Alma Mater Studiorum · Università di Bologna

SCHOOL OF ENGINEERING and ARCHITECTURE

Master of Science in Electronic Engineering

Deep Learning Architectures for Time of Arrival Detection in Acoustic Emissions Monitoring

MASTER THESIS IN

ANALOG CIRCUITS, SENSOR READOUT AND CONVERSION

Supervisor:
Prof. Luca De Marchi

Co-supervisors:
Dott. Federica Zonzini
Dott. Denis Bogomolov

Presented by:
Tanush Dhamija

Session III
Academic Year 2020/2021

Acknowledgements

I would like to express my sincere gratitude towards my thesis supervisor Prof. Luca De Marchi for giving me the opportunity to work on this very interesting research and for his support throughout the activity.

This project would have not been possible without the contributions and guidance of my co-supervisors Dott. Federica Zonzini and Dott. Denis Bogomolov. It was a great pleasure to collaborate with both of you and I am very grateful for your help from the start until the very end which included many discussions over the months. I have learnt a lot of things, not only academically but also how I would go about approaching a problem and presenting myself. Thank you for all the valuable lessons that I will take along with me long beyond the completion of this activity.

Lastly, I would like to thank my family members and good friends for being a continuous pillar of support not only during this activity but all throughout my Master's program. Given the very strange times we encountered all together, it was not easy but it was most definitely worth it.

Abstract

Acoustic Emission (AE) monitoring can be used to detect the presence of damage as well as determine its location in Structural Health Monitoring (SHM) applications. Information on the time difference of the signal generated by the damage event arriving at different sensors is essential in performing localization. This makes the time of arrival (ToA) an important piece of information to retrieve from the AE signal. Generally, this is determined using statistical methods such as the Akaike Information Criterion (AIC) which is particularly prone to errors in the presence of noise. And given that the structures of interest are surrounded with harsh environments, a way to accurately estimate the arrival time in such noisy scenarios is of particular interest.

In this work, two new methods are presented to estimate the arrival times of AE signals which are based on Machine Learning. Inspired by great results in the field, two models are presented which are Deep Learning models - a subset of machine learning. They are based on Convolutional Neural Network (CNN) and Capsule Neural Network (CapsNet). The primary advantage of such models is that they do not require the user to pre-define selected features but only require raw data to be given and the models establish non-linear relationships between the inputs and outputs.

The performance of the models is evaluated using AE signals generated by a custom ray-tracing algorithm by propagating them on an aluminium plate and compared to AIC. It was found that the relative error in estimation on the test set was $< 5\%$ for the models compared to around 45% of AIC. The testing process was further continued by preparing an experimental setup and acquiring real AE signals to test on. Similar performances were observed where the two models not only outperform AIC by more than a magnitude in their average errors but also

they were shown to be a lot more robust as compared to AIC which fails in the presence of noise.

Contents

Chapter 1

Introduction

Acoustic Emission (AE) detection is a Structural Health Monitoring (SHM) technique which can be used to perform continuous monitoring of structures to detect the presence of damage via permanently installed sensors. As a structure is subjected to mechanical load, AE stress waves are dynamically excited from defects such as cracks in metals or broken fibres and cracked matrix in composite materials. These waves propagate through the structure and can be detected by monitoring surface displacements at a particular location using piezoelectric or fibre optic transducers for example. In thin plate like structures these waves typically exhibit a characteristic behaviour where their wave velocity changes with frequency. This is also known as dispersion. The detection and location of damage by sensing AE signals could serve as a form of automated structural inspection, with the potential to reduce inspection time, maintenance cost and also increase availability. The location of the source of AE events (and hence the location of damage) can be determined using measurements of Difference Time of Arrival (dToA) of the signals detected at different sensors in an array. [1]

One of the features, the Time of Arrival (ToA) is of great importance as it is used to compute the difference time of arrivals which subsequently is needed during the process of localization. Hence it is quite relevant and worthy to retrieve the arrival time of AE signals. The most widely used ways to extract this information are the standard statistical methods such as the Akaike Information Criterion (AIC) which have proven to work quite well. But such methods are very sensitive

to noise and are unable to provide with enough precision the arrival times given that the structures of interest are often surrounded with harsh conditions. For this reason, it is essential to build algorithms that are not prone to noise and are able to accurately estimate the arrival times in a robust manner.

This brings us to the field of Machine Learning (ML), with a particular emphasis on Deep Learning, which is a subset of ML in which we do not explicitly provide features such as providing a predefined threshold, frequency monitoring, etc. to the algorithm. Only raw data is provided to the models along with the corresponding labels (outputs), which in this case is the arrival time and the model learns relationships between the inputs and outputs and finally provides predictions when provided with new inputs. Two models based on different architectures are presented in this work, namely Convolutional Neural Network (CNN) and Capsule Neural Network (CapsNet) to tackle the same problem of estimating the time of arrival and their results are compared with AIC.

1.1 Outline of the work

The structure and outline of the work presented is detailed here with a brief summary of every chapter and what is to be expected in them.

In Chapter 2, we start with an introduction to Structural Health Monitoring, giving background on the field along with why is it so important in today's world. Particular emphasis is laid on Acoustic Emissions in SHM, clearly outlining what it is and the procedure involved in it.

Chapter 3 deals with Machine Learning and how we apply it to our task. The problem statement is laid out and we then move on to describe how we generate the dataset needed to train the machine learning models including the environment considered, the challenges faced during building the dataset and it will become quite clear the very important role that data plays in the field of machine learning.

In Chapter 4, we go into specifies about the first model based on Convolutional Neural Networks starting by giving an introduction to them and then proceeding to build the model we use to estimate the time of arrival.

Chapter 5 talks all about Capsule Neural Networks and how they are different from CNN, their advantages over CNN and finally building a CapsNet model to

tackle the problem at hand. But before we can do this, some transformations to the dataset was required as CapsNet has a different architecture with respect to CNN.

In Chapter 6, we present all the results of the two models along with the results of AIC for the test set. And we make a comparison between all the three methods in performing the same task under varying noise conditions and how the performance varies for all three methods.

In the final Chapter 7, we extend the testing process to real experiments conducted in the laboratory and present the results from the signals acquired in the lab. We describe the experimental setup and the nature of experiments conducted and finally verify if the initial results from the test dataset are also matched on testing on real signals.

Final thoughts are given in the Conclusion taking into account all the various tests and experiments performed as part of this activity and what the results indicate along with a section on future work that can be carried out in the field.

Chapter 2

Structural Health Monitoring

Structural Health Monitoring (SHM) involves the observation and analysis of a system over time using periodically sampled response measurements to monitor changes to the material and geometric properties of engineering structures such as bridges and buildings. The SHM process involves selecting the excitation methods, the sensor types, number and locations, and the data acquisition/storage/transmittal hardware commonly called health and usage monitoring systems. Measurements may be taken to either directly detect any degradation or damage that may occur to a system or indirectly by measuring the size and frequency of loads experienced to allow the state of the system to be predicted.

It aims to give, at every moment during the life of a structure, a diagnosis of the "state" of the constituent materials, of the different parts, and of the full assembly of these parts constituting the structure as a whole. The state of the structure must remain in the domain specified in the design, although this can be altered by normal aging due to usage, by the action of the environment, and by accidental events. Thanks to the time-dimension of monitoring, which makes it possible to consider the full history database of the structure, and with the help of Usage Monitoring, it can also provide a prognosis (evolution of damage, residual life, etc.). [2]

If we consider only the first function, the diagnosis, we could estimate that Structural Health Monitoring is a new and improved way to make a Non- Destructive Evaluation. This is partially true, but SHM is much more. It involves the integration

of sensors, possibly smart materials, data transmission, computational power, and processing ability inside the structures. It makes it possible to reconsider the design of the structure and the full management of the structure itself and of the structure considered as a part of wider systems. This is schematically presented in Figure 2.1

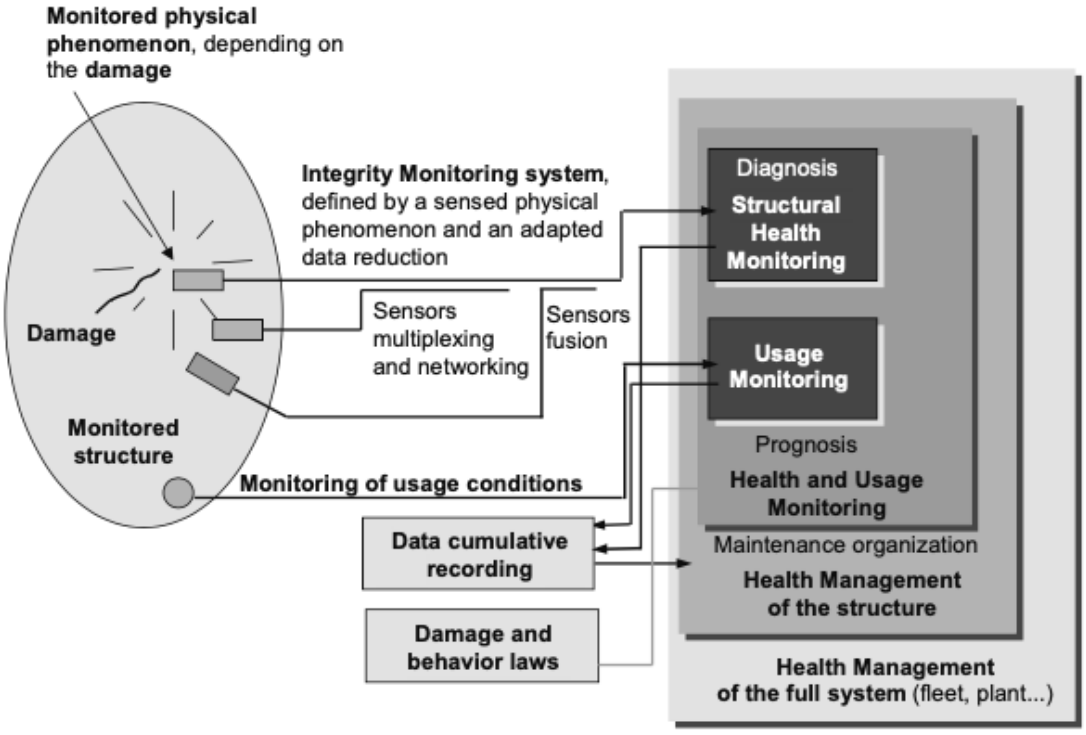

Figure 2.1: Principle and organization of a SHM system

2.1 Motivation for Structural Health Monitoring

Knowing the integrity of in-service structures on a continuous real-time basis is a very important objective for manufacturers, end-users and maintenance teams. In effect, SHM:

- allows an optimal use of the structure, a minimized downtime, and the avoidance of catastrophic failures,

- gives the constructor an improvement in his products,

- drastically changes the work organization of maintenance services. i) by aiming to replace scheduled and periodic maintenance inspection with performance-based (or condition-based) maintenance (long term) or at least (short term) by reducing the present maintenance labor, in particular by avoiding dismounting parts where there is no hidden defect; ii) by drastically minimizing the human involvement, and consequently reducing labor, downtime and human errors, and thus improving safety and reliability.

The improvement of safety seems to be a strong motivation, in particular after some spectacular accidents due to: i) unsatisfactory maintenance, for example, in the aeronautic field, the accident of Aloha Airlines (see Figure 2.2) or, in the civil engineering field, the collapse of the Mianus River bridge; ii) ill-controlled manufacturing process, for example, the Injaka bridge collapse (see Figure 2.3). In both fields the problem of aging structures was discovered and subsequent programs were established.

The economic motivation is stronger, principally for end-users. In effect, for structures with SHM systems, the envisaged benefits are constant maintenance costs and reliability, instead of increasing maintenance costs and decreasing reliability for classical structures without SHM (see Figure 2.4).

Figure 2.2: Aloha Airlines flight 243, April 29, 1988, due to corrosion insufficiently controlled by maintenance

Figure 2.3: Injaka bridge collapse, July 1998, due to a poorly controlled construction process

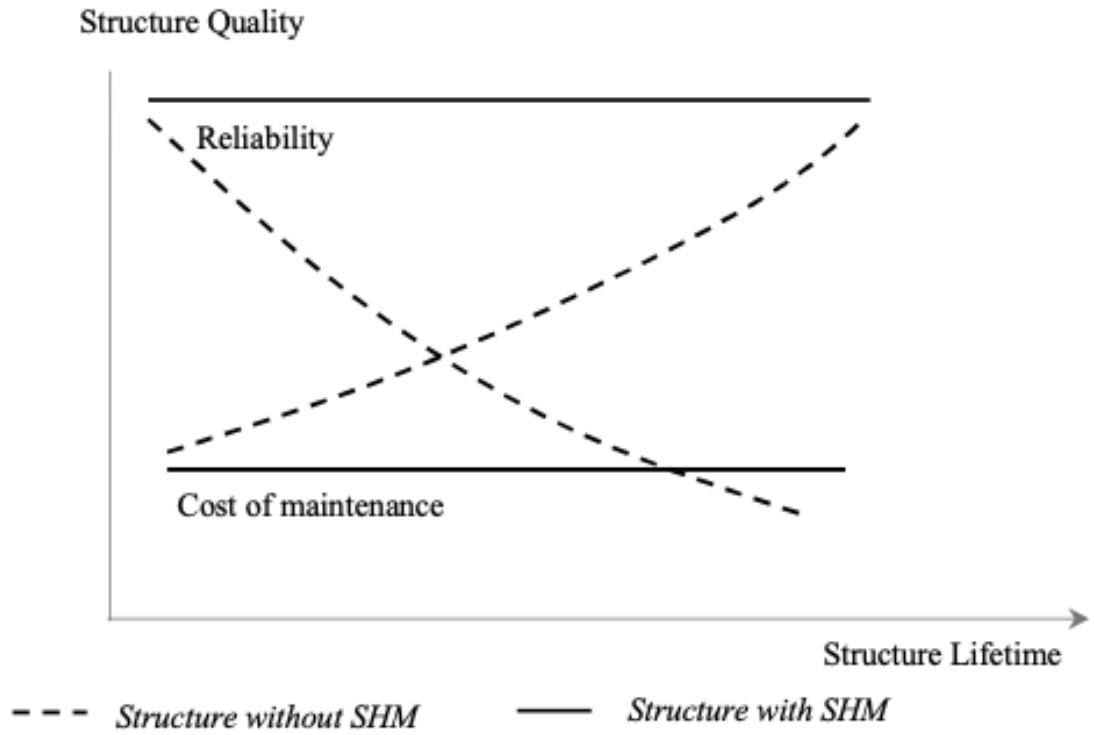

Figure 2.4: Benefit of SHM for end-users

2.2 Acoustic Emission Testing

2.2.1 What is Acoustic Emission?

Acoustic emission (AE) is the phenomenon of radiation of acoustic (elastic) waves in solids that occurs when a material undergoes irreversible changes in its internal structure, for example as a result of crack formation (see Figure 2.5) or plastic deformation due to aging, temperature gradients or external mechanical forces. In particular, AE is occurring during the processes of mechanical loading of materials and structures accompanied by structural changes that generate local sources of elastic waves. This results in small surface displacements of a material produced by elastic or stress waves generated when the accumulated elastic energy in a material or on its surface is released rapidly. The waves generated by sources of AE are of practical interest in structural health monitoring (SHM), quality control, system feedback, process monitoring and other fields. In SHM applications, AE is typically

used to detect, locate and characterise damage (see Figure 2.6).

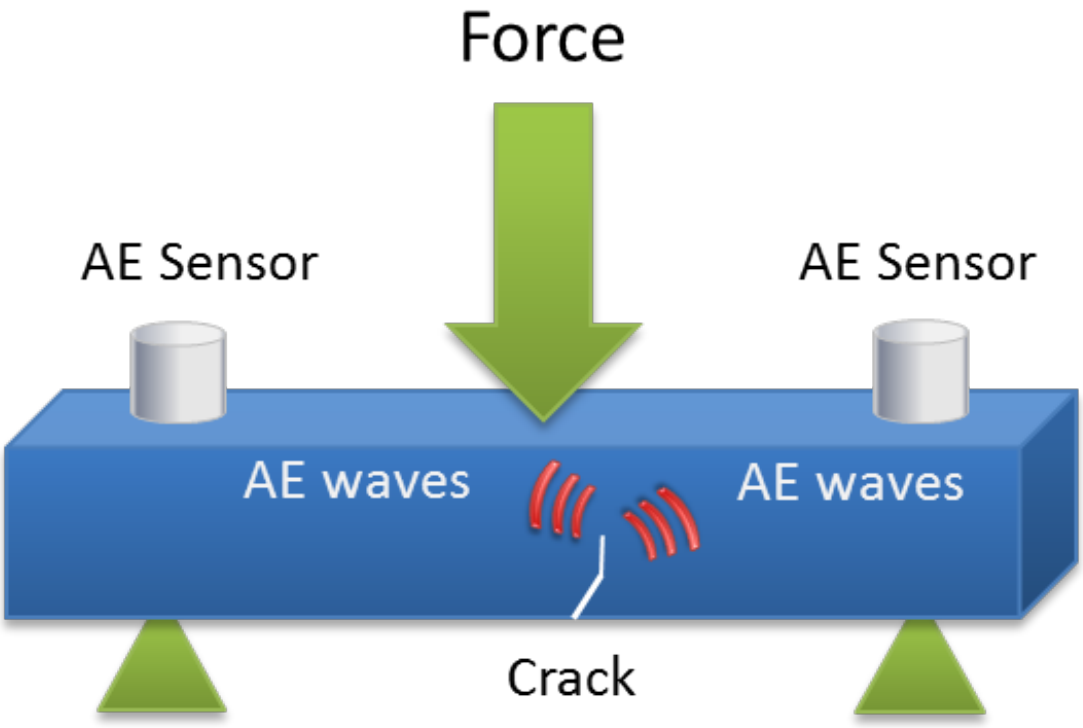

Figure 2.5: Acoustic emission due to crack growth in a solid material under stress

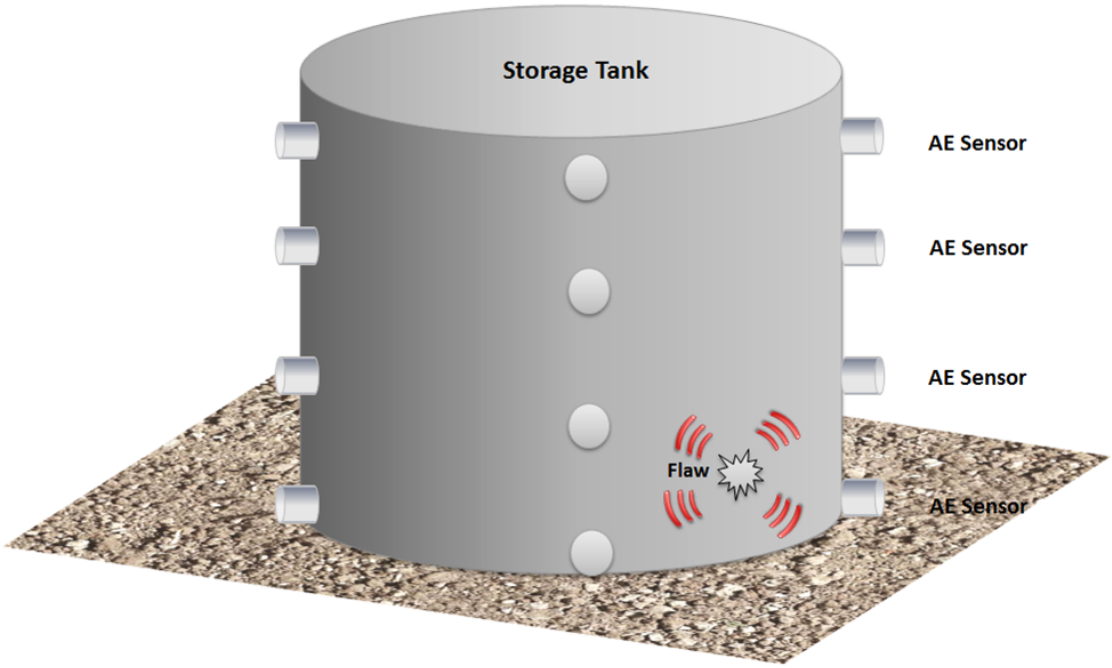

Figure 2.6: A storage tank under test

2.2.2 Acoustic Emission in SHM

Many conventional techniques are proposed by many engineers and scientists for health monitoring of structures, and the majority of them are nondestructive testing (NDT) type methods. Again, in many NDT systems, testing loads are applied before or after the testing and are widely used as an active method, where signals or energy is delivered from outside to the testing body. Contrary to active NDT, acoustic emission is widely used as a passive NDT method in structural health monitoring, where external energy is not needed to supply to the testing structure. The stimulated internal energy of the structure is received in this technique as health monitoring features. Due to this unique characteristic, acoustic emission has become very simple, however accuracy or acquisition sensitivity is very high. Therefore, acoustic emission technique is becoming popular day by day in all types of structural monitoring fields. [3]

2.2.3 AE technique

A fundamental AE testing system consists of a sensor (AE sensor), a pre-amplifier, main amplifier with appropriate filters, and a data acquisition system along with display (oscilloscopes, personal computer with data acquisition software, and data transferring devices like AD conversion system). As AE signals are very small, it is boosted by the pre-amplifier to gain at low signal-to-noise ratio. Later on, AE signals are amplified again and passed through band pass filters before storing them to a personal computer (PC) for analysis. This is described in Figure 2.7.

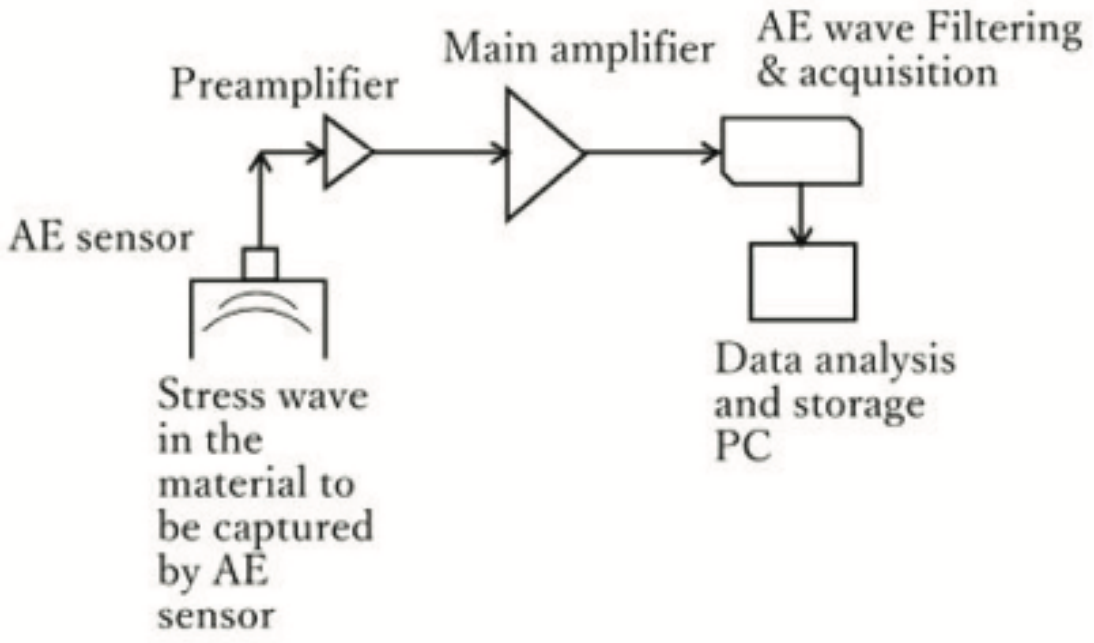

Figure 2.7: Fundamentals of AE testing

2.2.4 What is AET?

The term acoustic emission testing (AET) refers to the process of detecting and recording AE using specialized equipment. AET is a type of nondestructive test (NDT) that has various uses, including ensuring the structural integrity of vessels, monitoring weld quality and more. The process involves using sensors to detect AE and then converting the waves into electrical signals so that they can be recorded.

You can then analyze the results to assess a material's condition and locate any defects. The recorded information can provide potentially valuable information about the origin and significance of a defect in a structure. [4]

Chapter 3

Machine Learning applied to SHM

3.1 Intro

Pattern recognition implemented through machine learning algorithms is a mature discipline. In abstract terms the theory provides mathematical means of associating measured data with given class labels. In the context of SHM, one wishes to associate the measured data with some damage state, the simplest – and arguably most important – problem being that of distinguishing between the states 'healthy' and 'damaged' for a structure. In mathematical terms there are a number of distinct approaches to pattern recognition, the main ones being the statistical and neural approaches (Schalkoff, 1992). As all engineering problems are subject to various degrees of uncertainty, the statistical approach to pattern recognition appears to stand out as a natural approach for SHM purposes. But as it will be seen in later chapters, the statistical methods (such as the Akaike Information Criterion (AIC)) are extremely sensitive to noise which is evident due to the harsh environmental conditions where the structures of interest (such as bridges, storage tanks, etc.) are usually present. Hence a need for more robust methods is clearly evident and of much interest such as neural network approaches that offer a robust means of dealing with SHM problems.

The idea of machine learning can be simply stated: it is to 'learn' the relationship between some features derived from the collected data and the damaged state of the structure. If such a relationship between these two quantities exists, but

is unknown, the learning problem is to estimate the function that describes this relationship using data acquired from the test structure – the training data. [5]

3.2 Problem Statement

As seen in the earlier chapter, acoustic emission testing (AET) is of high practical use in the industry as it makes it possible to detect dangerous defects at an early stage of degradation [6]. This approach allows for the detection of weaknesses in large-bore structures, including pipelines, vessels and storage tanks, columns and reactors, etc. One of the main advantages of this method is the possibility to localize the source by passively capturing the acoustic waves propagating in the medium. Thus, a structure can be investigated in service when the AE system continuously monitors progressive damage [7].

One of the most important features to be extracted from an acoustic signal, the Time of Arrival (ToA), also known as onset time, requires primary attention since it is used for the subsequent localization step. Although widely used statistical algorithms for onset time determination, such as the Akaike Information Criterion (AIC) and the cross-correlation method, have proven their effectiveness even at low noise-to-signal ratios, unpredictable noise sources, such as electromagnetic interference, rain or random mechanical impacts in the vicinity of the AE-sounding system significantly affect the reproducibility and accuracy of ToA estimation results.

In recent years, machine learning methods capable of solving signal processing problems have increasingly found success. Noteworthy, powerful solutions to tackle ToA estimation in an automated and more robust manner were proposed in the seismology field. Seminal works in this direction include the improvement in the earthquake phase detection by means of template-waveform-based methods [8], that search continuous seismic data for signals that are similar to previously detected ones, while Chen (2018) proposed an unsupervised micro seismic picking algorithm that utilizes fuzzy clustering to identify ToAs [9]. Another example worthy of attention was examined in the work by Zachary E. Ross (2018) [10], where the ToA of the P-wave present in the seismogram was considered from the point of view of pattern recognition, in which a deep learning model was trained. The importance

of this work is that the feature extraction process was skipped, and time series data were provided directly as input of the model with little pre-processing.

Hence we will now see the application of Deep Learning methods such as Convolutional Neural Networks and Capsule Neural Networks applied to the problem of : retrieving the Time of Arrival (ToA) of an AE signal. But first, as is the case with every problem involving Machine Learning, even more so for a Deep Learning approach, we require training data from which the model can learn. As we will further see in this chapter, the quality of training data plays a very significant role in the performance of the model(s). But first let's talk about how the training data is collected.

3.3 Dataset Generation

Any Deep Learning task requires a lot of data and this being a supervised learning problem, we also require the data to be labelled accurately. In order to achieve this, the strategy employed was to collect the data using analytical simulations which allowed us to:

- have a faster collection time.

- label the ground truth accurately.

3.3.1 Simulation Environment

The setup involves an aluminium plate with dimensions of 1000 $mm \times 1000$ mm and a thickness of 3 mm. The actuation signal is a simple Gaussian modulated sine wave with central frequency 250 KHz as is seen in Figure 3.1. This signal is propagated through the aluminium plate and moreover in order to account for the dispersive and multi-modal propagation behaviour of guided waves, a custom ray-tracing algorithm (written in MATLAB) capable of handling this peculiar pattern and also account for the edge physical interaction up to the 4th reflection order, has been exploited for the purpose of signal generation. The propagated signal received, simulates an AE signal that constitutes the training set and is shown in Figure 3.2 along with it's ToA. Each signal consists of 5000 samples

acquired at a sampling frequency of 2 MHz. These quantities are compatible with commercial off-the-shelf sensors for AE monitoring and were, for this reason, chosen.

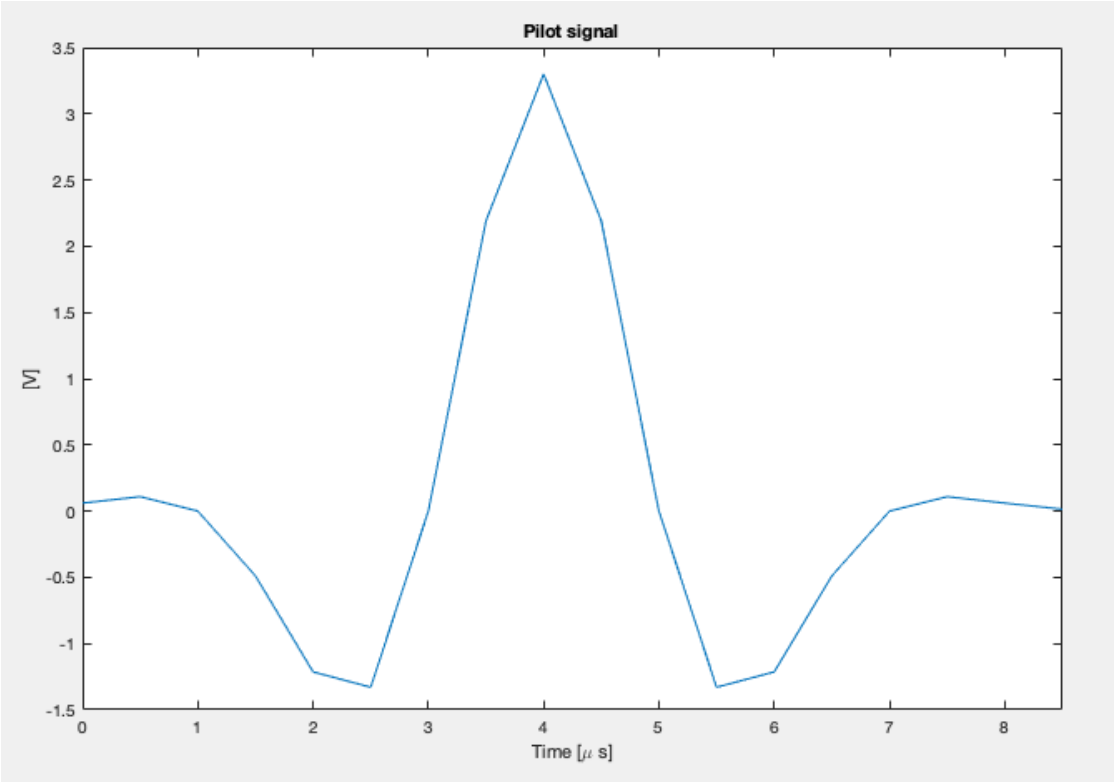

Figure 3.1: Actuation signal

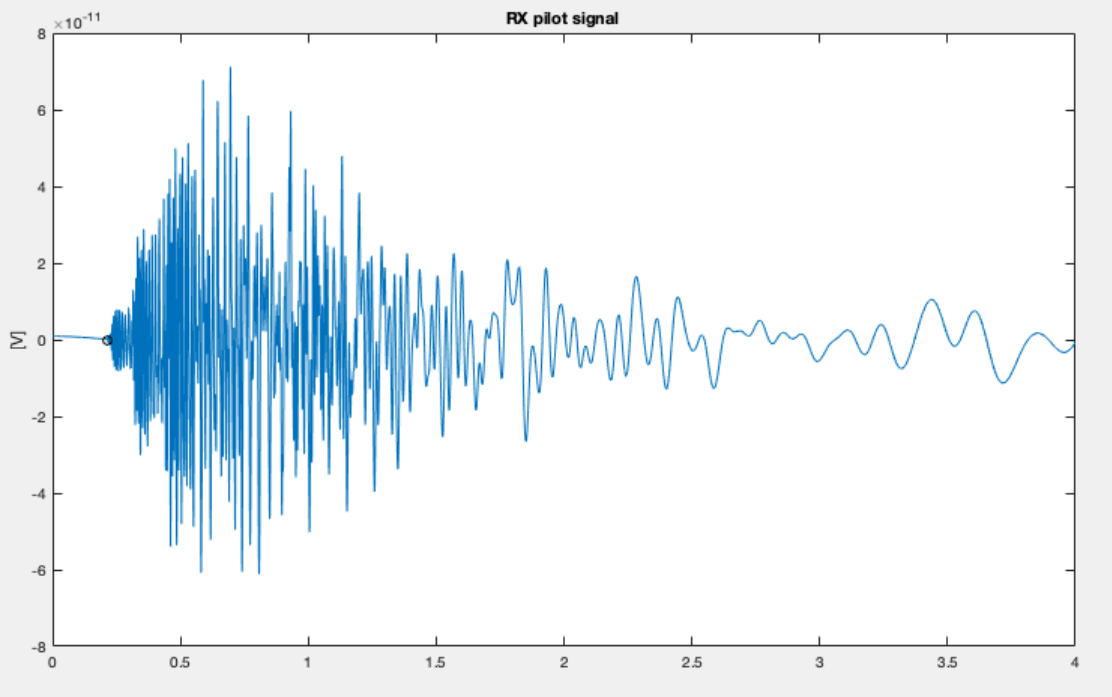

Figure 3.2: Simulated AE signal with it's ToA

A sampling grid with transmitters (TXs) and receivers (RXs) placed at various positions on the plate is created in order to collect signals with different arrival times that occur due to the varied distances between each TX-RX pair. A total of 40 TXs and 25 RXs are placed along the plate as shown in Figure 3.3. Lastly, and more importantly, since the primary goal is to have a better and more robust estimation of the arrival time in harsh environments where the presence of noise is apparent when compared to the statistical methods, it is of utmost importance that the training data consists of such noisy signals. Moreover, in order to have a diverse and varied dataset, we iterate over signals with signal-to-noise ratios (SNRs) starting from 1 to 30 dB in steps of 1 dB at a time. This when combined with the various TX-RX pairs gives us a good range and number of signals as a primary step for the training data.

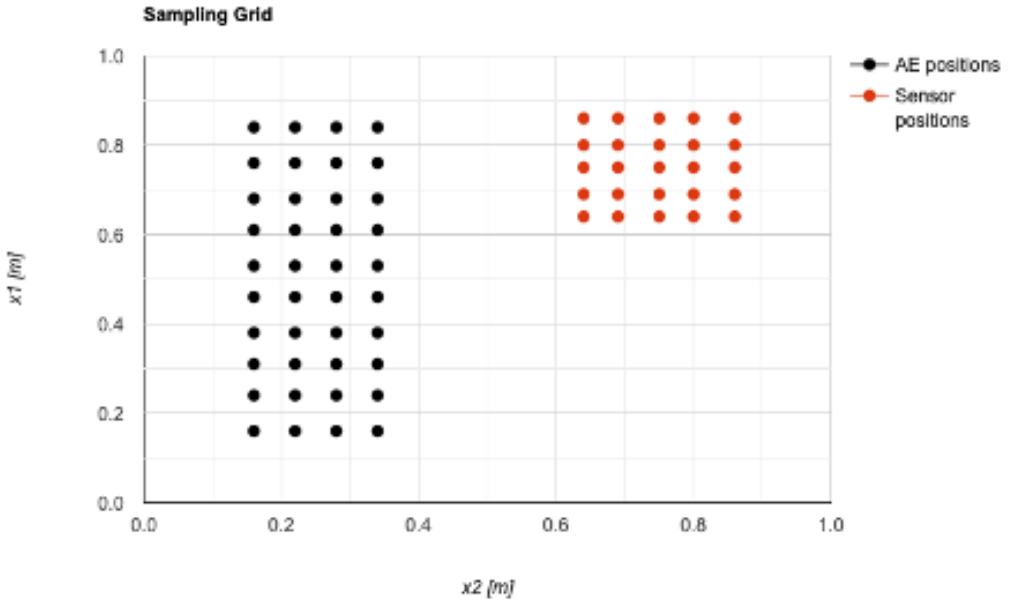

Figure 3.3: Sampling grid of TXs and RXs

3.3.2 Labelling the data

For supervised learning to work, we need a labeled set of data that the model can learn from to make correct decisions. Data labeling typically starts by asking humans to make judgments about a given piece of unlabeled data. The machine learning model uses human-provided labels to learn the underlying patterns in a process called "model training." The result is a trained model that can be used to make predictions on new data. [11]

In machine learning, a properly labeled dataset that we use as the objective standard to train and assess a given model is often called “ground truth.” The accuracy of the trained model will depend on the accuracy of our ground truth, so spending the time and resources to ensure highly accurate data labeling is essential. For this reason, the most robust and accurate way we could think of labelling our data is by using Akaike Information Criterion (AIC) to estimate the ToA in noise-free conditions, due to the proven high-quality estimates provided by this

method in presence of clear changes in the signal statistics.

Thus before noise of a certain SNR is added to the signal during the process of dataset generation, it's ToA is estimated by AIC. We then proceed to add noise to the signal and it is finally saved as part of the dataset. In this way we could practically have the ground truth to be as accurate as possible due to the fact being that it is estimated on the original noise-free signal by AIC.

3.3.3 Data augmentation

We now have our first training set as we have just seen. This comprises of simulated AE signals with varying SNRs (from 1 to 30 dB) and different AE and sensor positions (as seen in Figure 3.3). But what we noticed is that the different arrival times of the signals are still not diverse enough to generalize the model for test signals which will be later seen. This is true especially for signals with much higher pre-trigger times and. The training set just doesn't have signals with large pre-trigger times at the moment and this leads to bad predictions for the test signals with said higher pre-trigger times.

In order to account for this, we could either go back to the propagation algorithm and try out different AE and sensor positions in a way to maximize the distance between them in order to have signals with higher pre-trigger times but this would still limit us with the pre-trigger time values we could have given the compact geometry of the plate. Hence to overcome this what we proposed is to synthetically add onto our existing data which is built using the current data itself. The procedure to do this is laid out below:

1. Given a signal, estimate the variance of the part from the start of the signal until it's arrival time (label). This is done to avoid any local discontinuities introduced in the signal if we simply perform a copy and paste of the samples to increase the pre-trigger time of the signal.

2. Draw random samples from a normal (Gaussian) distribution with 0 mean and variance equal to the one calculated in the step above.

3. Construct a new signal starting from the random samples which were drawn and then append the remaining part with the original signal. And finally

calculate the new label accordingly to the number of samples that were added.

4. In order to add further randomness, the number of random samples added for each signal is also chosen randomly each time.

5. Repeat this procedure for every signal in the training set.

The end result of the procedure described above is shown in Figure 3.4. We see that given an original signal, the resulting transformed signal is the same but with an increased pre-trigger time. This gained us an advantage on two fronts:

- the training set now accounts for signals with a wide range of varied pre-trigger times.

- we have just doubled the size of our training set which plays a vital role as shall be seen later.

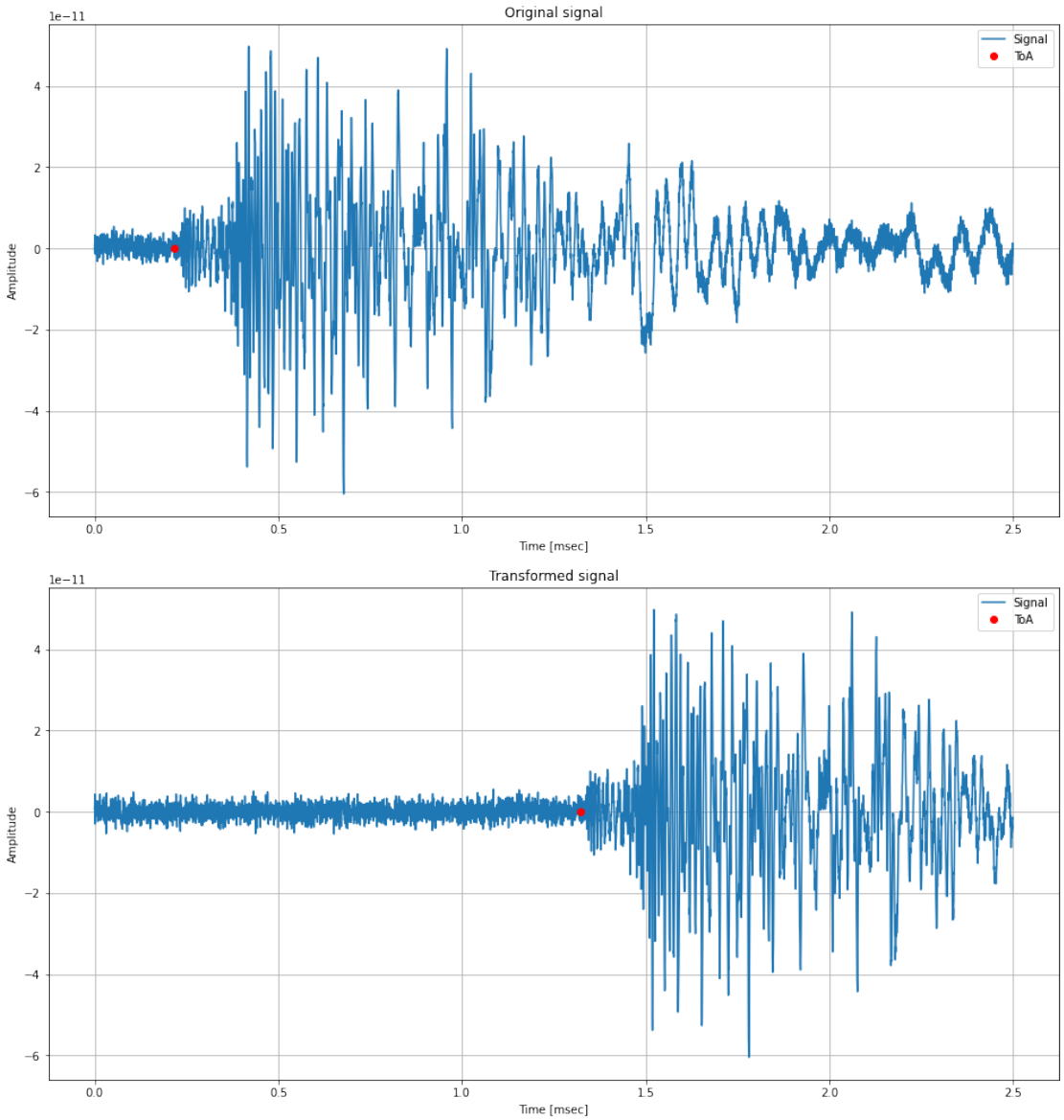

Figure 3.4: Synthetically increasing the pre-trigger time of a signal

Chapter 4

Convolutional Neural Networks

4.1 What are CNNs?

Convolutional Neural Network (CNN) is a class of artificial neural networks which can extract important and relevant information from raw data and retain it in the form of weights and biases and then use them to make classification and/or predictions on new data. This is performed through two stages, i.e., feature extraction and classification, which are further described below: [12]

Feature Extraction Here the convolution operations take place with the specific objective to extract all the important features from the inputs. Multiple convolutional layers are stacked one after the other and this enables each layer of the network to identify different properties of the input in a comprehensive manner. For the sake of an example, the first stage captures the low-level features, whereas the subsequent layers capture the high-level features. Convolutions are generally followed by pooling layers, that are responsible for reducing the spatial size of the convoluted features as well as for reducing the computational complexity of the involved operations and extracting dominant features, which are rotational and positional invariant.

Classification This stage consists of fully connected dense layers that are used to learn non-linear combinations of the high-level features from the output of the previous stage and finally make predictions, i.e., estimating the ToA in our case.

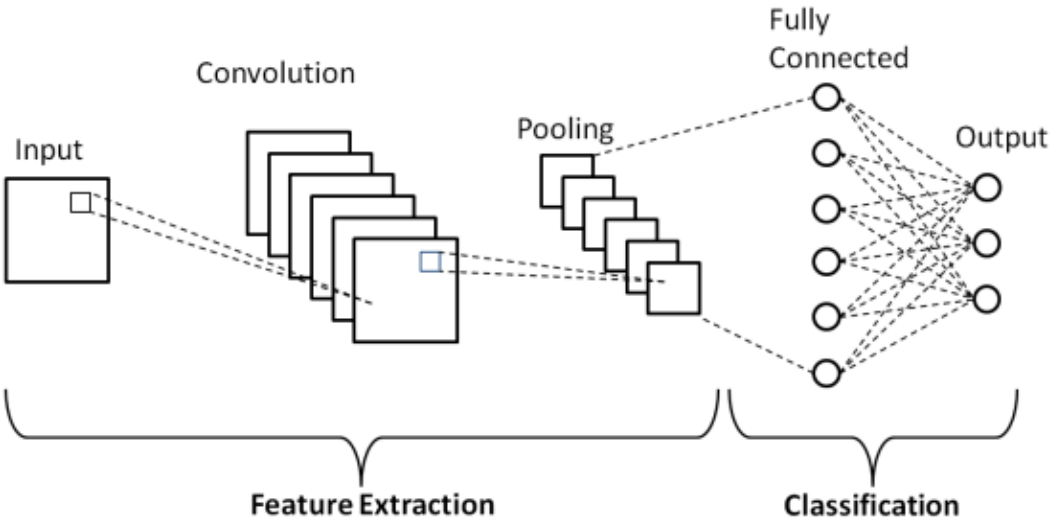

Figure 4.1: Outline of a CNN [13]

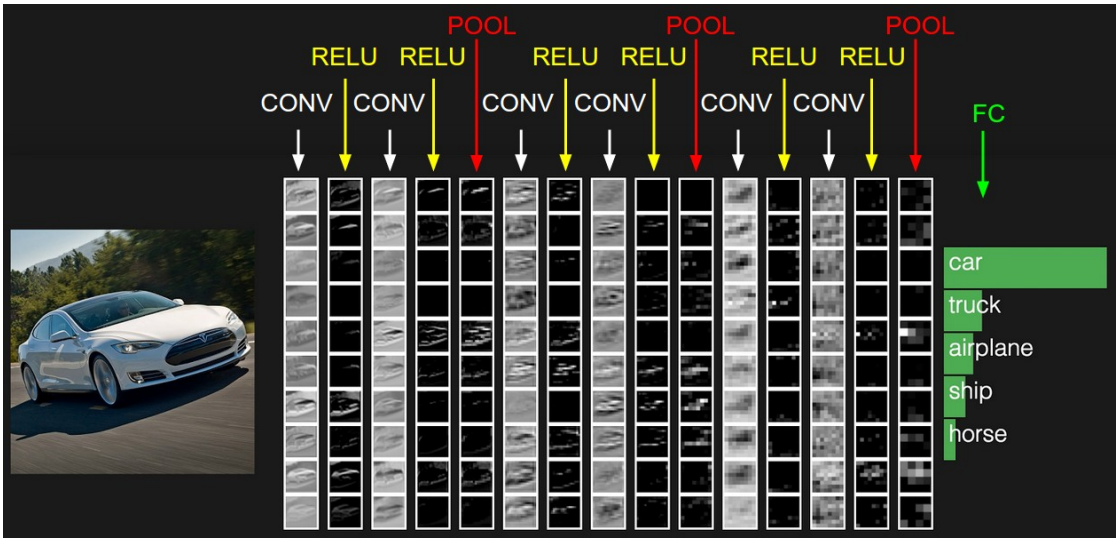

Figure 4.2: CNN example [14]

CNNs offer state-of-the-art performance on image data used for classification tasks. Since in our task we are dealing with one-dimensional AE signals, we can utilize the same principles and advantages of CNN using 1D convolutional layers. This is achieved by capturing the spatial and temporal dependencies in the input through the application of relevant filters.

4.2 CNN for estimating ToA

We stack five Conv1D + MaxPooling layers one after the other each with a kernel size of 10 x 10 and kernel strides of 2 samples. The number of filters in each layer increases step-by-step as can be seen in Figure 4.4: 50, 100, 150, 200, 250 have been employed, respectively, as each of the five layers. A non-linear activation function, Rectified Linear Unit (ReLU) is used at the output of each layer. We then add a Global Average Pooling layer before the fully connected Dense layer with 1024 units and ReLU activation, followed by an output Dense layer with just 1 unit and linear activation which finally gives us the ToA associated with the input signal.

A schematic of the 1D CNN model for retrieving the ToA is shown in Figure 4.3 and the layer sizes along with the number of parameters if the model is shown in Figure 4.4. With the following two figures it is easy to understand the process of how the CNN estimates the arrival time along with the dimensions of the model.

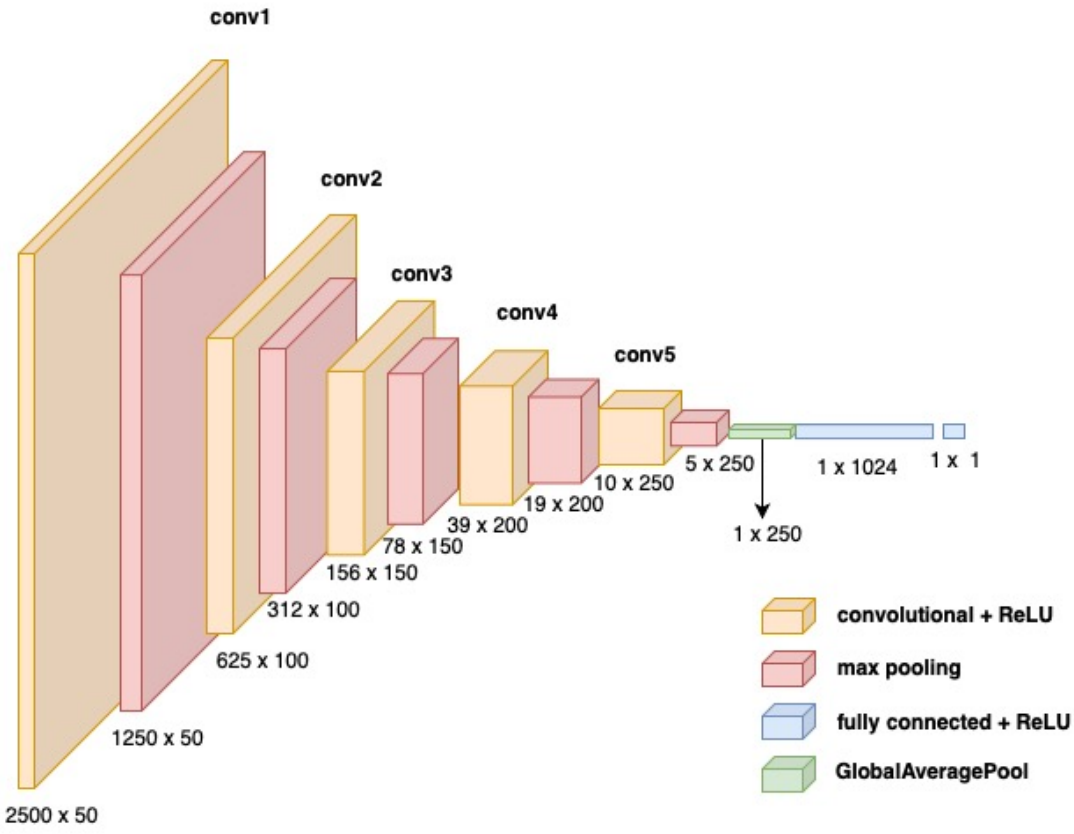

Figure 4.3: 1D CNN schematic for retrieving the ToA

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 2500, 50)	550
max_pooling1d (MaxPooling1D)	(None, 1250, 50)	0
conv1d_1 (Conv1D)	(None, 625, 100)	50100
max_pooling1d_1 (MaxPooling1	(None, 312, 100)	0
conv1d_2 (Conv1D)	(None, 156, 150)	150150
max_pooling1d_2 (MaxPooling1	(None, 78, 150)	0
conv1d_3 (Conv1D)	(None, 39, 200)	300200
max_pooling1d_3 (MaxPooling1	(None, 19, 200)	0
conv1d_4 (Conv1D)	(None, 10, 250)	500250
max_pooling1d_4 (MaxPooling1	(None, 5, 250)	0
dropout (Dropout)	(None, 5, 250)	0
global_average_pooling1d (Gl	(None, 250)	0
dense (Dense)	(None, 1024)	257024
dense_1 (Dense)	(None, 1)	1025

Total params: 1,259,299
Trainable params: 1,259,299
Non-trainable params: 0

Figure 4.4: Layers and dimensions of the CNN model for ToA estimation

The model performs at par with the statistical methods such as AIC under noise-free conditions and the real edge it has over AIC is under the influence of noise which is going to be present in real life scenarios. This is shown in Figure 4.5 in which CNN estimates the arrival time a lot more precisely when compared to AIC. The complete results along with the laboratory tests conducted will be discussed in the later chapters.

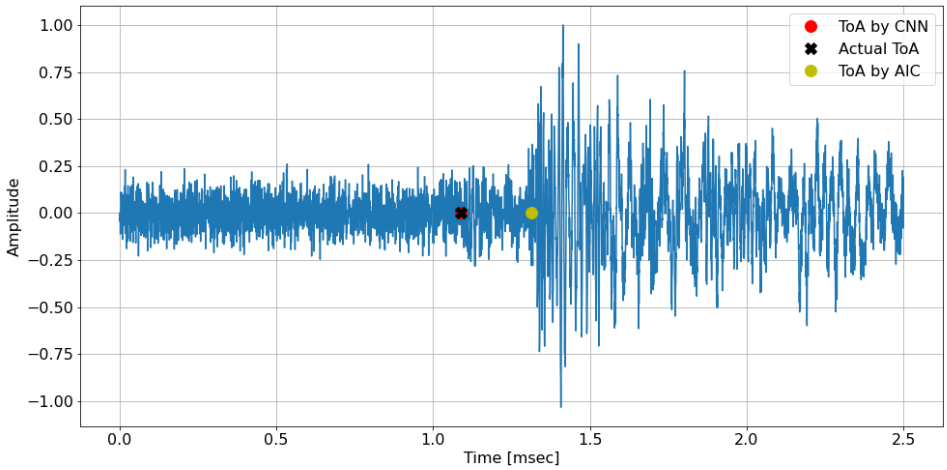

Figure 4.5: Test signal with predicted ToAs by CNN and AIC along with the actual ToA

Chapter 5

Capsule Neural Network

A Capsule Neural Network (CapsNet) is a machine learning system that is a type of artificial neural network (ANN) that can be used to better model hierarchical relationships. The approach is an attempt to more closely mimic biological neural organization. The idea is to add structures called "capsules" to a convolutional neural network (CNN), and to reuse output from several of those capsules to form more stable (with respect to various perturbations) representations for higher capsules. The output is a vector consisting of the probability of an observation, and a pose for that observation.

5.1 Why do we need Capsule Networks?

Convolutional Neural Networks (CNNs) no doubt offer state-of-the-art performance in Deep Learning problems involving detecting features from the input and performing classification and/or making predictions. But they also have their limits and some fundamental drawbacks in their architecture. Let us consider a simple example: Imagine a face. What are it's components? We have the oval face, two eyes, a nose and a mouth. For a CNN, a mere presence of these objects can be a very strong indicator to consider that there is a face in the image. Orientational and relative spatial relationships between these components are not very important to a CNN. This is depicted in Figure 5.1.

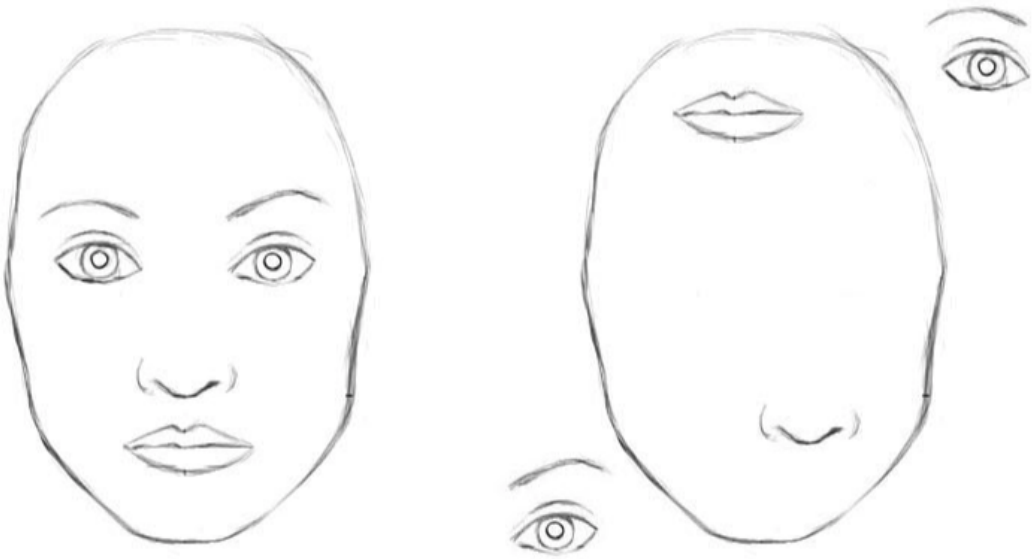

Figure 5.1: To a CNN, both pictures are similar, since they both contain similar elements.

We have seen how CNNs work in the previous chapter. The phenomenon described above is due to the fact that CNNs loose valuable information such as the **pose** (translational and rotational) relationship between the learned features. This is a result of the pooling operation present in CNNs. Here is a direct quote from Hinton, who is one of the creators of Deep Learning and has invented various algorithms that are widely used today including Capsule Neural Networks, Hinton: *"The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster."*

In the process of max-pooling, lots of important information is lost because only the most active neurons are chosen to be moved to the next layer. This operation is the reason that valuable spatial information gets lost between the layers. To solve this issue, Hinton proposed that we use a process called "routing-by-agreement" [15]. This means that lower level features (fingers, eyes, mouth) will only get sent to a higher level layer that matches its contents. If the features it contains resemble that of an eye or a mouth, it will get to a "face" or if it contains fingers and a palm, it will get send to "hand".

5.2 Understanding CapsNet

5.2.1 What is a capsule?

A capsule is a set of neurons that individually activate for various properties of a type of object, such as position, size and hue. Formally, a capsule is a set of neurons that collectively produce an *activity vector* with one element for each neuron to hold that neuron's instantiation value (e.g., hue). Artificial neurons traditionally output a scalar, real-valued activation that loosely represents the probability of an observation. CapsNets replace scalar-output feature detectors with vector-output capsules and max-pooling with routing-by-agreement [15]. Because capsules are independent, when multiple capsules agree, the probability of correct detection is much higher.

5.2.2 Architecture of CapsNet on MNIST

A simple CapsNet architecture is shown in Figure 5.2. It is shallow with only two convolutional layers and one fully connected layer.

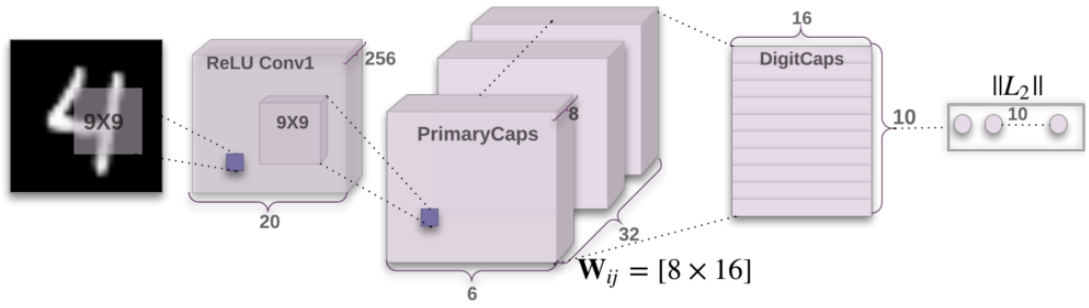

Figure 5.2: CapsNet architecture on MNIST (as proposed in Hinton, 2017)

There are four main components that are present in CapsNet which are listed below:

Matrix multiplication It is applied to the image that is given as input to the network to convert it into vector values to understand the spatial relationship.

Scalar weighting of the input It computes which higher-level capsule should receive the current capsule output.

Dynamic routing algorithm It permits these different components to transfer information amongst each other. Higher-level capsules get the input from the lower level. It is an iterative process.

Squashing function It is the last component that condenses the information. The squashing function takes all the information and converts it into a vector that is less than or equal to 1 while also maintaining the direction of the vector.

5.3 CapsNet for estimating ToA

Now that we have seen the basic principle and parts of Capsule Networks, we apply it to our case - estimating the ToA. But before we proceed with that we need to make some changes to our training data. As we have just seen, CapsNet in the present stage, performs well on classification tasks. Whereas we have initially used CNN as a regressor with the labels being ToA themselves. Hence we need to transform the labels as well as the input to use them as a classification task.

5.3.1 Transforming the training data

The training data we currently have is for the model to predict a continuous value (i.e. the ToA), making it a regression task. In order to turn it into a classification task, we need to transform the data accordingly. This includes both the input AE signals and the labels (which currently are ToAs).

- Input AE signals: For each signal, we only consider a small window of it which is centered around it's arrival time (label). The length of the window is fixed to 500 samples (or 0.5 msec). And for each signal we also generate an additional window of the same length comprising of random noise.
- Labels: The windows of the input signals are labeled as "1" and the noise windows are labeled as "0".

We now have the new transformed data consisting of valid AE event windows with a label 1 and noise windows with a label 0 which CapsNet will learn to distinguish between. The new training data is shown in Figure 5.3 below.

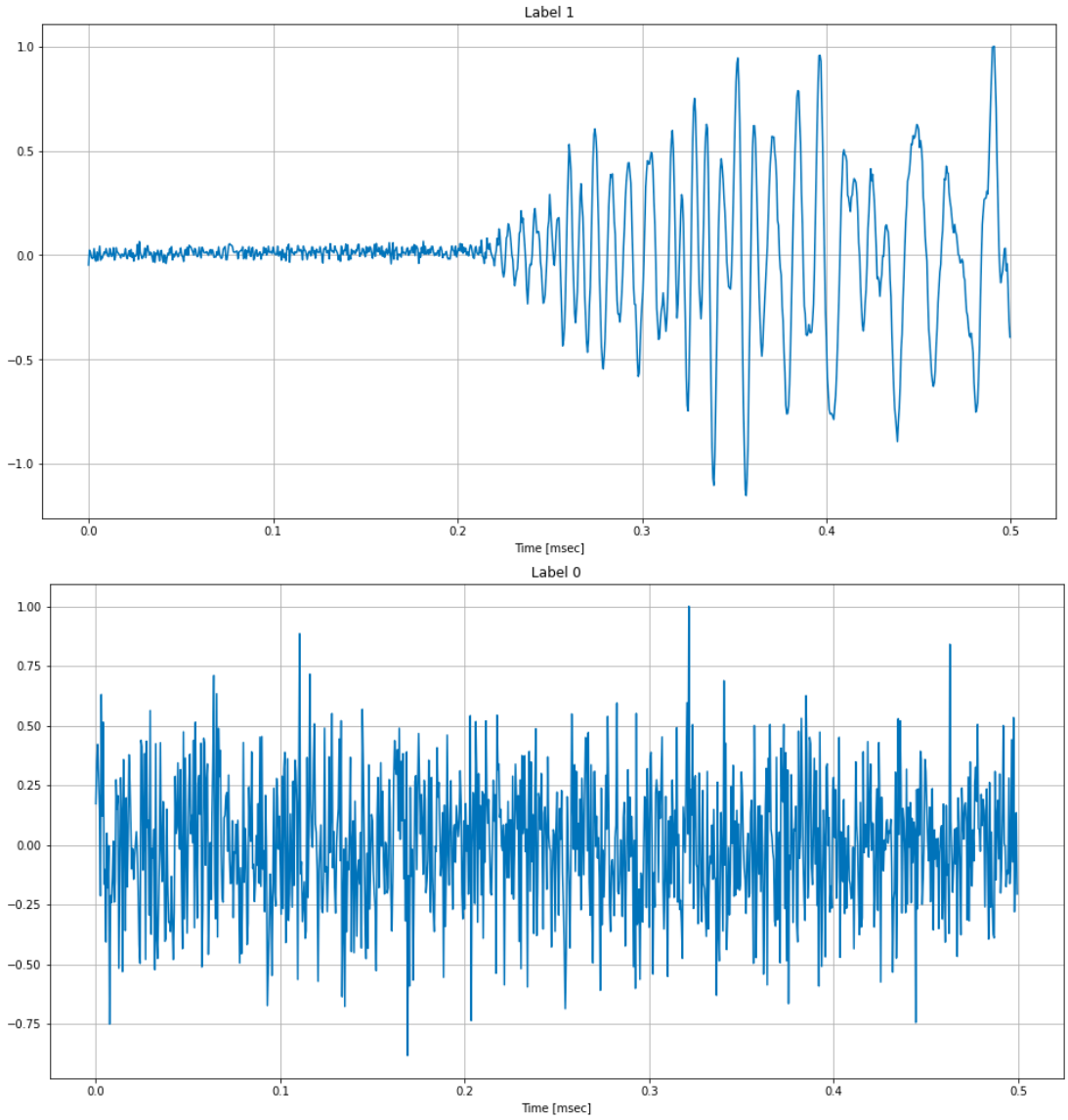

Figure 5.3: Train data for CapsNet showing the two classes

5.3.2 Defining the model

We implemented a CapsNet model compatible for our particular case from the official paper by Hinton, 2017. The key difference between the paper and our implementation is that we have a 1D Convolution at every stage as opposed to the 2D Convolution for the MNIST data as is in the paper.

It consists of two ordinary Conv1D layers with 64 and 128 filters respectively and a kernel size of 9 x 9 and kernel strides of 2 samples along with ReLU activation. This constitutes the first layer of the model.

The second layer (PrimaryCap) is a convolutional capsule layer with 16 channels of convolutional 8D capsules (i.e. each primary capsule contains 8 convolutional units with a 9×9 kernel size and a stride of 3). One can see PrimaryCap as a convolutional layer with the *"squashing"* function as its activation.

The final layer (DigitCaps) has one 8D capsule per digit class (i.e. 2 capsules since we have 2 classes) and each of these capsules receives input from all the capsules in the layer below. We have routing only between two consecutive capsule layers (e.g. PrimaryCapsules and DigitCaps). Since conv2 output is 1D, there is no orientation in its space to agree on. Therefore, no routing is used between conv2 and PrimaryCap. [15]

A simple plot of the architecture described above is shown in Figure 5.4.

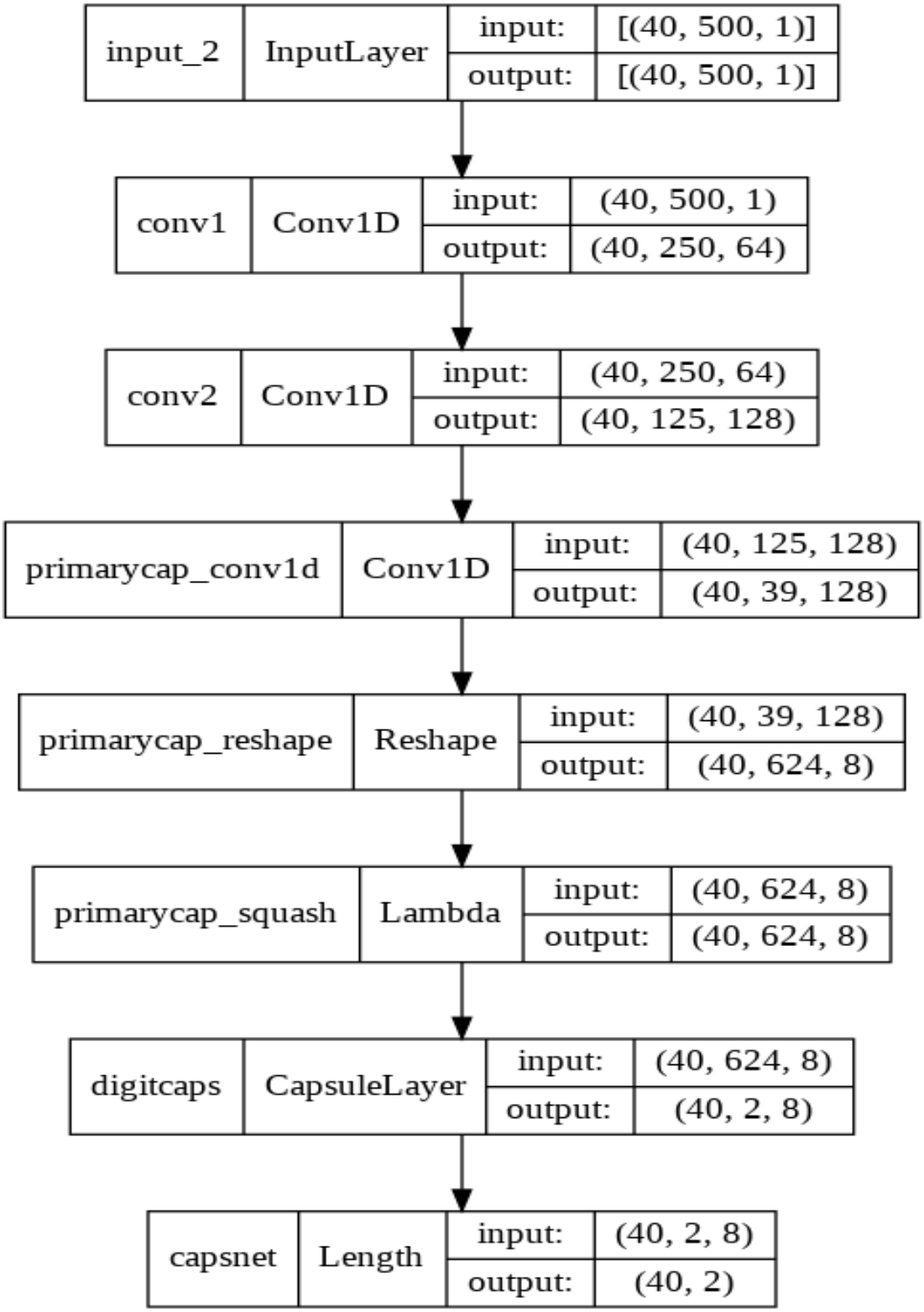

Figure 5.4: CapsNet model for estimating ToA

5.3.3 Retrieving the ToA

As described earlier, unlike the CNN model, which is a regression task and hence outputs the ToA directly given an input AE signal, what we have with CapsNet is a classification task and hence the model was trained to distinguish between valid AE events and background noise. More specifically, given an input window (similar to the training data), CapsNet outputs a probability (from 0 to 1) for the window where 1 being a valid AE event and 0 being noise.

Hence an additional post processing procedure is required that takes us from CapsNet output of window probabilities to the arrival time associated with the input AE signal. This procedure of retrieving the arrival time is laid out below:

1. First, the input AE signal is split into windows (of length 500, same as the model was trained on) as this is what CapsNet expects at it's input. The most important part of this step is that the signal is split into **overlapping windows** which strides by a small amount (10 samples) in order to have a good and smooth output of probabilities for the input signal.

2. After the above step, we have our input ready to be given to the model which outputs a number of probabilities for an input AE signal. This is equal to the number of windows generated in step 1. An input AE signal along with CapsNet output of probabilities for it is shown in Figure 5.5.

3. The final step is to estimate the arrival time from this list of probabilities. There are quite a few ways to go about this including a custom logic function which extracts the window that would most likely contain the arrival time and finally the ToA equals the mid point of the chosen window, as ToA was chosen to be in the middle of the window while training the model. However it is extremely difficult to build a custom logic function for this purpose which is able to generalize for all signals. The function would greatly depend on the type of input signals. This leads to getting great results in certain cases but equally bad results in other cases which is not acceptable. Thus in order to obtain the best possible outcome for all cases, we train a small version of our existing CNN model on the CapsNet output probabilities to estimate the

arrival time from it which we already have as labels from the original CNN training data.

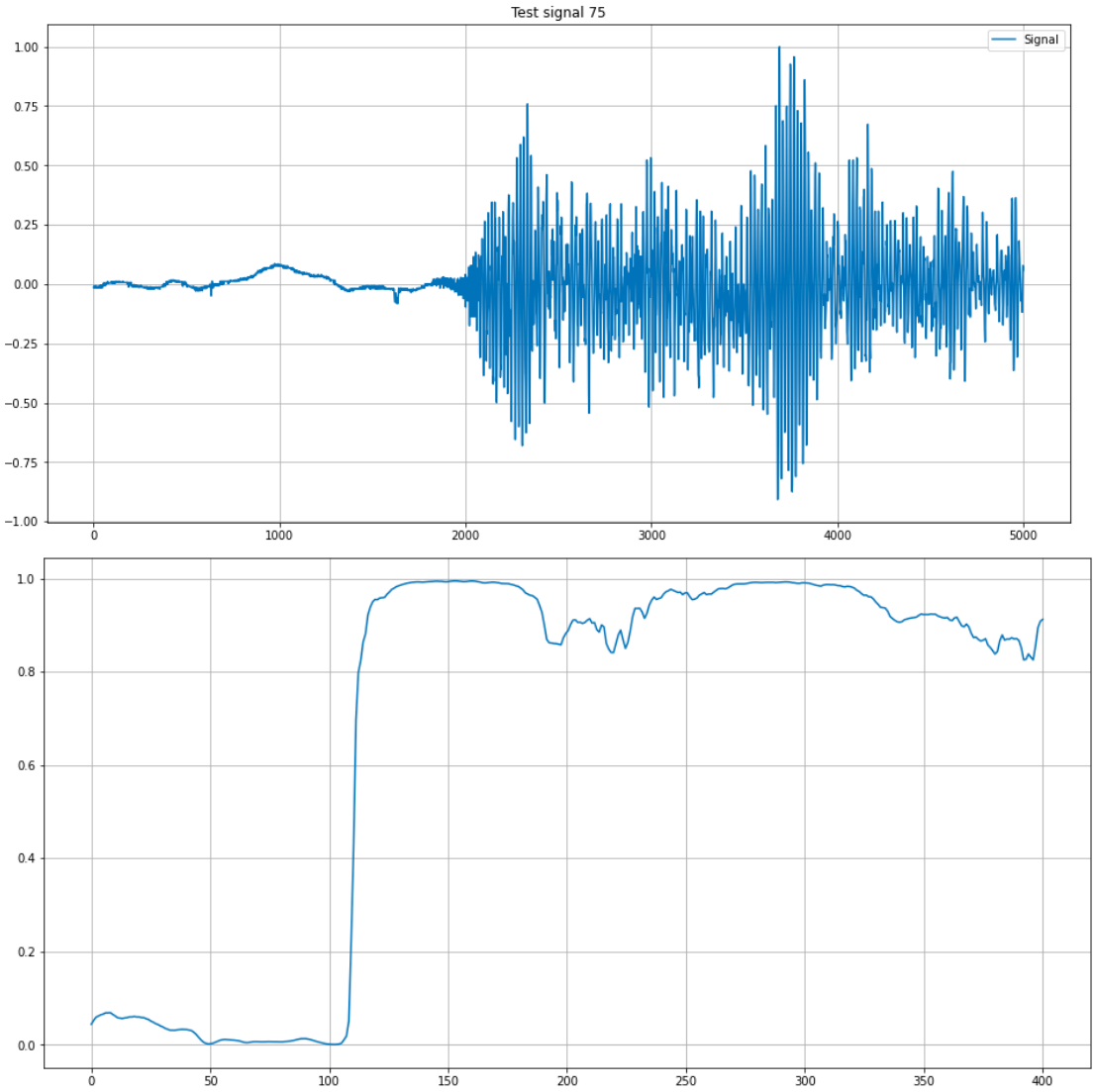

Figure 5.5: An example of CapsNet input and output

5.3.4 A smaller version of the CNN model

We will train a smaller version of the existing CNN model which will be trained on the output probabilities of CapsNet to estimate the final ToA associated with the signal.

The training set for the model is nothing but the output of CapsNet for the original training data (from SNRs 1 to 30 dB) and the labels stay the same which are the ToAs. We require a very small model to achieve this task and it is shown in Figure 5.6. Compared to the original CNN model it is ~ 10 times smaller.

```
Model: "CNN_small"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 200, 16)           176
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 100, 16)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 50, 32)            5152
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 25, 32)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 13, 64)            20544
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 6, 64)             0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 3, 64)             41024
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 1, 64)             0
_________________________________________________________________
global_average_pooling1d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1024)              66560
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 1025
=================================================================
Total params: 134,481
Trainable params: 134,481
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.6: A smaller version of the CNN model

This model performs relatively well for the task as we expected and the model's output on a sample test signal is shown in Figure 5.7.

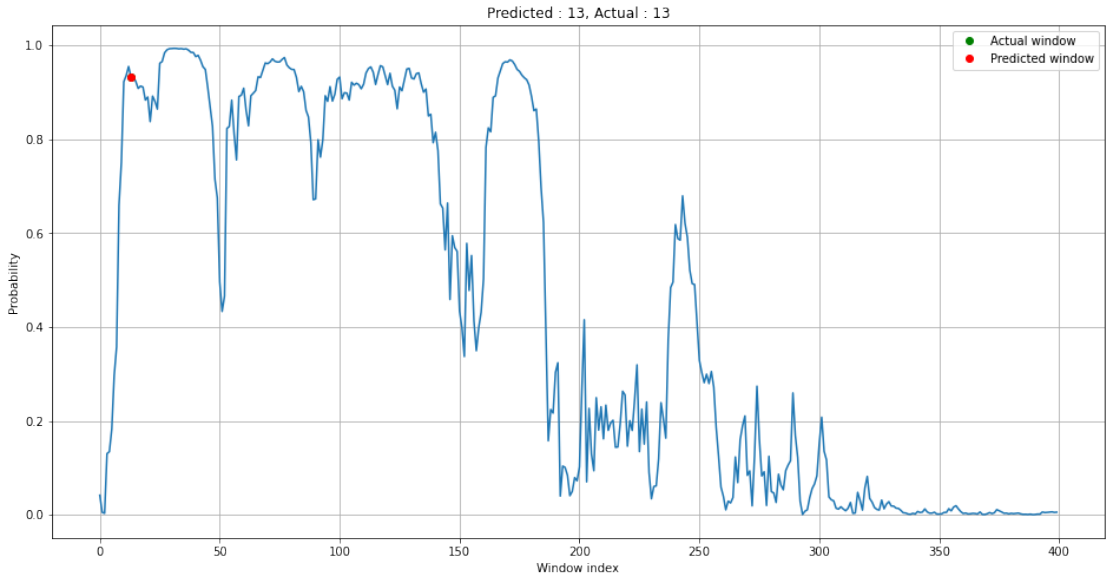

Figure 5.7: A sample output of the smaller CNN model

In this chapter, we have seen all about Capsule Neural Networks starting with what they are, how they work and finally applying them to the task of estimating the arrival time of an AE signal. And in the next chapters we shall see the results and comparisons between the different models we have built to accomplish the same task where we will talk about the pros and cons of each of them.

And we will also further include results from testing these models on real AE signals acquired in the laboratory and also extend them to the topic of localization which is one of the primary motivations for estimating the arrival time of acoustic waves and being able to localize the source of excitation (i.e. a defect in the structure) which is what's otherwise known as *Structural Health Monitoring*.

Chapter 6

Results and Comparison

We have now seen different models and their approach to the task of estimating the arrival time of AE signals. In this chapter, the results of the models are presented on the test data (which was separated from the training set before the model training begins) along with the results of AIC on the same data and we will do a comparison of their performance discussing about the pros and cons of the neural networks vs statistical methods.

6.1 Initial tests of CNN vs AIC

The very first set of tests were conducted to assess the performance of the CNN model on the test set and comparing them to the performance of AIC. Recalling from Chapter 3, the preliminary dataset we generated from the propagation algorithm consists of 20,000 AE signals.

We split this into train and test sets of 80% and 20% respectively which gives us 18,000 signals for training the model and 2,000 test signals. The model is trained for 20 epochs (i.e. the number of times the model sees and learns from the train set). The results are as follows:

```
[47] total_err = abs(y_pred - y_test) / y_test
     print(f'Average relative error : {total_err.mean()*100 : .2f} %')

     Average relative error :  2.46 %
```

```
total_aic_err = abs(y_aic_test - y_test) / y_test
print(f'Average AIC relative error : {total_aic_err.mean()*100 : .2f} %')

Average AIC relative error :  44.08 %
```

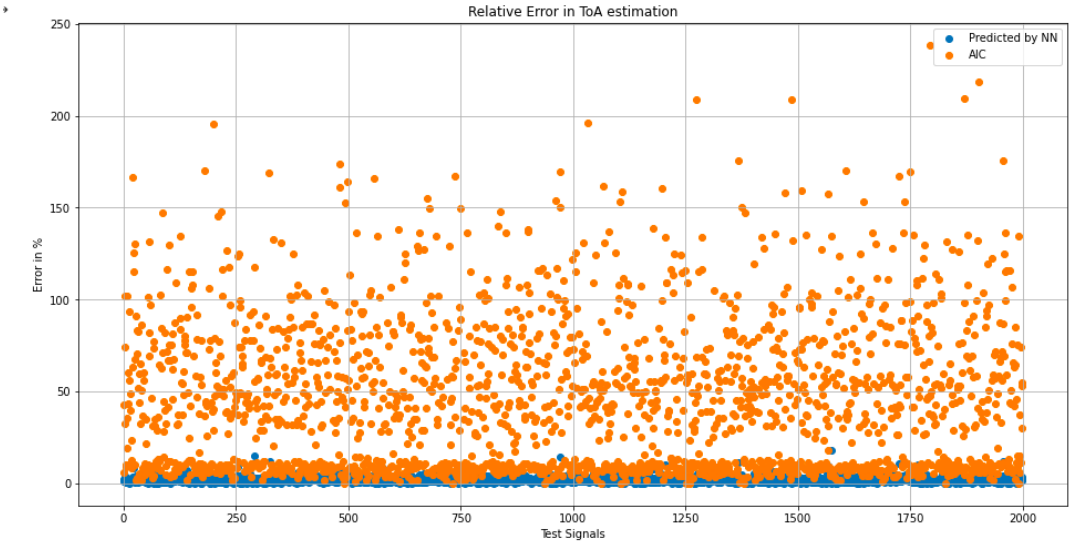

Figure 6.1: CNN vs AIC relative error on initial test set

This shows us a good performance when compared to AIC whose error goes very high for low SNR signals. And the quantity and quality of data played a very important role: initially we collected 10,000 signals with various SNRs from low to high (maximum of 30dB). The error of the model in this case was $\sim 20 - 30\%$. But upon doubling the dataset to 20,000 signals especially by including more lower SNR signals, the improvement was quite remarkable as is seen in Figure 6.1 where the average error by CNN is **less than 3%** compared to the **44% of AIC**. And it is clear from the plot that the errors by AIC are far more spread out compared to CNN. This is due to the presence of very low SNR signals in the test set where the performance of AIC drops whereas CNN stays considerably low. A more detailed table where the errors for each SNR value will be shown later where it becomes even more clear.

Another important evaluation of the performance we felt was to estimate the Gaussian distribution of the errors to better quantify the mean and standard deviation of the errors by CNN and AIC. It can be seen from Figure 6.2 the difference between the mean and standard deviation of errors by both methods. AIC has a much higher mean error and also a large deviation as compared to CNN. We can see a small peak of low AIC errors, this is due to the fact that the test set contains signals of low and high SNR values and we already know that AIC performs very well in noise free conditions. But the advantage of neural networks is apparent for low SNR signals (i.e. noisy scenarios).

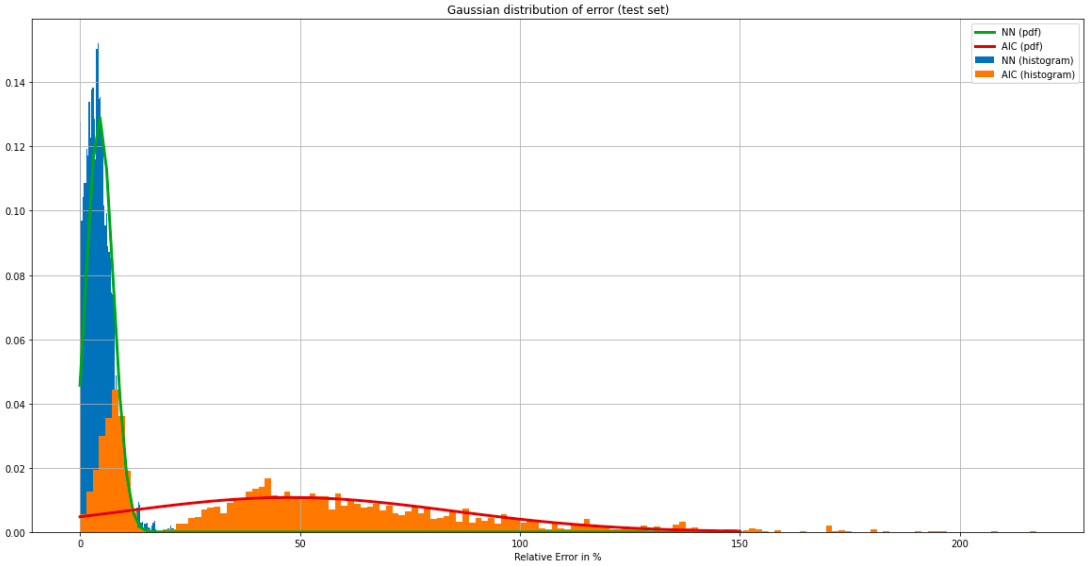

Figure 6.2: Gaussian distribution of CNN vs AIC relative errors

6.1.1 K-Fold Cross Validation

Evaluating a machine learning model can be quite tricky. Usually, we split the dataset into training and testing sets and use the training set to train the model and testing set to test the model. We then evaluate the model performance based on an error metric to determine the accuracy of the model. This is what we have seen until now. However, this method is not very robust as the accuracy obtained for one test set can be quite different sometimes to the accuracy obtained for a different test set. K-Fold Cross Validation (CV) provides a solution to this problem

by dividing the data into folds and ensuring that each fold is used as a testing set at some point.

K-Fold CV is where a given dataset is split into K number of sections/folds where each fold is used as a testing set at some point. Here we consider the case of 5-Fold cross validation (K=5) to test the model. Hence the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, the second fold is used as the testing set while the rest serve as the training set. This process is repeated until each of the 5 folds have been used as the testing set. In this way we can better evaluate the model performance avoiding any bias while making the split into train and test sets. The process of K-Fold CV is depicted in Figure 6.3.

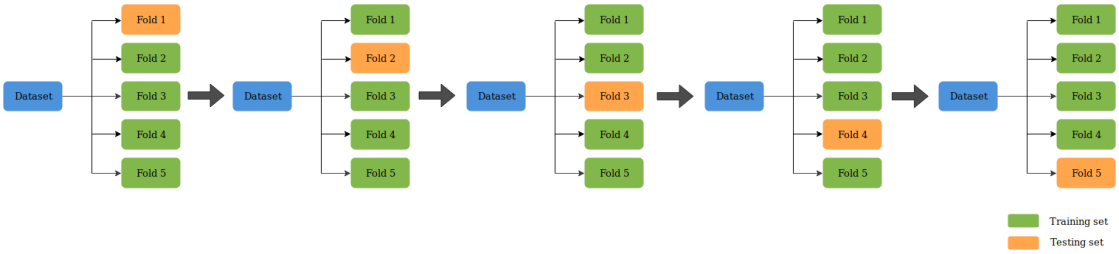

Figure 6.3: 5-Fold Cross Validation

We perform K-Fold Cross Validation on both CNN and AIC which gives us a better and more robust evaluation metric of the performance of both. The results are displayed in Figure 6.4. We can finally conclude that the average errors after testing on the entire dataset are about **2% for CNN** and about **47% for AIC**.

```
Running fold : 1/5
Average relative error :  1.54 %
Average AIC relative error :  47.62 %

Running fold : 2/5
Average relative error :  2.57 %
Average AIC relative error :  48.80 %

Running fold : 3/5
Average relative error :  1.83 %
Average AIC relative error :  47.35 %

Running fold : 4/5
Average relative error :  2.09 %
Average AIC relative error :  46.45 %

Running fold : 5/5
Average relative error :  2.01 %
Average AIC relative error :  46.49 %
```

```
err = np.array(err)
err_aic = np.array(err_aic)
print(f'Average error over {k} folds : {err.mean() : .2f} %')
print(f'\nAverage AIC error over {k} folds : {err_aic.mean() : .2f} %')
```

```
Average error over 5 folds :  2.01 %

Average AIC error over 5 folds :  47.34 %
```

Figure 6.4: K-Fold CV results of CNN vs AIC

6.2 Performance comparison : AIC vs CNN vs CapsNet

In the last section we saw the results of preliminary tests done on CNN and AIC and found that CNN was averagely performing about 20 times better than AIC overall.

We will now more closely look at the performance of each of them on every SNR value (i.e. from 1-30 dB) and we finally also include the results of CapsNet. In this way we have a complete picture of the performance of all the three methods at every SNR value. Tests were performed on two sets of data: the original signals

and on signals with increased pre-trigger times. This is done in order to cover all the possible real life scenarios and also to check if the methods show any difference in their performance between the two cases.

In addition to the relative error, we add another metric which is the Root Mean Squared Error (RMSE) in estimating the arrival time in order to better evaluate the performances due to the fact that the RMSE is a measure of the actual divergence with respect to the true values only whereas the relative error includes normalizing the divergence with the true labels. And the value of the true labels will be higher for increased pre-trigger signals which in turn reduced the relative error but the RMSE stays independent of it which gives us a better idea of the divergence from the true values.

6.2.1 Original signals dataset

Here, the relative errors and RMSE by the three methods on the original signals for every SNR value are displayed in Table 6.1 and Table 6.2 respectively.

What can be noticed is that the errors by all three methods are fairly similar around SNR=30 but a major difference is seen at low SNR values. The errors by AIC keep increasing as we further reduce the SNR whereas CNN and CapsNet increase by a very insignificant amount. The results by neural networks are a lot more stable.

SNR	AIC	CNN	CapsNet
1	64.38%	2.03%	5.17%
2	63.48%	1.96%	4.32%
3	63.12%	1.93%	3.78%
4	62.45%	1.91%	3.45%
5	61.69%	1.94%	3.22%
6	61.18%	1.98%	3.12%
7	60.53%	2.01%	2.90%
8	59.41%	2.03%	2.81%
9	57.45%	2.03%	2.73%
10	55.24%	2.04%	2.75%
11	52.39%	2.06%	2.69%
12	48.54%	2.04%	2.63%
13	42.96%	2.05%	2.62%
14	37.02%	2.04%	2.64%
15	29.96%	2.05%	2.60%
16	24.14%	2.04%	2.64%
17	19.52%	2.05%	2.65%
20	12.00%	2.05%	2.63%
25	6.22%	2.04%	2.74%
30	3.75%	2.03%	2.77%

Table 6.1: Relative errors on original signals dataset

SNR	AIC	CNN	CapsNet
1	243.18	10.31	26.25
2	240.27	10.07	21.21
3	238.72	9.53	18.26
4	236.77	8.95	16.96
5	234.61	8.92	15.89
6	233.19	9.09	15.19
7	231.92	9.09	14.41
8	229.94	9.15	13.79
9	226.52	9.15	13.37
10	222.56	9.07	13.5
11	217.7	9.16	13.22
12	210.23	9.14	12.95
13	196.85	9.13	13.01
14	181.1	9.08	13.11
15	157.54	9.1	13.04
16	135.85	9.06	13.08
17	116.74	9.11	13.19
20	76.85	9.08	13.28
25	34.96	9.06	13.86
30	16.02	9.03	14.05

Table 6.2: RMSE on original signals dataset

6.2.2 Increased pre-trigger signals dataset

We now see the results of the same metrics (relative error and RMSE) on the dataset of signals with their pre-trigger times increased. We can make two important observations from the results that are displayed in Tables 6.3 and 6.4:

1. As stated earlier, we will see a reduction in the relative errors. This is because of the way the metric is defined which normalizes the deviations with respect to the true values and in this case of increased pre-trigger signals, the true

values are greater compared to the original signals dataset. For this reason, RMSE is introduced and it remains fairly similar under both cases for AIC.

2. The second important observation is that in this case CapsNet outperforms CNN. The reason being that, CapsNet struggles a little for signals with low arrival times due to the fact that while training CapsNet, the input comprised of windows (of length 500 samples) and hence we don't end up with a clear probability curve at the output for signals with lower arrival times as the curve has a very high probability value from the start itself.

SNR	**AIC**	**CNN**	**CapsNet**
1	9.74%	3.99%	0.93%
2	9.69%	3.59%	0.78%
3	9.61%	3.66%	0.69%
4	9.56%	3.31%	0.64%
5	9.41%	3.54%	0.60%
6	9.32%	3.38%	0.55%
7	9.25%	3.34%	0.55%
8	9.12%	3.31%	0.57%
9	8.81%	3.34%	0.54%
10	8.47%	3.37%	0.55%
11	8.11%	3.12%	0.57%
12	7.43%	3.52%	0.56%
13	6.63%	2.96%	0.56%
14	5.53%	3.47%	0.57%
15	4.42%	3.30%	0.58%
16	3.44%	3.39%	0.57%
17	2.79%	3.21%	0.58%
20	1.69%	3.47%	0.58%
25	0.96%	3.25%	0.58%
30	0.60%	3.33%	0.63%

Table 6.3: Relative errors on increased pre-trigger signals dataset

SNR	AIC	CNN	CapsNet
1	243.18	179.87	29.33
2	240.27	165.96	23.78
3	238.72	170.53	21.46
4	236.77	158.95	19.77
5	234.61	167.17	18.25
6	233.19	158.63	17.17
7	231.92	158.29	16.97
8	229.94	158.64	17.61
9	226.52	159.95	16.71
10	222.56	162.33	16.6
11	217.7	154.71	17.66
12	210.23	170.15	17.42
13	196.85	148.28	17.09
14	181.1	163.86	17.45
15	157.54	163.58	17.93
16	135.85	162.78	17.46
17	116.74	157.92	17.69
20	76.85	163.67	17.51
25	34.96	159.31	17.89
30	16.02	160.23	18.85

Table 6.4: RMSE on increased pre-trigger signals dataset

We have seen that both the neural networks (CNN and CapsNet) outperform statistical methods like AIC in noisy scenarios. But we were dealing with signals from the synthetic dataset up until now. In the next chapter we are going to see results of the three models on real signals that were collected in the laboratory. This would be a real life test of the models in comparison to AIC.

Chapter 7

Laboratory Experiments

In this chapter, we will discuss about the experimental setup employed to generate and collect AE signals and test the performance of all the three methods on them. We also perform localization test using the difference time of arrival (dToA) estimation.

7.1 Experimental Setup

An aluminium plate of dimensions 1000mm x 1000mm and a thickness of 3mm was used as the medium of propagation similar to the one in the simulation environment. Sensor Node (SN) system was used to collect the signals. It consists of three channels that were placed on the plate. A total of nine excitation points were setup on the plate. The signal parameters are: 1V p-p sinusoidal, 100KHz, 10 cycles and the experimental setup is depicted in Figure 7.1.

The positions of the three channels (in meters) are:

- ch1: (0.5, 0.95)
- ch2: (0.5, 0.5)
- ch3: (0.95, 0.5)

And the positions of the nine excitation points (in meters) are:

- Point 1: (0.4, 0.4)

- Point 2: (0.55, 0.4)
- Point 3: (0.7, 0.4)
- Point 4: (0.4, 0.55)
- Point 5: (0.55, 0.55)
- Point 6: (0.7, 0.55)
- Point 7: (0.4, 0.7)
- Point 8: (0.55, 0.7)
- Point 9: (0.7, 0.7)

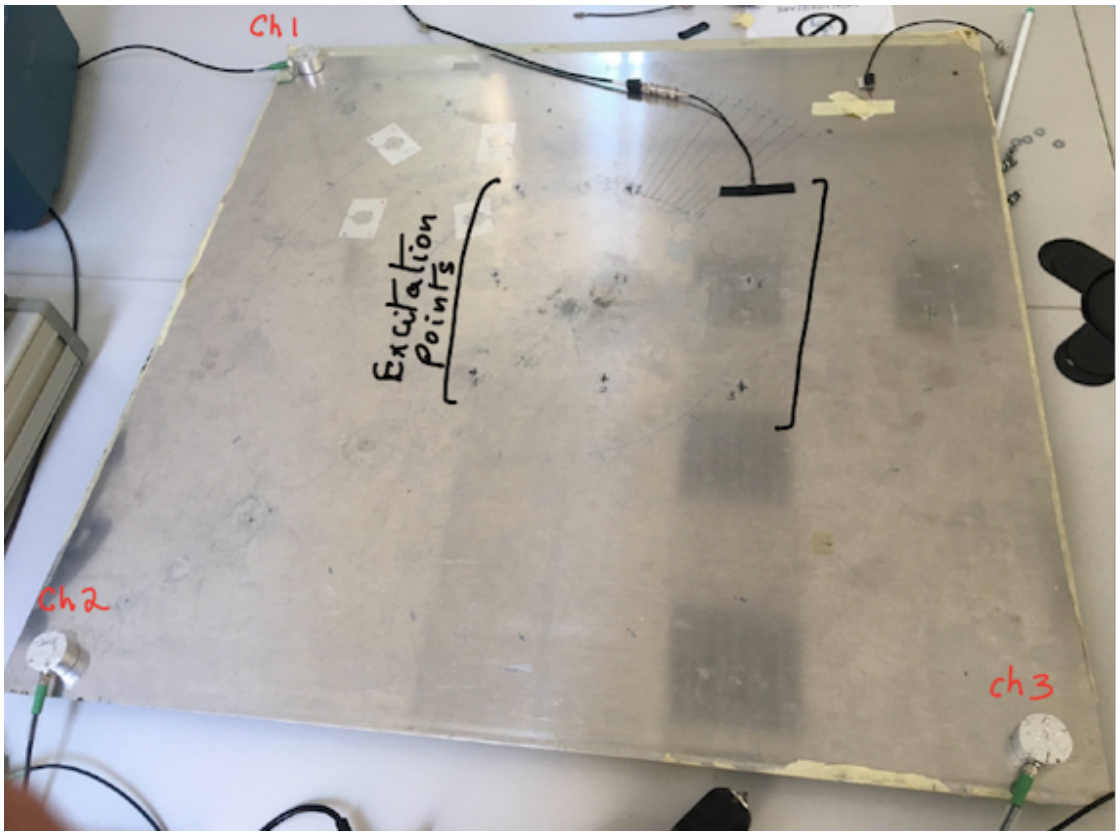

Figure 7.1: Experimental setup in SHM lab

7.2 Describing the experiments

1. **Localization experiment:** We collected three sets of signals at each excitation point. This gave us a total of 3 x 9(points) x 3(channels) = 81 signals.

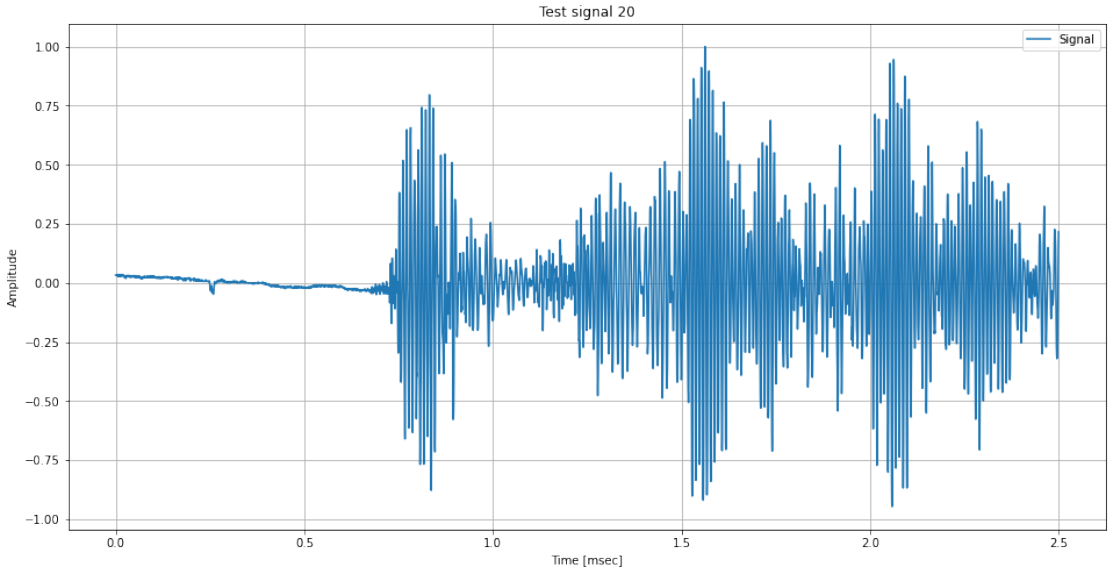

Figure 7.2: A test signal from localization experiment

2. **Drill experiment:** Excitation point 5 at (0.55, 0.55) was fixed and a total of 31 x 3(channels) = 93 signals were collected by changing their pre-trigger times from 1000:50:2500 samples with the addition of an externally induced noise source in the form of a power drill during the signal acquisition.

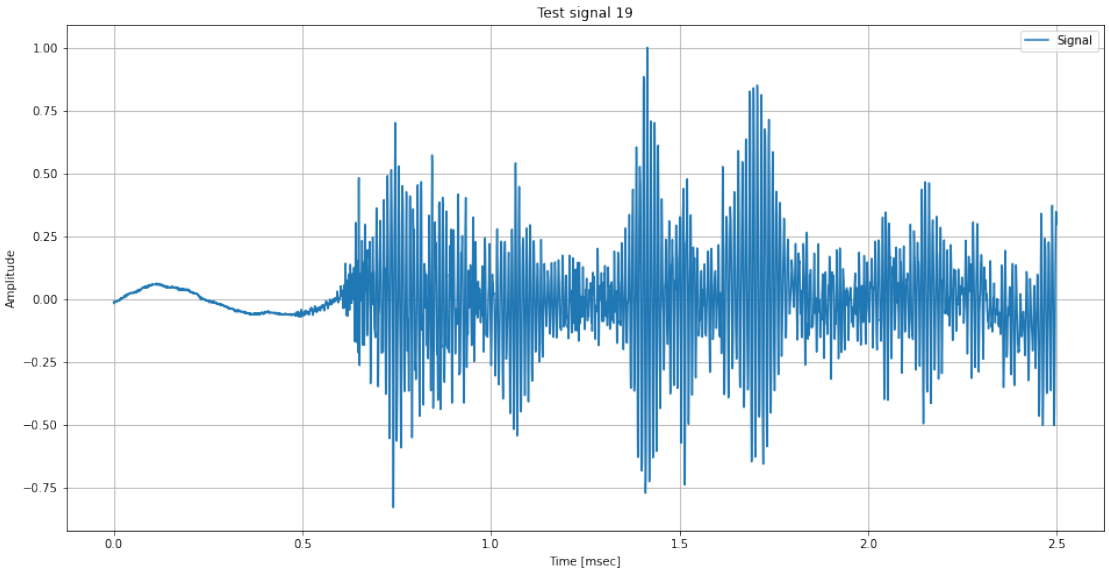

Figure 7.3: A test signal from drill experiment

3. **AWG noise experiment:** For this experiment, all the details of the drill experiment were kept the same but this time the noise source was replaced with an arbitrary noise output from a second AWG with 5V p-p and its output connected to the buzzer PZT disc placed on the upper right part of the sheet (as can be seen in Figure 7.1). Similarly, as before 31 x 3(channels) = 93 signals were collected by changing their pre-trigger times from 1000:50:2500 samples.

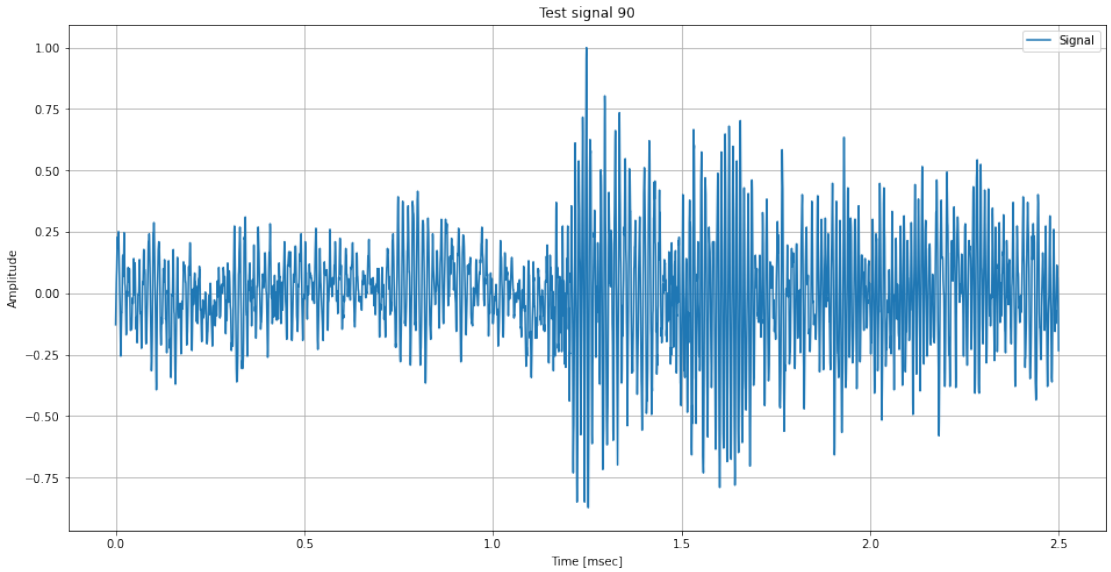

Figure 7.4: A test signal from AWG noise experiment

7.3 Initial Results

Recollecting from Chapter 3, in the section of data augmentation, it was mentioned that adding signals with high pre-trigger times was very important, we will talk about it now.

After having good results on the synthetic dataset, we proceeded to test the CNN model on the experimental signals *before* augmenting the dataset by adding signals with increased pre-trigger times. Upon looking at the results, they could be classified into two buckets : a good result and a bad result. The good results were for the signals with low arrival times and the bad results were for the signals with higher arrival times. This is depicted in Figure 7.5 where we see two such instances of a good and bad result.

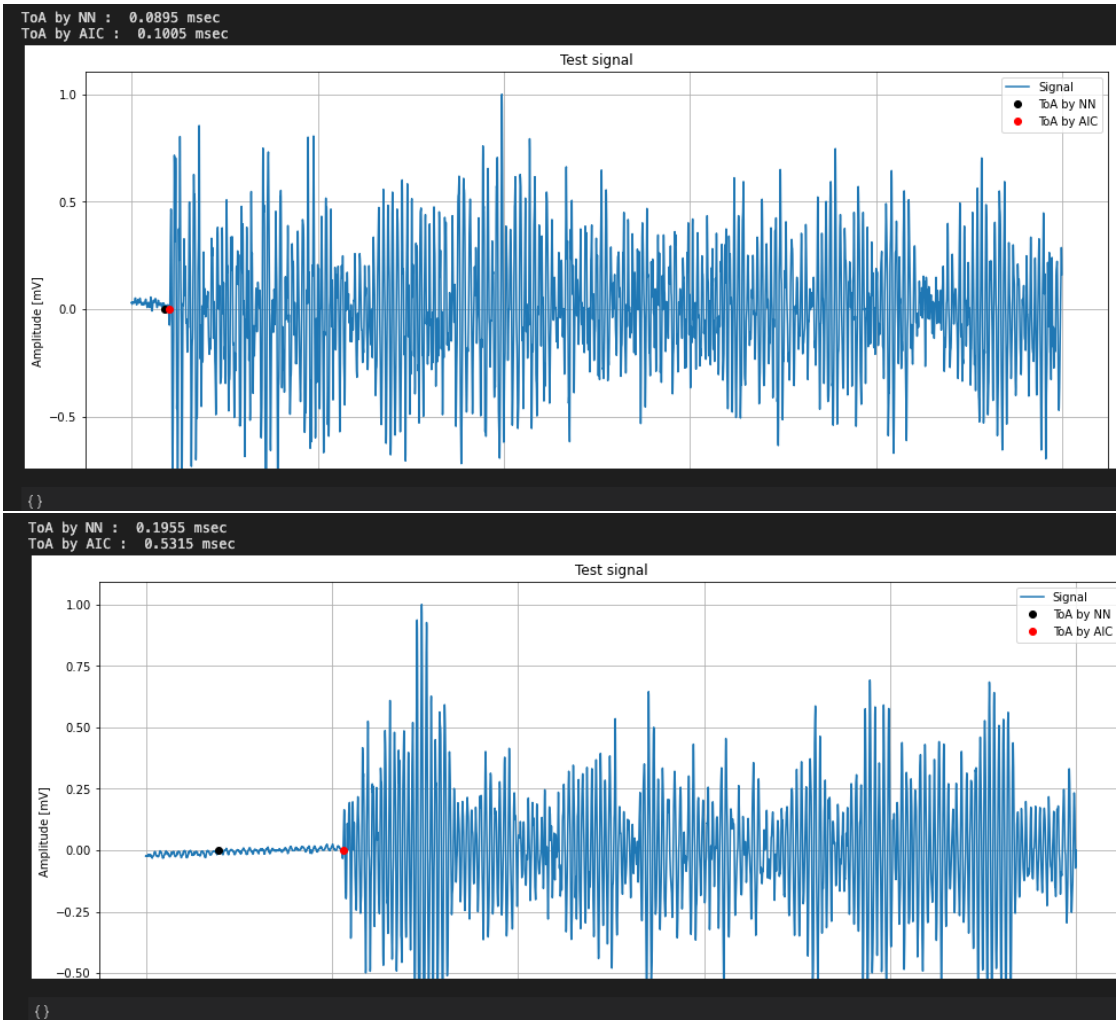

Figure 7.5: An example of a good result (above) vs a bad result (below)

It was clear now that we needed to include signals with higher arrival times to the training set as the model initially could never predict such high arrival times because it had never seen such kind of signals. After this addition, the model was able to estimate higher arrival time signals and the results were improved significantly as is shown in Figure 7.6.

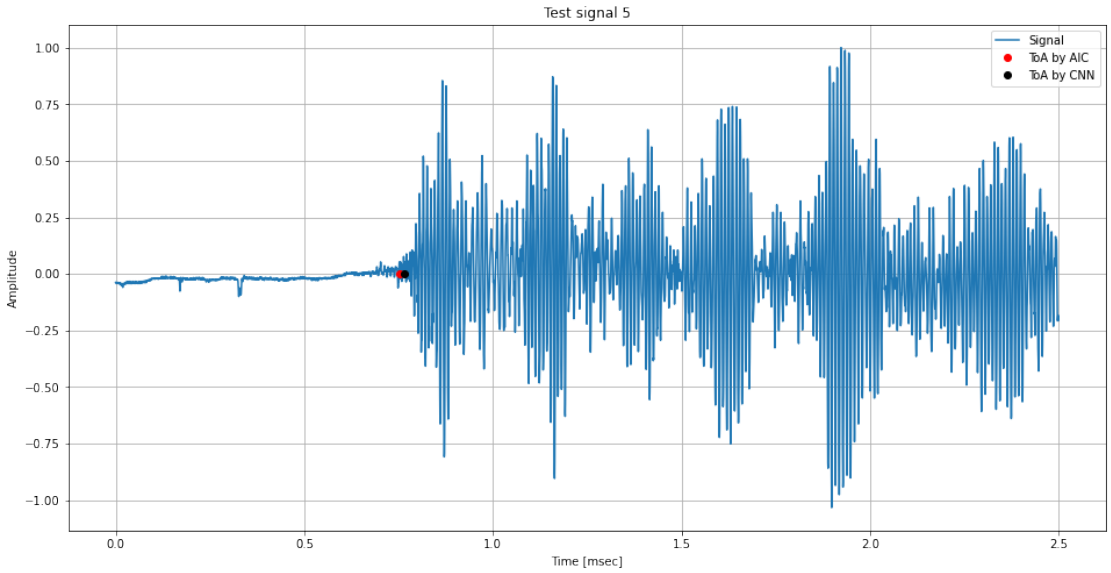

Figure 7.6: A good result for a signal with a higher arrival time

7.4 Difference Time of Arrival (dToA) tests

The primary purpose of this task of estimating the arrival times of AE signals is to be able to localize the source (excitation point) of the acoustic waves which occurs due to internal defects formed in the structure. For this purpose, we require to calculate the difference time of arrival for the 3-sensor array system we have in place. To be more specific, the two dToAs are defined as follows:

- **dToA21:** ToA of ch2 − ToA of ch1
- **dToA31:** ToA of ch3 − ToA of ch1

We estimate the two dToAs for each signal (consisting of 3 channels) acquired for the drill and AWG noise experiments and compare it with the theoretical dToAs which can be mathematically calculated as we know the distances between the 3 channels (sensors) and the location of the source (i.e. excitation point 5). And we also know the group velocity of the signals which is 5371 meters/sec from the dispersion curves on the aluminium plate.

Drill experiment

The average error and standard deviation results in estimating the two difference time of arrivals (dToA21 and dToA31) with respect to the theoretical values for the drill experiment by CNN, CapsNet, AIC are shown in Figures 7.7 and 7.8 below.

Average Error

- dToA Ch2-Ch1
  7.9E-05 (CNN) vs 12.9E-05 (AIC)
- dToA Ch3-Ch1
  1.6E-05 (CNN) vs 4.9E-05 (AIC)

Standard Deviation

- dToA Ch2-Ch1
  1.3E-05 (CNN) vs 19.1E-05 (AIC)
- dToA Ch3-Ch1
  1.4E-05 (CNN) vs 14.1E-05 (AIC)

Ratio Average NN / Average AIC	
dToA21	6.11E-01
dToA31	3.40E-01
Ratio Stan deviat NN /	AIC
dToA21	7.03E-02
dToA31	1.00E-01

Figure 7.7: Drill experiment: dToA error CNN vs AIC

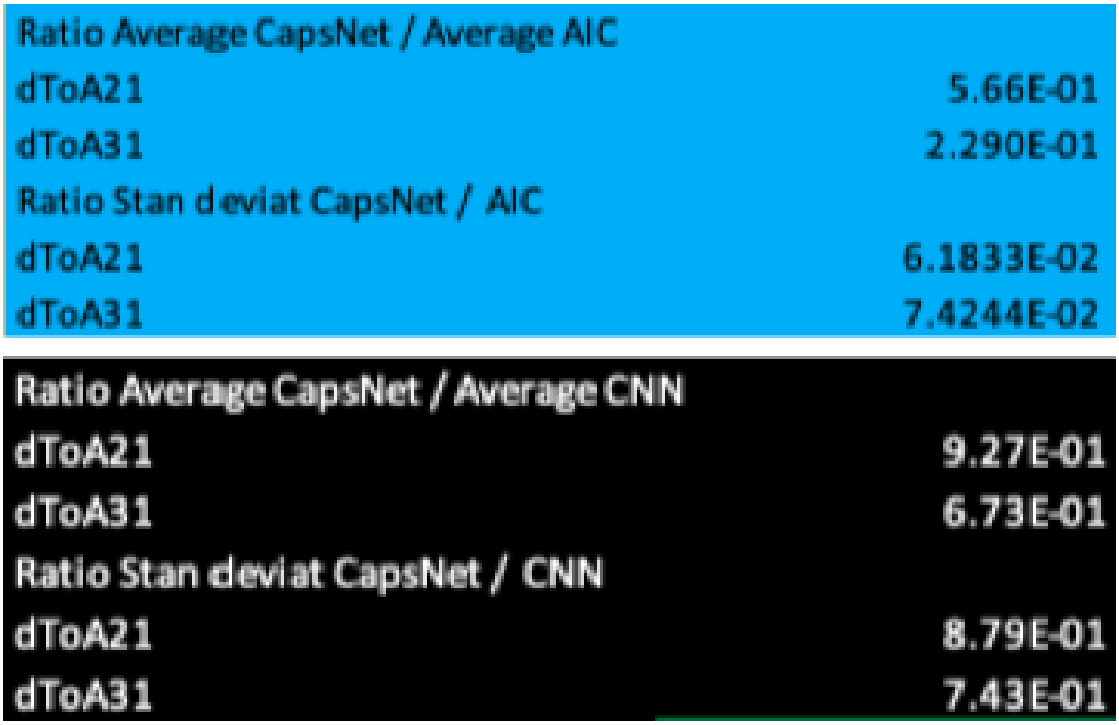

Figure 7.8: Drill experiment: dToA error CapsNet vs CNN vs AIC

AWG noise experiment

The same metrics are used to compare the performance of the three methods in estimating the two dToAs for AWG noise experiment which are shown in Figures 7.9 and 7.10.

Average Error

- dToA Ch2-Ch1
  9.9E-05 (CNN) vs 12.6E-05 (AIC)
- dToA Ch3-Ch1
  7.1E-05 (CNN) vs 17.9E-05 (AIC)

Standard Deviation

- dToA Ch2-Ch1
  8.7E-05 (CNN) vs 39.4E-05 (AIC)
- dToA Ch3-Ch1
  6.5E-05 (CNN) vs 44.6E-05 (AIC)

Ratio Average NN / Average AIC	
dToA21	0.785182361
dToA31	0.401024351
Ratio Stan deviat NN / AIC	
dToA21	0.221502527
dToA31	0.146705527

Figure 7.9: AWG noise experiment: dToA error CNN vs AIC

Ratio Average CapsNet / Average AIC	
dToA21	4.64E-01
dToA31	1.97E-01
Ratio Stan deviat CapsNet / AIC	
dToA21	1.57E-01
dToA31	7.36E-02

Ratio Average CapsNet / Average CNN	
dToA21	5.90E-01
dToA31	4.91E-01
Ratio Stan deviat CapsNet / CNN	
dToA21	7.10E-01
dToA31	5.02E-01

Figure 7.10: AWG noise experiment: dToA error CapsNet vs CNN vs AIC

It is quite clear from the results that the neural networks have more than an order better performance in estimating the difference time of arrival when compared to AIC. This applies to both the **average error and standard deviation.** And comparing CNN and CapsNet, **CapsNet performs better than CNN** in this line of testing.

The end results match the initial tests performed on the synthetic dataset that the real advantage of the neural networks over statistical methods is under the influence of noise where they outperform by a decent margin as is evident from both the tests performed on the synthetic dataset and further extended to tests on real signals collected in the laboratory is was seen in the last two chapters.

7.5 Influence of noise on AIC

To further demonstrate the above mentioned point, we can have a close look at the performance of AIC under noise-free conditions vs noisy scenarios. In Figure 7.11, we see the two cases for the same signal. One corresponds to AIC output on the original signal (i.e. noise-free) and the other is the same signal passed to AIC after adding synthetic noise to it.

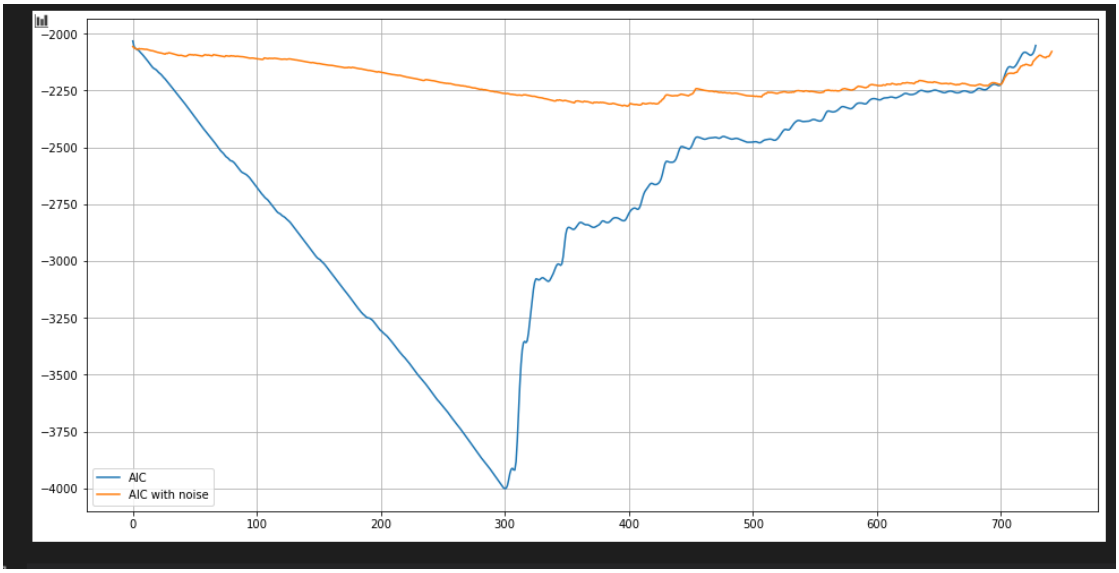

Figure 7.11: Influence of noise on AIC

It is quite evident from the AIC trend that in the noise-free case, there is a very clear global minimum which is taken to be the arrival time. But when noise is added, there is no clear global minimum in this case which leads to incorrect estimation of the arrival time.

Conclusion

7.6 Thoughts and Findings

This thesis work dealt with building and testing Deep Learning models to estimate the time of arrival of acoustic signals. After prototyping a few different models while deciding which could be best suited for the particular task, we presented two models based on Convolutional Neural Network (CNN) and Capsule Neural Network (CapsNet).

As is the case with every Deep Learning task, what was realized that data plays a very important role in the performance of the models. Not just the quantity of data but also the quality of data collected. The initial results presented showed that the error in estimating ToA was significantly reduced after training the models on a diverse set of data and by also increasing the amount to double the initial set. After the preliminary testing phase and making necessary changes to the dataset, we start to see good results.

It is a known fact that statistical methods like AIC have shown great results in estimating the arrival times of signals when there is a clear change in the signal statistics i.e. in the absence of noise. But these methods are prone to the addition of noise. The error in this case by AIC was found to be increased by **more than an order of magnitude** by reducing the SNR of the signals while CNN and CapsNet **show similar performance with only a slight increase in their errors** for the same low SNR signals.

The experiments conducted in the lab to acquire real signals for the models to test on showed a similar result. AIC proved to be very sensitive to a slight presence of noise which leads to incorrect estimation of ToA. While CNN and CapsNet not

only have their average errors much lower than AIC but they also have much stable performance as was measured by the standard deviation of the errors.

But as the saying goes: *"There is no such thing as a free lunch"*, this improvement in accuracy and robustness as compared to statistical methods also comes with it's drawbacks. These particularly deal with the huge memory requirements and computation power needed to deploy and make predictions using these models. This makes it a very difficult task to be able to deploy such models on embedded systems with extremely limited memories for instant and on device data processing without the need of sending the data over cloud services to a central computing station.

7.7 Future work

Given that these models have good results on the tests presented in this work and the fact that they require huge memory to deploy and make predictions, the areas to further work on could be the following:

1. **Optimizing the models:** Reducing the number of parameters and the subsequent size of the models with a slight decrease in performance could be effort worthy. There is a definitive trade-off between performance vs efficiency. It could vary depending on particular tasks and needs. And apart from this, further optimizations are possible through post-training quantization of the models that can reduce model size while also improving CPU and hardware accelerator latency, with little degradation in model accuracy.

2. **Improving the dataset:** The dataset can be further improved by collecting and adding more signals that could cover more scenarios of acoustic emissions. The dataset generated as part of this work, certainly does not cover every imaginable scenario. As we have already seen the improvement in model performance by increasing the dataset, this process can further go on and the results do indicate that time and effort spent on this is certainly worthy.

List of Figures

List of Tables

Bibliography

[1] F Bai, Daniel Gagar, Peter Foote, and Yifan Zhao. "Comparison of alternatives to amplitude thresholding for onset detection of acoustic emission signals". In: *Mechanical Systems and Signal Processing* 84 (2017), pp. 717–730.

[2] Daniel Balageas, Claus-Peter Fritzen, and Alfredo Güemes. *Structural health monitoring*. Vol. 90. John Wiley & Sons, 2010.

[3] Md. Tawhidul Islam Khan. "Structural Health Monitoring by Acoustic Emission Technique". In: *Structural Health Monitoring from Sensing to Processing*. Ed. by Magd Abdel Wahab, Yun Lai Zhou, and Nuno Manuel Mendes Maia. Rijeka: IntechOpen, 2018. Chap. 2. DOI: 10.5772/intechopen.79483. URL: https://doi.org/10.5772/intechopen.79483.

[4] Stress Engineering Services Inc. *An Overview on Acoustic Emission Testing*. Available at https://www.stress.com/acoustic-emission-testing-overview/. [Online; accessed 17-November-2021].

[5] Charles R Farrar and Keith Worden. *Structural Health Monitoring: A Machine Learning Perspective*. John Wiley & Sons, 2012.

[6] Samira Gholizadeh, Z Leman, and BT Hang Tuah Baharudin. "A review of the application of acoustic emission technique in engineering". In: *Structural Engineering and Mechanics* 54.6 (2015), pp. 1075–1095.

[7] Avraham Berkovits and Daining Fang. "Study of fatigue crack characteristics by acoustic emission". In: *Engineering Fracture Mechanics* 51.3 (1995), pp. 401–416.

[8] Zachary E Ross, Christopher Rollins, Elizabeth S Cochran, Egill Hauksson, Jean-Philippe Avouac, and Yehuda Ben-Zion. "Aftershocks driven by afterslip and fluid pressure sweeping through a fault-fracture mesh". In: *Geophysical Research Letters* 44.16 (2017), pp. 8260–8267.

[9] Yangkang Chen. "Automatic microseismic event picking via unsupervised machine learning". In: *Geophysical Journal International* 222.3 (2020), pp. 1750–1764.

[10] Zachary E Ross, Men-Andrin Meier, and Egill Hauksson. "P wave arrival picking and first-motion polarity determination with deep learning". In: *Journal of Geophysical Research: Solid Earth* 123.6 (2018), pp. 5120–5129.

[11] Amazon Web Services. *What is data labeling for machine learning?* Available at `https://aws.amazon.com/sagemaker/data-labeling/what-is-data-labeling/`. [Online; accessed 20-November-2021].

[12] F. Zonzini, D. Bogomolov, T. Dhamija, L. De Marchi, and A. Marzani. "Artificial Intelligence Algorithms for Time of Arrival Estimation in Acoustic Emission Signals". In: SAFAP 2021: Safety and Reliability of Pressure Equipment. 2021.

[13] Sai Balaji. *Binary Image classifier CNN using TensorFlow.* Available at `https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697`. [Online; accessed 17-November-2021].

[14] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition.* Available at `https://cs231n.github.io/convolutional-networks/`. [Online; accessed 17-November-2021].

[15] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic routing between capsules". In: *arXiv preprint arXiv:1710.09829* (2017).