Neural Network Classification of Seismic Events in a Volcanic Eruptive Setting

Royal Holloway, University of London

Richard Przemysław Strange

April 5, 2018

Abstract

Data collected by Goitom et al. (2015) and Hammond et al. (2013) of the June 2011 eruption of the Nabro Stratovolcano in Eritrea underwent complex signal processing and analysis to create an attribute database of noise and manually-picked events, for all collected data between the 29thAugust to the 7th September 2011 (Days 241-250). The attribute database was fed into a convolutional neural network, run on Python 3.6 using a *keras* front-end and *TensorFlow* back-end. An initial baseline model achieved 65% accuracy, and a further improved model achieved 100% accuracy within three epochs. Although there are no signs of overfitting, the database will need expanding to further settings, manual pickers and more noise data. The successful demonstration of a convolutional Neural Network in the field of volcanic seismology will allow further research in event type classification and event precursor signal identification applications.

Contents

List of Figures

List of Tables

1 Introduction

On 12th June 2011, the Nabro Volcano in Eritrea began to erupt resulting in the deaths of at least seven people (Eritrea - Ministry of Information, 2011). An unexpected eruption, there was little monitoring in place (Goitom et al., 2015). A team led by researchers from the universities of Bristol and Birkbeck completed an array of seismometers intended for monitoring the Afar Depression shortly after the start of the eruption recording the majority of the activity and resulting in a thorough dataset capturing the seismic behaviour of the erupting stratovolcano across multiple stations. (Hammond et al., 2013)

Following the capture of this data, analysis of the seismic data, both spatial and temporal, included an extensive programme of event identification and selection, or 'picking'. Given the time-intensive nature of manually picking events from the dataset, 'auto-picker' scripts are commonly used. These vary in accuracy and are not sophisticated enough to pick out events obscured by a high signal : noise environment reliably. In the case of Goitom et al. (2015), thousands of events were *manually* picked and extracted from the data — side-stepping the weaknesses of current auto-pickers — resulting in a further dataset of thousands of labelled events as a by-product. A dataset of this scale is unusual and allows the application of novel techniques, such as the use of an Artificial Neural Network (**ANN**) to learn from a researcher's manual picking patterns and develop a more sophisticated picking algorithm.

An ANN is a Deep Learning technique that mimics the mechanics of the human brain to "learn" complex datasets. Deep learning has moved into the reach of desktop machines following on from the release of CUDA-API (*Compute Unified Device Architecture - Application Programming Interface*), which enables graphics processing units to run machine learning on better-suited and faster GPUs, as opposed to CPUs. This change has led to the rapid and widespread application of Machine Learning and Deep Learning AI techniques in many areas of industry and academia, but has not seen much use in Geology and Geophysics, which historically has a stronger reliance on numerical modelling.

This paper aims to demonstrate that an ANN can improve on an auto-picker, as a technological demonstration of the use of ANNs in the fields of Geophysics and Geology.

The research aims of this project are to:

1. Analyse seismic data in a volcanic eruptive setting, with the intention of achieving a binary classification of intervals of seismic data as either discrete events or noise.
2. Demonstrate the use of a Convolutional Neural Network (CNN) as an effective and novel method in processing and classifying seismic datasets to near-human accuracy for bulk data.
3. Identify weaknesses in using seismic data in Machine Learning applications, mitigating issues where possible through thorough data preparation.

2 Layman's Summary

Volcanic eruptions need magma, which can only reach the central and side vents through upward migration through the *country rock* (the surrounding prior lithologies). Even with the heat and pressure of the magma, country rock rarely deforms in a purely plastic manner. As such, the migration results in constant brittle fracture failures leading up to, during, and after eruptions, propagating kinetic waves radially outward from the point of failure. These failures are caused by an intrusion pushing up as magmatic pressure rises until the pressure-driven stress overcomes the country rock's fracture toughness, releasing stress in a sharp discrete event.

Seismometers on the surface can measure these waves as seismic events. Rather than the immense regional failures across entire 50km$\leq$ faults, these smaller shallow internal failures result in much smaller localised earthquakes. Being able to monitor and understand these failures would allow the progression of magma to the surface, the likelihood of an eruption and a timescale of eruption to be estimated allowing smarter and more effective mitigative engineering, population and livestock evacuation, and wider-area hazard preparation.

Manual Classification of an event can take time, and in volcanic settings where hundreds of events can occur in an hour, it can sometimes be impossible to provide manual real-time analysis. Furthermore, smaller fractures may not be readily resolved against the background noise of constant tremors, magma movement, bubbling and background regional and global activity. Therefore, there is a need for a solution that can identify events quickly, with a higher sensitivity to events in noisy signals.

There have been attempts at using *heuristic* programs, which apply a defined set of parameters (amplitude, deviation, duration, for example) to signals to 'auto-pick' events. These tools are inflexible and their accuracy is uncertain. A set of heuristic rules cannot compete

with innate human performance in picking out intricate patterns in noise-rich domains, so a more sophisticated approach is needed if accurate analysis of seismic volcanic behaviour can be achieved in a real-time application.

Artificial Neural networks are a candidate solution to this gap in capability. Often abbreviated to **ANN**, (*or **cNN** for the more specific convolutional Neural Network used in this paper*) these models mimic the mechanics of the human brain to understand data better, from micro-scale neuron behaviour to macro-scale brain plasticity.

Neurons in the brain, on a simple level, operate by having neighbouring neurons pass them an impulse. If the impulse is great enough to overcome an internal threshold, the neuron triggers and propagates the impulse to its neighbours. The threshold can vary, and as a neuron can be connected to many neighbours, it can be just one impulse, or many from multiple other neurons at the same time that meet the threshold.

For example, nerve cells in skin are triggering continuously, sending sensory information on touch and temperature. The nerves send too few concurrent impulses to overcome a neuron threshold, halting the signal and meaning that the sensation isn't noticed. However, when there is a sharp pain, the higher and more strongly concurrent impulse exceeds the threshold, and the signal propagates to another cluster of neurons that might assess the pain as a sign of danger and release adrenaline. This cluster might sit behind another set of neurons that assess whether the pain is localised, or if there is any sound or smell information that suggests danger. They may have a greater or a lesser impact on the result —pain is more indicative of danger than just seeing a predator, for example — and varying thresholds in these neurons can control the weighting. This means that a '*flight-or-fight*' response isn't triggered from just a photograph of a wolf, providing a more accurate and assessed response to the situation.

These pathways grow during our lifetimes, with more used pathways growing stronger, and less used pathways being pruned or weakened. As we learn, abstract paths in our brains respond, spotting patterns that grow as we find the right answer, or weaken when we find the wrong answer. This 'plasticity' allows our limited biological resources to focus on valuable neuron pathways, and forms a vital part of the human learning process.

Likewise, we can construct artificial models that behave similarly. If we provide an ANN with information about a thousand dogs (Table 1) and let it explore every possible combination

of features, then the model might evaluate the hypothesis that:

$$All\ Dalmatians\ have\ spots \mid Golden\ Retrievers\ do\ not\ have\ spots$$

Almost every time it guessed using spots it was right, so it strengthened the pathway evaluating spots. Evaluating another hypothesis:

$$All\ Dalmatians\ like\ carrots \mid Golden\ Retrievers\ do\ not\ like\ carrots$$

would result in a weakened pathway, as these was no meaningful difference in the data, and the link was trivial. Another hypothesis:

$$All\ Dalmatians\ weigh\ more\ than\ 25kg \mid Golden\ Retrievers\ do\ not\ weigh\ more\ than\ 25kg$$

wouldn't work as well as spots, but the inverse would be partially strengthened. Abstracting up, the next level of neurons might look at combinations of spots and weight, and notice that, for example, non-spotted Dalmatians tend to be lighter, so if there is a non-spotted dog that is very light, it should guess that it is a Dalmatian, further improving the accuracy of the model.

Dog	Like Carrots? [%]	Have Spots [%]	Are Heavier Than 25kg [%]
Dalmatians	63	98	22
Golden Retrievers	67	0	84

Table 1: Table describing data provided to an ANN to classify dogs

Applying this to seismic data, we can draw information about the signal from the data, and present it to the neural network. Although the Dog example is more evident in its patterns, the abstract data from signal analysis of the seismic data is almost indecipherable to a researcher but can be easily processed by an ANN. This way it can find patterns 'invisible' to humans in data that would be missed by human-programmed heuristic programs.

3 Methodology

3.1 Seismic Recordings

Goitom et al. (2015) and Hammond et al. (2013), whose data we use for this project, describe the installation and use of the seismometer network emplaced through the Afar region. The seismometer network was deployed in June 2011, and coincidentally recorded the majority of the eruptive behaviour from the surprise activity of the Volcano. The seismic recordings began 11 days into the eruption, on the 23rd June 2011. Data from the 29thAugust to the 7th September (Days 241-250) was provided by the University of Bristol and Birkbeck, University of London, and has been used for this paper's model.

Six sensors were deployed to measure the Afar Depression as described in Hammond et al. (2013) *(see fig.1 for station locations)* across Eritrea, Yemen and Ethiopia, as the *"Horn of Africa"* seismic station network.

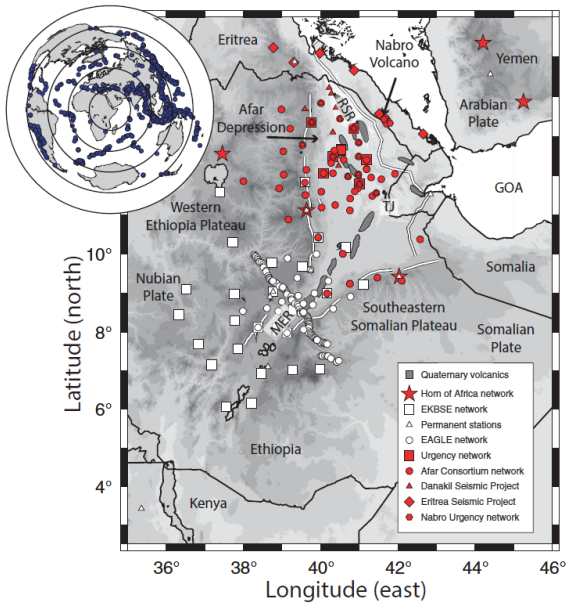


Figure 1: Map showing the seismic stations in the vicinity of Nabro - Note the red starred *Horn of Africa* network points used in this model. *(reproduced from Hammond et al. (2013))*

3.2 Data Description

Data files for days 241-250 of 2011 for the network were provided as both whole-day recordings, and as hand-picked events identified by B. Goitom *(School of Earth Sciences, University*

of Bristol). All data was provided in the standard *.sac* format[1] *(see (IRIS, 2013) for header documentation)* usable with the Incorporated Research Institutions for Seismology (**IRIS**) published Seismic Analysis Code (**SAC**) CLI-level program.

Event recordings consist of a leading 10 seconds prior to the beginning of the event, and the trailing 290 seconds after the marked beginning, for a total of 300 seconds per file. Recordings can contain additional events, which have their own overlapping records*(see fig.2 for an example of a manually identified and cropped seismic trace).*

Data Form	Recordings
Identified Events	2450
Whole Day Recordings	46
Identified Noise Recordings*	46650
Downsampled Noise Recordings*	2450

Table 2: Table showing the provided and calculated (*demarcated with an asterisk**) data used in the data preparation. Noise recording identification and downsampling are described in sections 3.3.5 and 3.3.8.

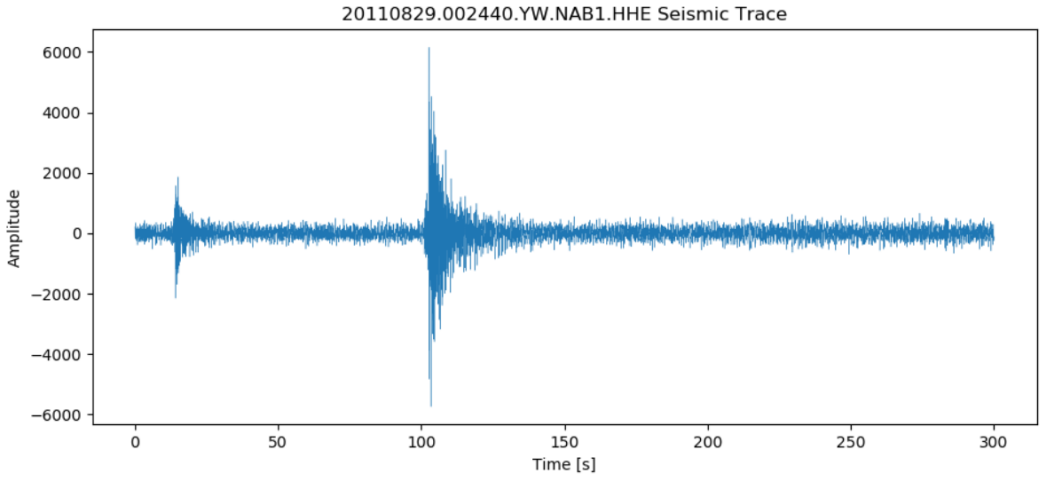


Figure 2: Specimen Data from event:*20110829.002440.YW.NAB1.HHE* from the identified events dataset. A recording showing two events within a single recording. Each recording captures ten seconds prior to, and the trailing 290 seconds after the event start, even if that includes a second event. The second event exist as its own recording.

3.3 Data Preparation Steps

The data was subjected to significant preparation before use in the Neural Network. In sequence:

[1] time-series data with the first 30 rows as fixed-width positional header fields, compressed

1. SAC Conversion
2. Header Removal and Column Flattening
3. Time-series Attachment
4. Noise Sampling
5. Kaiser Windowing
6. Fourier Processing
7. Frequency Binning
8. Attribute Database Compilation

We will describe each of these briefly - for more details and code examples refer to the *Jupyter* code notebooks.

3.3.1 Environments and Major Libraries

All of the data preparation work is coded using Python 2.7.14 running on an Anaconda distribution on a Windows 10 64bit OS unless otherwise stated. Internal libraries may not be explicitly described (e.g. **os**, the internal library for interacting with the container shell)

Pandas - derived from **Pan**el **Da**ta, is a standard Data Science library, used to introduce multidimensional dataframe functionality for handling data. Pandas is used for handling data in a tabular form, making data processing more manageable, and a key building block for constructing higher-dimensional tensor matrices required for Neural Networks. Often abbreviated in the code-base to an idiomatic *pd*.

Numpy - derived from **Num**erical **Py**thon, is another standard Data Science library, that provides additional mathematical functionality and new datatypes to Python for use in scientific applications. It also contains a new array iterable object type, closely linked with the Pandas' DataSeries object. It is a standard prerequisite for the vast majority of scientific libraries available in Python. Often abbreviated in the code-base to an idiomatic *np*.

SciPy - derived from **Sci**entific **Py**thon, like *numpy* it expands the scientific functionality for Python, providing functionality from image processing to statistical modules to calculus libraries. For Fourier transformation processing, we are using the *fftpack* (fast Fourier transform pack) sub-module.

3.3.2 SAC Conversion

The first step was run in a CentOS VirtualBox Virtual Machine to meet the operating system environment requirements of the Seismic Audio Code (**sac**) application, which requires a bash kernel in a Unix OS. The sac application runs on a terminal-level interface and has limited batch processing functionality, so a Python script was written and run to emulate the terminal commands through a pipe connection to drive the program. All the individual events and the whole-day records were translated into an equivalent ASCII-encoded format that was readable by other programs and scripts.

3.3.3 Header Removal and Column Flattening

Although the files are now readable, the files are meaningless as the data is arranged as 30 rows of headers and payload is arranged in an unparseable 5 column format (*see table 3 and Fig. 3, and see IRIS (2013) for full documentation*). There is some useful information in the headers, seen in bold in the table. The first cell ("**DELTA**") provides the time interval ($\Delta T = 0.01 seconds$), and the first cell on the 22$^{\text{nd}}$ row ("**LEVEN**") confirms that the time intervals are uniform (*Boolean: either "0" - uneven, or "1" - even*). This is constant throughout all of the recordings but is significant when the Time-series index is recalculated, and for the Fourier transform calculation steps.

The cells in bold on the 15$^{\text{th}}$ and 16$^{\text{th}}$ rows ("**NZYEAR**" through "**NZMSEC**") provide the Timestamp for the beginning of the recording, which is relevant for temporal analysis across different recordings, but will not be needed for now. Having determined that the ΔT is 0.01 seconds and uniform throughout each recording, and that the Timestamp does not need to be preserved, we can strip the header out.

Step two was run in the same CentOS VirtualBox Virtual Machine used in section 3.3.2, and made use of the *awk, tr* and *grep* functionality of the Bash kernel for bulk text-file editing,

driven through a Python script interacting with Bash.

To flatten the multi-columnar data format to a simple uni-columnar arrangement, each file was piped through a series of *awk, tr* and *grep* Bash statements to strip the first 30 rows (the header); replace any pre-existing troublesome carriage returns, line breaks or new lines; then with *regex* find and substitute any number of consecutive spaces that delimit the data for a newline character. This is then piped into a new headless and flattened csv flat-file.

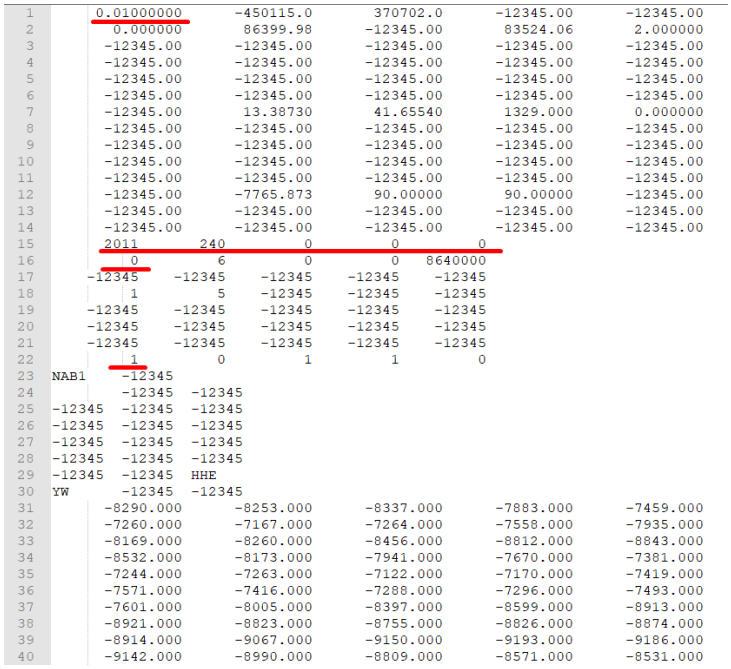


Figure 3: First 40 lines of event 20110829.002440.$YW.NAB1.HHE$ as a Specimen ASCII Data Format. Note the fields underlined, correlating to those indicated in Table 3.

3.3.4 Time Series Attachment

From this point in the Data Preparation pipeline, scripts are run in the standard Windows 10 x64 environment (see section 3.3.1). With the data now flattened and headless, it is readily digestible. Knowing that the ΔT is 0.01 seconds, and without Timestamp data to anchor the dataset to an originating datetime, we have an arbitrary time-series array in 0.01 second intervals that we attach to the dataset as the index of the pandas dataframe.

3.3.5 Noise Sampling

A roughly equal number of Noise samples to Event samples would be needed to provide the base dataset for the Neural Network — around 2450. Event data is pre-selected, in the format

DELTA	DEPMIN	DEPMAX	SCALE	ODELTA
B	E	O	A	INTERNAL
T0	T1	T2	T3	T4
T5	T6	T7	T8	T9
F	RESP0	RESP1	RESP2	RESP3
RESP4	RESP5	RESP6	RESP7	RESP8
RESP9	STLA	STLO	STEL	STDP
EVLA	EVLO	EVEL	EVDP	MAG
USER0	USER1	USER2	USER3	USER4
USER5	USER6	USER7	USER8	USER9
DIST	AZ	BAZ	GCARC	INTERNAL
INTERNAL	DEPMEN	CMPAZ	CMPINC	XMINIMUM
XMAXIMUM	YMINIMUM	YMAXIMUM	ADJTM	-
-	-	-	-	-
NZYEAR	**NZJDAY**	**NZHOUR**	**NZMIN**	**NZSEC**
NZMSEC	NVHDR	NORID	NEVID	NPTS
NSPTS	NWFID	NXSIZE	NYSIZE	-
IFTYPE	IDEP	IZTYPE	-	IINST
ISTREG	IEVREG	IEVTYP	IQUAL	ISYNTH
IMAGTYP	IMAGSRC	-	-	-
-	-	-	-	-
LEVEN	LPSOL	LOVROK	LCALDA	-
KSTNM	KEVNM	-	-	-
KHOLE	KO	KA	-	-
KT0	KT1	KT2	-	-
KT3	KT4	KT5	-	-
KT6	KT7	KT8	-	-
KT9	KF	KUSER0	-	-
KUSER1	KUSER2	KCMPNM	-	-
KNETWK	KDATRD	KINST	-	-
PAY1	PAY2	PAY3	PAY4	PAY5
PAY6	PAY7	PAY8	PAY9	PAY10
PAY11	PAY12	PAY13	PAY14	PAY15
PAY16	PAY17	PAY18	PAY19	PAY20
PAY21	PAY22	PAY23	PAY24	PAY25
-	-	*etc...*	-	-

Table 3: Table illustrating the standard SAC ASCII file header formatting and payload arrangement. Note the sections in **bold**. (IRIS, 2013)

of a 300-second recording, with the peak of the primary event occurring at +10 seconds. Noise data from the eruption has not been the focus of prior research, and so a dataset of noise recordings comparable to the event dataset does not exist. Given the scope of the project and time available, it is not realistically possible to sample every recording manually. Instead, approaches to systematic sampling of the noise dataset were explored, with a few specimen noise samples selected from the whole-day recordings *(see Fig. 4)* for 'ground-truthing' of any sampling method.

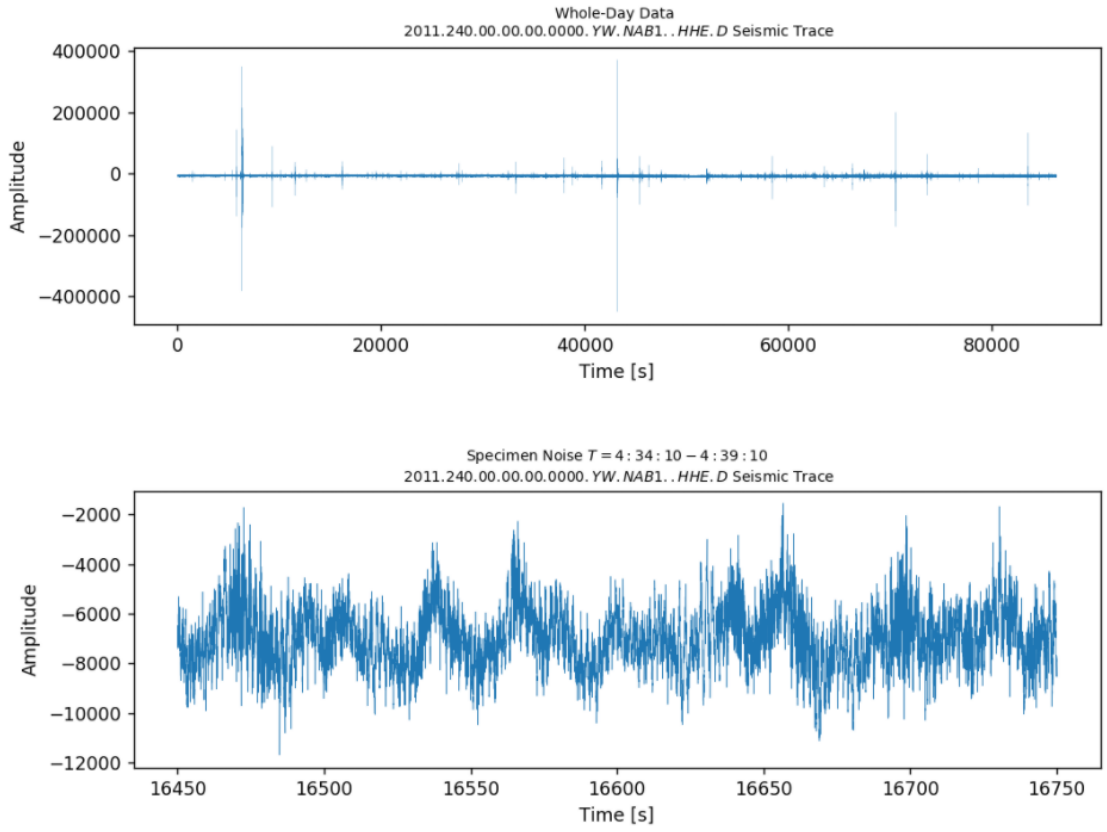


Figure 4: Specimen Data of Whole-Day recording, *reference: 2011.240.00.00.00.0000. YW.NAB1..HHE.D* from the whole-day dataset for the day 240 (*28th August*) against a manually picked noise specimen found at $T4:34:10-4:39:10$. Although the noise waveform does show some underlying very low frequency oscillations (*peaks appear at 30-60 second periods* $= frequency^{-1} = 1-2Hz$), the y-scale change in amplitude is minuscule in comparison to positive event patterns

One option is to use a rolling mean at a window size of 300 seconds. The window size should match the event recording length to have a comparable usable frequency floor as the sampling rate and recording length control the upper and lower frequency limits achievable without aliasing (distortion and misrepresentation of frequency amplitudes) *(see Section 3.5.2*

and Eq. 5) (H. Nyquist, 1928) (C. E. Shannon, 1948). It is also good design to keep input data features as homogeneous as possible — normally negligible variations caused by two disparate window sizes may result in overfitting.

In practice, this proves to be ineffective at differentiating between windows with and without events as significant smoothing does not occur at a 300-second window size, making a positive identification of a window without events difficult (*see Fig. 5*). Also, the average signal amplitude varies between more proximal and more distal seismometer sites - a neural network is highly sensitive to any form of bias, and correcting for the signal floor may distort low amplitude features. As windows containing events are expected to be more chaotic and with a greater variance between average and maximum amplitude, rolling standard deviation is an alternative to the rolling mean. Using standard deviation provides better selection and control, as seen in *Fig 5* where there is a well-expressed discrimination between windows containing events and those just containing noise.

Given the clear demarcation showing with Standard Deviation, a cut-off value can be specified to label windows into noise- or event-containing windows(*see Fig. 6*). Noise-containing windows will then be processed into a noise auto-picked dataset to complement the event dataset provided, with the audio from the window forming each noise recording.

3.3.6 Kaiser Windowing

Although the Event and Noise datasets can be run through a Fourier Transform, the signal clarity benefits from a degree of filtering, typically a form of *apodization* where the foot of the waveform is removed to filter noise and to smooth transitions between beginnings and ends of samples (Marshall and Stutz, 2012). There are several filtering methods, but the *Kaiser* function has an added benefit of a variable filter curve through a changable '$\beta - max$' value (see α in section 3.5.1), allowing it to mimic the other typical filtering methods (*see Table 4, Fig. 7a, 7b*); provide a custom filter; and allow calibration where needed.

The standard $\beta - max = 14$ curve has been used and although high model accuracy (see section 4 for results) has meant that $\beta - max$ tuning wasn't needed, reuse of the code for new datasets may need further calibration of $\beta - max$ to better filter noise. Where the window size is significantly smaller, the effect that apodization has on the signal to noise ratio is more

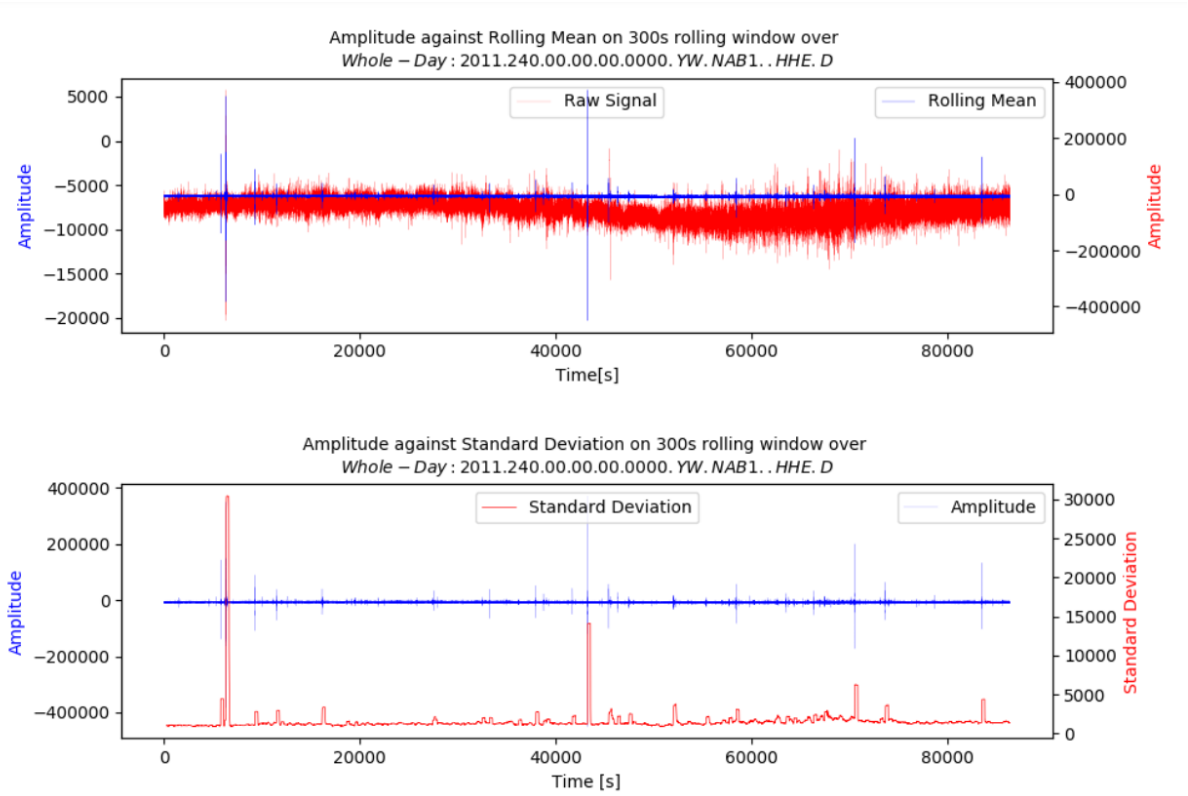


Figure 5: *Upper:* Specimen Data from Whole-Day recording, *2011.240.00.00.00.0000. YW.NAB1..HHE.D* from the whole-day dataset for the day 240 (*28th August*) in blue, against a rolling mean at a 300s window size in red. Although there is some correlation with 150s offset spikes between both datasets, there is no clear pattern to positively select noise events. *Lower:* Standard deviation in the same window against signal, showing clear event demarcation for positive noise window selection.

noticeable and more sensitive to tuning — the DFFT functions by slicing a signal by every possible frequency periodicity and repeating a signal fragment to better and more quickly resolve the frequency amplitude — as the smaller windows would have a greater ratio of artefacts to signal. (Cooley and Tukey, 1965)

Filter	Equivalent $\beta - max$ value
Rectangular	0
Hanning	5
Hamming	6
Blackman	8.6

Table 4: Table listing common signal filters and an equivalent Kaiser $\beta - max$ value

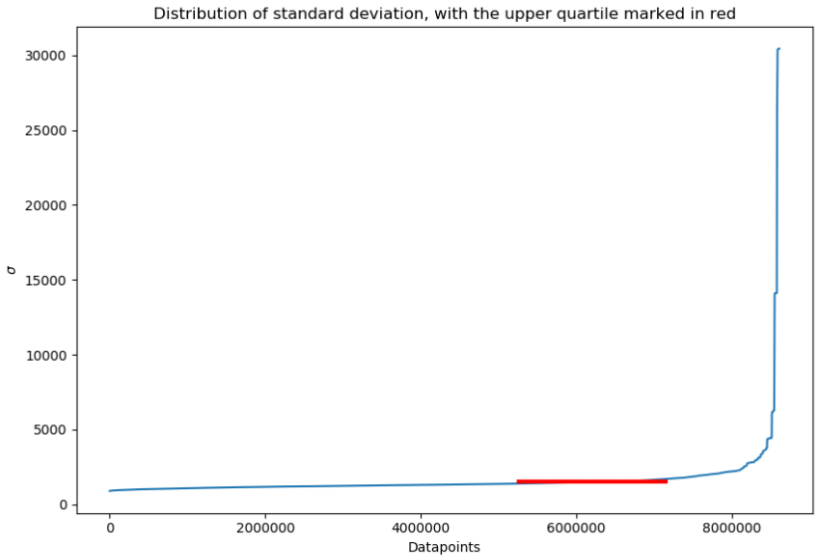


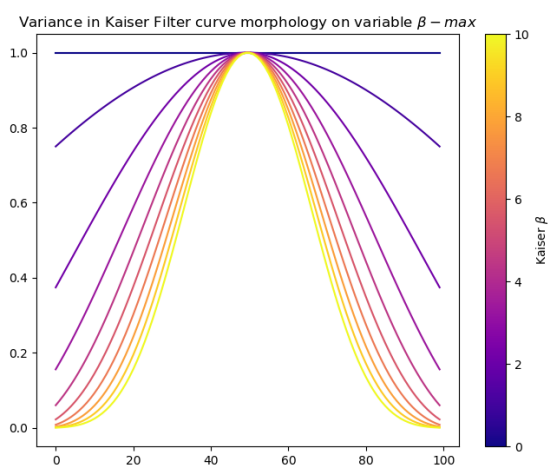
Figure 6: Distribution of standard deviation, showing the sharp variation between noise and events, and a sparse event occurrence. An arbitrary upper-quartile (shown by the red line threshold has been chosen as the cut-off value.

3.3.7 Fourier Transform

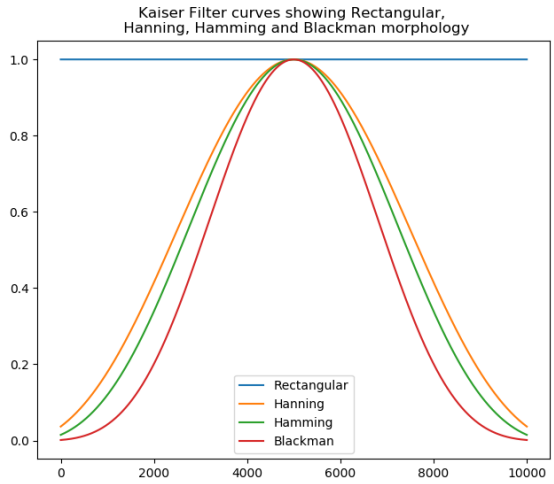
For the Fourier Transform calculation, the SciPy library (a common python library with various signal processing functions, *see section 3.3.1*) is used. Processing an event recording (*Fig. 8a*) using *fftpack.fft()*[2] provides a basic output, shown in Figure *8b*. Although the y-axis data is valid, the move from time-domain to frequency-domain results in a meaningless x-axis. In addition, the transformation has produced a symmetrical image in the plot. Inferring the sample frequencies with *fftpack.fftfreq()*[3] grounds the data in the frequency domain (*Fig. 8c*), bounded at 50Hz and -50Hz, the Nyquist Limit (*see Section 3.5.2, (H. Nyquist, 1928) (C. E. Shannon, 1948)*) showing the additional symmetrical component as the negative frequency mirror. This is to be expected as the signal is real — only complex signals are expected to show variation in the negative Hz range. As the imaginary negative frequency components are identical, they can be trimmed to show the resulting plot in Figure *8d*.

With the Fourier Transform convolutions summarised, the Kaiser filter function is applied. Overall signal morphology is maintained but cleaned, with an overall reduction in signal amplitude as a result of the apodization (*8e*). Finally, the signals are normalised by translating them

[2]fft — Fast Fourier Transform
[3]fftfreq — Fast Fourier Transform Sample Frequencies



(a) Kaiser Curve Morphology with Varying $\beta - max$



(b) Equivalent Kaiser Curve Morphology (see table 4)

Figure 7: *Sub-figure a)* showing the change in Kaiser Morphology when $\beta - max$ is changed, used to fine-tune the filter; *Sub-figure b)* showing the equivalent morphologies for the alternative filtering methods listed in Table 4

to a decimal fraction of the maximum amplitude observed in each recording, which is then converted to decibels to better express the upper (35Hz upwards) range (*See Eq. 1*). In addition to a better expression of the higher-range frequency components, normalisation removes signal volume bias, and forces every recording to a similar scale of magnitude, which is another vital preprocessing step to improve neural network accuracy.

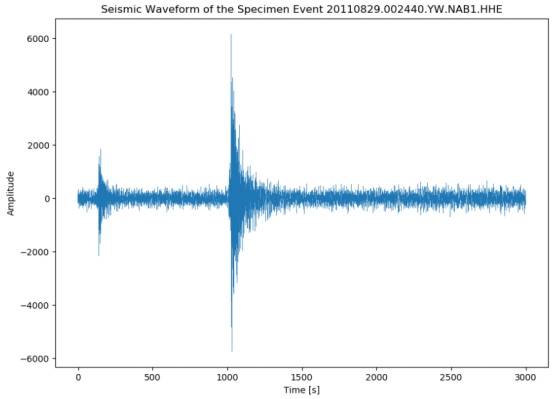
$$Decibels = 20 \times \log_{10}\left(\frac{Amplitude}{Maximum\ Amplitude}\right) \tag{1}$$

For a mathematical overview of Fourier and Kaiser filtering steps, please refer to section 3.5.3.

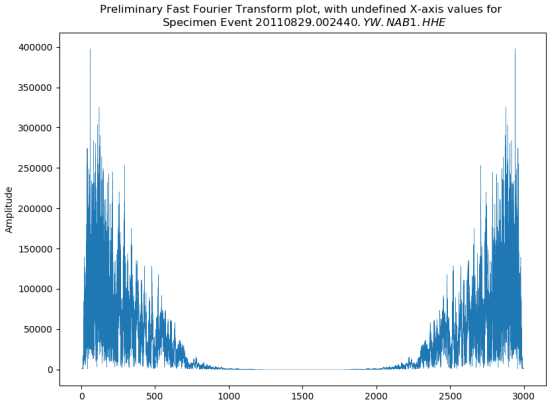
3.3.8 Noise Downsampling

In section 3.3.5, the sampling can select any window that meets the threshold. Processing all 46 whole-day recording produces somewhere in the region of 46650 noise samples. As we need an equal number of event and noise datapoints, we will need only 2450, as providing a significantly greater number would result in significant bias in the resulting model and loss in accuracy.

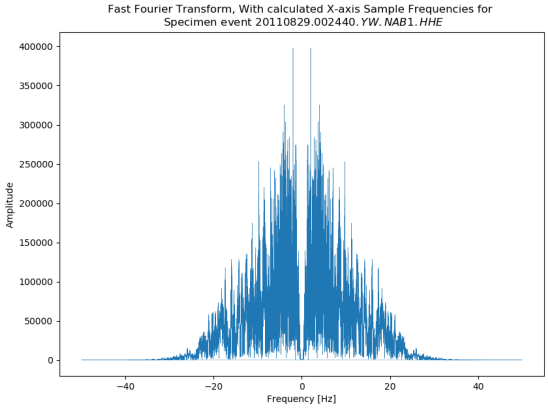
Where a random guess in a binary classification might have a likelihood of $p = 0.5$ of



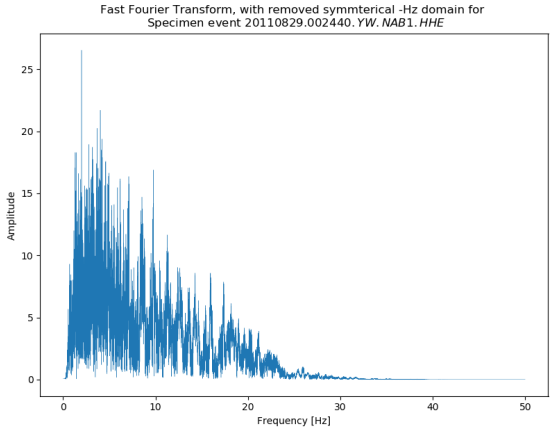
(a) Original Waveform



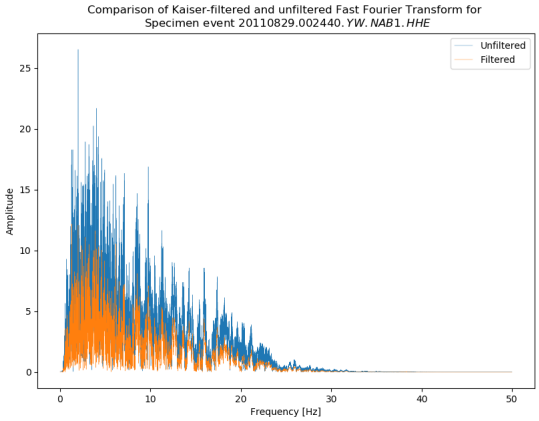
(b) Initial *scipy.fftpack.fft()* output



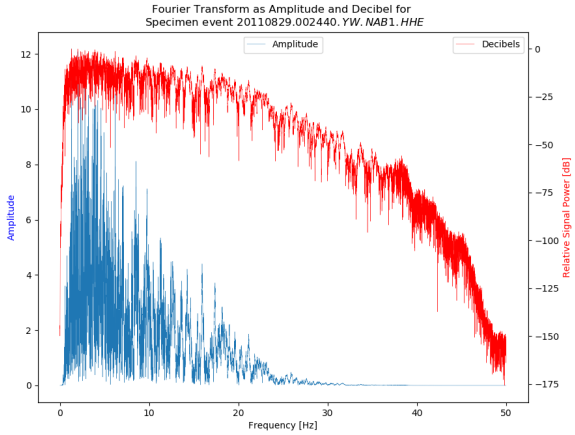
(c) With sample frequencies inferred from *scipy.fftpack.fftfeq*



(d) Symmetrical Alias removed



(e) Passed through a Kaiser Filter



(f) Normalised into a Decibel scale of $-20\log_{10}\left(\frac{Amplitude}{MaxAmplitude}\right)$

Figure 8: Intermediate Processing Stages for Fourier Transformation

correct labelling, so a poorly performing neural network would maintain its accuracy at this level throughout all layers and epochs. Any improvement above this baseline would signify a better-than-random model. If there is a significant asymmetry in test sets — for example, a noise:event %90:%10 distribution — where a simple random guess would maintain a $p = 0.5$ performance, a poorly fit neural model would learn to guess "noise" every time and achieve a comfortable $p = 0.9$ performance in test data, but a very poor performance in validation, and a useless model. This is a simple form of overfitting.

To avoid this issue, we randomly sample 2450 samples from all noise recordings using the *numpy.choice.random* sub-module, selecting by the index number to produce a reduced dataset in a single dataframe, ready for neural network ingestion.

3.3.9 Frequency Binning

Although it is entirely possible to feed in every data-point as an attribute, the performance demands of running a 5000-row, 50000-attribute dataset is considerable. For example, one hidden layer in a Neural Network that evaluates the weighting of any two of n attributes in conjunction would require x rows $\times\, n(n-1) \div 2$ operations, (effectively, an $O = [N^2]$ order of operations) is exponentially sensitive to an increase in attributes. To reduce the number of computations needed, attributes can be binned, reducing the overall size of the data fed into the model. This reduces the granularity of the data provided to the CNN, therefore a compromise between the data granularity and processing demands needs to be found. Ultimately, 200 equally spaced bins of 0.25Hz width from zero to 50 Hz were found to be a good compromise (*see Jupyter code notebooks and fig 9*).

3.4 Convolutional Neural Network

The design of a Convolutional network revolves around several variables and hyperparameters that change during model evolution, from a simplistic baseline model — which represents a basic network — that is used to gauge the effectiveness of model adaption, to more complex iterations of the neural network. Specifically, these variables and hyperparameters are:

- Input Dimensions

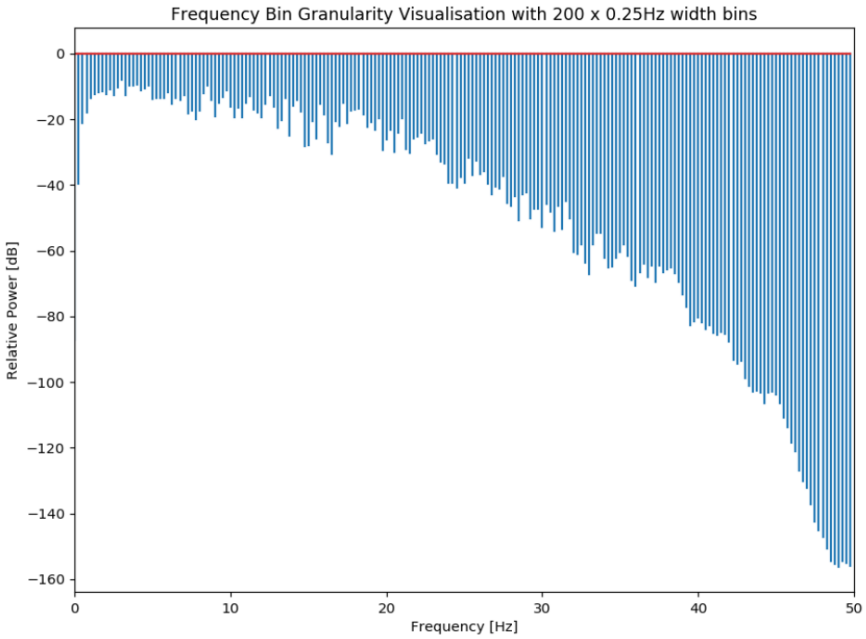


Figure 9: Representative graph of the binned frequency data resolution for event $20110829.002440.YW.NAB1.HHE$

- Output Dimensions
- Number of Hidden Layers
- Layer Types
- Activation Functions
- Number of Epochs
- Batch Size
- Validation Methodology
- Learning Rate
- Degree of Dimension Bottle-necking

We will explore these in the following sections.

3.4.1 Dimensions and Layers

All attribute data within the cNN is treated as mathematical tensors — higher-dimensional mathematical matrix objects. (Schouten, 1989) (*See section 3.5.4*) The behaviour and manipulation of tensors depends on the design of the cNN. Our objective — the discrimination between events and noise — is a form of binary classification, which is simpler use case. Given

the discrete variation between the frequency component profiles of noise and event recordings, a simple 3-layer baseline model was used, as shown in *fig 10*.

The baseline model consists of an input layer of 202 dimensions - one for each of the attributes (*200 frequency bins, an overall mean and median*), a hidden layer of the same dimensionality comparing the relative weighting of each of the input neurons, and a summation to a one-dimensional output: whether a prediction is noise, or an event. More complex topological designs can be used if this arrangement delivers poor model performance, but are usually avoided for baseline models.

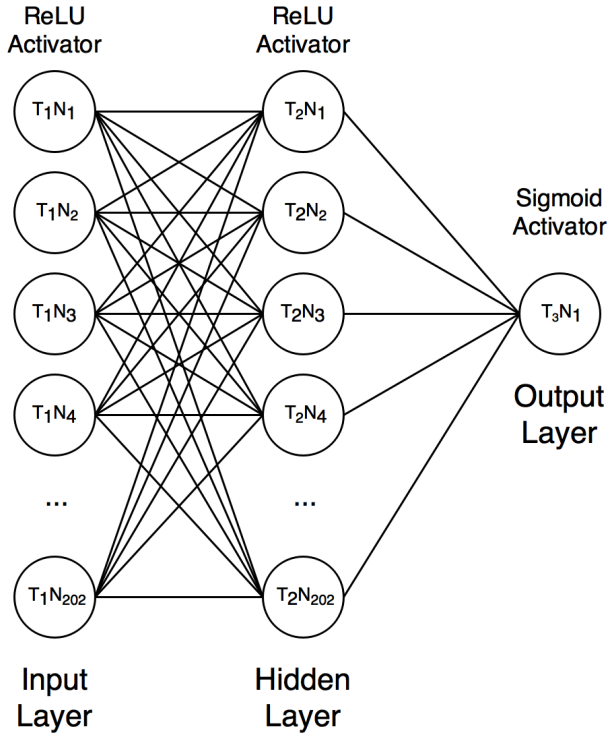


Figure 10: Schematic Topology of the initial "baseline" dense-point convolutional Neural Network

Hidden layers are where the majority of the processing occurs, without any major user control beyond parameter calibration - hence the term "deep" learning. A deep neural network requires at least one hidden layer, so for a baseline model, a single hidden layer is used.

3.4.2 Layer Types

There are several ways to link between layers in cNN models. The classic fully-linked mesh between neurons in two layers is referred to as a **dense** layer, but other forms of layers may also be used. To help tackle overfitting by reliance on only a few strong attributes, **dropout** layers

can be used to force a random selection of data-points to 0, forcing redundancy in the model, significantly demonstrated in Srivastava et al. (2014) (*see also Hinton et al. (2012)*). Where "bottlenecking" (reduction in dimensionality, *see section 3.4.7*) occurs, pooling layers are used to summate *pools* — smaller clusters of neurons — into a single neuron in the following layer. In computer vision, the maximum value in the pool is typically selected as a **maxpool** layer, demonstrated well in Krizhevsky et al. (2012), an early and notable winner of the ImageNet challenge; even for this project's use-case, max-pooling would offer the best performance as it can select the most strongly expressed patterns across multiple neurons. Although max-pooling is the most common version of pooling, any valid summarisation can be used (*e.g. mean, minimum, sum*).

The baseline model will only consist of dense layers, however if the model would need to grow more complex, then bottlenecking with pooling (to compensate for greater overfitting risk as a result of an increase in model size) would be included in the model design.

3.4.3 Neuron Activation Functions

As important as the types of layers are the weights and values passed between them. These can be controlled through the use of activators: functions that define the output of a neuron. In terms of a biological parallel, a true neuron can receive an impulse propagating from a pre-synaptic axon which it can choose to propagate onwards as an action potential if it satisfies the threshold potential of the cell, acting as a binary function — [*propagate—arrest*] (Hodgkin and Huxley, 1952). This is what is known as a linear activation function, and early neural networks used electronic switches at weighted thresholds to propagate or terminate electric signals. However this severely limits the flexibility and scope of a neural network model (Wu, 2009). Non-linear functions allow a gradient-based approach to weighting, with last-generational neural network models using *tanh* and *sigmoid* activations heavily. (Snyman, 2005)

More recent models have started using the Rectified Linear Unit (**ReLU**) function, which outperform the incumbent standard *tanh* function, with a significant decrease in epochs needed to to reach the same error rate, as seen in fig 11. The baseline model will therefore use a combination of ReLU and Sigmoid functions.

ReLU does tend towards fragility when passed extremely strong outliers with high learning

Activation	Function
Linear/Binary Step	$f(x) = \binom{0 for x<\theta}{0 for x \geqq \theta} where\ \theta = threshold$
Tanh	$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(1+e^{-x})}$
Sigmoid	$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$
ReLU	$f(x) = \binom{x for x<0}{0 for x \geq 0}$

Table 5: Table listing applicable common Activation Functions *(see section 3.5.4 for a more extensive exploration)*

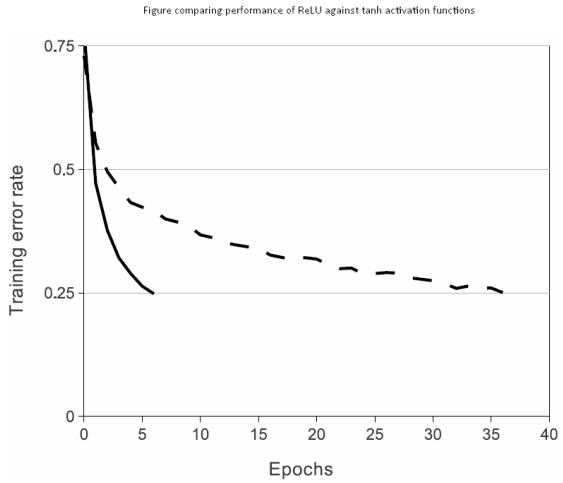


Figure 11: Figure demonstrating improved model performance using ReLU (solid line) against tanh (dashed) activation functions. ReLU achieves a 25% error rate within 6 epochs — six times faster than tanh in the **CIFAR-10** model in Krizhevsky et al. (2012), from which this figure is reproduced

rates (*see section 3.4.6 for tuning learning rates*) as the combination of a numerically larger variation with an outlier and higher weighting change can 'jam' the ReLU threshold so high or so low that it is permanently open or closed — referred to as a 'dying' ReLU neuron (*see section 3.4.4 for more detail*). These can be corrected for with ReLU variants, such as the 'leaky' ReLU. (Sutskever and Hinton, 2010)(Wu, 2009) Given the scope of the project, standard ReLU functions are sufficient, but further model improvement and expansion would include introducing leaky activator functions.

3.4.4 Number of Epochs and Batch Size

A cNN runs through a series of common steps for every iteration that is run. For each layer, data is ingested, passed forward to the layer, and from the data a series of weights and gradients

are calculated. The micro-parameters are then passed-back or *back-propagated* to re-evaluate training data that has been already processed. This occurs for every iterative batch of the training set, and for every repetition of the training set (or *Epoch*).

Batch Size On first assumptions, an entire training dataset should be passed through in one go to minimise the number of back-propagation calculations required. However, it is likely that the scale of data needed for Neural Network training means that the amount of RAM needed is prohibitively large, or that the dataset isn't large enough. To address this, data can be released in batches of training data with back-propagation applied after each batch.

The size of the batch can also dictate the sensitivity of the model to individual training examples: a strong outlier would be more strongly expressed in a smaller batch, resulting in a stronger change back-propagated which may compound itself if the outlier is strong enough, especially in ReLU neurons. This results in neurons so skewed that they are either forced open or closed, referred to as "dying ReLU" neurons. (Chawla and Wang, 2017)(He, 2017) Conversely, where signals are more subtle against noise, a smaller batch size may result in a a better sensitivity and therefore accuracy. There is a trade-off between more sensitive and more robust batch sizes (with better processing cost) which is dependent on the level of pre-processing, the data domain, and the learning rate. A lower rate of $lr = 0.01$ was chosen for the baseline model.

Epochs As modern Neural Networks use gradient descent optimisation for weighting calculations, the entire training set is typically applied repeatedly as far as possible. Although repeated application can improve the performance of a cNN, it also increases the likelihood of overfitting as the cNN overspecialises on non-significant features of the dataset. Standard practice is to separate the dataset into a large training set, and a small validation set that isn't used to train the data, but is kept hidden and tested against later. The next step is to run a number of Epochs and to monitor the trend in both the training set's, and the validation set's accuracy and degree of loss. While the cNN specialises on significant patterns that are general, rather than specific to only the test set, then the accuracy of the training and the validation set should increase together. Once the cNN begins to overfit on the training data after several epochs, then it should further improve in the training set's accuracy, but would perform worse on the

validation set. By finding the maximum number of epochs prior to a consistent deviation in Training and Validation accuracy and loss, we can determine the best performing number of epochs for the cNN.

3.4.5 Validation Methodology

To validate the accuracy of the cNN, normal practice is to segregate a fraction of the data to be used as validation data, and a fraction to be used as training data. As the model 'sees' the training data during training and fits its model around it, it cannot be used to measure the effectivity of the cNN on a generic dataset. Instead, the validation data (which has never been seen before by the model) is used. The process of validation does not ingest the validation data into the model, and does not change any of the weighting in the neural network. It only runs alongside the Training measure to see how accurate it really is. Typically a ratio of 70% training data to 30% validation data, or 80% to 20% is used.

Comparing the accuracy of *both* sets of data is important for detecting overfitting. The classic sign is that training accuracy continues to improve through progressive epochs as the model begins to overspecialise, finding trivial patterns in the data endemic to the training set while the validation accuracy decreases as decisions based on these trivial patterns begins to impinge on the classification effectivity. In short, *where training and and validation accuracy consistently diverge, overfitting has occurred*.

However, the segregation of data does lead to an increased risk of overfitting as the subset may contain patterns or features that by chance do not occur in validation set, and that smaller training sets are inherently more at risk to overfitting. Conversely, a smaller validation sets can result in less representative validation accuracy. To address both issues, *k-fold* validation is used, a method that randomly splits the entire testing set into k number of sets, and then k number of independent isolated neural networks in parallel with $k - 1$ sets as training data, and the last set as a validation dataset. This way, every piece of data is, in some form or combination, ingested into the model, and every piece of data is also used to validate the data in the model. K-fold validation provides a better validation accuracy for a more representative performance, and the model itself becomes more resilient to overfitting.

In the baseline model, 10-fold validation was used, with 10% data used for validation in

each case. Given the simplicity of the model this is computationally possible, even though it increases the processing time ten-fold. On more complex or even larger datasets, the number of folds is typically reduced as the processing cost involved is prohibitively high.

3.4.6 Learning Rate

A further calibrable model hyperparameter, the learning rate is the rate at which weighting changes for each dataset. High learning rates can 'overshoot', overwriting old valid trained features when a small set of outliers are encountered, instead defining the classification on the outlier features. Low learning rates may not respond sufficiently to features in a minority of valid datasets, resulting in poorer accuracy. In addition, the number of epochs needed to train the model increase significantly with a low rate. There is another trade-off between a fast and flexible, and a slow and reliable model, usually leaning closer to lower learning rates (Dangeti, 2017). See Fig. 12.

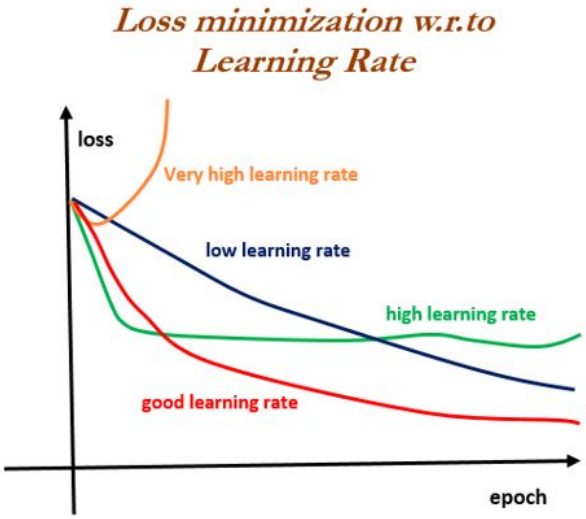


Figure 12: Effects of learning rates on loss, *reproduced from Dangeti (2017)*

3.4.7 Degree of Dimension Bottle-necking

Although larger models may look to "bottlenecking" to lower dimensionality layers (*see fig 13*) to improve performance and explore higher-level abstraction, the computational cost of our baseline model is not so complex as to require bottlenecking for the purpose of reducing processing costs. Later topological designs may include bottlenecking, although it is more common for computer vision applications, in combination with pooling layers.

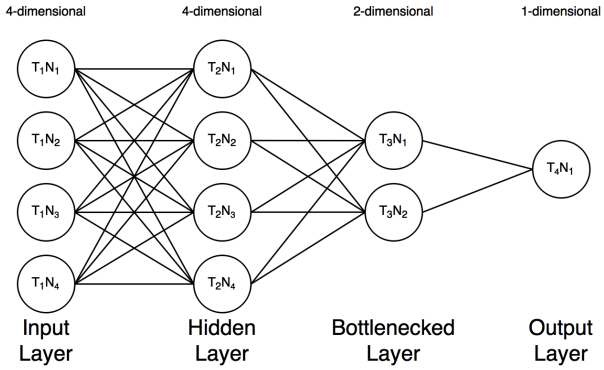


Figure 13: Representative Schematic Topology of a "bottlenecked" convolutional Neural Network

3.5 Mathematical Overview

3.5.1 Kaiser Windowing

A kaiser windowing filter function is a Bessel function used in signal processing.

$$\omega[n] = \begin{cases} \frac{I_0\left[\alpha\sqrt{1-\left(\frac{2n}{N-1}-1\right)^2}\right]}{I_0[0]}, & 0 \leq n \leq N-1, \\ 0, & otherwise, \end{cases} \tag{2}$$

where I_0 is a zero$^{\text{th}}$-order bessel function, N is the window duration and α is the non-negative real '$\beta - max$' value. (Kaiser and Schafer, 1980)

3.5.2 Nyquist's Theorem

Nyquist's Theorem focuses on identifying the range of frequencies that can be resolved without risk of aliasing, given the sampling frequency. Files are recorded at a 0.01-second interval (this can be found in the header information for the recordings) - the sampling frequency is, therefore:

$$Sampling\ Frequency = \frac{1}{Sample\ Interval\ (0.01seconds)} = 100Hz \tag{3}$$

The sample rate (*Eq.3*) controls the amount of information we can draw from the signal. In particular, it controls the frequency ceiling before we start experiencing aliasing - known as the Nyquist Limit. We can also calculate the lower frequency limit, as a waveform must appear at least twice within a window of time to be identified:

$$Nyquist\ Limit = \frac{Sample\ Frequency}{2} \tag{4}$$

$$Signal\ Floor = \left(\frac{Window\ Duration}{2}\right)^{-1} \tag{5}$$

(H. Nyquist, 1928) (C. E. Shannon, 1948)

$$\begin{aligned} &Applying\ (3)\ to\ (4):\\ &Nyquist\ Limit = \frac{100Hz}{2} = 50Hz\ ceiling \end{aligned} \tag{6}$$

$$\begin{aligned} &(5)\ for\ a\ 300\ second\ window:\\ &Signal\ Floor = \left(\frac{300}{2}\right)^{-1} = \frac{1}{150}\ or\ 0.00\overline{6}Hz \end{aligned} \tag{7}$$

We cannot meaningfully find frequencies outside $\frac{1}{150}$Hz to 50Hz. This will limit the range of frequencies we can use to analyse the seismicity.

3.5.3 Fourier Transform

The Fourier Transform is a method to transform a signal within the time domain to its component frequencies and forms the cornerstone of the data preparation prior to ANN processing.

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi}\, dx, \tag{8}$$

$\hat{f}$, the Fourier transform of f in respect to time x gives us the component frequencies ξ of a given signal (*Eq.8*). This may be inversed by changing the polarity and direction, as in *Eq.9* to find the inferred signal from the frequency components. Note that the frequency is typically normalised as $\frac{radians}{sample}$ with the periodicity of 2π (seen in (*Eq.8, Eq.9*).

Throughout the paper and documentation, the signal, and any temporal data will be referred to as *in the time domain*, with the extracted frequencies as in the *frequency domain*, as the signal components have no temporal dimension.

$$\hat{f}(x) = \int_{-\infty}^{\infty} f(\xi) e^{2\pi i x \xi} \, dx, \tag{9}$$

It should be noted that rather than the continuous forms of the Fourier transform in *Eq.8, Eq.9,* the computationally cheaper *Discrete Fast* Fourier Transform (**DFFT**) is used (*Eq.10*) — common practice for computational methods. The DFFT does assume that temporal sampling is continuous and uniform. The seismometer readings meet both requirements. Molnár et al. (2005) describes briefly an application of the short-time Fourier Transform, a variant of the DFFT suited for non-continual signals, and although focusing on muzzle blast detection, does offer some additional signal processing techniques to explore in further research.

$$X_{2\pi}(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-i\omega n} \tag{10}$$

The *discrete-time Fourier Transform*, (*Eq.10*) finds the periodic function of a component of a set of discrete samples $X[n]$, for all integers n. The frequency ω is again expressed in $\frac{radians}{sample}$. The DFFT is a variant using the *Cooley-Tukey* algorithm, (Cooley and Tukey, 1965) to improve computational efficiency by breaking a DTFT into many smaller DTFTs — A $N \times N$ matrix would classically require N^2 operations, but the *Cooley-Tukey* algorithm instead requires $N \log N$ operations.

3.5.4 Neural Networks

Tensors All attributes handled in a Neural Network are abstracted down to mathematical objects of any given dimensionality. These *tensors*, as they are formally referred to, are defined as the generalisation of vectors and scalars - vectors are 1st rank tensors with 2 dimensions of information - magnitude and direction. Likewise, Scalars, with only 1 dimension of information (magnitude) are considered 0th rank tensors. (Schouten, 1989) (Feynman et al., 1965) Given the 200 frequency bin and mean and median columns, the system consists of 201st rank tensors.

Loss The pivotal function that drives any neural network, loss is the numerical representation of the optimisation of the network. Every change in weighting is driven to decrease the loss down a gradient - the basis for *gradient descent* methodology. Loss in cNNs is strictly the *Zero-One loss function* (*see Eq. 11, 12*).

$$L(i,j) = \begin{cases} 0, \ where\ i = j, \\ 1, \ where\ i \neq j, \end{cases} \\ i,j \in M \tag{11}$$

where M is the data set and i, j are binary classifiers

$$L(\hat{y}, y) = I(\hat{y} \neq y) \tag{12}$$

where I is the indicator *or predicate* function

Activation Functions A neuron receives input vectors x and weighting w from one or many preceding neurons, as a sum input u for n inputs, as seen in Eq. 13.

$$u = \sum_{i=1}^{n} w_i x_i \tag{13}$$

(Anthony, 2001)(Dangeti, 2017)

Expanding on Eq. 13, the sum input u is then processed by an activation function φ to provide an output y, as shown in Eq. 14 in terms of the output for the k^{th} neuron.

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right) \tag{14}$$

where there are m inputs

(Anthony, 2001)(Dangeti, 2017)

Expanding on section 3.4.3, and table 5, φ can be varied according to the function of the neuron and the layer. The two activation functions used in the baseline model are the *Rectified*

Linear Unit (**ReLU**) and *Sigmoid* functions (*Eq. 15, 16*). ReLU is used to propagate a tensor onwards, zeroing any negative values and preserving weighted positive inputs. A Sigmoid function fits an input to a value within 0 — 1, fitting strongly to either extreme. Effectively a more smoothed step function, (such as the Linear Activation Function 17), it is used as a final-stage neuron for binary classification where a Boolean [0 | 1] output is expected, or any intermediary-stage end neuron where a Boolean output is expected.

$$f(x) = \begin{cases} x, for x < 0, \\ 0, for x \geq 0, \end{cases} \tag{15}$$

Commonly written as $x = max(0, x)$

(Nair and Hinton, 2010)

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{16}$$

(Chow and Cho, 2007)

$$f(x) = \begin{cases} 0, for x < \theta, \\ 1, for x \geq \theta, \end{cases} \tag{17}$$

$$where\, \theta = \ threshold$$

Chow and Cho (2007)

It should be noted that ReLU has only been adopted widely within the last few years, having been proven to outperform the standard *tanh* activation (Eq. 18) with fewer epochs needed to achieve the same accuracy. (Krizhevsky et al., 2012) (Nair and Hinton, 2010)

$$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(1 + e^{-x})} \tag{18}$$

Chow and Cho (2007)

4 Summary of Results

The results are split into two sections: the initial baseline model and the later improved model.

Data preparation is the same in both examples, with the following parameter settings:

- Window size of 300 seconds
- Kaiser function of $\beta_{max} = 14$
- Frequency bin width of 0.25Hz
- Noise selection threshold at the upper quartile of distribution of standard deviation
- Normalisation function of $20 \log_{10} \left(\frac{Amplitude}{Maximum\ Amplitude} \right)$

4.1 Baseline Model

The Baseline model design is shown in Figure 14. The baseline model was run with the following parameters:

- 3 Layers
- 202—202—1 Topology
- Learning rate = 0.01
- 10-fold validation
- ReLU - Sigmoid Activation

The cNN was run for approximately two hours on a CUDA-enabled NVIDIA GeForce GTX 760 GPU, achieving a best accuracy of 65%. Figure 15 shows training and validation accuracy and loss over the first five epochs. The cNN reached maximum accuracy in three epochs and minimum loss in four epochs.

4.2 Final Model

The Final model design is identical in topology to the baseline model in Figure 14. The model was run with the following parameters:

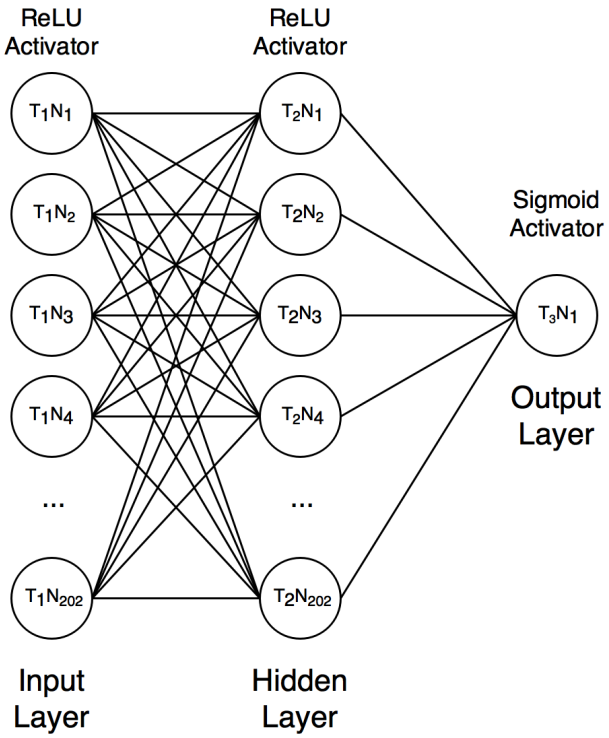


Figure 14: Schematic Topology of the initial "baseline" dense-point convolutional Neural Network. The final model shares the same topology.

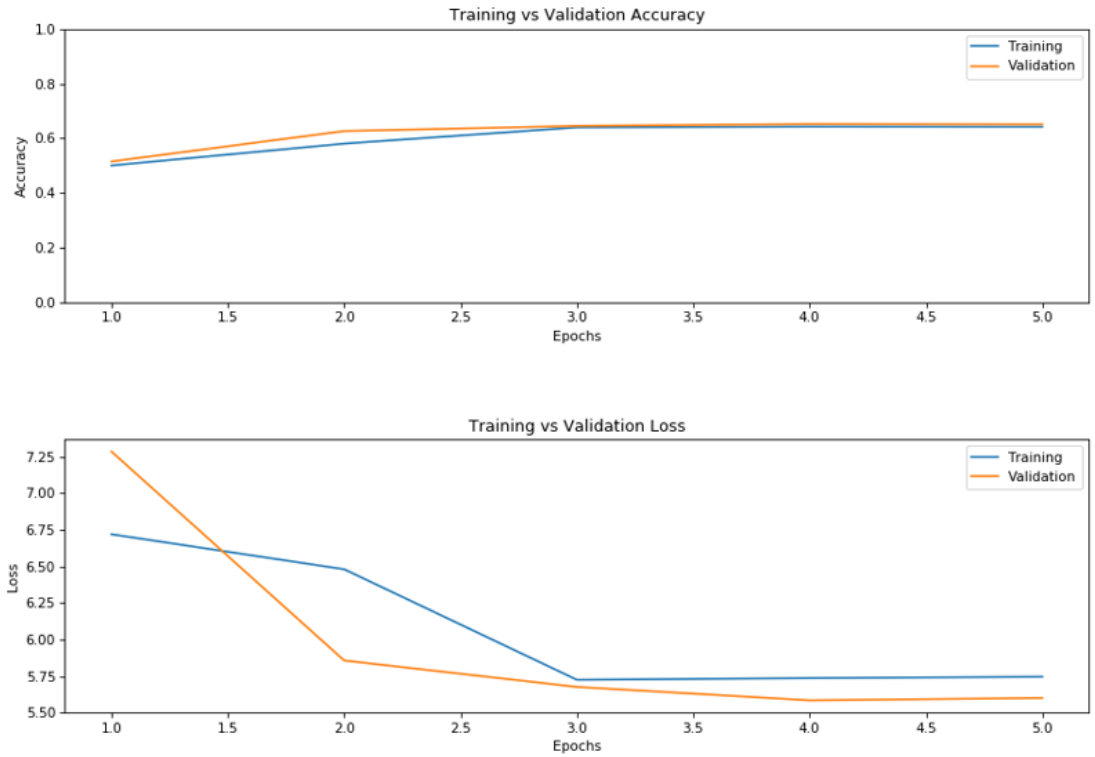


Figure 15: Mean training and validation accuracy and loss of the 10-fold validation for the first five epochs of the baseline model

- 3 Layers
- 202—202—1 Topology
- Learning rate = 0.01
- 10-fold validation
- ReLU - Sigmoid Activation
- **StandardScaler() Pipeline function**

To improve on the baseline model, further inter-layer standardisation was applied through a *scikit.learn StandardScaler()* pipeline function, which attempts to fit the dataset at each layer back to a Gaussian distribution with a 0 mean and unit variance. The cNN was run for approximately two hours on a CUDA-enabled NVIDIA GeForce GTX 760 GPU, achieving a best accuracy of 100%. Figure 16 shows training and validation accuracy and loss over the first five epochs. The cNN reached maximum accuracy in two epochs and minimum loss in three epochs.

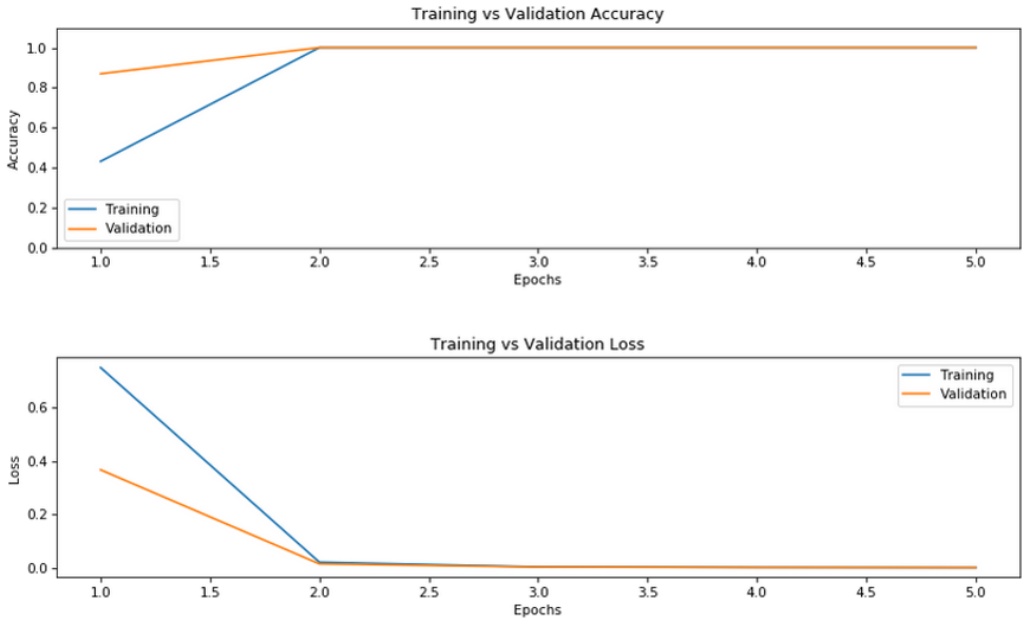


Figure 16: Mean training and validation accuracy and loss of the 10-fold validation for the first five epochs of the final model

5 Discussion

5.1 Aims

The research aims of this project were to:

1. Analyse seismic data in a volcanic eruptive setting, with the intention of achieving a binary classification of intervals of seismic data as either discrete events or noise.
2. Demonstrate the use of a Convolutional Neural Network (CNN) as an effective and novel method in processing and classifying seismic datasets to near-human accuracy for bulk data.
3. Identify weaknesses in using seismic data in Machine Learning applications, mitigating issues where possible through thorough data preparation.

The results presented in section 4 satisfy aims 1 and 2. The model was able to successfully digest, analyse and provide classification of events and noise, when given seismic data. The model was able to do so through effective use of a convolutional Neural Network, and did so with a very high level of accuracy.

100% accuracy is an anomalously high accuracy level. One explanation of this level of performance is that the model experienced overfitting — that the model over-generalised non-relevant features endemic to only the training set, rather than features important to wider application of the model. Although this cannot be fully dismissed as the model was based on data from one setting, there is no evidence of overfitting in Figures 15 and 16 (overfitting would be expressed as a trend of deviation of validation accuracy and loss from training accuracy and loss over successive epochs). Even in Figure 15, where the training set loss seems to move away from the validation set in epoch 2, both sets still show a trend in decreasing loss.

Even without any evidence of overfitting, further work is still needed in expanding the dataset to include different settings, eruptive behaviours, geologies and seismometer emplacements to further generalise the model. This would reduce the risk of overfitting as the model develops.

100% accuracy does not necessarily mean that the model can spot every single event within a set of seismic data. The data it trained on was hand-picked, and so the accuracy, consistency and bias of the human picker are inherited in the model. All that the model can achieve is

to mimic the performance of the human expert picker as closely as possible. If data from a multitude of different experts was collected and used to train a model, it is reasonable to expect the biases from individual pickers to be reduced and the different strengths of different experts absorbed into the model to the point where it may achieve a better performance than any one one of the constituent pickers. Therefore, another expansion to the research might be to include additional data from other researchers.

The biggest weakness of the model is that, given the time constraints and author expertise, the noise dataset was not manually picked. Ideally, the noise data should be manually picked by the same picker as the event data, however this was not possible. Nor was the author able to pick out 2450 300-second noise windows given the timeframe of the work, and a lack of ability. Instead, noise data was systematically selected, with a care to select only the stronger candidates for noise windows. This type of function can perform well when given more extreme parameters to select by, so it is unlikely that any events were selected. As a significant buffer was ensured between where the events were observed to start, and where the threshold was placed in Figure 6, efforts were made to avoid any ambiguous weak events that may have been on the lower boundary of the observed spike, where a human picker might perform better.

The data preparation (Section 3.3) was decided on with the express aim of mitigating the weaknesses of ANNs as analysis tools (aim 3). In particular, the consistent steps to normalisation were chosen to minimise the effects of systematic bias on the cNN. Areas of normalisation included:

- Window sizes (*section 3.3.5*)
- Signal Amplitude Normalisation (*section 3.5.3*)
- Decibelisation (*section 3.5.3*)
- Number of noise vs event samples passed to the cNN (*section 3.3.8*)
- cNN pipeline normalisation to Gaussian mean and unit variance[4].

[4]StandardScaler function

5.2 Previous Literature

In comparison to previous research in applying Machine Learning techniques to volcanic and seismic settings, this paper has approached the aim of better interpreting volcanic behaviour differently. Many of the publications focus on the applications of Neural Networks on regional seismology, rather than volcanic seismology. Moustra et al. (2011) reached a 84.01% accuracy by assessing the statistical temporal distribution of regional seismic events, and using *Seismic Electric Signal* data (the supposed fluctuations in the Earth's magnetic field prior to an earthquake (*See Uyeda (1998), Uyeda et al. (1999)*)), however much of the SES was estimated and back-generated from the magnitudal data. Without SES data, the models only achieved a 60% accuracy on the statistical analysis of magnitudal data alone, with poorer accuracy above $M_W 5.2$.

Likewise, Adeli and Panakkat (2009) attempted to estimate the probability of regional seismicity through temporal analysis of the stress released and the magnitude, to varied success, predicting one of two $M_W 6.5 \leq 7.5$ events, and predicting 102 of 127 smaller events.

Following this trend, Reyes et al. (2013) focused on temporal analysis of regional seismicity in terms of energy release and time elapsed between events, to an average accuracy of 70.5%. Again, the focus is on large statistical temporal analysis, with none of the literature investigating the actual signal. This paper may be one of the first in the seismology field to analyse seismic data through complex signal analysis, and apply the information to an ANN. It is even more likely to be one of the first papers to do so in the field of volcanic seismology.

6 Conclusions

To conclude:

1. Noise and Event data was successfully differentiated
2. Differentiation was successfully automated
3. Use of a cNN in analysing seismic data was effective
4. Expansion of the datasets is needed to limit any risk of overfitting and improve flexibility of the model.

The model would benefit from improvements in manual noise-picking, data from multiple pickers, and from multiple settings before it can be considered robust enough for widespread application. However, the results from this initial attempt are a strong indication that ANNs are valuable tools in better understanding and working with seismic data in a volcanic setting, and even in a regional seismic setting in the future.

In addition to manually-picking noise data, using multiple pickers and using data across several settings, further work would focus on maturing the model by expanding from binary event—noise classification to event type classification, differentiating between VT, LF, ULF, Tremor and Hybrid events. Last-seconds analysis differentiating the 10 seconds prior to an event to noise data may also be an area of future research.

References

H. Adeli and A. Panakkat. A probabilistic neural network for earthquake magnitude prediction. *Neural Networks*, 22(7):1018–1024, Sept. 2009. ISSN 08936080. doi: 10.1016/j.neunet.2009.05.003.

M. Anthony. *Discrete Mathematics of Neural Networks: Selected Topics*. SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001. ISBN 978-0-89871-480-7.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656, Oct. 1948. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1948.tb00917.x.

N. Chawla and W. Wang, editors. *Proceedings of the 2017 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, Philadelphia, PA, June 2017. ISBN 978-1-61197-497-3. doi: 10.1137/1.9781611974973.

T. W. S. Chow and S.-Y. Cho. *Neural Networks and Computing: Learning Algorithms and Applications*. Number 7 in Series in electrical and computer engineering. Imperial College Press, London, 2007. ISBN 978-1-86094-969-2 978-1-86094-758-2. OCLC: 253863027.

J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

P. Dangeti. *Statistics for Machine Learning*. 2017. ISBN 978-1-78829-575-8. OCLC: 1015996007.

Eritrea - Ministry of Information. Volcanic Eruption in Southern Red Sea Region creates new landmass —. http://www.shabait.com/news/local-news/6072-volcanic-eruption-in-southern-red-sea-region-creates-new-landmass, June 2011.

R. P. Feynman, R. B. Leighton, and M. L. Sands. *The Feynman Lectures on Physics. Vol. 2: Mainly Electromagnetism and Matter*. Basic Books, New York, NY, new millenium ed edition, 1965. ISBN 978-0-465-02494-0 978-0-465-02416-2 978-0-465-02562-6. OCLC: 838503554.

B. Goitom, C. Oppenheimer, J. O. S. Hammond, R. Grandin, T. Barnie, A. Donovan, G. Ogubazghi, E. Yohannes, G. Kibrom, J.-M. Kendall, S. A. Carn, D. Fee, C. Sealing, D. Keir, A. Ayele, J. Blundy, J. Hamlyn, T. Wright, and S. Berhe. First recorded eruption of Nabro volcano, Eritrea, 2011. *Bull Volcanol*, 77(10):85, Oct. 2015. ISSN 0258-8900, 1432-0819. doi: 10.1007/s00445-015-0966-3.

H. Nyquist. Certain Topics in Telegraph Transmission Theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, Apr. 1928. ISSN 0096-3860. doi: 10.1109/T-AIEE.1928.5055024.

J. Hammond, J.-M. Kendall, G. Stuart, C. Ebinger, I. Bastow, D. Keir, A. Ayele, M. Belachew, B. Goitom, G. Ogubazghi, and T. Wright. Mantle upwelling and initiation of rift segmentation beneath the Afar Depression. *Geology*, 41(6):635–638, June 2013. ISSN 0091-7613. doi: 10.1130/G33925.1.

F. He. Real-time Process Modelling Based on Big Data Stream Learning. *Mälärdälen University, Thesis*, page 45, May 2017.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, Aug. 1952. ISSN 00223751. doi: 10.1113/jphysiol.1952.sp004764.

IRIS. SAC Data File Format. https://ds.iris.edu/files/sac-manual/manual/file_format.html, Nov. 2013.

J. Kaiser and R. Schafer. On the use of the I0-sinh window for spectrum analysis. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):105–107, Feb. 1980. ISSN 0096-3518. doi: 10.1109/TASSP.1980.1163349.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

G. F. Marshall and G. E. Stutz, editors. *Handbook of Optical and Laser Scanning*. Optical science and engineering. CRC Press, Boca Raton, FL, 2nd ed edition, 2012. ISBN 978-1-4398-0879-5.

K. Molnár, A. Lédeczi, L. Sujbert, and G. Peceli. Muzzle blast detection via short time Fourier transform. In *Proc. 12th PhD Mini-Symp*, pages 16–17, 2005.

M. Moustra, M. Avraamides, and C. Christodoulou. Artificial neural networks for earthquake prediction using time series magnitude data or Seismic Electric Signals. *Expert Systems with Applications*, 38(12):15032–15039, Nov. 2011. ISSN 09574174. doi: 10.1016/j.eswa.2011.05.043.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

J. Reyes, A. Morales-Esteban, and F. Martínez-Álvarez. Neural networks to predict earthquakes in Chile. *Applied Soft Computing*, 13(2):1314–1328, Feb. 2013. ISSN 15684946. doi: 10.1016/j.asoc.2012.10.014.

J. A. Schouten. *Tensor Analysis for Physicists*. Dover Publications, New York, 2nd ed edition, 1989. ISBN 978-0-486-65582-6.

J. A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Number v. 97 in Applied optimization. Springer, New York, 2005. ISBN 978-0-387-24348-1 978-0-387-29824-5 978-0-387-24349-8.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

I. Sutskever and G. Hinton. Temporal-Kernel Recurrent Neural Networks. *Neural Networks*, 23(2):239–243, Mar. 2010. ISSN 08936080. doi: 10.1016/j.neunet.2009.10.009.

S. Uyeda. VAN method of short-term earthquake prediction shows promise. *Eos, Transactions American Geophysical Union*, 79(47):573–580, Nov. 1998. ISSN 00963941. doi: 10.1029/98EO00417.

S. Uyeda, K. S. Al-Damegh, E. Dologlou, and T. Nagao. Some relationship between VAN seismic electric signals (SES) and earthquake parameters. *Tectonophysics*, 304(1-2):41–55, Mar. 1999. ISSN 00401951. doi: 10.1016/S0040-1951(98)00301-1.

H. Wu. Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions. *Information Sciences*, 179(19):3432–3441, Sept. 2009. ISSN 00200255. doi: 10.1016/j.ins.2009.06.006.