NHL STENDEN
computer vision & data science

Automated Analysis of Time Series Data

NHL Stenden Centre of Expertise in Computer Vision & Data Science

Emiel Bosman

Supervisors: Ioannis Katramados, Hossein Rahmani, Jaap van de Loosdrecht

Abstract— Time series data is used in many areas of research and various ways to represent this kind of data is available. However not a lot of research has been done to determine which kind of data representations leads to the best results when performing automated analysis of time series data. Five different neural networks have been tested with three different types of data representation as input in order to determine which of them leads to the best results. In order to further improve network performance four different kinds of data augmentations have also been tested in order to show which of these is most effective in improving network performance. The dataset used contains data from various earthquakes in the Groningen seismological region. Using spectrograms as input leads to a small performance increase in all cases when compared to the other data representations. Thus, spectrograms should be the first data representation to look at when deciding which data representation to use for analysing time series data.

Keywords—[Time series, waveform, spectrogram, CNN, earthquake]

1 Introduction

Time series data is one of many types of data found in the world of Data Science. Time series data has a natural temporal ordering. This makes time series analysis distinct from other types of analysis such as object detection and recognition where the data can be sampled in any order without losing implicit information.

Time series are used in many fields where future events need to be predicted or where past events are relevant to the current situation. Examples include earthquake prediction, predictive maintenance of hardware, healthcare, weather forecasting and many others.

Because of the temporal component inherent to time series data, analysis of this data requires a different kind of network from the Convolutional Neural Networks (CNNs) which are often used in image classification. Or the data needs to be transformed into a representation that encodes the temporal component into the image itself. Studies have used various pre-processing methods such as spectrograms (D. Park[1], Ibrahim et al.[2]), raw waveforms (T. Perol et al.[3], L. Zhu[4]) and RGB spectrograms or wavelets (W. Jiang et al.[5], S. van der Heide and E. de Wit[6]) to represent time series data.

This paper compares different types of neural networks and data representations and attempts to answer the following research question: *Which combination of data representation for time series data and type of Neural Network leads to the most accurate results when doing automated analysis of time series data?*

- *Emiel Bosman is a Computing Science student at the NHL Stenden University of Applied Sciences, E-mail: emiel.bosman@student.nhlstenden.com.*
- *Ioannis Katramados is a lector at the NHL Stenden Centre of Expertise in Computer Vision & Data Science, E-mail: ioannis.katramados@nhlstenden.com.*
- *Hossein Rahmani is a researcher at the NHL Stenden Centre of Expertise in Computer Vision & Data Science, E-mail: hossein.rahmani@nhlstenden.com.*
- *Jaap van de Loosdrecht is a lector at the NHL Stenden Centre of Expertise in Computer Vision & Data Science, E-mail: jaap.van.de.loosdrecht@nhlstenden.com.*

2 State of the Art

Before neural networks the standard approach to analysing time series data was to manually find an equation that would fit most data points and extrapolate from there.

When detecting earthquakes the standard approach has been to either use large databases of past earthquakes and try to match these to incoming data from seismological stations to determine whether an event can be classified as an earthquake or not. Or to convert the data to a spectrogram which encodes the temporal component of time series data into an image and allows for visual analysis of earthquakes.

With the advent of affordable computing power and neural networks becoming more popular, there have been several studies on using deep learning to classify and detect various types of events using time series data or spectrograms.

A recent project at NHL Stenden by S. van der Heide and E. De Wit[6] compared several neural networks and their performance when classifying earthquakes using spectrograms. They found that SpectroidNet, developed by G. van Straaten[7], is the most accurate in identifying earthquakes with a F1-score of 0.93. This architecture will serve as a baseline for the classification of events using spectrograms.

M. Al Ibrahim, J. Park and N. Athens[2] have compared three different types of networks that can be used to detect or predict earthquakes. These networks (Figure 1) are respectively a 1D CNN in combination with raw time series, 2D CNN on spectrogram data and a Recurrent Neural Network (RNN) on spectrogram data.

All networks in this study showed an accuracy of more than 95% when attempting to detect earthquakes. However when attempting to predict an earthquake the highest accuracy was 56.4% by the RNN.

Ibrahim et al. theorize that this low accuracy is due to the quality of the data, because of which the precursor signal cannot be sufficiently isolated. They suggest to manually pick the arrival times of earthquakes, which may result in better accuracy than the method they used for this study, allowing them to experiment with smaller warning periods.

This study provides baselines for both prediction and classification of events as well as an example of network architectures that can be used in combination with both spectrogram data and raw time series.

Another paper that has attempted to predict earthquakes and has reported promising results is one by D. Jozinović, A. Lomax, I. Štajduhar and A. Michelini[8]. They used a 2D CNN

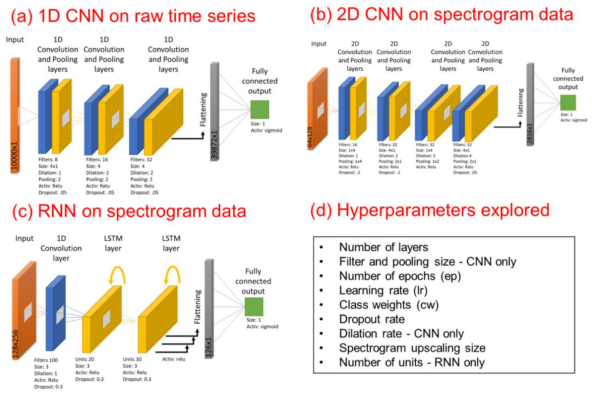

Fig. 1: The three neural networks explored by Ibrahim et al.[2]

on waveform data. This waveform data consisted of 3-channels from 39 stations. Resulting in a $39 \times N \times 3$ input where N is the amount of samples in the waveform.

They proved that earthquake prediction using waveform data is possible, provided that there is enough data available. Their network was able to predict an earthquake 10 seconds before it arrived as well as predict ground movement 15-20 seconds after arrival. This approach may not be feasible everywhere due to the large amount of data required.

3 Materials and Methods

In the following section, the choice of hardware, software, architectures, datasets and augmentations will be explained.

3.1 Hardware and software

Training neural networks is computationally expensive. As such a suitable environment is needed. The networks discussed in this paper have been trained on a virtual machine with access to 12 CPU threads, 16GB of RAM and a NVIDIA RTX 2070 with 8GB of VRAM. This ensures that training networks with many layers and large datasets can be done in a reasonable amount of time.

All networks in this paper have been built using `Python 3.8.2` and `PyTorch 1.5.0`. Datasets were loaded using `numpy 1.18.4` or `torchvision 0.5.0`.

3.2 Architectures

To analyse time series data several networks will be tested. Such as SpectroidNet, several 1D convolutional networks and a recurrent neural network.

In order to compare the performance of these networks against each other the metrics recall, precision and f1 will be used (Figure 2). The F1 score for both training and evaluation will be used in order to determine whether the model is capable of generalizing the data.

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{1}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{2}$$

$$f1 = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{3}$$

Fig. 2: Formulas for recall, precision and f1

3.2.1 SpectroidNet

When analysing spectrograms using a convolutional network de Wit and van der Heide[6] have demonstrated that SpectroidNet is fairly good at the classification of spectrograms. The network consists of four convolutional layers and three fully-connected layers (Figure 3).

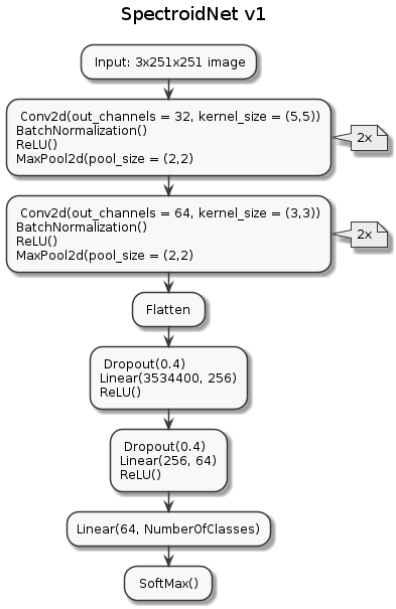

Fig. 3: Architecture of SpectroidNet

3.2.2 OneNet

Six different architectures for use with raw waveform data have been created for this research. The starting point for these networks are SpectroidNet and network (a) from Figure 1.

The networks are shown in Appendix B. In section 4.2 the three best networks will be determined and used in the following experiments.

OneNet version 1 is a network with three convolutional layers and one fully connected layer. The last two convolutional layers have a stride of four and two in addition to the MaxPooling operator having a stride of two in order the reduce the size of the input.

OneNet version 2 has four convolutional layers and one fully connected layer. These convolutional layers follow the same structure as SpectroidNet v2. With the first two layers having 32 output channels and the last two 64 output channels.

Versions 3 and 4 use dilation in their convolutional layers. These networks are based the network for 1D CNN used by Ibrahim et al[2], see also Figure 1. Version 4 uses a dropout layer of 0.05 in order to attempts to improve network robustness and an additional fully-connected layer that should improve network performance.

OneNet v5 uses three convolutional layers, the first two using stride. The third convolutional layer has a reduced kernel size in order to facilitate the reduced tensor size by that point. An additional fully connected layer was used in order to attempt to improve classification performance.

The last OneNet version that will be tested has six convolutional layers and one fully connected layer. The first three layers once again use a stride of two in order to reduce the size of the input. With the last two layers having 128 output channel in order to facilitate feature extraction.

3.3 Dataset

The dataset used in this study consists of seismological measurements made by various seismological stations in Groningen and is provided by ID3AS[9]. There are 3810 samples in this dataset. Of which 876 are labelled as earthquake events and 2934 are labelled as non earthquake events. During experiments a random 75% of the samples in this dataset is used for training. The remaining 25% is used to evaluate the network.

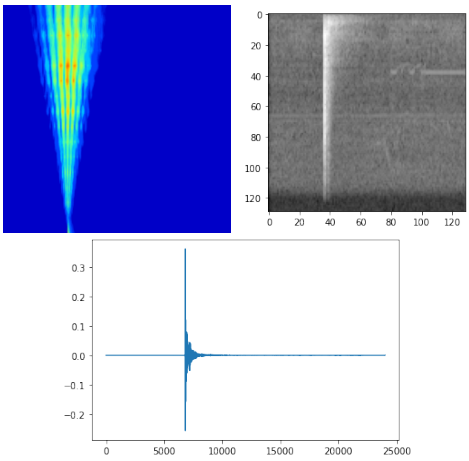

Fig. 4: Sample of Groningen Earthquake dataset, wavelet, spectrogram and waveform representation respectively

Each sample (Figure 4) is a recording of two minutes of seismological activity with 200 Hz sampling. These recordings have been made during an earthquake, a storm, a time during which there was a lot of traffic and other noise and a time during which there was a low amount of noise.

Two version of this dataset have been provided, raw waveforms and wavelet images. The raw waveforms contain the raw seismological data from each station without any post-processing. The wavelet images are Morlet wavelets created using the `pywt`[10] library. These were then turned into RGB images using false colour mapping in order to make the image easier to interpret for researchers.

As described in section 1 a third option is regularly used to represent time series data. This being spectrograms. These spectrograms have been created using the `MelSpectrogram` module in the `torchaudio.transforms`[11] library.

3.4 Augmentations

In order to prevent network overfitting, augmentations are applied during training. Due to the nature of time series data the type of augmentations that can be applied are limited. Image augmentations such as rotation or perspective manipulation do not make sense for time series as time series data does not contain this information. Park et al.[1] developed a set of four augmentations specifically for time series data. These augmentations are time warping, time masking and Frequency Masking. Additionally van der Heide and de Wit[6] used Salt and Pepper noise as an augmentation for images and will be used to augment wavelets spectrogram images. A noise augmentation for raw time series data and spectrograms will also be tested.

3.4.1 Time warping

By shifting the data in time, specific events in the data will appear to take place earlier or later depending on the value of the time warp (Figure 5). The time warp is done as either a random horizontal translation of the image or by shifting the values in raw time series a random amount of steps towards the start or end of the series.

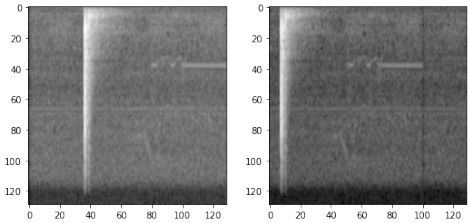

Fig. 5: Unaugmented spectrogram, Time warped spectrogram

3.4.2 Time masking

Time masking works by taking two random points in time in the data and changing all values between these two points to 0. Resulting in a horizontal bar when looking at the resulting image. See Figure 6 for an example.

3.4.3 Frequency masking

Frequency masking follow the same principle as time masking but instead of masking a time range, it masks all values the fall between two random frequencies.

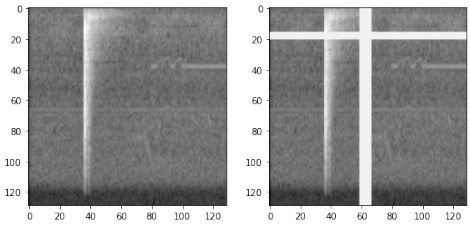

Fig. 6: Unaugmented spectrogram, masked spectrogram
Horizontal line is a frequency mask, vertical line is a time mask

3.4.4 Noise augmentations

Salt and pepper noise works by changing a percentage of all pixels in an image to either black or white.

Adding noise to time series data is implemented by adding a random value to all values in the data.

4 Experiments

In this section the planned experiments for this paper will be described. The datasets used for these experiments are described in section 3.3.

4.1 SpectroidNet architecture

The current SpectroidNet (Figure 3) uses more than 7982 MiB during classification of 251×251 images with a batch size of 1. As a result, it is not possible to use this network on the available hardware. Decreasing the image size to 128 x 128 decreases memory usage to 5593 MiB at a batch size of 1. In order to make SpectroidNet usable it needs to have its memory usage lowered.

By increasing the stride of one or more convolutional layers, the input is down sampled and memory usage is reduced. Two approaches are tested. The first approach (Figure 7) adds a stride of 2 to the first two convolutional filters of SpectroidNet, this network is referred to as SpectroidNet v2. The second approach (Figure 8) adds a stride of 2 to all convolutional filters and is referred to as SpectroidNet v3.

The experiment will be run using 128 x 128 images for all networks as well as 251 x 251 images for the two revised networks. All other hyperparameters are kept the same

throughout the experiment. The only augmentation used is Salt and Pepper noise. The top two architectures will be used in the other experiments.

4.1.1 Results

Table 1: Results of SpectroidNet architecture experiment

Network	Mem. (MB)	Precision	Recall	F1	Eval f1
128x128					
SpNet v1	7785	0.860	0.773	0.814	0.791
SpNet v2	1195	**0.967**	**0.973**	**0.970**	**0.965**
SpNet v3	**927**	0.947	0.948	0.948	0.946
251x251					
SpNet v2	2823	**0.971**	**0.976**	**0.973**	**0.967**
SpNet v3	**1507**	0.961	0.962	0.962	0.961

Both SpectroidNet v2 and SpectroidNet v3 perform significantly better than version 1 while also using up to eight times less memory. SpectroidNet v2 performs better than version 3, while version 3 has a lower memory footprint.

4.2 OneNet architecture

Because a waveform is a single dimensional input, conventional image classification networks cannot be used. As such a new network is needed in order to process this kind of data. In this experiment the networks described in section 3.2.2 are trained for 20 epochs with a batch size of 32. Noise and TimeShift augmentations are used. The best three networks will be used in all following experiments.

4.2.1 Results

Table 2: Results for OneNet architecture experiment

Network	Precision	Recall	F1	Eval f1
OneNet v1	0.860	0.773	0.814	0.791
OneNet v2	**0.904**	0.836	0.869	**0.901**
OneNet v3	0.870	0.782	0.824	0.732
OneNet v4	0.820	0.735	0.775	0.798
OneNet v5	0.898	**0.845**	**0.871**	0.878
OneNet v6	**0.933**	**0.884**	**0.908**	**0.899**

OneNet versions 2, 5 and 6 have the best performance in one or more metrics. However their performance is still lower than the performance of SpectroidNet. These three versions will be used in all following experiments.

4.3 Data pre-processing

In this experiment all networks will be trained for 20 epochs and a batch size of 32, with no augmentations. SpectroidNet will be trained using both RGB images and the Spectrograms created using the `torchaudio` library.

4.3.1 Results

Using spectrograms as input for the model leads to the best results (Table 3), wavelet inputs show the second best performance, closely followed by waveform input.

Table 3: Results for Data pre-processing experiment

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.992**	**0.998**	**0.995**	**0.987**
SpectroidNet v3	**0.984**	**0.994**	**0.989**	**0.987**
Waveform				
OneNet v2	0.974	0.944	0.959	0.955
OneNet v5	0.898	0.898	0.926	0.960
OneNet v6	0.967	0.967	0.975	0.965
Wavelet				
SpectroidNet v2	0.980	0.980	0.986	0.970
SpectroidNet v3	0.968	0.974	0.971	0.965

4.4 Augmenting the dataset

As discussed earlier, augmenting a dataset is one possible solution to increase network performance[12] and reduce overfitting. In this experiment different combinations of the augmentations described in section 3.4 are tested in order to determine which combination results in the highest performance with the least amount of delta between training and evaluation metrics. Each network is trained ten times on each set of augmentations after which the results are averaged to reduce the impact of random number generation on the results. Each network is trained for 30 epochs and use a batch size of 32.

4.4.1 Results

The tables in Appendix C show the complete results for each experiment. All results show the average of 10 training sessions of 30 epochs. SpectroidNet v2 in combination with spectrograms as input results in the highest performance in nearly all experiments and metrics.

Table 4: Best networks for Augmentation set:
Noise, TimeShift, FrequencyMask, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.942**	**0.960**	**0.951**	**0.956**
Waveform				
OneNet v6	**0.939**	0.895	**0.917**	**0.920**
Wavelet				
SpectroidNet v2	0.912	0.915	0.913	0.916

Waveform representation does not perform very well. Ending up with a single set of augmentations that gave it a top two performance in most metrics when using Noise, TimeShift, FrequencyMask and TimeMask augmentations (Table 4). Out of the networks using waveforms as input OneNet v6 has consistently the best performance. Additionally this augmentation set performs on par with the augmentation set of Noise and TimeShift (Table 8).

Table 5: Best networks for Augmentation set:
Noise, FrequencyMask, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.976**	**0.985**	**0.981**	**0.980**
Waveform				
OneNet v6	**0.983**	0.967	0.975	0.967
Wavelet				
SpectroidNet v2	**0.976**	**0.985**	**0.981**	0.973

Wavelet representation never exceeds the performance of using spectrograms. With only the noise, FrequencyMask and

TimeMask augmentations (Table 5) resulting in performance approaching the spectrogram input. Just as with spectrogram input SpectroidNet v2 has better performance than SpectroidNet v3 in all sets of augmentations.

Table 6: Best networks for Augmentation set: Noise

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.984**	**0.995**	**0.990**	**0.987**
Waveform				
OneNet v6	**0.987**	0.972	0.980	0.971
Wavelet				
SpectroidNet v2	0.979	0.988	0.984	0.974

The noise augmentation shows the best results (Table 6). With 0.990 and 0.987 F1-score for training and evaluation respectively when using Spectroidnet v2 with spectrogram input. OneNet v6 also show very good results with this augmentation. Resulting in a precision of 0.987. Which is the highest of all tested networks. However its recall is not high enough to warrant a good f1 score during either training or evaluation.

Table 7: Best networks for Augmentation set: TimeShift

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.975**	**0.988**	**0.981**	**0.982**
Waveform				
OneNet v6	0.939	0.893	0.915	0.918
Wavelet				
SpectroidNet v2	0.933	0.942	0.937	0.933

Time warping (Table 7 has the second best performance when using spectrograms, with 0.981 and 0.982 F1-score in training and evaluation. However performance for both waveform and wavelet is significantly lowered. Combining the noise and timeshift augmentations (Table 8) results in lower performance in most metrics while positively impacting evaluation performance when compared to training performance.

Table 8: Best networks for Augmentation set: Noise, TimeShift

2 Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.970**	**0.984**	**0.976**	**0.980**
Waveform				
OneNet v6	0.936	0.892	0.914	0.920
Wavelet				
SpectroidNet v2	0.920	0.927	0.924	0.920

Frequency masking (Table 9) and time masking (Table 10) both heavily reduced network training performance while having a small positive impact on evaluation performance. With time masking reducing the train F1 score of SpectroidNet v2 with spectrogram input to 0.972 compared to the baseline of 0.981 when only using the TimeShift augmentation.

Table 9: Best networks for Augmentation set: TimeShift, FrequencyMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.968**	**0.982**	**0.975**	**0.978**
Waveform				
OneNet v6	0.706	0.723	0.714	0.773
Wavelet				
SpectroidNet v2	0.924	0.932	0.928	0.923

Table 10: Best networks for Augmentation set: TimeShift, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.964**	**0.980**	**0.972**	**0.969**
Waveform				
OneNet v6	**0.939**	0.894	0.916	0.924
Wavelet				
SpectroidNet v2	0.923	0.932	0.928	0.929

This can also be seen in the other experiments using the time mask augmentation. Such as the ones in Table 11 and Table 4. With both of them having significantly lower performance in all metrics when compared to an experiment that did not use the time mask augmentation such as the one in Table 9.

Table 11: Best networks for Augmentation set: TimeShift, FrequencyMask, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.953**	**0.970**	**0.962**	**0.965**
Waveform				
OneNet v6	**0.938**	0.894	0.916	0.919
Wavelet				
SpectroidNet v2	0.922	0.922	**0.932**	0.927

5 Discussion

This study found that there is a significant benefit to using spectrograms over waveforms and wavelets when analysing time series data. Additionally, the reduction in performance due to augmentations was significantly larger when using wavelet or waveform representations. Why this is the case is not entirely clear.

Unlike the results found in SpecAugment[1], time warping is most effective in improving results in this study. This is probably due to the fact that there is one specific feature the network needs to identify in this dataset. Additionally, this feature is small enough that moving it to an earlier or later time in the sample has little impact on the feature itself. While the background noise is low enough to not create features the network will pick up on.

Time masking significantly reduces performance and frequency masking shows a small negative impact. This might be due to the specific dataset used for training the network as the features needed to correctly identify earthquakes are found in a small slice of time in each sample. Randomly masking this domain introduces features that either fully mask the feature the network needs to identify or in the case of a negative sample creates a mask that is too similar to a positive sample.

Much like the results found in 'Earthquake classification using deeplearning'[6], the experiments in this study confirm

that using a noise augmentation improves the network's generalization capabilities, while having a minimal impact on overall performance.

The dataset used in this study consists of single channel earthquake recordings. As such it is not known whether these results can be generalized to multi-channel data or data from different types of events.

6 Future Work

As stated in the previous section, only one dataset was used during this study. Further research could be done to determine whether data such from different sources or data with multiple channels also show the same performance benefits when using spectrograms or that raw waveforms or wavelets show better performance in these cases.

Additionally, due to time constraints it was not possible to do hyperparameter tuning for the networks could possibly improve the network performance further.

7 Conclusion

In this study several deep neural networks have been compared. In section 4.3 we show that using spectrograms to train neural networks is more effective than either waveform or wavelet representations. With waveform representation having the worst performance of the three.

In section 4.4 eight combinations of augmentations were tested on all networks. From the results of these experiments we can conclude that noise and time warping augmentations show the largest performance increase in network performance. While frequency and time masking show the largest decrease in network performance. Noise in combination with either the time warp augmentation or the frequency and time masking augmentations result in the highest performance when more than one augmentation is used.

Based on these results it is recommended to use spectrograms when possible for automated time series analysis in combination with the noise and time warping augmentations during training.

Acknowledgements

I would like to thank the Centre of Expertise in Computer Vision and Data Science for allowing me to take on this project and for offering technical support in order to work remotely during the Covid-19 lockdown. I want to thank Hossein Rahmani and Ioannis Katramados in particular for their suggestions and feedback regarding the paper.

This project was commissioned by ID3AS, with the goal of creating a service that would integrate with SensEQuake.

References

[1] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, Sep 2019.

[2] M. Al Ibrahim, J. Park, and N. Athens. Earthquake warning system: Detecting earthquake precursor signals using deep neural networks, Sep 2018.

[3] Thibaut Perol, Michaël Gharbi, and Marine Denolle. Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2), 2018.

[4] Lijun Zhu, Zhigang Peng, James McClellan, Chenyu Li, Dongdong Yao, Zefeng Li, and Lihua Fang. Deep learning for seismic phase detection and picking in the aftershock zone of 2008 m7.9 wenchuan earthquake. *Physics of the Earth and Planetary Interiors*, 293:106261, Aug 2019.

[5] Weiheng Jiang, Xiaogang Wu, Bolin Chen, Wenjiang Feng, and Yi Jin. Time-frequency analysis based blind modulation classification for multiple-antenna systems, 2020.

[6] Sjoerd van der Heide and Emiel de Wit. Earthquake classification using deep learning, 2019.

[7] Guido van Straaten. Classification and regression by performing deeplearning on spectrogram images, 2018.

[8] Dario Jozinović, Anthony Lomax, Ivan Štajduhar, and Alberto Michelini. Rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network, 2020.

[9] ID3AS. Over ons.

[10] Filip Wasilewski. Pywt: Wavelets.

[11] PyTorch. Torchaudio.

[12] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017.

A Experiment SpectroidNet: New architectures

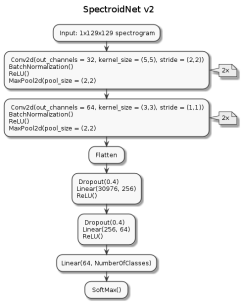

Fig. 7: Architecture of SpectroidNet v2

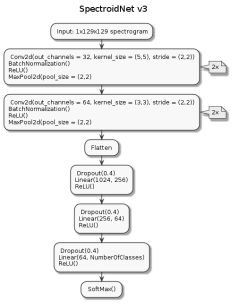

Fig. 8: Architecture of SpectroidNet v3

B Experiment One Net: Architectures used

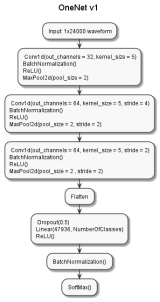

Fig. 9: Architecture of OneNet v1

Fig. 10: Architecture of OneNet v2

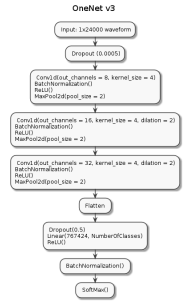

Fig. 11: Architecture of OneNet v3

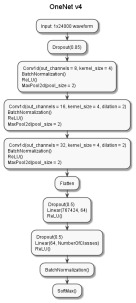

Fig. 12: Architecture of OneNet v4

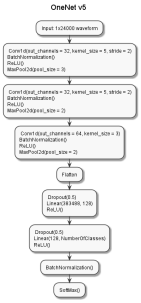

Fig. 13: Architecture of OneNet v5

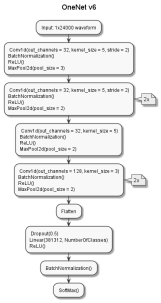

Fig. 14: Architecture of OneNet v6

C Augmentation experiment results

Table 12: Results for Augmentation experiment: Noise, TimeShift, FrequencyMask, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.942**	**0.960**	**0.951**	**0.956**
SpectroidNet v3	0.893	**0.926**	0.909	0.918
Waveform				
OneNet v2	0.904	0.831	0.866	0.877
OneNet v5	0.899	0.825	0.860	0.895
OneNet v6	**0.939**	0.895	**0.917**	**0.920**
Wavelet				
SpectroidNet v2	0.912	0.915	0.913	0.916
SpectroidNet v3	0.899	0.892	0.895	0.888

Table 13: Results for Augmentation experiment: Noise, FrequencyMask, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.976**	**0.985**	**0.981**	**0.980**
SpectroidNet v3	0.965	0.978	0.971	**0.975**
Waveform				
OneNet v2	0.973	0.950	0.961	0.961
OneNet v5	0.956	0.906	0.930	0.959
OneNet v6	**0.983**	0.967	0.975	0.967
Wavelet				
SpectroidNet v2	**0.976**	**0.985**	**0.981**	0.973
SpectroidNet v3	0.966	0.971	0.968	0.966

Table 14: Results for Augmentation experiment: Noise

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.984**	**0.995**	**0.990**	**0.987**
SpectroidNet v3	0.979	**0.992**	**0.985**	**0.983**
Waveform				
OneNet v2	0.977	0.955	0.966	0.961
OneNet v5	0.961	0.919	0.940	0.961
OneNet v6	**0.987**	0.972	0.980	0.971
Wavelet				
SpectroidNet v2	0.979	0.988	0.984	0.974
SpectroidNet v3	0.969	0.974	0.971	0.966

Table 15: Results for Augmentation experiment: TimeShift

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.975**	**0.988**	**0.981**	**0.982**
SpectroidNet v3	**0.964**	**0.983**	**0.973**	**0.981**
Waveform				
OneNet v2	0.914	0.841	0.876	0.874
OneNet v5	0.903	0.834	0.867	0.890
OneNet v6	0.939	0.893	0.915	0.918
Wavelet				
SpectroidNet v2	0.933	0.942	0.937	0.933
SpectroidNet v3	0.921	0.919	0.920	0.919

Table 16: Results for Augmentation experiment: Noise, TimeShift

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.970**	**0.984**	**0.976**	**0.980**
SpectroidNet v3	**0.961**	**0.979**	**0.970**	**0.978**
Waveform				
OneNet v2	0.911	0.842	0.875	0.877
OneNet v5	0.905	0.837	0.870	0.895
OneNet v6	0.936	0.892	0.914	0.920
Wavelet				
SpectroidNet v2	0.920	0.927	0.924	0.920
SpectroidNet v3	0.903	0.899	0.901	0.893

Table 17: Results for Augmentation experiment: TimeShift, FrequencyMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.968**	**0.982**	**0.975**	**0.978**
SpectroidNet v3	**0.953**	**0.976**	**0.964**	**0.971**
Waveform				
OneNet v2	0.690	0.671	0.680	0.676
OneNet v5	0.690	0.664	0.677	0.755
OneNet v6	0.706	0.723	0.714	0.773
Wavelet				
SpectroidNet v2	0.924	0.932	0.928	0.923
SpectroidNet v3	0.912	0.910	0.911	0.903

Table 18: Results for Augmentation experiment: TimeShift, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.964**	**0.980**	**0.972**	**0.969**
SpectroidNet v3	0.933	**0.962**	**0.948**	**0.960**
Waveform				
OneNet v2	0.910	0.835	0.871	0.887
OneNet v5	0.903	0.838	0.869	0.890
OneNet v6	**0.939**	0.894	0.916	0.924
Wavelet				
SpectroidNet v2	0.923	0.932	0.928	0.929
SpectroidNet v3	0.916	0.914	0.915	0.909

Table 19: Results for Augmentation experiment: TimeShift, FrequencyMask, TimeMask

Network	Precision	Recall	F1	Eval f1
Spectrograms				
SpectroidNet v2	**0.953**	**0.970**	**0.962**	**0.965**
SpectroidNet v3	0.912	**0.948**	0.930	**0.941**
Waveform				
OneNet v2	0.906	0.831	0.867	0.868
OneNet v5	0.905	0.834	0.868	0.890
OneNet v6	**0.938**	0.894	0.916	0.919
Wavelet				
SpectroidNet v2	0.922	0.922	**0.932**	0.927
SpectroidNet v3	0.909	0.903	0.906	0.902