolute amplitude profile with respect to 3-6c. A darker background in the noise distribution plot indicates more effective noise removal.

3-2-4 Weak Microseismic Events Analysis

To address the limitations of the STA/LTA method in detecting weak microseismic events with low SNR, we focus on enhancing events with SNR below 1.9. To simulate this challenge, Figure 3-13 presents two scenarios with strong white Gaussian noise (150-500 Hz) added to the original profile, resulting in SNRs of 0.8 and 0.02. Figure 3-13e and f show corresponding signal-to-noise ratios with first arrival intensity as reference (value 1). Compared to Figure 3-6c, background noise approaches 1 in Figure 3-13e and the signal is completely obscured in Figure 3-13f. This noise is added to assess algorithm robustness, followed by post-filtering (10-150 Hz) to remove artificial noise and evaluate signal coherence preservation which the inherent coherence within the signal should be preserved, even if it is weak.

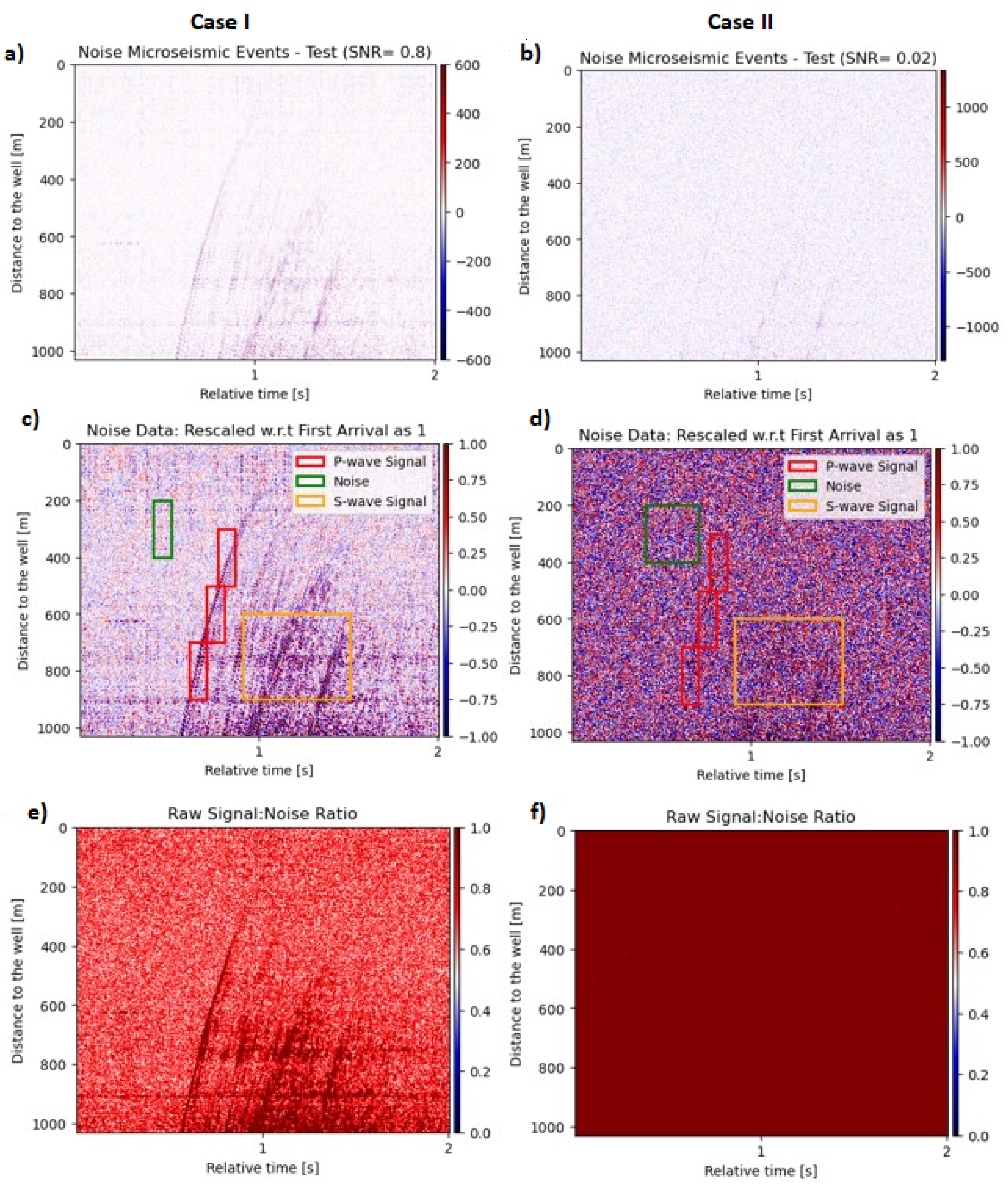


Figure 3-13: (a) and (b) are the profile of a weak FORGE microseimsic event with white random noise added which the corresponding SNR ratio is 0.8 (Case I) and 0.02 (Case II) respectively. (c) and (d) are the rescaled versions of Case I and Case II with normalisation by the amplitude of the first P-wave arrivals. The red block, green block and yellow block represent the part extracting the P-wave signal, Noise and S-wave Signal respectively. (e) and (f) are the absolute value of the data in (c) and (d) used to facilitate qualitative assessment. In (f) it shows that this test case it is a completely hidden signal with respect to first arrival amplitude.

Table 3-3 summaries SNRs before post-filtering. Strategy B remains as the best performance in enhancing SNR ratio in both cases, with an increased of SNR ratio from 0.0853 to 23.664 and 1.436 to 45.622 in P-wave SNR and S-wave SNR respectively in case I. In case

II, it improves by a factor of 1000, reaching 14.948 and 32.373, respectively. The second best performance is Strategy C, showing an improvement at a factor around 6 in case I and a factor around 60 in case II for both P-wave SNR and S-wave SNR respectively. The Strategy A and Strategy D show slight improvement in case I and greater improvement in case II. Strategy F decreases P-wave SNR and shows insignificant S-wave SNR improvement in both cases. Compared to Table 3-2, Strategy F performs better on strong events, while Strategy C excels on weak events. Other strategies show similar performance across both event types.

Strategy	**case I P-wave SNR**	**case I S-wave SNR**	**case II P-wave SNR**	**case II S-wave SNR**
Raw Data	0.853	1.436	0.028	0.0485
A (Baseline)	1.436	2.151	0.436	0.641
B	23.664	45.622	14.948	32.373
C	6.027	8.990	1.704	3.158
D	1.275	1.634	0.908	0.874
F	0.254	1.887	0.0176	0.084

Table 3-3: Comparison of P-wave and S-wave SNR values for different processing strategies.

Figures 3-14 to 3-17 present denoised and post-filtered results for both weak microseismic events. For case I, all methods preserve the main signal, with clear P-wave and S-wave first arrivals shown in Strategy A, Strategy B and Strategy C. Strategy F shows a clear P-wave first arrival, particularly between channels 500 and 900 (Figure 3-18a), but S-wave arrival is less distinct due to noise interference. Strategy D is able to detect all the arrivals, but the amplitude is distorted and it is widely covered by noise.

For case II, only Strategy B is able to capture a clear P and S wave arrival with high amplitude, as shown in Figure 3-15d. Strategy A, Strategy B and Strategy F partially recover the signal, with less clear P-wave arrival in Strategy F. Strategy D fails to recover the signal.

Post-filtering is applied to all methods to evaluate the effectiveness of random noise removal, as noise was manually introduced at a specific frequency. Strategy B shows an excellent performance in capturing the long-range coherence value. This method effectively denoises the data while preserves the signal, as evident in Figures 3-15e and f for both case I and case II. Strategy A and Strategy C perform well for case I but tend to remove very weak microseismic events, as shown in 3-14h and 3-16h respectively. Similarly, Strategy F performs well in case I but misses the P-wave first arrival in case II(see Figures 3-18f and h. Strategy D completely misses case II (Figure 3-17f) and preserved a weaker signal compared to other methods in case I (Figure 3-17e).

A common artifact, circled in white dotted lines in Figures 3-14c, d, 3-18c,d, and 3-16c, d, is observed in the results of Strategy A, Strategy C, and Strategy F. These artifacts are more pronounced for events with lower SNRs and exhibit varying temporal occurrences across methods. While Strategy A and Strategy C tend to produce artifacts before and after the signal, Strategy F exhibits artifacts around the 1.0-1.5 second mark, potentially obscuring the S-wave arrival.

Interestingly, artifacts observed in Strategy A and Strategy C are successfully mitigated by the post-filtering step (Figures 3-14g and h and 3-16g and h. In contrast, artifacts persist in Strategy F even after post-filtering (Figure 3-18g and h). This suggests that artifacts in Strategy F are not solely caused by introduced noise but may indicate an underlying algorithmic issue.

In conclusion, Strategy B shows the best results among the all methods, while the Strategy D demonstrates the poorest performance in blind denoising. The presence of artifacts in certain methods, particularly Strategy F, significantly impacts the ability to calculate SNR accurately. These artifacts, unlike those in Strategy A and Strategy C remain after post-filtering, raising concerns about their algorithmic origin and hindering the reliability of Strategy F as a denoising strategy.

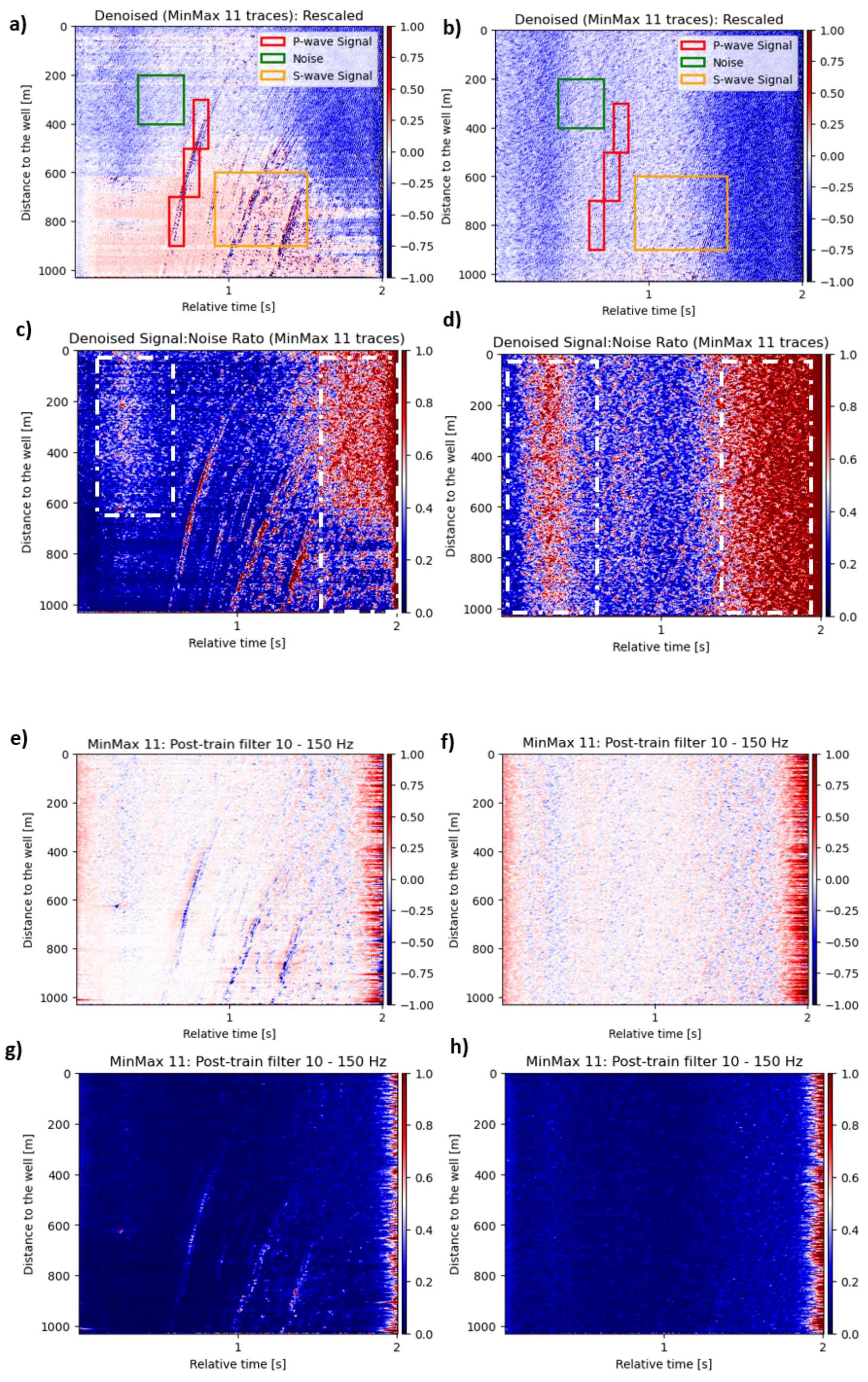


Figure 3-14: Test cases I and II with respect to Strategy A: (a) and (b) are the denoised profile with respect to 3-13c and 3-13d respectively. (c) and (d) are the denoised absolute profile with respect to 3-13e and 3-13f. Note that the white rectangular block with dotted line shows undesirable artifact in denoising. (e) and (f) are the post-filtered profile with range 10 to 150 Hz with respect to (a) and (b). (f) and (g) are the post-filtered absolute profile with respect to (e) and (f).

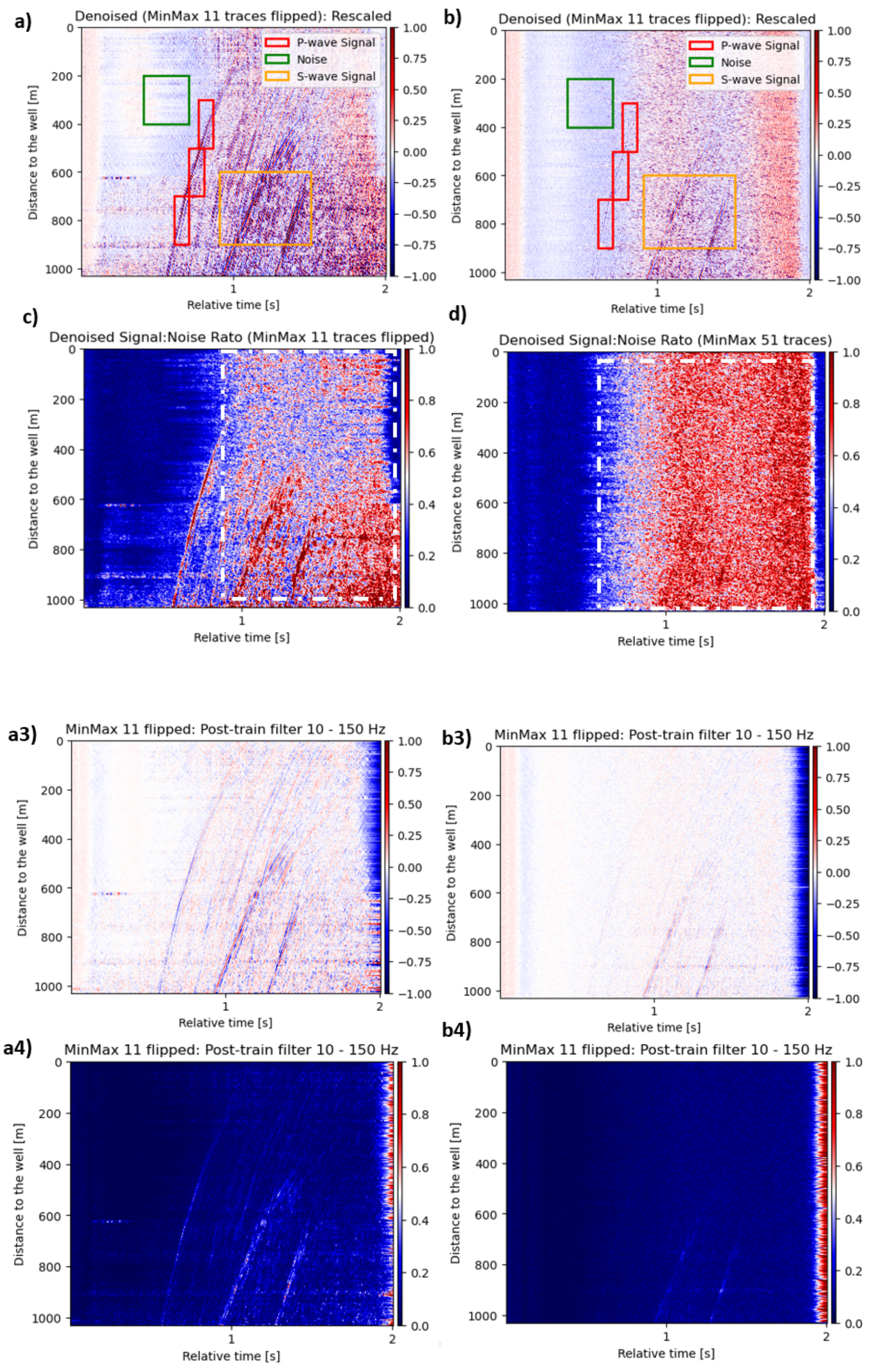


Figure 3-15: Test cases I and II with respect to Strategy B: (a) and (b) are the denoised profile with respect to 3-13c and 3-13d respectively. (c) and (d) are the denoised absolute profile with respect to 3-13e and 3-13f. Note that the white rectangular block with dotted line shows undesirable artifact in denoising. (e) and (f) are the post-filtered profile with range 10 to 150 Hz with respect to (a) and (b). (f) and (g) are the post-filtered absolute profile with respect to (e) and (f).

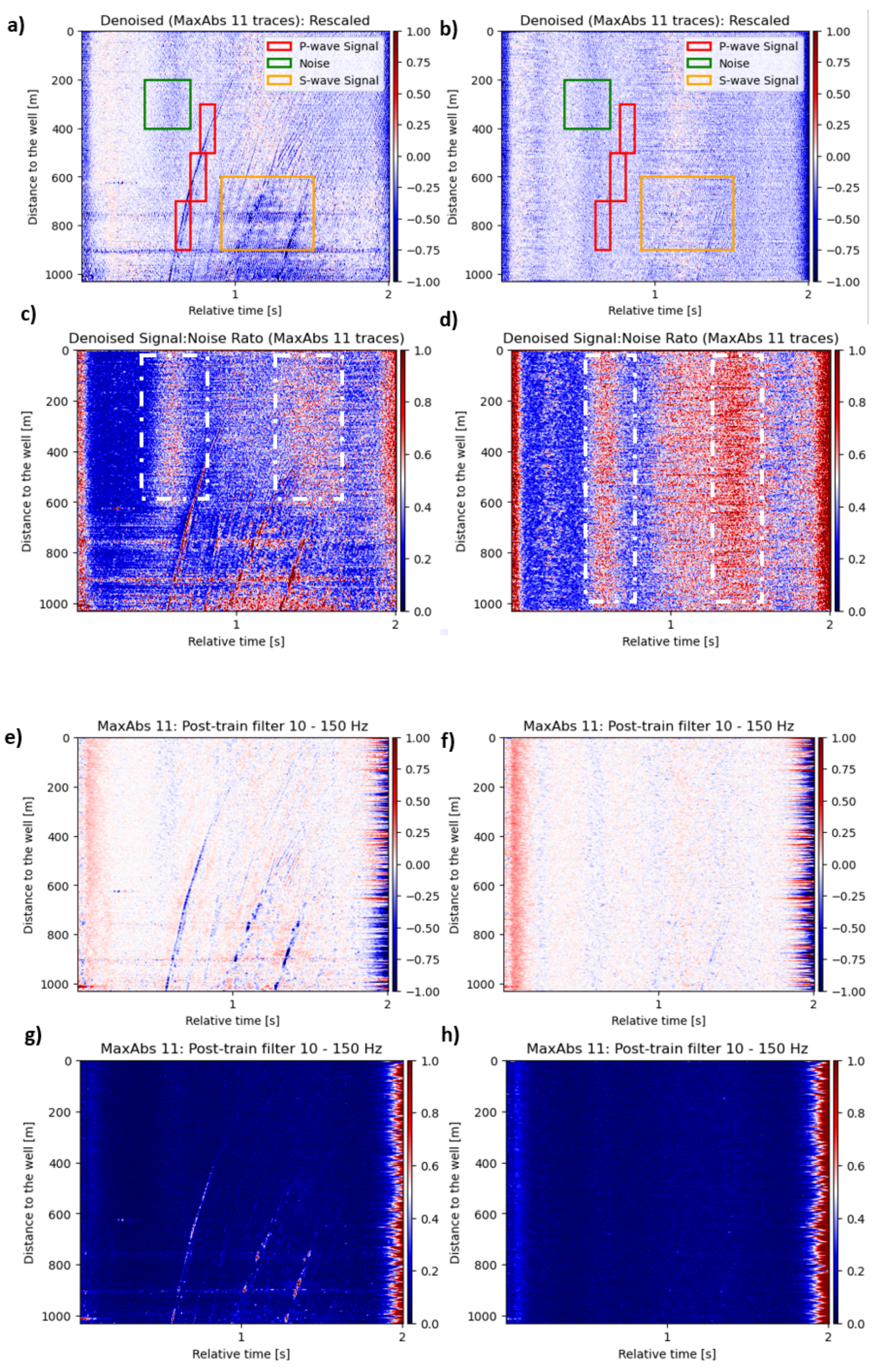


Figure 3-16: Test cases I and II with respect to Strategy C: (a) and (b) are the denoised profile with respect to 3-13c and 3-13d respectively. (c) and (d) are the denoised absolute profile with respect to 3-13e and 3-13f. Note that the white rectangular block with dotted line shows undesirable artifact in denoising. (e) and (f) are the post-filtered profile with range 10 to 150 Hz with respect to (a) and (b). (f) and (g) are the post-filtered absolute profile with respect to (e) and (f).

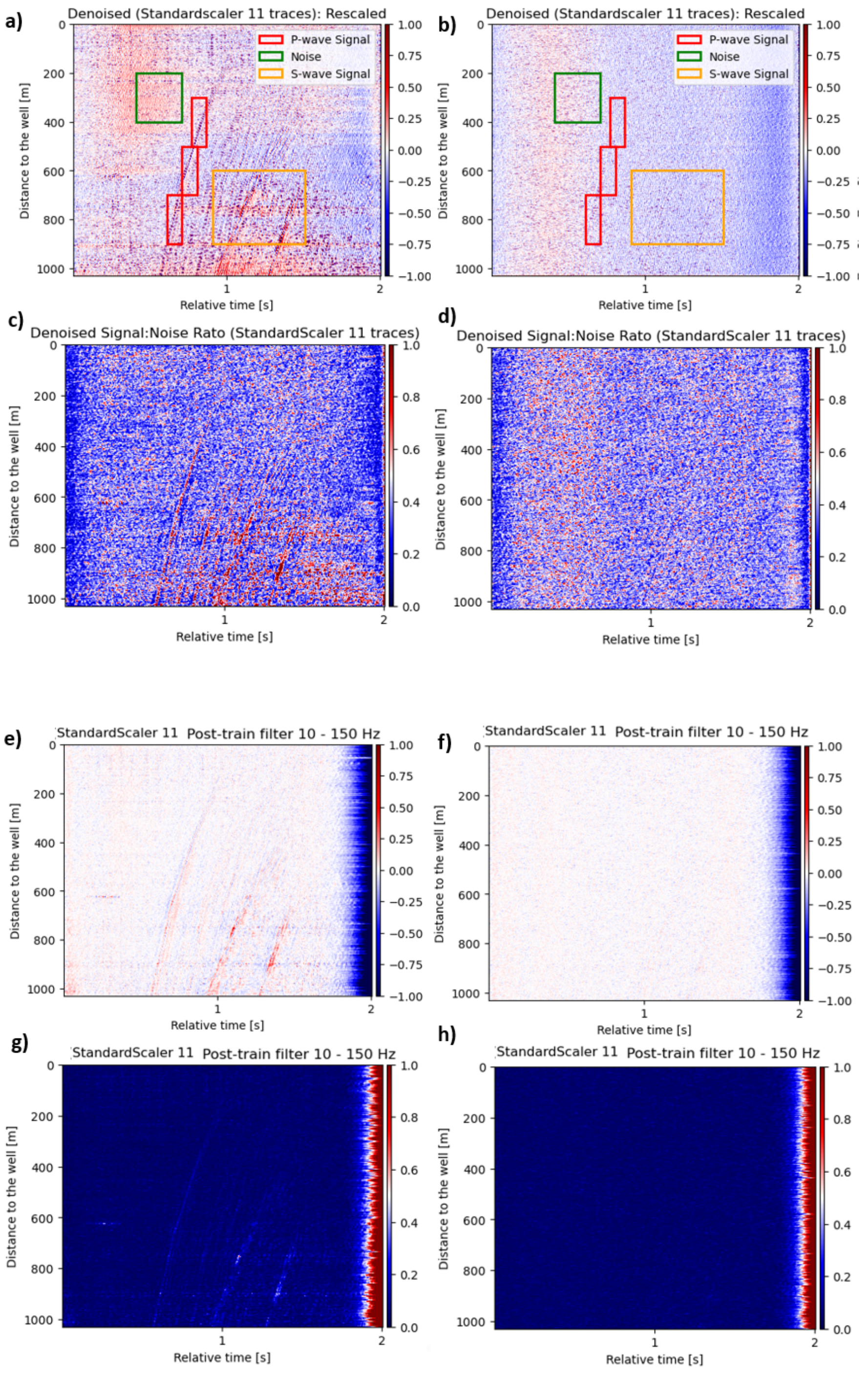


Figure 3-17: Test cases I and II with respect to Strategy D: (a) and (b) are the denoised profile with respect to 3-13c and 3-13d respectively. (c) and (d) are the denoised absolute profile with respect to 3-13e and 3-13f. (e) and (f) are the post-filtered profile with range 10 to 150 Hz with respect to (a) and (b). (f) and (g) are the post-filtered absolute profile with respect to (e) and (f).

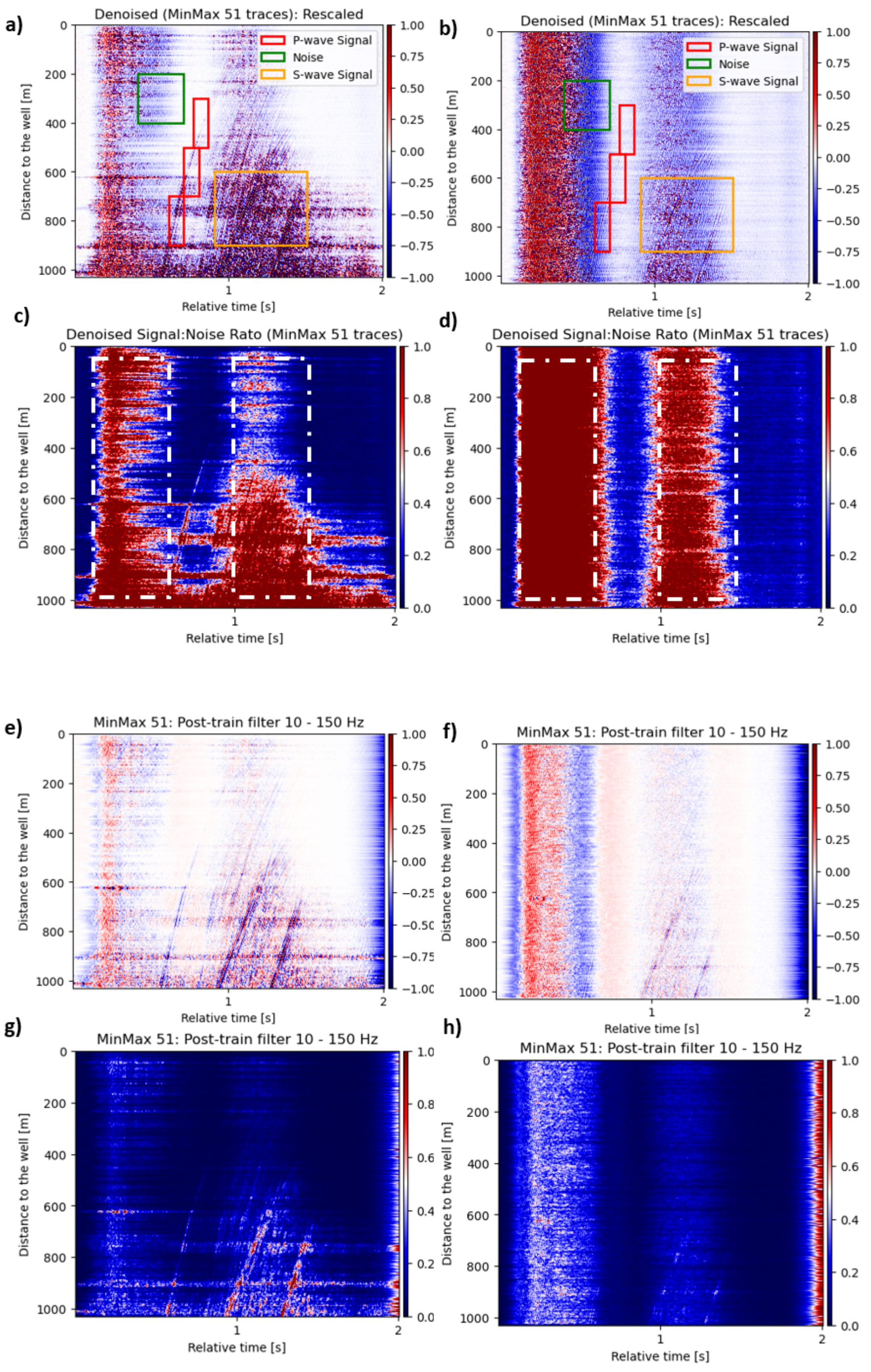


Figure 3-18: Test cases I and II with respect to Strategy F: (a) and (b) are the denoised profile with respect to 3-13c and 3-13d respectively. (c) and (d) are the denoised absolute profile with respect to 3-13e and 3-13f. Note that the white rectangular block with dotted line shows undesirable artifact in denoising. (e) and (f) are the post-filtered profile with range 10 to 150 Hz with respect to (a) and (b). (f) and (g) are the post-filtered absolute profile with respect to (e) and (f).

3-3 Microseismic Detection Analysis

To demonstrate the impact of denoising on microseismic detection, we reprocess a 10-minute FORGE dataset from 16:00:00 to 16:10:00 UTC on April 22, 2022. We compare STA/LTA detections from raw and denoised data, using the HECTOR algorithm for validation.

We present a microseismic detection example using FORGE data from 16:00:09.398 to 16:00:39.398, with a 30-second interval. We set the short-term moving window to 0.5 seconds and the long-term moving window to 10 seconds, following criteria in [Trnkoczy, 2009] for local earthquake detection and seismic ambient noise representation. A limitation of the STA/LTA algorithm is that they cannot perform before the LTA is constructed, therefore the real STA/LTA detection starts after 10 seconds.

According to Porras et al., (2024), a common detection occurs between HECTOR and Silixa (using STA/LTA) [LLC, 2022] at 16:00:34.323 with an SNR ratio of 3.013, while a weak event detected solely by HECTOR appears at 16:00:30.691. We set the event detection threshold at 2.8 (STA amplitude twice LTA amplitude) and the off-threshold value at 1.9 to mark event endings.

Figure 3-19 presents the re-stimulation of both STA/LTA and Hector methods applied to the raw data. This analysis focuses on seven traces (numbered 520, 570, 670, 720, 770, 820, and 870). An event is considered real if detected by at least four of these traces. Both

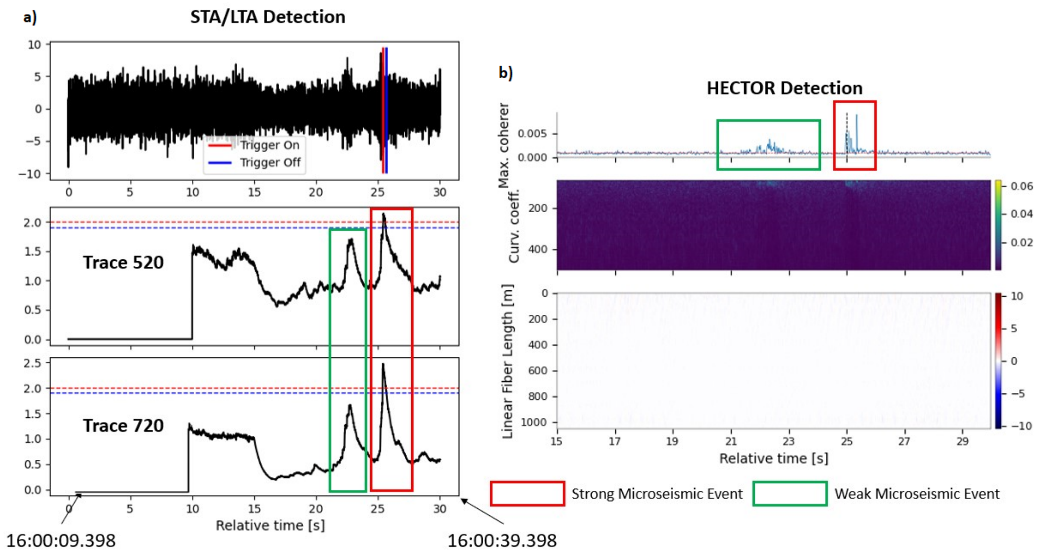


Figure 3-19: Illustration of a microseismic detection. (a) The STA/LTA algorithm which the top panel represents the trace amplitude and the following two panels show the detection from trace 520 and 720 respectively. Detection starts at 10s after the LTA can be constructed. A strong event is detected with the trigger on at 25s, as the amplitude level is stronger than the threshold over 2.0. (b) The detection from HECTOR between relative time 15s to 30s, which detects the same event with a strong coherence matrix. The green box marks the time location for the weak microseismic events, where the HECTOR could able to detect with coherence matrix, while the STA/LTA could only see a small spike but the amplitude level is not strong enough to detect it.

methods detect strong events around a relative time of 25 seconds. STA/LTA achieves a peak amplitude factor of 2.5 at trace 720, while Hector shows a coherence matrix value exceeding 0.005. However, for the weak event between 21 and 23 seconds, Hector exhibits a peak coherence matrix value of only 0.0025, while STA/LTA shows a small amplitude spike, insufficient for event classification.

Microseismic detections are conducted using various denoising strategies outlined in Section 3.2. These strategies generally amplify signal amplitude levels. Consequently, the on-threshold is adjusted to 4.0 and the off-threshold to 3.8 to minimise false positives. Figure 3-20 presents detection results for the Strategy F (left panels) and Strategy B (right panels) strategies. Focusing on the 20s to 25s timeframe, both strategies successfully detect both strong and weak events across different traces. Notably, both approaches effectively remove random noise surrounding weak microseismic signals and enhance their amplitude levels, making them comparable to strong signals. This aligns with observations from Section 3.2.

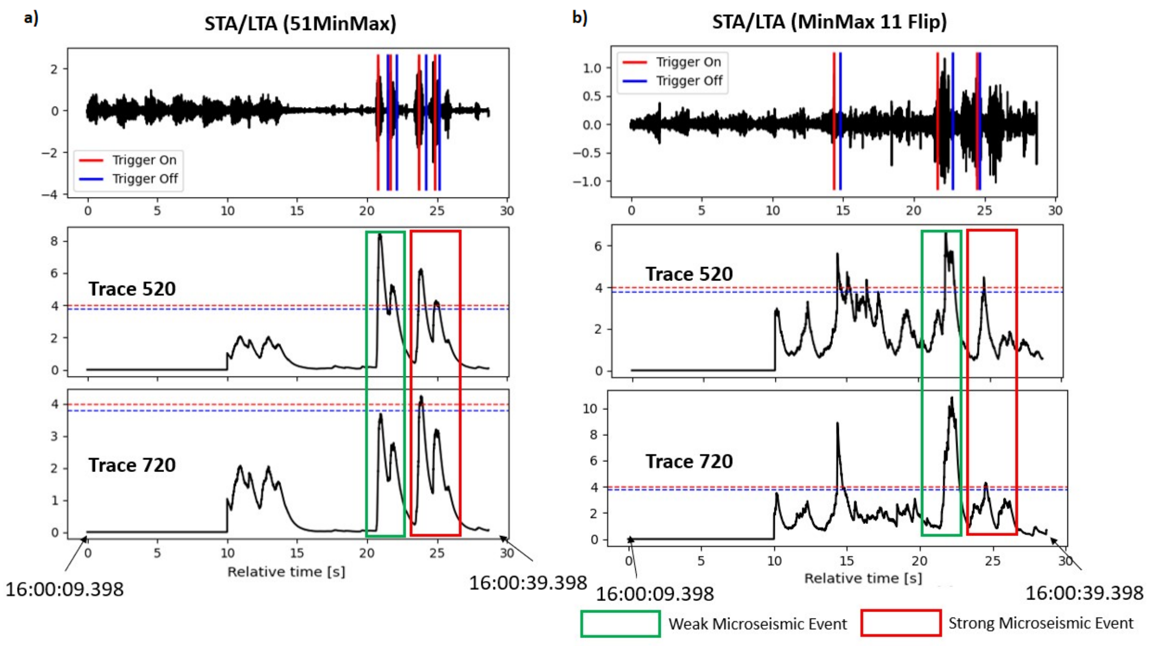


Figure 3-20: Microseismic detections for Strategy F and Strategy B: The green boxes highlight weak microseismic events which can be detected by both methods. The strong microseismic events (red boxes) remain to be detected.

Figure 3-21 demonstrates detections with the remaining strategies. Selecting the trace with the best microseismic detection performance for each method, the Strategy A detects both weak and strong events but introduces significant edge effects when denoising large data chunks. This results in false detections, as shown by the blue region in the Figure 3-21a. The edge effect is even more pronounced with the Strategy C strategy, leading to an artificial increase in both short-term and long-term amplitude levels. Consequently, weak microseismic events become hidden. The Strategy D shows false detections between 16 and 18 seconds while also failing to detect weak events.

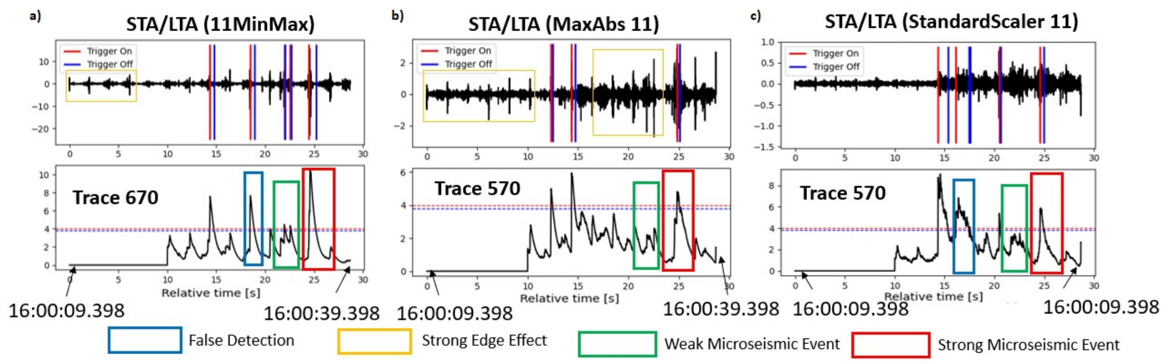
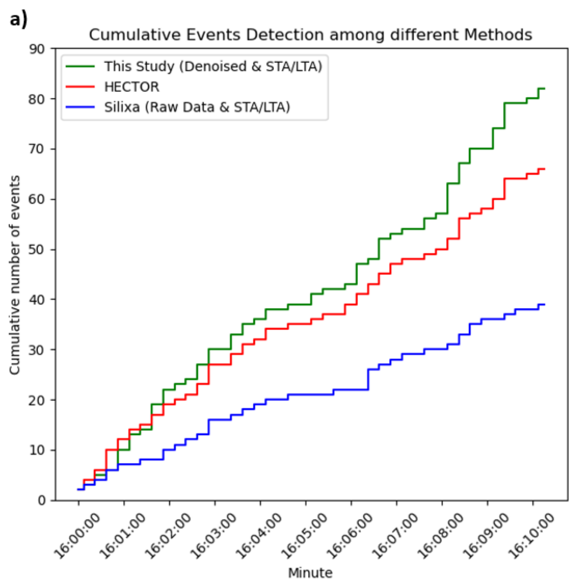
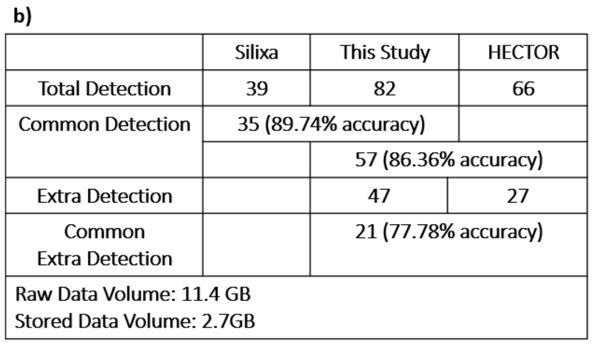


Figure 3-21: Microseismic detections for (a) Strategy A, (b) Strategy C and (c) Strategy D: The green boxes highlight weak microseismic events only detectable by the Strategy A. Yellow boxes in Strategy A and Strategy C indicate edge effects that negatively impact detection. Additionally, blue boxes in Strategy A and Strategy D show areas with false detections. Notably, all strategies successfully detect strong microseismic events (red boxes).

Based on the above results, the Strategy F exhibits a smoother amplitude level curve across different traces when there is no signal. This characteristic indicates a lower likelihood of false detections compared to other strategies. Therefore, it would be the chosen denoised strategy for running the 10 minutes detection stimulation.

Figure 3-22 presents the detection results and a statistical comparison among three studies: Silixa (raw data & STA/LTA), HECTOR, and our method (denoised data & STA/LTA). In general, our study follows a similar trend as HECTOR in microseismic




b)

	Silixa	This Study	HECTOR
Total Detection	39	82	66
Common Detection	35 (89.74% accuracy)		
		57 (86.36% accuracy)	
Extra Detection		47	27
Common Extra Detection		21 (77.78% accuracy)	

Raw Data Volume: 11.4 GB
Stored Data Volume: 2.7GB

Figure 3-22: Microseismic Event Detection Comparison. (a) Cumulative number of detected events within a 10-minute interval using three methods. (b) Performance metrics comparing the three approaches.

detections along the time profile, indicating that the proposed denoising method effectively enhances the detectability of microseismic events. Our method has the highest detection by identifying 82 events compared to the other methods. Compared to the original STA/LTA algorithm, our method yields a 43% increase in the detection of real effective signals. Among them, 35 out of 39 events are commonly detected by STA/LTA on denoised data, with an additional 21 real events verified by the HECTOR method, indicating a 68 % accuracy in real event detections with a significant improvement in weak event detection. Furthermore, our approach offers data reduction. Considering the corresponding 4 seconds stored per triggered event, our method saves 76.32% of data (2.7GB out of 11.4GB) while capturing 86.36% of the effective targets with reference to HECTOR's detection of both strong and weak events.

Chapter 4

Real-Time Denoising Experiment

While seismic data processing packages like pyseistr [Chen et al., 2023] or DAS-N2N [Lapins et al., 2023] offer denoising and interpolation capabilities for multichannel data, their high computational requirements render them unsuitable for real-time applications. In this experiment, we input our denoising algorithm with the Febus DAS interrogator which the fiber optic are linked and implemented inside the building of TNO Utrecht, at Princetonlaan 6, 3584 CB Utrecht. It is a real-time measuring device and we run the workflow on 60 seconds of the data on 7th December, 2023 at time 13:48:02. 167 channels were implemented and the operational frequency would 20kHz, and it would store 6.819 GiB raw data per day.

The denoising algorithm is implemented on a computer equipped with an NVIDIA Quadro K420 graphics card, a 12-core CPU, 64 GB of memory, and 2 GB of dedicated GPU memory. To facilitate real-time processing, the data is downsampled by a factor of 10, reducing the sampling frequency to 2.1 kHz. Data is logged every second. We adopt our baseline model as the algorithm to denoise, emphasising the feasibility of real-time processing, as different strategies can be implemented within the same computational time. Figure 4-1a illustrates the original and denoised profiles of the raw data. To evaluate the algorithm's performance, synthetic DAS signals are embedded into the data at a random time series. Figures 4-1b-f showcase comparisons of raw and denoised data at various time samples.

Figure 4-2 shows the computational and logging results during the 60-second streaming session. Data requests begin at 11.27 seconds, with an average interval of 1.07 seconds between requests and subsequent data transmission. A total of 46 requests are made, with an average processing time of 0.81 seconds for the basic denoising algorithm (without additional functions). From figure 4-2b, none of the 46 requests require processing time more than 1 seconds, and it would reach the minimum of using 77.6 % of the processing time required before the next message queue.

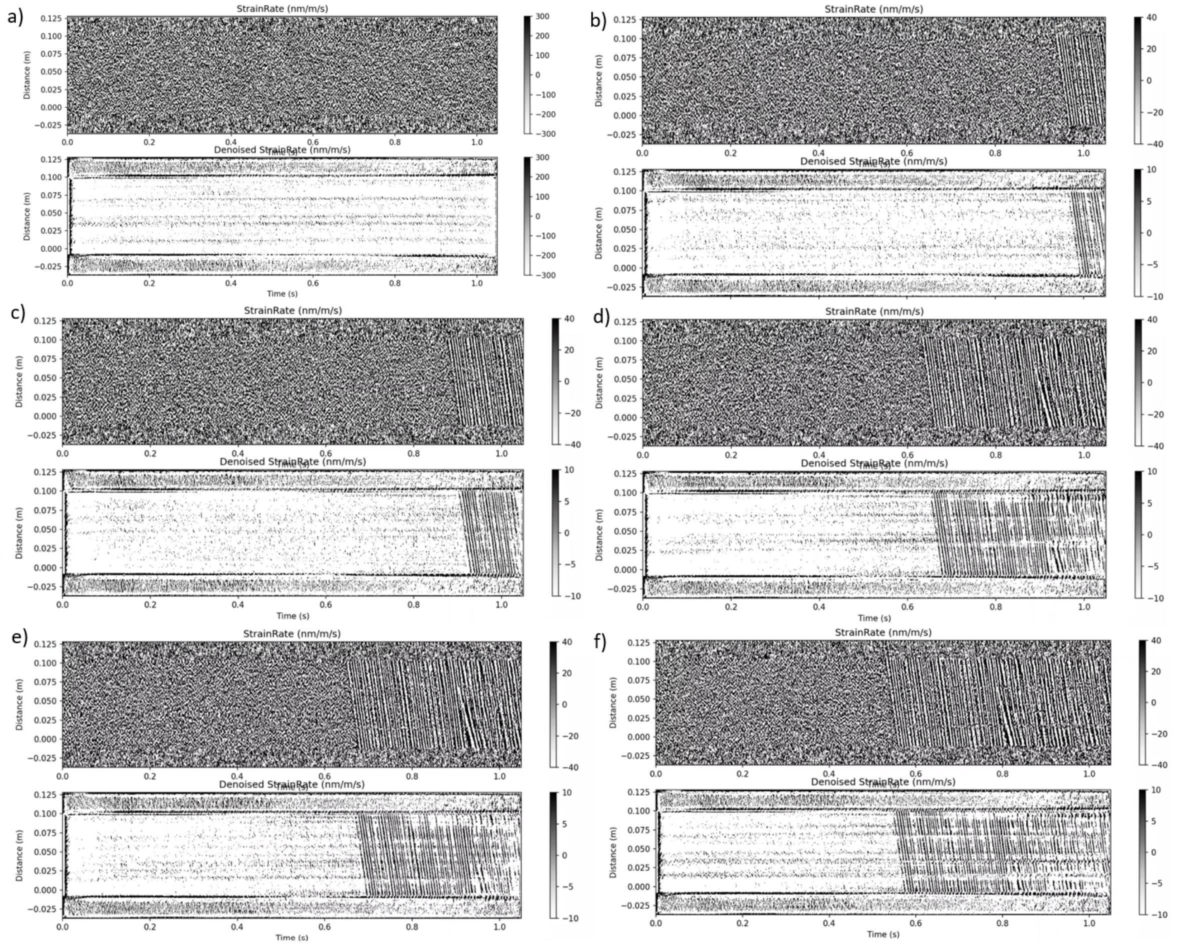
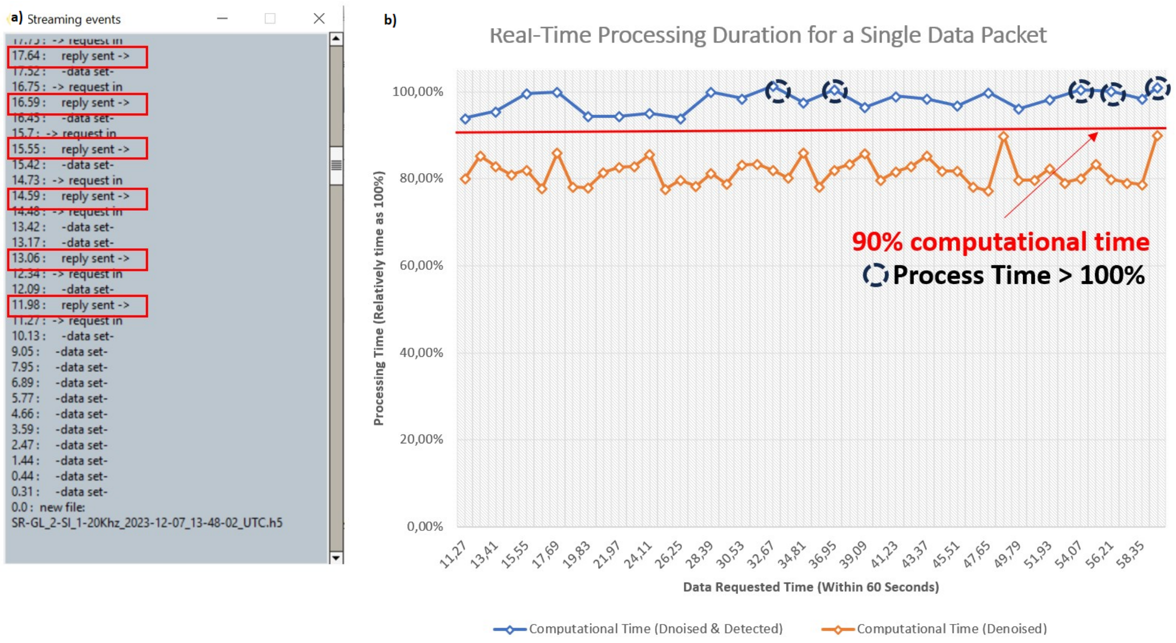


Figure 4-1: Real-Time Denoising Profile: The upper profile shows the raw data collected and the lower profile shows the denoised results. Note that only data logged with relative distance 0m to 0.1m are denoised. (a) shows the profile with raw data while (b) to (f) shows the profile with synthetic microseismic data happening at random time.

Screening Experiment

We evaluate the feasibility of integrating the detection algorithm with real-time denoised data. We achieve this by injecting synthetic data randomly into a real-time data stream. A STA/LTA algorithm is employed based on the denoised data profile. To minimise computational demands, the STA/LTA is implemented on a single trace, resulting in an additional processing time of only 0.05 seconds per data point on average. However, additional considerations impact real-time performance. Random synthetic data generation and file saving introduce an additional 0.7 seconds of processing time, exceeding the real-time limit in certain scenarios. The experiment processes 24 data patches at a time interval of 2 seconds. Twelve synthetic data points are randomly added to the dataset. We configure the STA window as a factor of 20 compared to the LTA window, with an on-threshold of 1.9 and an off-threshold of 1.0.

Table 4-1 presents the performance of pre-screening using the detection profile without denoising. As expected, all data is saved, offering no improvement in data storage reduction. With the denoised algorithm applied (Table 4-2), we detect and save 11 out of 12 events. While one true event profile is missed, and two false detections occur, the overall accuracy reaches 87.5% (calculated by summing true positives and true negatives). Additionally, the denoising algorithm achieves a 75% reduction in unnecessary data, significantly improving storage efficiency. However, as illustrated in Figure 4-2, four out of 24 events slightly exceed the target processing time of 100%. The maximum observed processing time is 101.3%. This time constraint may result in some data being missed during the message cue send and receive process.



```
17.73 : -> request in
17.64 :    reply sent ->
17.52 :    -data set-
16.75 : -> request in
16.59 :    reply sent ->
16.45 :    -data set-
15.7 : -> request in
15.55 :    reply sent ->
15.42 :    -data set-
14.73 : -> request in
14.59 :    reply sent ->
14.48 : -> request in
13.42 :    -data set-
13.17 :    -data set-
13.06 :    reply sent ->
12.34 : -> request in
12.09 :    -data set-
11.98 :    reply sent ->
11.27 : -> request in
10.13 :    -data set-
9.05 :   -data set-
7.95 :   -data set-
6.89 :   -data set-
5.77 :   -data set-
4.66 :   -data set-
3.59 :   -data set-
2.47 :   -data set-
1.44 :   -data set-
0.44 :   -data set-
0.31 :   -data set-
0.0 :  new file:
SR-GL_2-SI_1-20Khz_2023-12-07_13-48-02_UTC.h5
```

Figure 4-2: (a)Steam Data Logging Profile: the message queue was requested at time 11.27 seconds and was sent in 11.98 seconds. The red block indicates the message sent in time for each request and the time interval would be between 1s to 2s. (b) Single Data Packet processing Time: the processing is plotted with the relative processing time at each requests. The computational time with denoising only is plotted in orange while denoising and detection is plotted in blue. Note that none of the request for doing denoising exceed 90% of the processing time.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive (12)	False Negative (0)
	Negative	False Positive (12)	True Negative (0)

Table 4-1: Confusion Matrix for the STA/LTA running in the raw data profile

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive (11)	False Negative (1)
	Negative	False Positive (2)	True Negative (10)

Table 4-2: Confusion Matrix for the STA/LTA running in the denoised profile

Chapter 5

Discussion and Outlook

5-1 Model Comparison

This study examines three primary scaling methods: MinMaxScaler, MaxAbsScaler, and StandardScaler. MinMaxScaler achieves the lowest data loss and delivers the highest SNR, particularly for strong microseismic events. Combined with "flipping" training data, it performs well in capturing sharp first arrivals, crucial for accurate signal start time determination in microseismic detection, as illustrated in Figures 3-7a and 3-14a. MaxAbsScaler effectively scales data points near background noise to zero, benefiting weak microseismic events. However, it suffers from information loss due to its focus on data magnitude, hindering noise-signal distinction, especially in unsupervised learning. Figure 3-16a shows all remaining data preserved in single polarity, complicating subsequent frequency analysis. Single trace analysis in Figure 5-2 reveals a significant bias towards negative values with MaxAbsScaler, particularly in trace 540. StandardScaler offers a gentler scaling approach, effectively eliminating outliers under the assumption of Gaussian data distribution. Yet, it retains significant noise in both strong and weak event scenarios, resulting in minimal SNR improvement and near-complete loss of coherent signal in weak events.

Moreover, scaling approaches significantly influence activation layer performance. Our study observes early stopping during training with StandardScaler, as shown in Figure 3-4. This suggests a potential issue with negative values in the data and the activation function used. In U-Nets, training with data containing negative values can lead to the "dying ReLU" problem, which occurs because ReLU neurons can learn a large negative bias, causing them to become permanently inactive and always output zero for any input [Lu et al., 2020]. While our research observes similar behaviour with Swish, Tanh, and ReLU activation functions, we recommend excluding negative values entirely before feeding the data into the U-Net.

An interesting counterintuitive observation emerges regarding the relationship between loss and denoising effectiveness. Strategy B - "MinMax 11 traces with flipping" achieves the best denoising results despite exhibiting a higher training and validation loss compared to

Strategy A - "MinMax 11 traces" and Strategy C - "MaxAbs 11 traces." This phenomenon can be attributed to the nature of our unsupervised learning approach. Unlike supervised learning where synthetic data with ground truth is used, we employ real data with inherent noise, despite at an extremely smaller scale compared to the strong target signal introduced for testing. In this context, a loss function approaching zero would theoretically imply perfect reconstruction of the input traces, including the existing noise. Therefore, it is important to minimise the loss for reconstructing similar traces while a relatively small loss would imply an overfitting towards the model, results in a less accurate denoising algorithm. The "flipping" step in Strategy B introduces a randomisation element. This prevents the data from exhibiting a consistent pattern (e.g., P-wave, S-wave, coda waves) and promotes model flexibility in capturing the long-term coherent value within the signal. As a result, flipping helps the model generalise better and avoid overfitting to specific noise patterns.

Similar behaviour is found in the experiments of using different input patch sizes (11, 31, and 51 traces). While larger patches (31 and 51 traces) achieve lower loss values (as shown in Figure 3-5), they exhibit limitations in denoising performance. Specifically, these methods yield less sharp P-wave arrivals before channel 400 and retained noise, particularly in dense signal areas (below channel 700, before and after the target signal) as seen in Figures 3-11b and 3-12b. We hypothesise that smaller input patches (11 traces) capture a more focused local trend compared to the global features captured by larger patches. This focus benefits the denoising of weak microseismic events and the top channels, where the SNR is generally lower compared to the bottom channels. Given our primary objective of achieving accurate pre-screening results (identifying as many weak microseismic events as possible to prevent data loss), prioritising 11 input traces offers a clear advantage for this specific application.

Our findings align with observations from [van den Ende et al., 2023b], who reported the creation of unexplained artifacts during denoising of data with strong noise (similar to Figures 3-14 to 3-16c and d. They proposed introducing Gaussian random noise at 1-10 Hz followed by a post-filtering step with a low-cut frequency of 10 Hz. In our experiment, we adopt a similar strategy, but with Gaussian random noise between 10-150 Hz and a post-filtering range of 10-150 Hz. However, this approach failed to fully eliminate the artifacts in the case of Strategy F, even leading to a decrease in the true SNR.

The detection performance is influenced by the denoising algorithm's characteristics. For instance, Strategy F excels in deeper channels but struggles with shallower depths, while Strategy B achieves high SNR but requires a higher detection threshold to mitigate false alarms. Strategies A and C which adopt 11 input traces, effectively enhance first arrivals at shallower depths but it would suffer from surface-correlated noise-induced false detections as shown in Figure 5-4. We need to balance the use of multiple traces at varying depths so as to optimising detection accuracy, and also prioritising channels where the denoising algorithm performs optimally.

5-2 Real-Time Detection Feasibility

Section 4 presents an experiment exploring real-time denoising capabilities. With the equipment setup it could perform blind denoising and detection in the real-time, yet this approach introduces additional processing time for data saving before the arrival of the next data patch. Furthermore, data size constraints impose a maximum sampling frequency of 2100 Hz. Capturing higher resolution data per second would require creating additional data chunks and doubling the denoising processing time. It's important to note that these limitations are specific to the current hardware configuration, and performance might vary depending on programming language, which in our scenarios is python.

Our real-time detection algorithm utilises a single-trace STA/LTA implementation. This is a simpler approach compared to the 7-trace detection employed in Section 3.3. While the maximum trace analysis for this equipment setup might be 3 traces, this would significantly increase processing time, leading to a more general scenarios in exceeding real-time constraints and leading to missed data. Notably, even with the single-trace approach, the algorithm achieves a remarkable accuracy of 87.5%. However, increasing the number of traces analysed by the STA/LTA algorithm has the potential to reduce false positives caused by instrument faults or poor channel coupling. Figure 5-1 exemplifies this concept. The red line shows a false positive generated by the single-trace STA/LTA implementation. This occurs because the issue is a localised phenomenon. On the other hand, the blue region exhibits less correlated noise across channels which if we take more account of traces in detection the events can be detected.

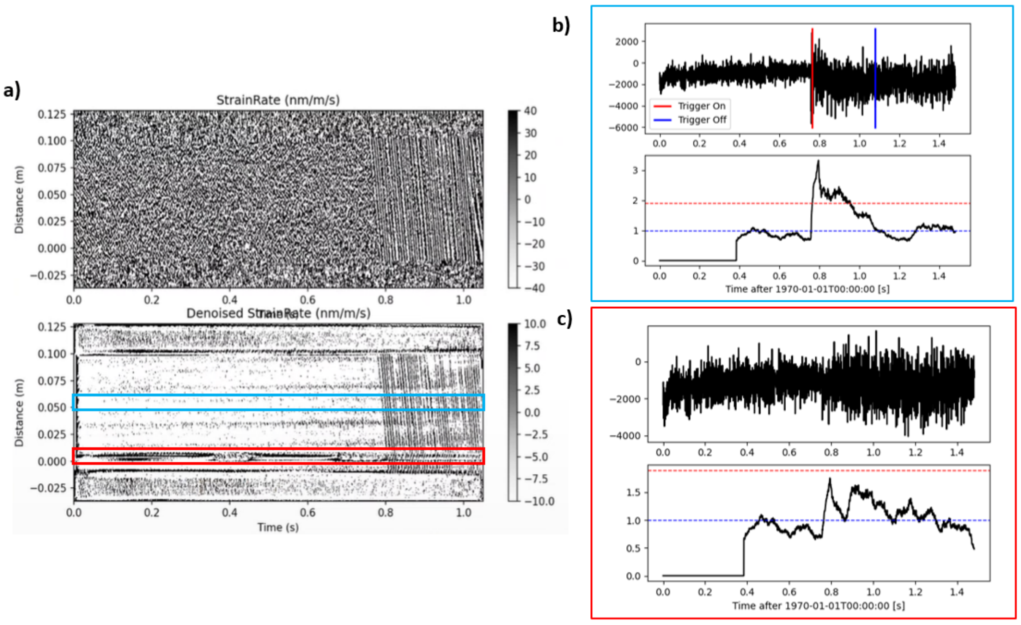


Figure 5-1: Example of a false positive in real-time experiment: (a) demonstrates a scenario with real-time denoising, and the trace location where STA/LTA is applied in 2 different case. (b) demonstrates the event can be detected if traces are selected in the blue region. (c) demonstrates the event cannot be detected if traces are selected in the red region.

To potentially improve accuracy, one strategy could involve lowering the on-threshold requirement. While this might increase false positives, it aligns with the screening purpose of capturing as many potential microseismic events as possible to avoid missing critical data. In this context, false alarms are acceptable as long as the algorithm avoids missing true positives and offers a significant advantage over storing all raw data.

5-3 Limitations

Data Chunk Constraints

One harsh condition for using U-Net with blind denoising is the size of the data sample. It could only process when the patch of the data is sufficient with 2048 time sample points, and it would also process with the multiple of 2048. This limitation leads to edge effects when the total data length exceeds this threshold, as shown in Figure 5-2. These effects arise because the network reconstructs boundaries based on interpolations from adjacent windows, resulting in data discontinuities. Depends on the chosen algorithm, edge effects

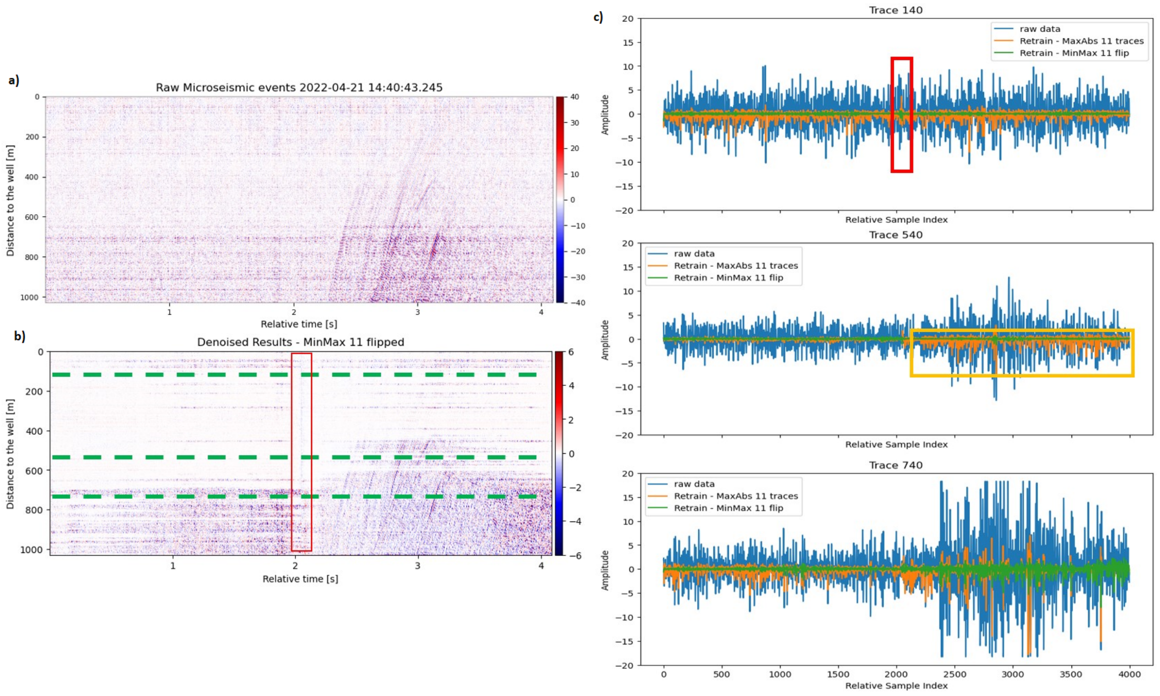


Figure 5-2: Real-Time Denoising Profile and Single Trace Analysis: (a) The raw microseismic events. (b)The denoised profile when the input file is more than 2048 time samples, and an edge effect would be created at the boundary of each denoising windows (i,e, multiple of 2048) as shown in the red block. (c) The single trace comparison among the raw data, Strategy B and Strategy C at trace 140, 540 and 740. As shown the amplitude is distorted among all denoised profile, and the edge effect as circled in red is more obvious among weaker traces (a spike in the orange profile). For MaxAbs scaling, it tends to reconstruct the amplitude in single polarity as circle in orange block in trace 540, which hinders the extraction of phase information afterwards

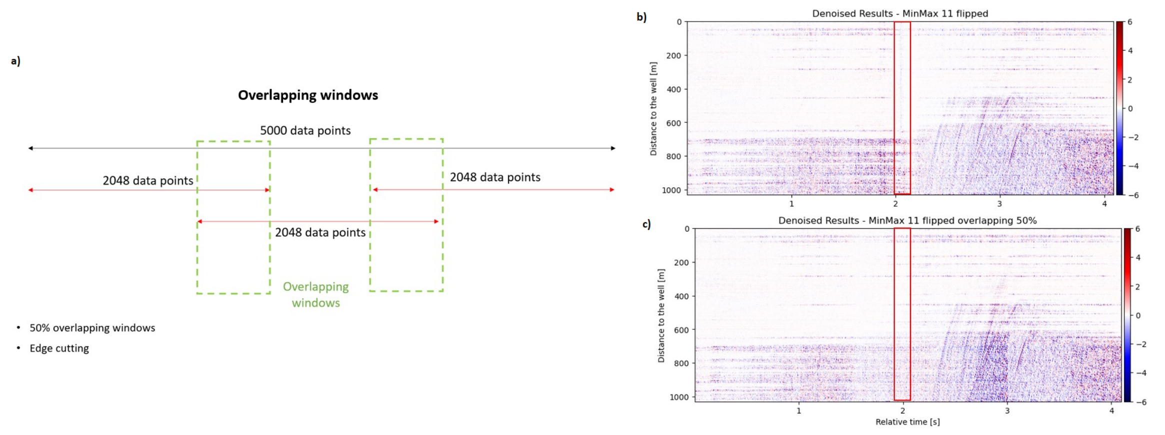


Figure 5-3: (a)Conceptual ideas of overlapping examples and the data demonstration: We would expand our data with an 50% overlapping windows before computing denoising, and then recombine the data without selecting the edge from each data drunk. (b)The denoised profile. (c) The denoised profile with overlapping windows. The edge effect would be gone as shown in the red block.

are more significant in methods like Strategy A and Strategy C, while it was less when we consider flipping and increase the side of the input traces. Yet. the edge effects can lead to significant false detection rates, particularly when spurious spikes appear across multiple individual traces.

One possible approach would be constructing overlapping windows during data pre-processing, which the concepts and the examples are shown in Figure 5-3. This method expands the data size and excludes edge points from the final reconstruction process, ensuring the original data length is preserved. In our experiment, we implemented a 50% overlap between windows, effectively eliminating edge effects. Yet, for n number of data patch, it would create an additional of (n-1) set of data which significantly increase the processing time for the data. Fortunately, edge effects are a predictable consequence of the U-Net architecture and data processing limitations, which in detection algorithm we can exclude the data that is sampled at the factor of 2048 manually, avoiding the creation of false detection.

Amplitude Distortion

Similarly finding as [van den Ende et al., 2023b], challenges remain in estimating the true signal amplitudes accurately, especially if the noise and the signal are at similar level. As shown in figure 5-2, in trace 140 where it contains mainly noise and in trace 740 where noise are significantly before the first arrival, especially with Strategy B, the amplitude of signal are diminished compared with the raw data. As in our experiment, this distortion appeared to be more pronounced in cases with high initial SNR (raw data) compared to low SNR scenarios.

Correlated Noise Mitigation

The underlying principle of the blind denoising algorithm relies heavily on spatial-temporal signal coherence, which is very useful for removing incoherent noise while coherent noise like ocean gravity waves will be remained and these require separate denoising techniques. Figure 5-4 illustrates a false detection resulting from surface-correlated noise, during the microseismic event detection experiment outlined in Section 3.3. Despite denoising efforts, the coherent nature of this noise prevents its complete removal.

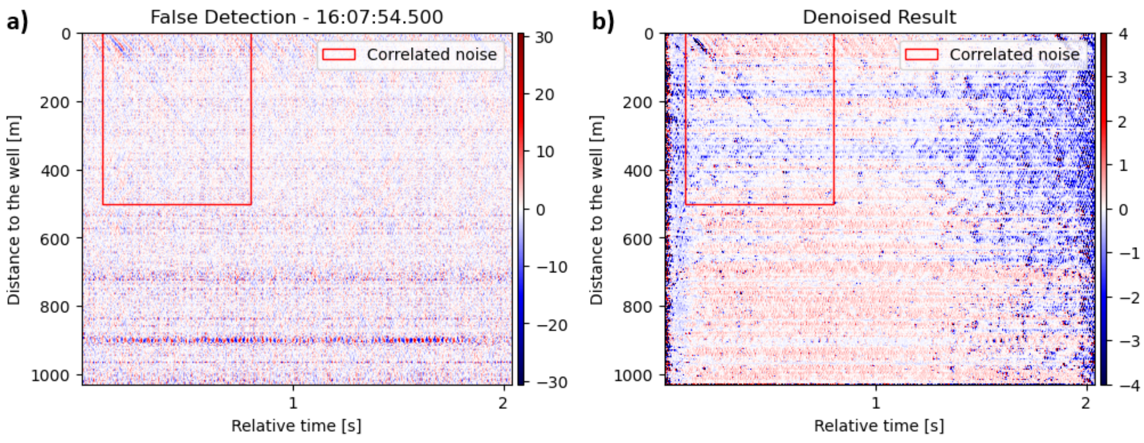


Figure 5-4: (a) False detection caused by surface-correlated noise (marked in red) in the microseismic event detection experiment (Section 3.3). (b) Denoised profile with surface-correlated noise remained.

5-4 Future Outlook

Machine learning algorithms for Distributed Acoustic Sensing (DAS) applications often exhibit dataset and setup specificity. For example, jDAS [van den Ende et al., 2023a] a is trained on earthquake events, making it unsuitable for microseismic monitoring. Likewise, DAS-N2N [Lapins et al., 2023] requires specific data logging conditions, limiting its applicability to single-direction data collection. Similarly, our algorithm demonstrates effectiveness with microseismic events acquired using our specific equipment setup and data logging approach. However, it would require retraining for other applications, such as large-magnitude earthquakes or Distributed Temperature Sensing (DTS) data analysis. The development of universally applicable blind denoising algorithms which is capable of handling diverse datasets would be a desirable goal with less overfitting with specific dataset.

More advanced denoising algorithms exist, such as PySeis [Chen et al., 2023]. This package offers structural denoising and interpolation of multi-channel seismic data, and it is effective in removing both random and correlated noise. Yet with the current computational power and time it could not be done in real time. Future advancements in computing power, including the exponential speedup promised by quantum computing [Liu, 2021], could enable real-time utilisation of these advanced denoising methods.

While our algorithm primarily focuses on data screening, its applications could be extended to include a ”traffic light” system for deep geothermal well stimulation. This system would automatically shut down operations when induced seismicity reaches predefined thresholds [Ader et al., 2020]. Additionally, the algorithm could be used in fluid pressure monitoring and carbon injection for Carbon Capture and Storage (CCS) projects [Grab et al., 2022].

Chapter 6

Conclusion

This study investigates the application of unsupervised learning for blind denoising of DAS data based on the concept of J-invariance. Unsupervised learning offers the advantage of bypassing the need for prior noise knowledge and noise-free ground truth data. We incorporated J-invariance principles into a U-Net CNN architecture by inputting a series of seismic traces, masking one, and reconstructing it based on neighbouring traces.

Utilising field DAS data (i.e, FORGE data) containing strong microseismic events for model training, we successfully captured long-term signal coherence and boosted SNR for both strong and weak microseismic events. Hyperparameter analysis revealed that Min-Max scaling generally outperforms Max-Abs scaling and standardisation in terms of SNR, while preserving amplitude polarity. The number of input traces influenced feature extraction, with fewer traces capturing local details effectively, particularly for the P-wave arrival in shallower depths, and more traces capturing global trends and waveform characteristics in deeper channels. Time flipping, although increasing training loss and validation loss, enhanced data diversity and mitigated overfitting.

We evaluated our denoised data by implementing a microseismic event detection algorithm. A data screening process triggered additional detections, resulting in a 43% increase in identified real microseismic events while maintaining an overall detection accuracy of 68%. By implementing a data separation and saving function triggered by event detection, we achieved a 75% reduction in data volume, effectively addressing the storage challenges associated with continuous monitoring campaigns.

We further implemented the proposed method in a real-time demonstration. Despite a data logging rate of one second, the denoising function was integrated into the workflow, operating at a maximum sampling rate of 21 kHz without exceeding 90% of the processing time before the next data input. While denoising and detection occasionally pushed processing time to 101.3%, the rapid advancements in GPU and quantum computing technologies offer promising potential to the exponentially accelerate computational speeds, enabling higher real-time processing stability.

Bibliography

[Ader et al., 2020] Ader, T., Chendorain, M., Free, M., Saarno, T., Heikkinen, P., Malin, P., Leary, P., Kwiatek, G., Dresen, G., Blümle, F., and Vuorinen, T. (2020). Design and implementation of a traffic light system for deep geothermal well stimulation in finland. *Journal of Seismology*, 24.

[Anikiev et al., 2023] Anikiev, D., Birnie, C., bin Waheed, U., Alkhalifah, T., Gu, C., Verschuur, D. J., and Eisner, L. (2023). Machine learning in microseismic monitoring. *Earth-Science Reviews*, 239:104371.

[Batson and Royer, 2019] Batson, J. and Royer, L. (2019). Noise2self: Blind denoising by self-supervision. *CoRR*, abs/1901.11365.

[Binder and Tura, 2020] Binder, G. and Tura, A. (2020). Convolutional neural networks for automated microseismic detection in downhole distributed acoustic sensing data and comparison to a surface geophone array. *Geophysical Prospecting*, 68(9):2770–2782.

[Chen et al., 2018] Chen, J., Chen, J., Chao, H., and Yang, M. (2018). Image blind denoising with generative adversarial network based noise modeling. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3155–3164.

[Chen et al., 2023] Chen, Y., Savvaidis, A., Fomel, S., Chen, Y., Saad, O. M., Oboué, Y. A. S. I., Zhang, Q., and Chen, W. (2023). Pyseistr: A Python Package for Structural Denoising and Interpolation of Multichannel Seismic Data. *Seismological Research Letters*, 94(3):1703–1714.

[Correa et al., 2017] Correa, J., Egorov, A., Tertyshnikov, K., Bona, A., Pevzner, R., Dean, T., Freifeld, B., and Marshall, S. (2017). Analysis of signal to noise and directivity characteristics of DAS VSP at near and far offsets ??A CO2CRC Otway Project data example. *The Leading Edge*, 36(12):994a1–994a7.

[Dou S, 2017] Dou S, Lindsey N, W. A. D. T. F. B. R. M. P. J. U. C. M. E. A.-F. J. (2017). Distributed acoustic sensing for seismic monitoring of the near surface: A traffic-noise interferometry case study. *Sci Rep. 2017 Sep 14;7(1):11620. doi: 10.1038/s41598-017-11986-4. PMID: 28912436; PMCID: PMC5599533.*

[Durall et al., 2024] Durall, R., Ghanim, A., Ettrich, N., and Keuper, J. (2024). An in-depth study of u-net for seismic data conditioning: Multiple removal by moveout discrimination. *GEOPHYSICS*, 89(1):WA233–WA246.

[Foulger et al., 2018] Foulger, G. R., Wilson, M. P., Gluyas, J. G., Julian, B. R., and Davies, R. J. (2018). Global review of human-induced earthquakes. *Earth-Science Reviews*, 178:438 – 514. Cited by: 340; All Open Access, Hybrid Gold Open Access.

[Gholamalinezhad and Khosravi, 2020] Gholamalinezhad, H. and Khosravi, H. (2020). Pooling methods in deep neural networks, a review.

[Grab et al., 2022] Grab, M., Rinaldi, A. P., Wenning, Q. C., Hellmann, S., Roques, C., Obermann, A. C., Maurer, H., Giardini, D., Wiemer, S., Nussbaum, C., and Zappone, A. (2022). Fluid pressure monitoring during hydraulic testing in faulted opalinus clay using seismic velocity observations. *GEOPHYSICS*, 87(4):B287–B297.

[Hudson et al., 2021] Hudson, T. S., Baird, A. F., Kendall, J. M., Kufner, S. K., Brisbourne, A. M., Smith, A. M., Butcher, A., Chalari, A., and Clarke, A. (2021). Distributed acoustic sensing (das) for natural microseismicity studies: A case study from antarctica. *Journal of Geophysical Research: Solid Earth*, 126(7):e2020JB021493. e2020JB021493 2020JB021493.

[Ilesanmi, 2021] Ilesanmi, A.E., I. T. (2021). Methods for image denoising using convolutional neural network: a review. *Complex Intell. Syst*, 7(2179-2198):17–23.

[Iqbal et al., 2018] Iqbal, N., Liu, E., McClellan, J., Al-Shuhail, A., Kaka, S., and Zerguine, A. (2018). Detection and denoising of microseismic events using time–frequency representation and tensor decomposition. *IEEE Access*, PP:1–1.

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[Kuijpers, 2020] Kuijpers, D. (2020). Seismic data reconstruction using deep learning. *Utrecht University*.

[Lapins et al., 2023] Lapins, S., Butcher, A., Kendall, J.-M., Hudson, T. S., Stork, A. L., Werner, M. J., Gunning, J., and Brisbourne, A. M. (2023). DAS-N2N: machine learning distributed acoustic sensing (DAS) signal denoising without clean data. *Geophysical Journal International*, 236(2):1026–1041.

[Lellouch et al., 2021] Lellouch, A., Schultz, R., Lindsey, N., Biondi, B., and Ellsworth, W. (2021). Low-magnitude seismicity with a downhole distributed acoustic sensing array—examples from the forge geothermal experiment. *Journal of Geophysical Research: Solid Earth*, 126(1):e2020JB020462. e2020JB020462 2020JB020462.

[Li et al., 2022] Li, X., Wu, B., Zhu, X., and Yang, H. (2022). Consecutively missing seismic data interpolation based on coordinate attention unet. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5.

[Li et al., 2021] Li, Y., Karrenbach, M., and Ajo-Franklin, J. B. (2021). *A Literature Review*, chapter 17, pages 229–291. American Geophysical Union (AGU).

[Lindsey et al., 2020] Lindsey, N. J., Rademacher, H., and Ajo-Franklin, J. B. (2020). On the broadband instrument response of fiber-optic das arrays. *Journal of Geophysical Research: Solid Earth*, 125(2):e2019JB018145. e2019JB018145 10.1029/2019JB018145.

[Liu, 2021] Liu, Y., A. S. . T. K. (2021). A rigorous and robust quantum speed-up in supervised machine learning. *Nat. Phys. 17, 1013–1017.*

[LLC, 2022] LLC, S. (2022). Utah forge: Well 16a(78)-32 2022 stimulation microseismic report.

[Lu et al., 2020] Lu, L., Shin, Y., Su, Y., and Karniadakis, G. (2020). Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28:1671–1706.

[Ma, 2024] Ma, L. (2024). Noise classification and comparison of denoising methods for distributed acoustic sensing data (das). *RWTH Aachen: Research Module in Applied Geophysics.*

[Martin, 2022] Martin, Taylor, N. G. (2022). Utah forge: High-resolution das microseismic data from well 78-32. *FORGE*, 1(1).

[Mohan et al., 2020] Mohan, S., Kadkhodaie, Z., Simoncelli, E. P., and Fernandez-Granda, C. (2020). Robust and interpretable blind image denoising via bias-free convolutional neural networks.

[Moore et al., 2023] Moore, J., McLennan, J., Pankow, K., Finnila, A. B., Dyer, B., Karvounis, D., Bethmann, F., Podgorney, R., Rutledge, J., Meir, P., Xing, P., Jones, C., Barker, B., Simmons, S. D., and Damjanac, B. (2023). Current activities at the utah frontier observatory for research in geothermal energy (forge): A laboratory for characterizing, creating and sustaining enhanced geothermal systems. *All Days.*

[Porras et al., 2024] Porras, J., Pecci, D., Bocchini, G. M., Gaviano, S., De Solda, M., Tuinstra, K., Lanza, F., Tognarelli, A., Stucchi, E., and Grigoli, F. (2024). A semblance-based microseismic event detector for DAS data. *Geophysical Journal International*, 236(3):1716–1727.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

[Szandala, 2021] Szandala, T. (2021). *Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks.* Springer Singapore.

[Tian et al., 2019] Tian, C., Xu, Y., Fei, L., Wang, J., Wen, J., and Luo, N. (2019). Enhanced cnn for image denoising. *CAAI Transactions on Intelligence Technology*, 4(1):17–23.

[Trnkoczy, 2009] Trnkoczy, A. (2009). Understanding and parameter setting of sta/lta trigger algorithm. */.*

[Vaezi and Van der Baan, 2015] Vaezi, Y. and Van der Baan, M. (2015). Comparison of the STA/LTA and power spectral density methods for microseismic event detection. *Geophysical Journal International*, 203(3):1896–1908.

[van den Ende et al., 2023a] van den Ende, M., Ferrari, A., Sladen, A., and Richard, C. (2023a). Deep deconvolution for traffic analysis with distributed acoustic sensing data. *IEEE Transactions on Intelligent Transportation Systems*, 24(3):2947–2962.

[van den Ende et al., 2023b] van den Ende, M., Lior, I., Ampuero, J.-P., Sladen, A., Ferrari, A., and Richard, C. (2023b). A self-supervised deep learning approach for blind denoising and waveform coherence enhancement in distributed acoustic sensing data. *IEEE Transactions on Neural Networks and Learning Systems*, 34(7):3371–3384.

[Wang et al., 2023] Wang, C., Huang, X., Li, Y., and Jensen, K. (2023). Removing multiple types of noise of distributed acoustic sensing seismic data using attention-guided denoising convolutional neural network. *Frontiers in Earth Science*, 10.

[Xu et al., 2024] Xu, G., Wang, X., Wu, X., Leng, X., and Xu, Y. (2024). Development of skip connection in deep neural networks for computer vision and medical image analysis: A survey. *ArXiv*, abs/2405.01725.

[Yang et al., 2022] Yang, L., Wang, S., Chen, X., Saad, O. M., Chen, W., Oboué, Y. A. S. I., and Chen, Y. (2022). Unsupervised 3-d random noise attenuation using deep skip autoencoder. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16.

[Zhan, 2019] Zhan, Z. (2019). Distributed Acoustic Sensing Turns Fiber?ptic Cables into Sensitive Seismic Antennas. *Seismological Research Letters*, 91(1):1–15.

[Zhang et al., 2022] Zhang, Z., Yu, P., Zhang, L., Zhao, C., Wang, Y., and Xu, Y. (2022). Application of u-net for the recognition of regional features in geophysical inversion results. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–7.

[Zhu et al., 2023a] Zhu, D., Fu, L., Kazei, V., and Li, W. (2023a). Diffusion model for das-vsp data denoising. *Sensors*, 23(20).

[Zhu et al., 2023b] Zhu, Y., Cao, J., Yin, H., Zhao, J., and Gao, K. (2023b). Seismic data reconstruction based on attention u-net and transfer learning. *Journal of Applied Geophysics*, 219:105241.

Appendix A

Data Management Plan

1. Is TU Delft the lead institution for this project?

No - This project is in collaboration with TNO, the geophysical survey of the Netherlands and TU Delft. TNO is the leading institution. It is an independent statutory research organization in the Netherlands that focuses on applied science. My main role in the project is to develop an deep learning algorithm that can denoise the DAS and implemenet in various TNO ongoing DAS Project.

2. If you leave TU Delft (or are unavailable), who is going to be responsible for the data resulting from this project?

My supervisor are Dr. Deyan Draganov, Boris Boullenger and Vincent Vandeweijer. They will be responsible for the data resulting from the project.

3. Where will the data (and code, if applicable) be stored and backed-up during the project lifetime?

TNO will have a secured dara server, on which the data will be stored.

You could find the scripts, retraining model, detection results, and the real-time denoising demo in my personal GitHub: LongSang-RealTimeDAS. All the materials will be available since 12th August, 2024

4. How much data storage will you require during the project lifetime?

< 2 TB
It requires a full data storage of data for the model training, but it will require less data storage on the final model and report.

5. What data will be shared in a research data repository?

The data set (i,e FORGE) is open accessed and can be stored. The list of detections from HECTOR and the comparison with the Silixa's detections for the same period is freely available in Bocchini et al. (2023).Moreover, the final trained model, the code and the final master thesis report will be shared on the TU Delft research repository.

Due to confidentiality, the real-time data remains accessible only on the TNO server. However, a demonstration video showcasing the results will be made available on my Git account.

6. How much of your data will be shared in a research data repository?

$<$ 2 TB
Only the open accessed FORGE data, the code, the demonstration video and the final thesis report will be shared. The internal TNO data will not be stored.

7. How will you share your research data (and code)?

Data obtained by TNO will be stored on their secure server. I will upload my code to Github.

8. Does your research involve human subjects? No

9. Will you process any personal data? No

Appendix B

U NET Model Summary

B-1 Base Strategy

Model: "model_1"

Layer (type) Output Shape Param # Connected to

==

input_4 (InputLayer) [(None, 11, 2048, 1)] 0 []

input_3 (InputLayer) [(None, 11, 2048, 1)] 0 []

tf_op_layer_Mul_2 (TensorFlowOpLayerlowOpLayer) (None, 11, 2048, 1) 0 ['input_4[0][0]', 'input_3[0][0]']

conv2d_19 (Conv2D) (None, 11, 2048, 4) 64 ['tf_op_layer_Mul_2[0][0]']

activation_18 (Activation) (None, 11, 2048, 4) 0 ['conv2d_19[0][0]']

conv2d_20 (Conv2D) (None, 11, 2048, 4) 244 ['activation_18[0][0]']

activation_19 (Activation) (None, 11, 2048, 4) 0 ['conv2d_20[0][0]']

max_pooling2d_4 (MaxPooling2D) (None, 11, 2045, 4) 0 ['activation_19[0][0]']

tf_op_layer_Conv2D_21 TensorFlowOpLayer(None, 11, 512, 4) 0 ['max_pooling2d_4[0][0]']

conv2d_21 (Conv2D) (None, 11, 512, 8) 488 ['tf_op_layer_Conv2D_21[0][0]']

activation_20 (Activation) (None, 11, 512, 8) 0 ['conv2d_21[0][0]']

conv2d_22 (Conv2D) (None, 11, 512, 8) 968 ['activation_20[0][0]']

activation_21 (Activation) (None, 11, 512, 8) 0 ['conv2d_22[0][0]']

max_pooling2d_5 (MaxPooling2D) (None, 11, 509, 8) 0 ['activation_21[0][0]']

tf_op_layer_Conv2D_23 TensorFlowOpLayer(None, 11, 128, 8) 0 ['max_pooling2d_5[0][0]']

conv2d_23 (Conv2D) (None, 11, 128, 16) 1936 ['tf_op_layer_Conv2D_23[0][0]'] activation_22 (Activation) (None, 11, 128, 16) 0 ['conv2d_23[0][0]']

conv2d_24 (Conv2D) (None, 11, 128, 16) 3856 ['activation_22[0][0]']

activation_23 (Activation) (None, 11, 128, 16) 0 ['conv2d_24[0][0]']

max_pooling2d_6 (MaxPooling2D) (None, 11, 125, 16) 0 ['activation_23[0][0]']

tf_op_layer_Conv2D_25 (TensorFlowOpLayer)(None, 11, 32, 16) 0 ['max_pooling2d_6[0][0]']

conv2d_25 (Conv2D) (None, 11, 32, 32) 7712 ['tf_op_layer_Conv2D_25[0][0]']

activation_24 (Activation) (None, 11, 32, 32) 0 ['conv2d_25[0][0]']

conv2d_26 (Conv2D) (None, 11, 32, 32) 15392 ['activation_24[0][0]']

activation_25 (Activation) (None, 11, 32, 32) 0 ['conv2d_26[0][0]']

max_pooling2d_7 (MaxPooling2D) (None, 11, 29, 32) 0 ['activation_25[0][0]']

tf_op_layer_Conv2D_27 (TensorFlowOpLayer) (None, 11, 8, 32) 0 ['max_pooling2d_7[0][0]']

conv2d_27 (Conv2D) (None, 11, 8, 64) 30784 ['tf_op_layer_Conv2D_27[0][0]']

activation_26 (Activation) (None, 11, 8, 64) 0 ['conv2d_27[0][0]']

conv2d_28 (Conv2D) (None, 11, 8, 64) 61504 ['activation_26[0][0]']

activation_27 (Activation) (None, 11, 8, 64) 0 ['conv2d_28[0][0]']

up_sampling2d_4 (UpSampling2D) (None, 11, 32, 64) 0 ['activation_27[0][0]']

concatenate_4 (Concatenate) (None, 11, 32, 96) 0 ['up_sampling2d_4[0][0]', 'activation_25[0][0]']

conv2d_29 (Conv2D) (None, 11, 32, 32) 46112 ['concatenate_4[0][0]']

activation_28 (Activation) (None, 11, 32, 32) 0 ['conv2d_29[0][0]']

conv2d_30 (Conv2D) (None, 11, 32, 32) 15392 ['activation_28[0][0]']

activation_29 (Activation) (None, 11, 32, 32) 0 ['conv2d_30[0][0]']

up_sampling2d_5 (UpSampling2D) (None, 11, 128, 32) 0 ['activation_29[0][0]']

concatenate_5 (Concatenate) (None, 11, 128, 48) 0 ['up_sampling2d_5[0][0]', 'activation_23[0][0]']

conv2d_31 (Conv2D) (None, 11, 128, 16) 11536 ['concatenate_5[0][0]']

activation_30 (Activation) (None, 11, 128, 16) 0 ['conv2d_31[0][0]']

conv2d_32 (Conv2D) (None, 11, 128, 16) 3856 ['activation_30[0][0]']

activation_31 (Activation) (None, 11, 128, 16) 0 ['conv2d_32[0][0]']

up_sampling2d_6 (UpSampling2D) (None, 11, 512, 16) 0 ['activation_31[0][0]']

concatenate_6 (Concatenate) (None, 11, 512, 24) 0 ['up_sampling2d_6[0][0]', 'activation_21[0][0]']

conv2d_33 (Conv2D) (None, 11, 512, 8) 2888 ['concatenate_6[0][0]']

activation_32 (Activation) (None, 11, 512, 8) 0 ['conv2d_33[0][0]']

```
conv2d_34 (Conv2D) (None, 11, 512, 8) 968 ['activation_32[0][0]']

activation_33 (Activation) (None, 11, 512, 8) 0 ['conv2d_34[0][0]']

up_sampling2d_7 (UpSampling2D) (None, 11, 2048, 8) 0 ['activation_33[0][0]']

concatenate_7 (Concatenate) (None, 11, 2048, 12) 0 ['up_sampling2d_7[0][0]', 'activa-
tion_19[0][0]']

conv2d_35 (Conv2D) (None, 11, 2048, 4) 724 ['concatenate_7[0][0]']

activation_34 (Activation) (None, 11, 2048, 4) 0 ['conv2d_35[0][0]']

conv2d_36 (Conv2D) (None, 11, 2048, 4) 244 ['activation_34[0][0]']

activation_35 (Activation) (None, 11, 2048, 4) 0 ['conv2d_36[0][0]']

tf_op_layer_Sub_1 (TensorFlowOpLayer) (None, 11, 2048, 1) 0 ['input_4[0][0]']

conv2d_37 (Conv2D) (None, 11, 2048, 1) 61 ['activation_35[0][0]']

tf_op_layer_Mul_3 (TensorFlowOpLayer) (None, 11, 2048, 1) 0 ['tf_op_layer_Sub_1[0][0]',
'conv2d_37[0][0]']
==================================================================

Total params: 204729 (799.72 KB)

Trainable params: 204729 (799.72 KB)

Non-trainable params: 0 (0.00 Byte)

-------------------------------------------------------------------
```