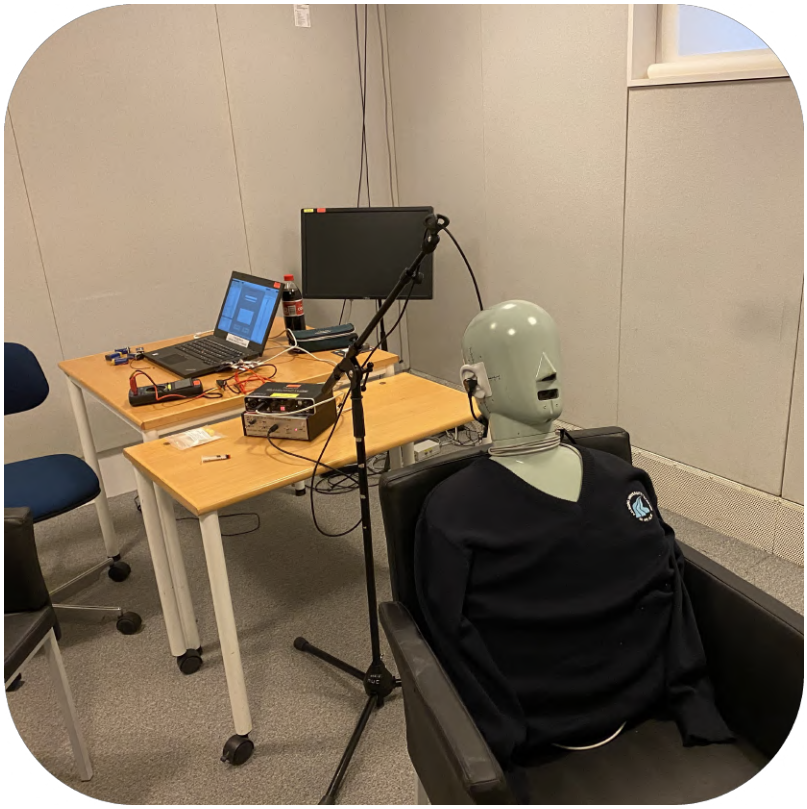
Using Transient Evoked Otoacoustic Emissions For Biometric Identification

A 8th semester's Computer Engineering project

Project Report Group 842



Aalborg University
Department of Electronic Systems

Department of Electronic Systems
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Using Transient Evoked Otoacoustic Emissions For Biometric Identification

Theme:
Scientific Theme

Project Period:
Spring Semester 2023

Project Group:
Group 842

Participant(s):
Damian Dariusz Herman
Kazim Kerem Kocan
Laurits Winter
Sebastian Michael Ramm

Supervisor(s):
Dorte Hammershøi

Copies: 1

Page Numbers: 92

Date of Completion:
May 19, 2024

Abstract:

This report investigates the possibility of using a 1-Dimensional CNN to identify a person based on their Transient Evoked Otoacoustic Emissions. This is done by first analysing the properties of Otoacoustic Emissions, CNNs and Biometric authentication.
From the knowledge gathered there, the problem statement of *"Given a personal device meets the technical constraints measuring TEOAEs and the user has healthy ears, is it possible to create a TEOAE-based deep learning model to identify and authenticate a user for the device within a practical time frame and memory amount?"* was created.
After that, a dataset of TEOAEs from 39 different people was created by combining new data with an existing dataset of 15 individuals. This data was then used to train and test a 1D-CNN model which showed that TEOAEs are usable as a biometric identifier in a deep-learning model.

Contents

Preface

This report was written as part of the 8th semester in the computer engineering masters programme specialising in AI, vision, and sound at Aalborg University, Aalborg. This report which is called "Using transient evoked otoacoustic emissions for biometric identification" was made in the semester module for "Acoustics, Sound Processing and Sound Perception" in spring 2023 for 15 ECTS. The project was supervised by Dorte Hammershøi.

Our deepest gratitude goes to Rodrigo Ordoñez for his guidance in the project as well as for providing the code for measuring OAEs and the base dataset, which we have updated and expanded. We would also like to acknowledge the many students and staff at AAU for providing their time and ears to help expand the previously mentioned TEOAE dataset.

Aalborg University, May 19, 2024

Damian Dariusz Herman
<dherma22@student.aau.dk>

Kazim Kerem Kocan
<kkocan19@student.aau.dk>

Laurits Winter
<lwinte19@student.aau.dk >

Sebastian Michael Ramm
<SRamm19@student.aau.dk>

Chapter 1

Introduction and Motivation

For the past decade, interest in biometrics has slowly been on the rise [1]. With further improvements to existing biometrics, such as fingerprints, Apple's release of Face ID back in 2017 and Microsoft's Windows Hello [2], biometrics have become mainstream and have been integrated into many devices, especially smartphones. Despite the general trust in biometrics, researchers and developers in the field have long known that biometrics is not a perfect system, but is prone to make mistakes, and have been working hard to find solutions. Among these, the approach is generally split between refining the existing biometrics and diversifying the field, by finding new distinct characteristics which can be used for identification. With the rising utilization of biometrics, in particular fingerprints, some researchers have recently been attempting to develop cyber-attack software to breach the system, with the latest, as of writing being BrutePrint, which can breach current biometrics, sometimes in as little as 40 minutes [3]. One approach to solving issues such as this is to rely on more than one type of biometric. With multi-factor authentication, the level of security grows exponentially with the amount of biometrics involved, relative to their individual level of security. In the context of sound and acoustics, the Electronic systems department at AAU has suggested researching the viability of using Oto-Acoustic Emissions as a potential biometric for authentication. The scope of the project is not necessarily to develop a full implementation, but rather to test the viability, and Find potential challenges, of such a system. Based on the proposed problem, an initial problem statement is formed:

1.1 Initial problem statement

What are the characteristics and traits of OAE, in the context of biometric authentication?

Chapter 2

Problem analysis

The following chapter analysis the initial problem statement, by first explaining the auditory system in general, then what otoacoustic emissions are and different ways to measure them. Lastly, the chapter explains the steps that are required for making biometric authentication, as well as OAEs viability in being a biometric identifier through possibly machine learning.

2.1 Auditory System

To understand the nature of OAEs, a general understanding of the auditory system, as well as the source of OAEs is required. To achieve this, a general description of the functions of the auditory system and its parts has been made.

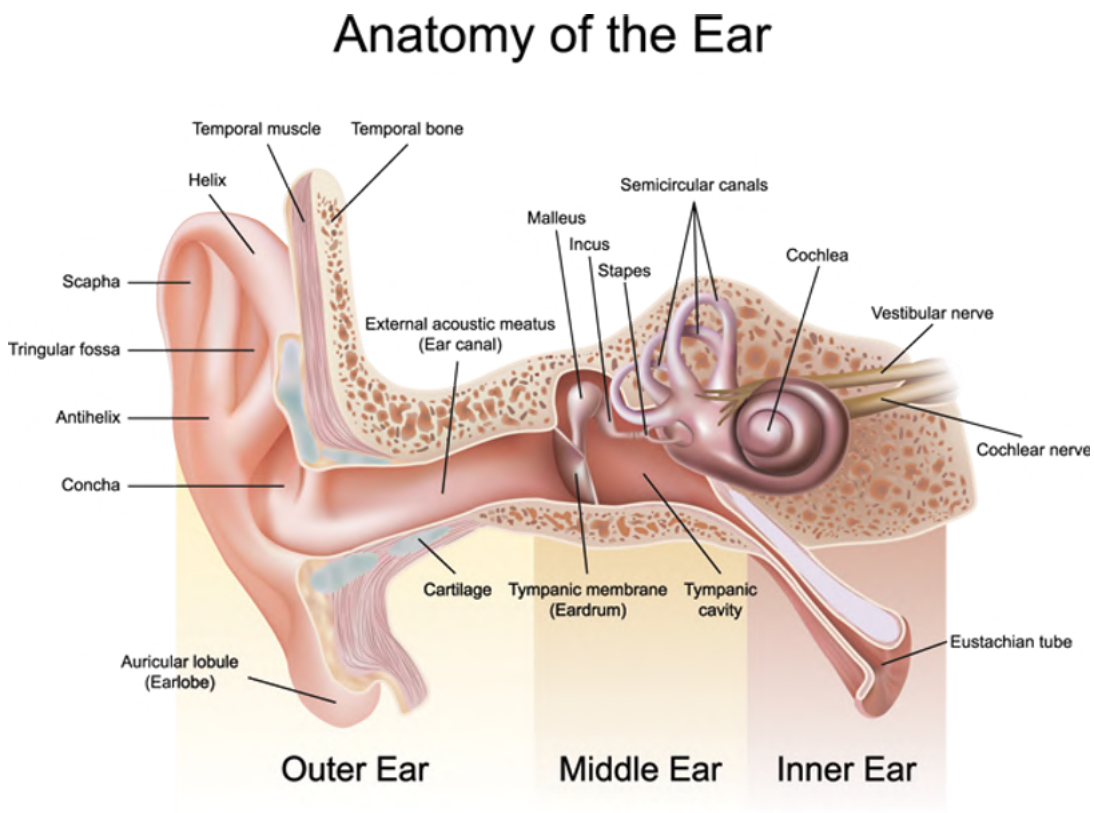


Figure 2.1: Graphic illustration of the auditory system[4]

2.1.1 The outer ear

The outer ear consists primarily of the Auricle and the Ear canal. The Auricle encompasses the exposed part of the ear, from the earlobe to the helix. The auricle is specialized to help determine the directional orientation of the sound, to locate its source. It achieves this, by reflecting the sound off of the various folds and ridges, such that the reflected sound will follow the initial sound wave into the ear canal with a slight delay. The offsets vary, depending on which part of the Auricle caused the reflection, which can be used to determine the direction of the sound [5].
The ear canal serves as a pathway, to move these sounds to the middle ear, located deeper in the skull.

2.1.2 The middle ear

As the offset sound wave pairs enter the middle ear, where we find the Tympanic membrane, the Tympanic cavity, the Eustachian tube, and the Ossicles, consisting of the Malleus, Incus, and Stapes. As can be seen in 2.1 the sound waves first, coming from the Ear canal collide with the Tympanic membrane This causes the membrane to vibrate in accordance with the sound wave. This movement is then directly translated to a mechanical movement by the Ossicles, as this configuration of bones moves in tandem. The Ossicles then convert this movement back to waveform, when impacting the Oval window with the Stapes. The Eustachian tube provides pressure regulation for the Tympanic cavity, which is needed for the Tympanic membrane to vibrate properly. It also provides space for the Round window to flex, and for the inner ear to regulate its internal pressure [5].

2.1.3 The inner ear

As the Stapes impact the Oval window, the sound waves now propagate into the inner ear. Here, we find two main components, namely the Vestibula, and the Cochlea. The Vestibula utilizes the semicircular canals, to find the orientation of the head. As this is not directly relevant to sound, we will not discuss it further.

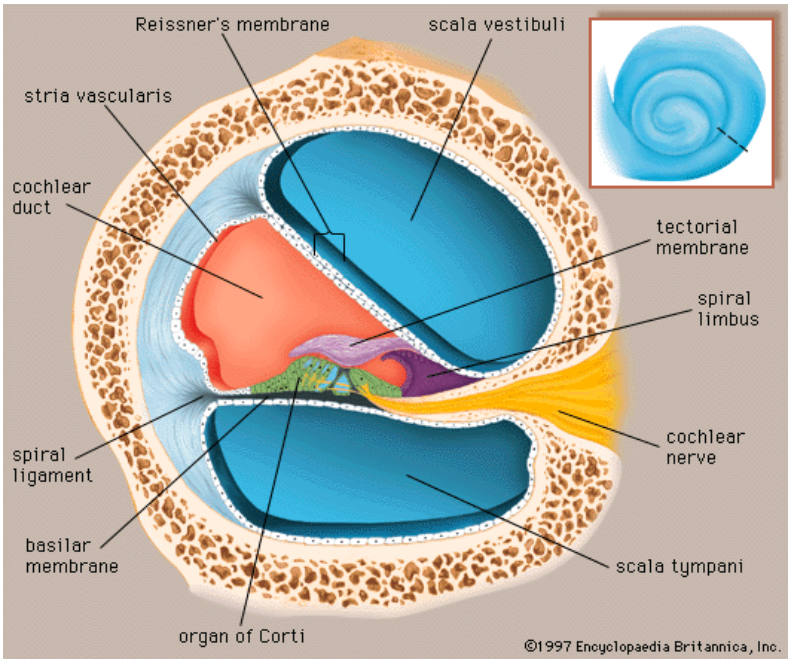


Figure 2.2: Cross section of the Cochlea [6].

The cochlea is an intricate organ. The wave itself propagates from the Oval window, through the Scala Vestibuli, which continues into the Scala tympani, and finally, the wave reaches the Round window and terminates in the middle ear. The Scala tympani is effectively just the extension of the Scala Vestibuli, as the fluid-filled tube wrap around the end of the cochlear duct, and returns. The interface between the Cochlear duct, and the Scala tympani is called the Basilar membrane. It is important to note, the interface with the Scala Vestibuli is not the Basilar membrane, but the Reissner's membrane. When the Basilar membrane moves, it shifts the Organ of Corti, which is the specific organ responsible for the perception of sound. On this organ, there is a row of incredibly fine hair cells, which is what produces the neural response of sound, when interacting with the Tectoral membrane. The sound perceived is dependent on where along the Basilar membrane the sound wave interacts, which is frequency dependent. The Organ of the Corti is able to manipulate these hairs, by shifting their position, to amplify sound.
It is this amplification procedure, which is responsible for Oto-acoustic emissions. The motion induces new sound waves, which can be perceived faintly, as it reverberates back out through the ear [5].

2.2 Otoacoustic Emissions

Oto acoustic emissions (OAE) are auditory events in the impulse response of the human ear. The OAEs are present in people with normal functioning hearing, they are absent in the case of cochlear deafness [7]. OAEs can be emitted from the ear with, and without external stimulation [8].

2.2.1 OAE measurement methods

All OAE measurement requires a dedicated system. Usually, such a system is composed of a specialised probe composed of a miniature microphone and speaker inserted into the patient's middle ear, then the specific stimuli is being played into the ear, evoking an OAE. The OAE is then being recorded and processed by the microphone in the probe. In the case of SOAEs, just a dedicated, miniature microphone is sufficient, since there are no stimuli to play into the ear [9].

2.2.2 SOAE

Spontaneous Oto Acoustic Emissions (SOAE) are emitted from the ear in the absence of an external stimulus. In about 60% of healthy ears, SOAE can be measured [10], and they can be audible for some people and be perceived as tinnitus [11].

2.2.3 TEOAE

Transient Evoked Oto Acoustic Emission (TEOAE) are a type of OAE evoked by a click. The measurement is processed by a Fast Fourier Transform (FFT) and then assessed [12]. TEOAE, like other OAEs, are the direct result of the cochlear amplifier function. The bandwidth of recorded TEOAEs depends heavily on the specific setup. Different studies have different TEOAE spectra and intensities. The TEOAE pattern depends heavily on the environment in the measurement was taken and the hearing state [13] [14] of the measured person, showing a promise that TEOAE is going to be useful in biometrics, as there might be different TEOAE patterns from different people. Overall, they are evoked by stimulating the ear with a tone burst.

The study [15] recorded TEOAE between 0.7 kHz and 8 kHz, and another study [13] measured them in 0. 5 kHz and 6 kHz spectrum. The paper [13] further process the recorded stimuli by dividing them into blocks A and B. To measure the repeatability, the absolute value of Pearson's correlation coefficient was calculated between the groups. High correlation implies a good, strong OAE. A low correlation implies a weak OAE.

2.2.4 DPOAE

Distortion Product Oto Acoustic Emission is OAEs evoked by two pure tones differing slightly in frequency between $\frac{f1}{f2} = 1.20 - 1.25$. To extract the OAE from two measurements, a subtraction operation is performed $f_3 = 2 * f_1 - f_2$. f_2 has to be greater than f_1 [12].

Paper [13] describes a measurement protocol, with $\frac{f_1}{f_2} = 1.22$ and levels $f_1 = 65dB, f_2 = 45dB$. Sixteen pairs of linearly separated spaces were created with distortion product frequency from 900.8 Hz to 3301.5 Hz. The system reproduced the set of stimuli for 1.3 s, simultaneously recording OAEs in the ear canal. The time-domain signal is processed via FFT, transforming the signal to the frequency-domain. All distortion product frequencies corresponded to an FFT bin, thus no averaging across FFT bins was required.

2.3 Machine learning

There are 3 common types of problems in machine learning, these are classification, density estimation, and regression. A regression problem is the most commonly known of the three problems, the purpose of regression analysis is to find the relationship between dependent variables and independent variables, ie. fitting a function to some sampled data [16] as shown in figure 2.3.

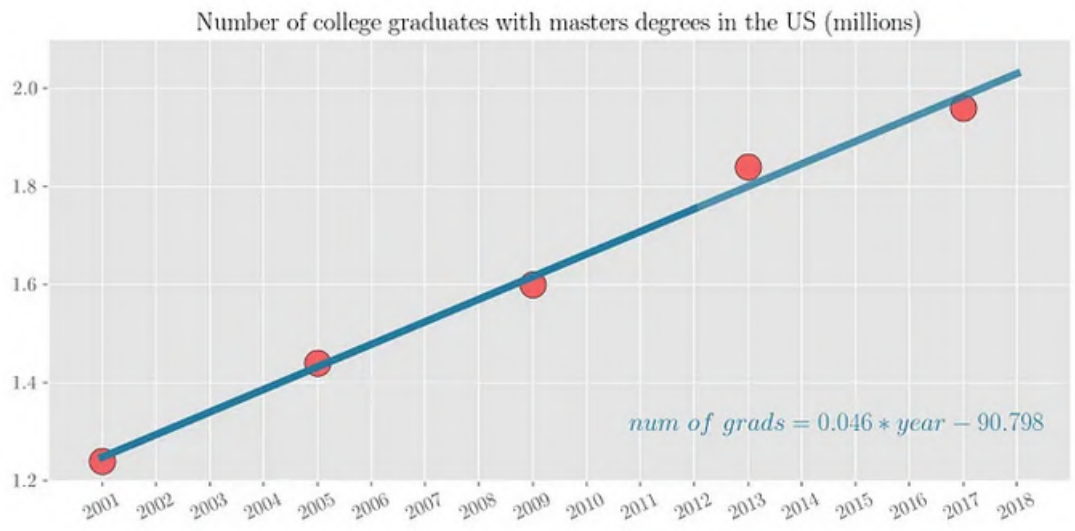


Figure 2.3: An example of fitting a line to a sample of the number of college graduates with masters degrees in the USA per year in millions [16]

The purpose of density estimation is to find the probability where the data is most dense, this can easily be achieved with a histogram plot [17] as shown in figure 2.4

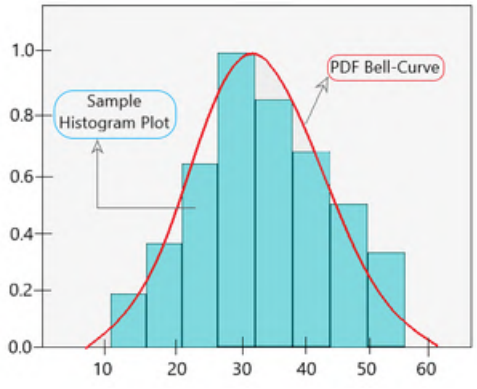


Figure 2.4: A probability density function fitted over histogram plot with one peak around 30, the figure is from [17]

A classification problem is to categorise a new sample of data that comes from the same label as a previous sample of data. The classification problem is what this project will try to solve for OAE biometrics. In the following section what the classification problem involves will be expanded.

2.3.1 Classification

As mentioned previously classification is to categorise a sample of data from the same label based on a previously labelled sample. An example of this can be face recognition, categorizing different animals, vehicles, people etc. which are formally called "objects". An example of labelled data can be seen in figure 2.5.

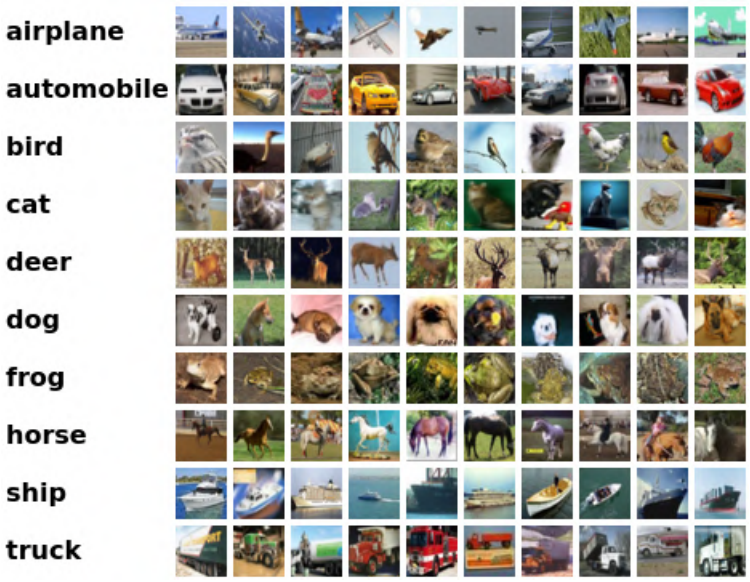


Figure 2.5: Different labeled data from the CIFAR-10 dataset [18]

With the labelled data from figure 2.5 it is possible to use machine learning or a deep learning method to predict an input and then classify it as the same class as the labelled data. When doing classification there are 3 types of methods, these are binary, multi-class, and multi-label classification [19]. Binary classification is used to classify between 2 options, ie. whether it is or is not this class. A multi-class classification is used to classify between multiple options, ie. whether or not the input is part of one class out of multiple classes. Multi-label classification is like multi-class classification, but instead of the input only being part of one class, it can have multiple classes. When evaluating the methods there are two types of errors that can be encountered these are type I and type II errors which are also called False Positives (FP) and False Negatives (FN). In binary classification, a false positive would be classifying the input as part of the class, ie. classifying something as a dog when it is not a dog, if it is correctly classified as not a dog it is called a True Negative (TN). A false negative is the opposite where the input is not classified as part of the class when it is, ie. classifying a dog as not a dog, if it is correctly classified as a dog it is called a True Positive (TP). With the 4 possible outcomes it is possible to place them in something like a "lookup table" called a confusion matrix, where an example of a binary classification can be seen in table 2.1. For multi-class and multi-label classification, a multidimensional confusion matrix can be created. This differs from a binary classification confusion matrix where instead of having predicted positives and negatives versus actual positives and negatives, there are predicted classes versus actual classes. Another difference is that a multidimensional confusion matrix can not be shown with TP, FP, FN, and TN values as that would

	Predicted positives	Predicted negatives
Actual positives	TP	FN
Actual negatives	FP	TN

Table 2.1: An example of a confusion matrix

Class 1		Predicted classes		
		1	2	3
Actual classes	1	TP	FN	FN
	2	FP	TN	TN
	3	FP	TN	TN

Table 2.2: An example of a multi-dimensional confusion matrix for class 1, if all outcomes were true

be different for each class, instead, the number of occurrences can be counted where it is then possible to turn that into a binary confusion matrix or only show it for one class which can be seen in table 2.2 From table 2.2 it can be seen that if all outcomes were true that there is always a lot more outcomes for errors, and a lot more negative type outcomes whereas there is only 1 possibility of it being TP. If the method did the classification correctly there would be 1 positive and 8 negatives, meaning there is a ratio of $\frac{1}{classes^2-1}$ positives to negatives making the dataset unbalanced against TP numbers. To counteract this it is possible to use the 4 possible outcomes to calculate the true positive rate also called recall, the positive predictive value also called precision, the accuracy and the balanced accuracy F1-score for each class using the following equations:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{P + N}$$

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Precision is a metric which shows how many correct predictions were done out of all the possible positives there are, ie. how many are correctly classified as a dog out of all the dogs there are. This makes precision to be more important in a multi-class classification as there are more possible negatives than positives. The recall shows how many correct classifications were done out of all the positive predictions, ie how many were correctly classified as a dog out of all the predictions that were classified as a dog. The accuracy shows how many correct predictions were done

out of all predictions. If the dataset is unbalanced, which it will be for a multi-class classification the accuracy will not show the whole picture and can therefore be a bad metric to use, since it does not take FP and FN into account and positives and negatives are weighed the same. Instead of accuracy, the F1-score can be utilised to highlight the FP and FN since it is the harmonic mean between precision and recall.

2.3.2 Convolutional Neural Networks

One of the most common neural networks is Convolutional Neural Networks (CNN) and they are distinguished from other types of neural networks by having superior performance with images, speech, and audio signals [20]. A convolutional neural network has 3 types of main layers, these being:

- Convolutional layer
- Pooling layer
- Fully connected layer

The convolutional layer is the first layer in a CNN and can be followed by other convolutional layers or a pooling layer, but the fully connected layer is the final layer [20]. After each convolution layer an activation function introduces non-linearity, some examples of these are:

- Rectified Linear Unit (ReLU): $ReLU = (x)^+ = max(0, x)$
- Hyperbolic Tangent (Tanh): $tanh(x) = \frac{exp(x)-exp(-x)}{exp(x)+exp(-x)}$
- Sigmoid: $\sigma(x) = \frac{1}{1+exp(x)}$
- Softmax: $Softmax(x) = \frac{exp(x_i)}{\sum_j exp(x_j)}$

ReLU is the most common activation function used for CNNs since it has fewer vanishing gradient problems. This is because it consists of 2 linear pieces, where the negative values are 0 and the positive values are the input itself. This means that if the input is a negative value the output will be zero, and if it is a positive value it will output itself. The only problem with ReLU is that at 0 it does not perform any learning, this can be mitigated by several methods [21].

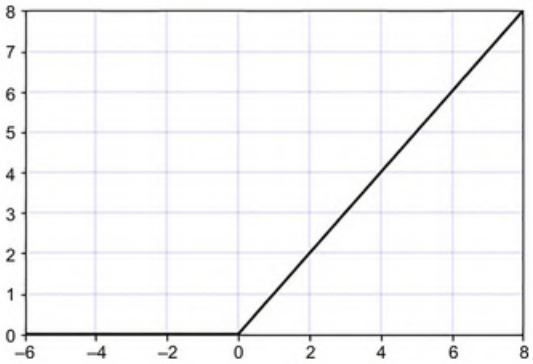


Figure 2.6: The ReLU activation function [21]

Before the release of ReLU, Tanh and Sigmoid were the most common activation functions. Sigmoid can be used as an output unit for a binary classifier, but the drawback of using sigmoid is that it is very flat with high positive and negative values, which is also true for Tanh. This drawback can be resolved by normalizing the input to somewhere between -1 and 1, although this doesn't fix the problem where both sigmoid and tanh are very sensitive around 0 [21]. The Sigmoid and tanh functions can be seen in figure 2.7

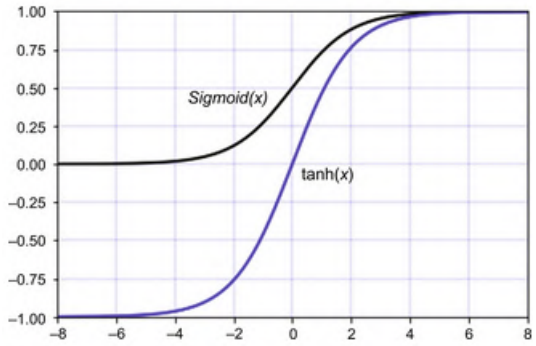


Figure 2.7: The Sigmoid and tanh activation functions [21]

Softmax is only able to be used to get the output from the model and is a generalization of sigmoid, where instead of giving the probability like a binary classifier it gives the probability of all classes which sum to 1.

2.4 Biometric authentication

This section will look into what biometrics are as well as some biometric authentication already in use, and then explore the possibility of using OAEs for this purpose as another source for authentication.

2.4.1 Biometric modalities

A biometric is defined as a "measurable physical characteristic or personal behavioral trait used to recognize the identity of an applicant" [22] and is useful for both identification and verification of an individual's identity for security, convenience or legal purposes. When input into a biometric system, the type of data input into the system is called the biometric modality, such as a fingerprint.

Biometric categories

From the definition, biometric modalities can be divided into two different categories, physiological and behavioral biometrics. With the most known being fingerprints as a physiological biometric modality and signatures as a behavioral biometric modality. These as well as some other examples can be seen in figure 2.8.

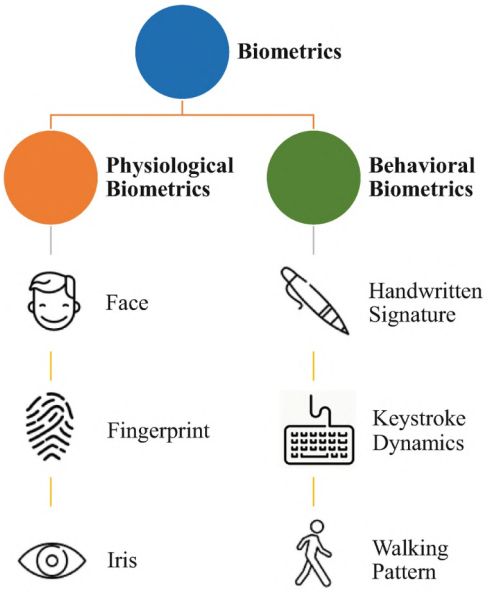


Figure 2.8: Examples of the two different biometric categories and examples of modalities linked to them [23].

As these examples show, the physiological biometric modalities are characteristics that vary based on the shape of the individual body parts, and are usually fairly

non-invasive to collect data. The behavioral biometric modalities however, are the patterns of how an individual does an action, either by the outcome of the action such as by identifying the features of a signature image and comparing it to previous ones, or by the process of the action itself, such as looking at the features of a signing process. These features can include the speed, pressure, directions and pauses [23]. While the shape of the ear is a physiological characteristic, OAE is a behavioral one, like a heartbeat.

Both of these categories have different strengths and weaknesses, as behavioral characteristics are usually more difficult to spoof, but require more analysis and have lower accuracy and performance than most physiological characteristics for biometrics n [23]. However, within the categories, strengths and weaknesses of different biometric modalities can also have a much bigger effect on the performance of the identification, based on the use case.

Suitability of biometrics

When deciding the suitability of a biometric modality for real-world usage some properties should be considered [23]:

- Universality of the modality - How common the given biometric is for the possible users.
- Uniqueness of the modality - How easy it is to differentiate between the users.
- Collectability of the modality - How easy it is to collect data from the biometric.
- Permanency of the modality - How much the biometric changes during the lifetime of the user.
- Acceptability of the modality - How willing possible users would be to use the biometric.

While the perfect biometric would score high on every one of these properties, no such biometric modality is in use. A comparison between these biometric properties of the modalities from figure 2.8, can be seen in table 2.3.

Biometrics	Universality	Uniqueness	Collectability	Permanency	Acceptability
Fingerprint	Medium	High	Medium	High	Medium
Face	High	Medium	High	Medium	High
Iris	High	High	Medium	High	Low
Signature	Medium	Low	High	Low	High
Keystroke	High	Medium	Medium	Low	Medium
Gait	High	Medium	Low	Low	Medium

Table 2.3: Comparison of biometric modalities based on the biometric Properties [23].

Biometric systems

Once a biometric modality is chosen, it is used as the input into a biometric system. These can use various different architectures, but follows the same general structure of a binary trained classifier as seen in figure 2.9, where the upper part of the flowchart is the enrollment or registration, where the data is stored and label to be compared later with the lower part, which is the actual authentication. These are then compared and are either identified as matching a registered individual or not [23].

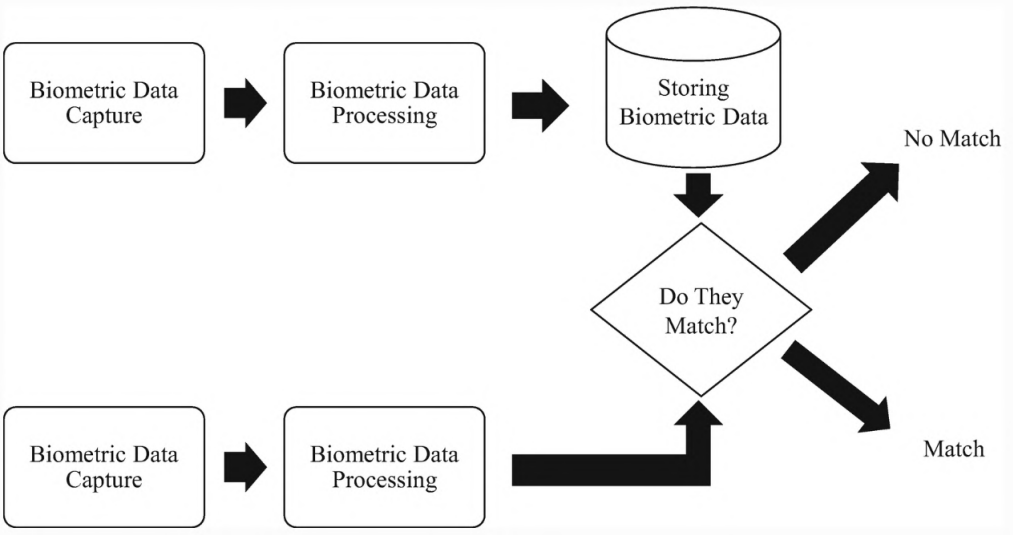


Figure 2.9: General structure of biometric systems [23].

Performance metrics

Much like with typical classification, a biometric system has various different metrics used to decide how well they perform, which can be tuned by changing the features looked at, the data used or the sensitivity of the system.

For the registration process, the metrics measured are the *Failure-to-Enroll-Rate* (FTE) and *Failure-to-Acquire-Rate* (FTA), among others such as average time to enrol and size of the biometric templates. When the universality of the modality used in a system is low, then the *Failure-to-Enroll-Rate* (FTE) will be high. This metric measures the proportion of people from whom the system fails to extract any biometric information, such as with a gait system attempting to extract any biometric information from a paralysed person. Whereas the *Failure-to-Acquire-Rate* (FTA) is the proportion of people where the data captured was unusable. Such as if trying to scan a fingerprint with a wet finger. And this metric is closely linked to the Collectability of the modality [23].

For the accuracy of the authentication, 3 different metrics are used which must be balanced by changing the sensitivity of the system [23]:

- *False Acceptance Rate* (FAR) is the rate at which the system allows unauthorized individuals access. Where a high FAR means that the sensitivity of the system is too low, and the security of the system is compromised. The FAR can be calculated by

$$FAR = \frac{FalseAcceptances}{UnauthorizedAttempts} * 100$$

- *False Rejection Rate* (FRR) is the rate at which the system denies authorized individuals access. Where a high FRR means the sensitivity of the system is too high and the usability of the system is compromised. The FRR can be calculated by

$$FRR = \frac{FalseRejections}{AutorisedAttempts} * 100$$

- *Genuine Acceptance Rate* (GAR) is the rate at which the system correctly authorized individual access and is negatively proportional to the FRR. Meaning

$$GAR = 100 - FRR$$

Tuning the sensitivity threshold

When deciding a threshold the FAR, FRR and GAR need to be balanced against each other since a low FAR means a high FRR and low GAR. One way to do this is by finding the *Average Classification Error* (ACE) and the *Equal Error Rate*(EER). These can be found by first finding the FAR, FRR and GAR of a test set for each different threshold. Then for each of the thresholds the ACE is calculated using the equation [24]:

$$ACE = \frac{FAR + FRR}{2}$$

The threshold where $ACE_T = FAR_T = FRR_T$ is the EER, as seen in figure 2.10 and if error at that threshold is low enough for each of the parameters and the threshold is high enough, it is a usable threshold to set as it should minimize the error. Usually, a threshold higher than 65% confidence is preferred if possible. If the EER is lower than 65% confidence, the threshold where $FRR_T = GAR_T$ is also a possibility. Lastly, the threshold where the F1 score of the system is lowest is also a possibility.

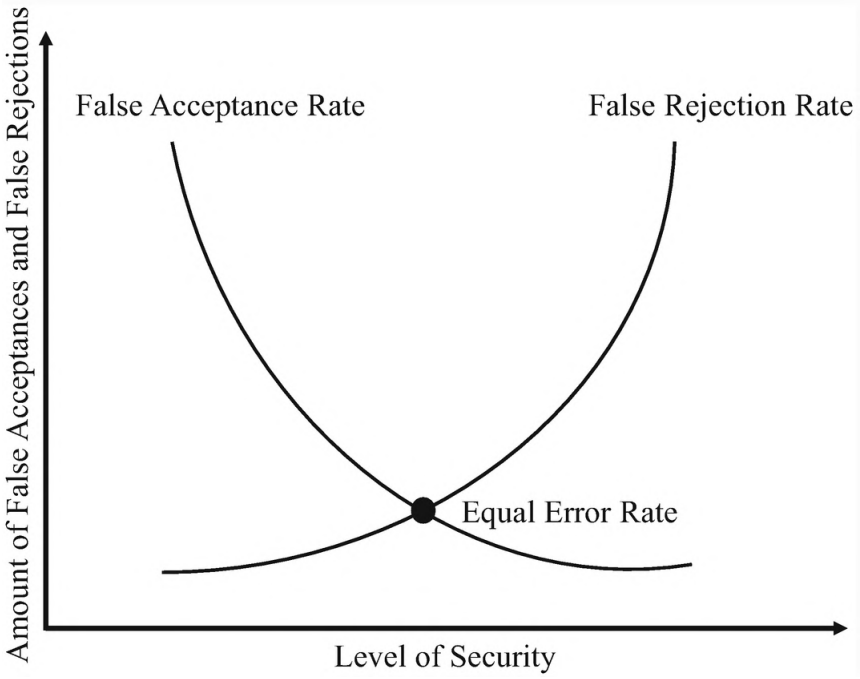


Figure 2.10: Graph of the Equal Error Rate compared to the False Acceptance Rate and False Rejection Rate [23].

So this gives the following four usable thresholds: The lowest ACE value, the EER, the lowest F1-score and the intersection between FRR and GER [23]. However, while these are some usable thresholds, the specific ones that best suit the system are dependent on the level of security required compared to convenience.

2.4.2 Example of biometrics: Fingerprints

Fingerprints are the most popular biometric indicator used for identification and have been in use since 1897 CE for crime investigations [25]. The dermal ridge of a finger is what creates a fingerprint, and this dermal ridge is formed by genetic and environmental factors [26]. These environmental factors are specific events that happen in the womb such as the position of the fetus, where the fingerprint will be fully formed at about 7 months of fetus development [26]. The way that fingerprints are formed makes them so unique that even identical twins have different fingerprints [27], and they do not change over a person's lifetime with the only exception being damaging the finger.

Features and classification

The challenging problem with the uniqueness of fingerprints is creating a model that can obtain the variability of features. This is because fingerprints are represented by a large number of features [28]. These features are categorized into 3 different levels according to the resolution needed from 1 to 3 where for level 3 a 1000 DPI is needed. Level 1 features are macro details and therefore primarily used for classification rather than recognition. Level 2 features refer primarily to minutiae and are the most distinct and stable features and can be accurately extracted at a resolution of 500 DPI, making them used in nearly all Automated Fingerprint Identification Systems (AFIS). Level 3 features are generally defined as the dimensional attributes of ridges. Some of the features in these 3 categories are [27]:

- Level 1 features
  - Fingerprint classes ie. right or left loop, whorl, or arch
  - Singular points ie. core(s) or delta(s)
  - External fingerprint shape, orientation, and frequency maps of fingerprint ridges
- Level 2
  - ridge endings and ridge bifurcations
- Level 3
  - Finger sweat pore
  - width, curvature, edge contours etc.

which is used to assess the individuality of the fingerprints [28]. An example of fingerprints with different classes can be seen in figure 2.11 and the 3 categories can be seen in figure 2.12

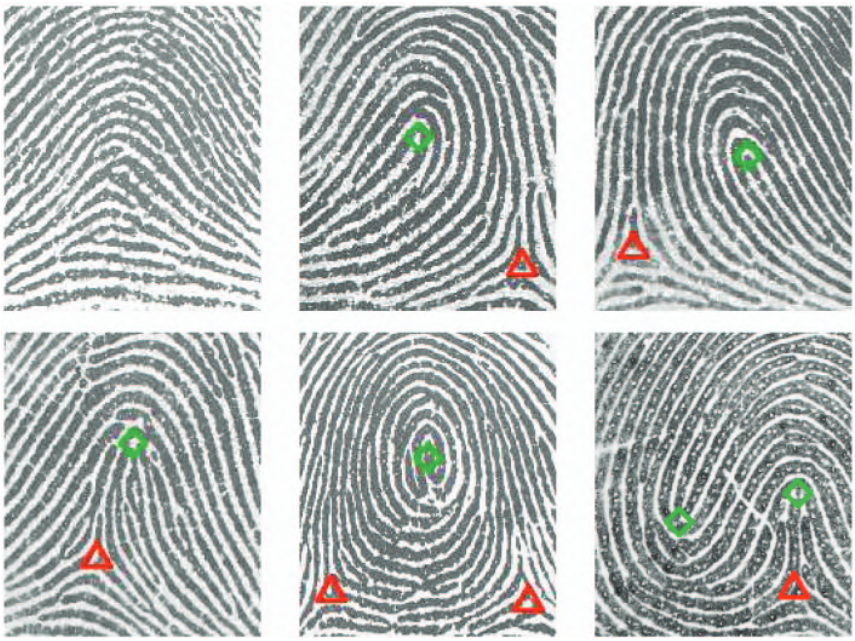


Figure 2.11: Image of fingerprint classes where in the top row from left to right is an arch, left loop, and right loop. In the bottom row from left to right are a tented arch, whorl, and twin-loop. Triangles mark deltas and diamonds mark cores. The figure is from [29].

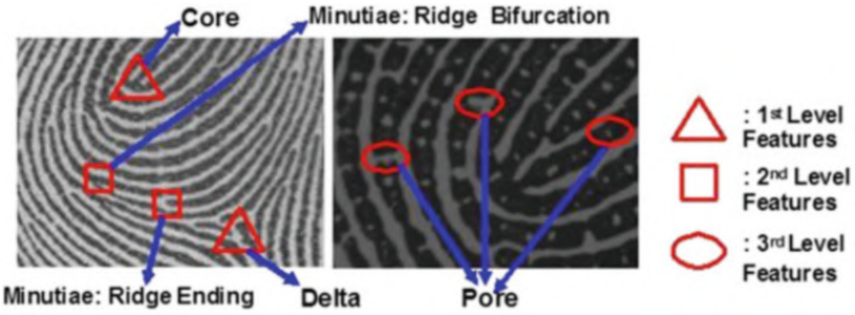


Figure 2.12: The 3 levels of fingerprint features, the figure is from [29].

2.4.3 OAE as Biometrics

While Otoacoustic Emissions(OAE) are useful for determining how well an individual's inner ear is working, a few studies have also tested OAEs capabilities for biometric identification [30], [31]. However, compared to the other biometrics mentioned in this section, OAE-biometrics are mostly an unexplored topic. To measure the OAE for biometrics, both DPOAE and TEOAE can be used. However, TEOAE does seem to offer better performance for biometrics, likely due to DPOAEs reduced degrees of freedom, which was limited in the study by the ILO292 system [30].

Universality of OAEs

OAEs are not present in every individual, as it is closely linked to having a functional cochlear amplifier [7]. In other words, this means OAEs are only expected to be a useful biometric modality if the individuals have normal hearing, and as such only have a medium universality

Uniqueness of OAEs

From figure 2.13 it can be seen that the DPOAE of different individuals does appear unique, as the peaks and valleys of the graphs are at completely different positions, as well as strengths. This can also be seen for TEOAE in figure 2.14, where the differences are also pronounced. This is supported by the fact that previous studies have found them distinctive [30], it has also been shown that the EER for both ears is 2.64%±10% FAR and 2.65%±30% FRR. When the threshold had been lowered to minimise the FAR it is possible to get a FAR at 0.47%±10% and a FRR at 4.83%±30%. If the threshold was raised to minimise the FRR the results showed that the FRR was 1.61%±30% and the FAR was 12.31%±10% [32]. So they at least have a medium to high uniqueness, especially TEOAE [30].

Collectability of OAEs

Stimuli-evoked OAEs are being collected by a specialized and dedicated system, consisting of a probe with a miniature speaker and microphone built-in, inserted into the ear canal. The stimuli are being played into the ear and subsequent OAEs are being recorded. For DPOAEs compared to TEOAEs, there is a need for more complex hardware, with 2 speakers, as opposed to 1, making it harder to obtain data for DPOAEs compared to TEOAEs [31]. There is also a large difference in the complexity of the signal, one being continuous sweeps and the other being simple clicks. making it TEOAEs more easily collectable Spontaneous OAEs are being recorded with a probe with just a microphone built-in. There is no need for a speaker since there are no stimuli to be played. [9] The measurements are however very dependent on the level of background noise [30]. This means the collection of data is prone to errors and requires specialised equipment, and as such has a medium to low collectability.

Permanency of OAEs

While the distinctiveness of features is the most important part of the viability of a biometric identifier, it is also important that those distinctive features are robust and not changing rapidly. For TEOAE, this was tested in 'N. J. Grabham Et al.' [30], by making TEOAE recordings of an individual 6 months apart. The visual results of the test can be seen in figure 2.15.

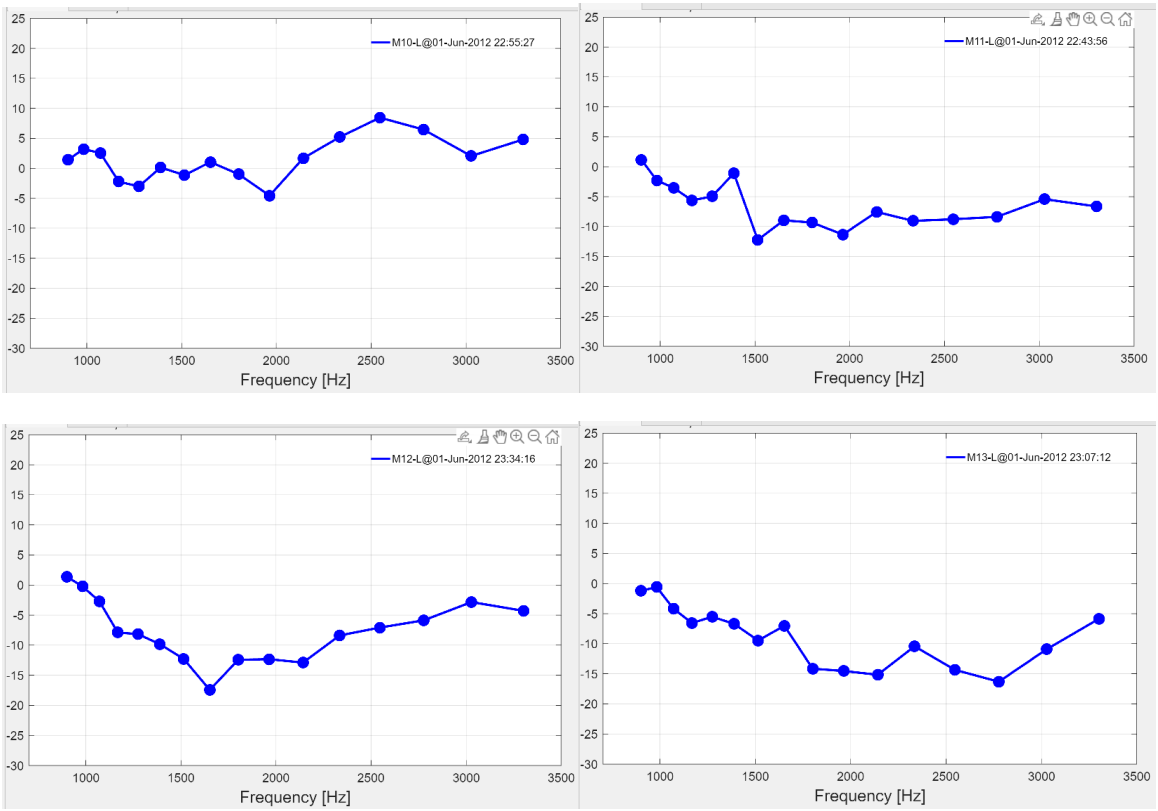


Figure 2.13: DPOAEs captured from 4 different individuals [13].

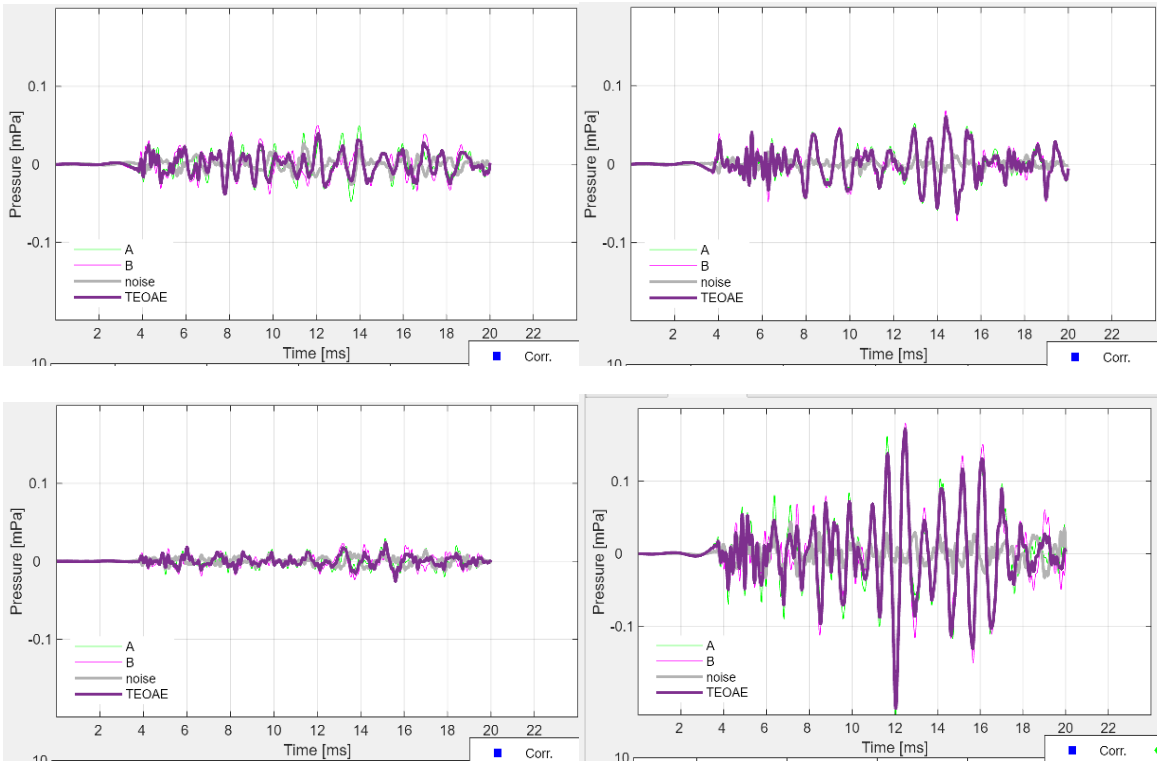


Figure 2.14: TEOAEs captured from 4 different individuals [13].

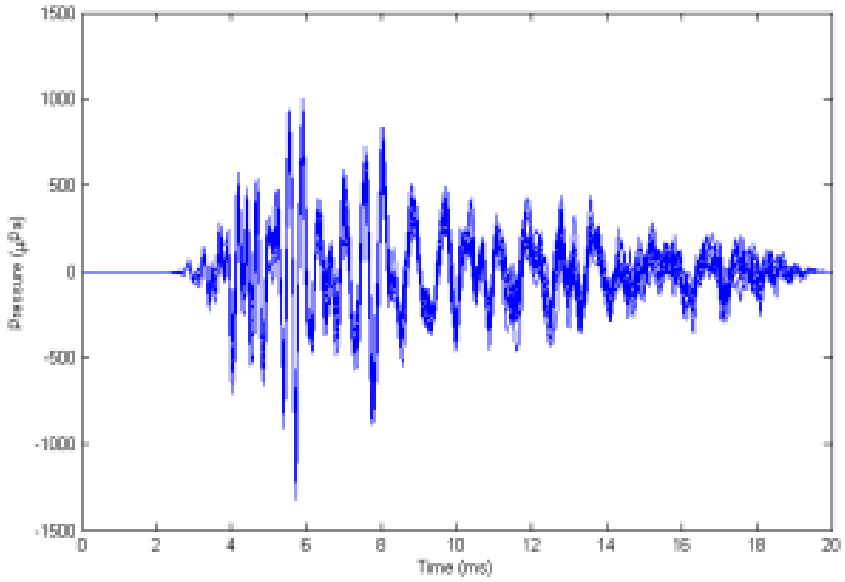


Figure 2.15: Ten TEOAE recordings from the same individual comprising two sets of five recordings taken 6 months apart. [30]

As seen in the figure, the shape is almost identical. Suggesting that within a short term, TEOAEs are highly repeatable [30]. However the OAEs do degrade with an individual's hearing ability, and as such has a medium permanency.

Acceptability of OAEs

OAEs require a probe to be inserted into the ear as far as comfortable, and to not swallow or make a sound while measuring. This can be an intrusive procedure and troublesome for some individuals, and as such the Acceptability would be low. This could change if the data acquisition could be done with less intrusive instruments.

Considerations as a biometric identifier

Inserting the properties of OAEs as a biometric modality into the property table from section 2.4.1, the table would look like table 2.4:

Biometrics	**Universality**	**Uniqueness**	**Collectability**	**Permanency**	**Acceptability**
Fingerprint	Medium	High	Medium	High	Medium
Face	High	Medium	High	Medium	High
Iris	High	High	Medium	High	Low
Signature	Medium	Low	High	Low	High
Keystroke	High	Medium	Medium	Low	Medium
Gait	High	Medium	Low	Low	Medium
OAE	*Medium*	*Medium/High*	*Low/Medium*	*Medium*	*Low*

Table 2.4: OAEs compared to the other biometric modalities from table 2.3 [23].

These points suggest that OAEs and specifically TEOAEs should be usable as a biometric identifier. However, the biggest concern for using it is its collectability and acceptability, which would mean that it might only see use in situations where sound is already a parameter. In other words, collectability and acceptability would increase if using an earphone is already a factor.

Chapter 3

Problem statement and requirements

In this chapter, a final problem statement and corresponding requirement specification are made using the problem analysis.

3.1 Final problem statement

Using the problem analysis, the problem can be narrowed down to a project with the following problem statement which will be followed in the later part of this report:

Given a personal device meets the technical constraints measuring TEOAEs and the user has healthy ears, is it possible to create a TEOAE-based deep learning model to identify and authenticate a user for the device within a practical time frame and memory amount?

3.2 Requirements

Based on the final problem statement, some requirements can be made as seen in table 3.1, as there is no previous biometric identification for OAEs. These are designed to test whether TEOAE is usable for biometric authentication. The single-class precision, recall, accuracy and F1-score are required to make sure that none of the classes is too big of an outlier. Precision is a more important metric than recall, this is because there are a lot more possible negative values that can be classified as wrong. Specifically, there is 1 possible positive value for every $class^2 - 1$. This also

means that accuracy is not going to show much and therefore the F1-score is used as a balanced accuracy metric.

The false acceptance rate and the false rejection rate are specified to define the required maximum failure rate of the biometric system. The number has been chosen based on subsection 2.4.3 and since it is fine for a user to be rejected a few times instead of having an unauthorised user get access, the false acceptance rate is a more important metric to have fewer errors on.

The last two parameters are to make sure the time and memory use is within a practical limit.

Functional Requirements	
Single-Class Precision:	>0.75
Single-Class Recall:	>0.5
Single-Class Accuracy:	>0.65
Single-Class F1-score:	>0.65
False Acceptance Rate:	<0.01
False Rejection Rate:	<0.05
Time spent for 95% of the identifications:	<2 sek
Memory used for 95% of the identifications:	<1 GiB

Table 3.1: Requirement list for the solution

Chapter 4

Technical analysis

4.1 Audio features extraction

Audio features are descriptions or parameters of an audio signal that are being used in Deep Learning. There are a couple of audio feature types:

- Time domain features - audio processing methods considering the waveform or time signal of the audio. They reflect the amplitude and frequency changes of the signal over time. Typical time domain feature consist of Short-term energy, Zero crossing rate and Entropy of energy [33]
- Frequency Domain features - analysis of the frequency spectrum to obtain features like formant or bandwidth. Such features include Short-time Fourier transform, Spectral Centroid and Spectral Bandwidth [33].

4.1.1 Time domain features

Short-term energy

Short-term energy is an audio feature defined as [34]:

$$E_m = \sum_{n=0}^{W_l} |x_i s(n)|^2 \tag{4.1}$$

where $x_i(n)$ is the sequence of audio samples to the ith sample, W_l the count of samples. The variations in the energy will indicate variations in the signal intensity.

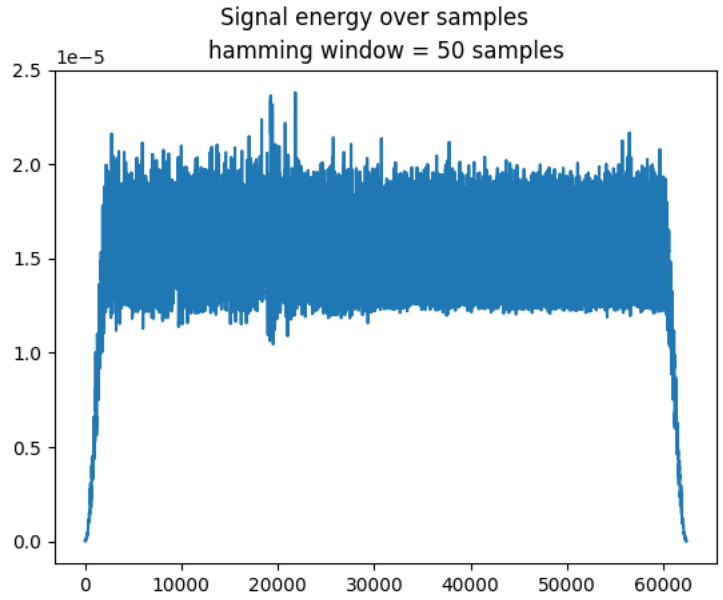


Figure 4.1: Short-term energy plot created from one of the data points in OAEDataset.

Analysing 4.1 plot we can see that the signal is relatively constant in amplitude/intensity, which is also seen in the time signal.

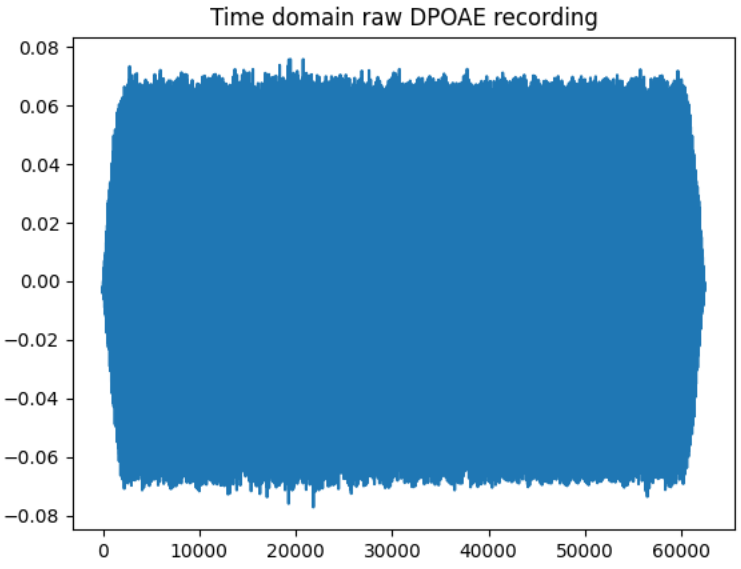


Figure 4.2: Source signal for the Short-term energy plot 4.1

Zero crossing rate

Zero crossing rate is a measure which denotes how many times the rate of value sign changes in a signal. Its mathematical formula is defined as [34]:

$$Z(i) = \frac{1}{2 * W_l} \left| \sum_{n=0}^{W_l} sign[x(n)] - sign[x(n-1)] \right| \tag{4.2}$$

where $sign(.)$ function is defined as:

$$sign[x(n)] = \begin{cases} 1 & x_i(n) \geq 0 \\ 0 & x_i(n) < 0 \end{cases} \tag{4.3}$$

and $x_i(n)$ is the sequence of audio samples to the ith sample, W_l the count of samples. The more noise in the signal, the bigger the Zero-crossing rate is. A relatively constant rate of zero crossings also will show that the signal is relatively constant over time [34].

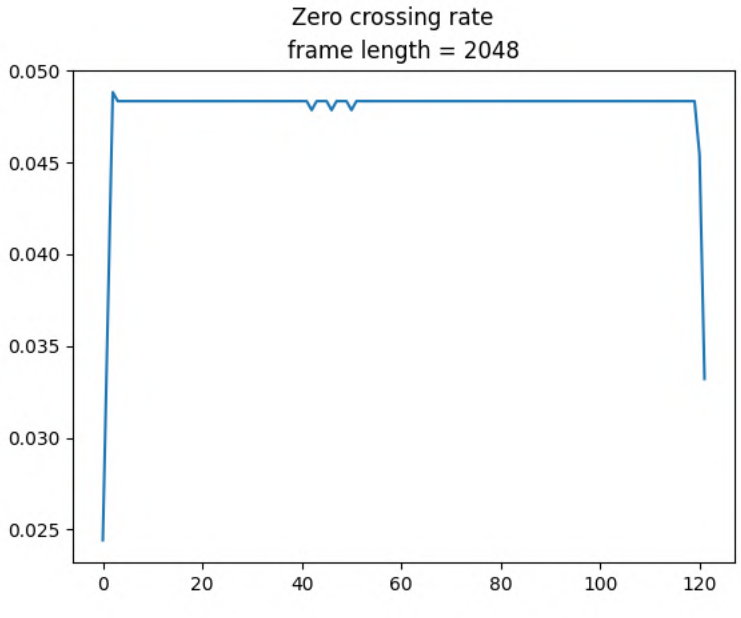


Figure 4.3: Zero crossing rate for source signal 4.2

A brief look at the plot 4.3 shows that the signal 4.2 is relatively constant over time and might not contain too much noise.

Entropy of Energy

The short-term entropy of energy can be interpreted as a measure of sudden changes in the energy level of a signal. It is defined as [34]:

$$e_j = \frac{E_{subFrame_j}}{E_{shortFrame_i}} \tag{4.4}$$

$$E_{shortFrame_i} = \sum_{k=1}^{K} E_{subFrame_j} \tag{4.5}$$

$$H(i) = -\sum_{j=1}^{K} e_j * \log 2(e_j) \tag{4.6}$$

where e_j is the sequence of entropy probabilities, $E_{shortFrame_i}$ is total energy of the short frame, $H(i)$ is the calculated entropy.

4.1.2 Frequency-time

Continuous Wavelet Transform

Continuous wavelet transform is an audio signal transform used to extract out frequency over time content. CWT is defined as:

$$C(a, b; f(t), \psi(t)) = \int_{-\infty}^{\infty} f(t) \frac{1}{a} \psi(\frac{t-b}{a}) dt \tag{4.7}$$

where, $f(t)$ is the analyzed signal in time, ψ is the used wavelet, a is the transform's scaling component, b is the translation component [35]. The scale parameter controls the 'stretching' of the wavelet, used in the analysis.

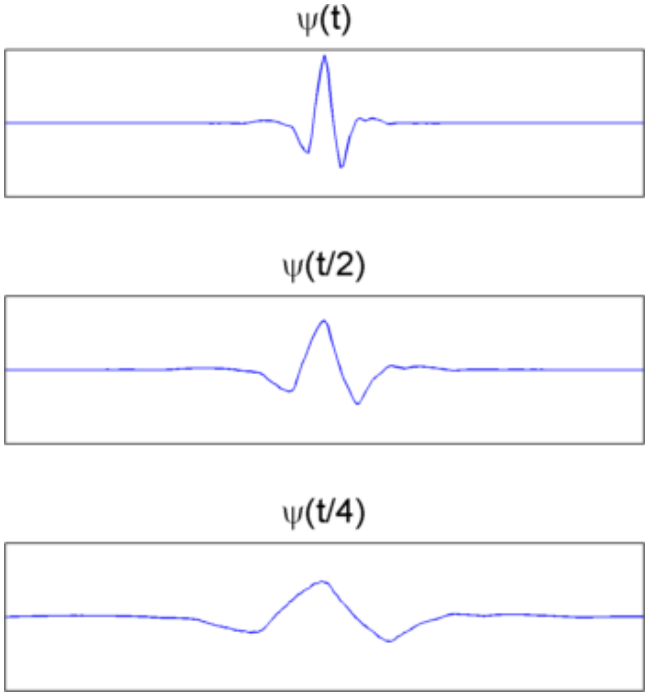


Figure 4.4: The influence of parameter a on an example wavelet [35]

In all, a small parameter a value yields a compressed wavelet, enabling finer features bu the wavelet coefficients. The opposite applies as well [35]. MATLAB implementation of continuous wavelet transforms offers three wavelets: the Morse, the Morlet and the bump wavelets.

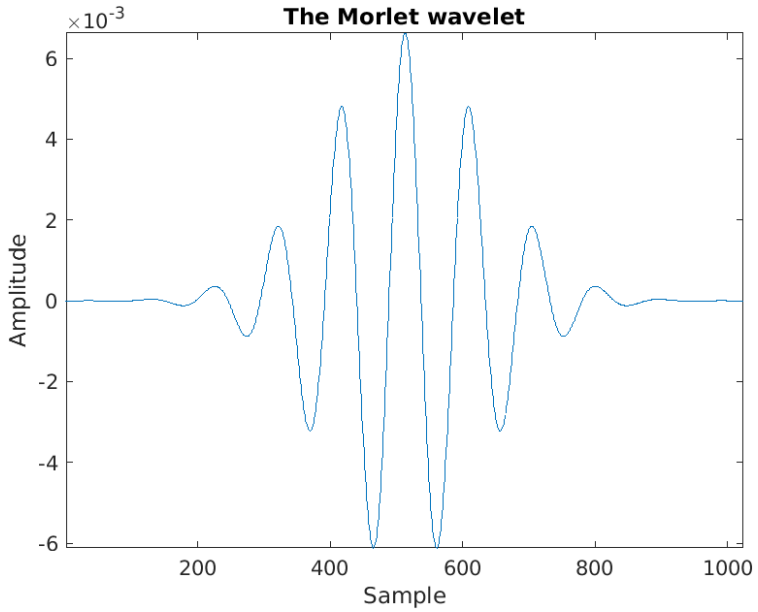


Figure 4.5: The Morlet wavelet generated in MATLAB.

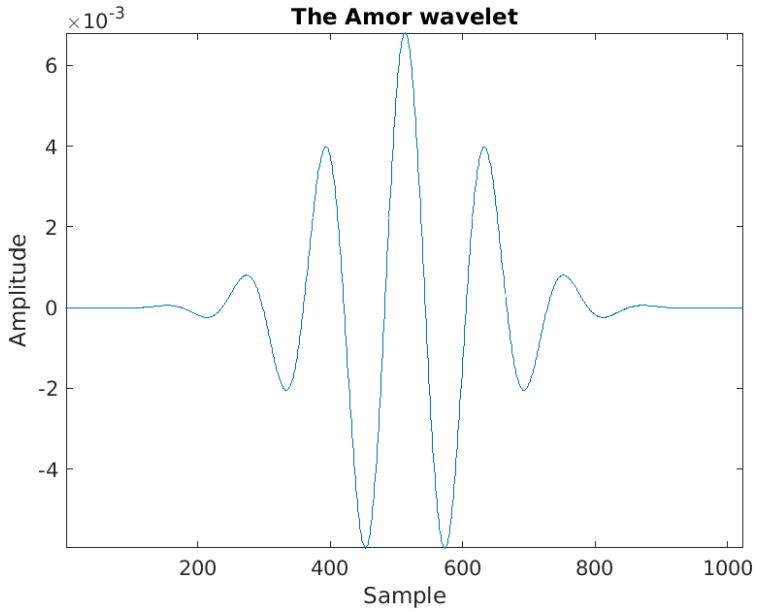


Figure 4.6: The Analytic Morlet (amor) wavelet generated in MATLAB.

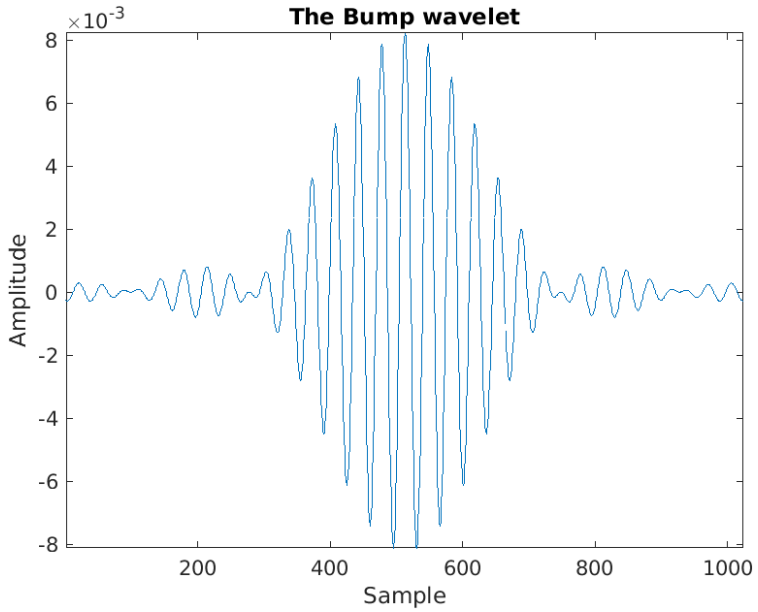


Figure 4.7: The Bump wavelet generated in MATLAB.

Different wavelets will yield different filters used to transform input signal [36].

The result of the transform is a time-frequency representation of a target signal:

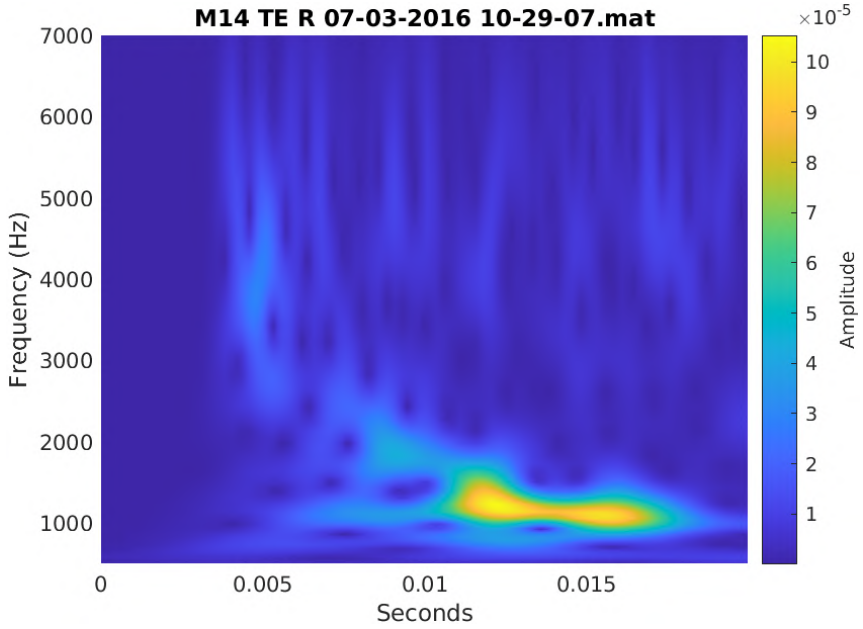


Figure 4.8: CWT generated from a subject's TEOAE, using the Morse wavelet

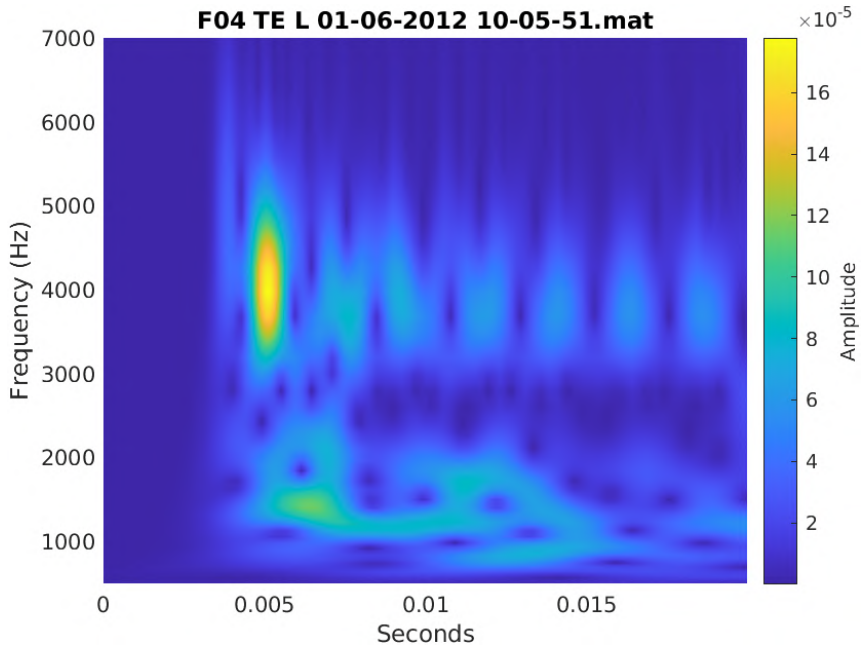


Figure 4.9: CWT generated from a subject's TEOAE, using the Morse wavelet

Comparing figures 4.9 and 4.8 we can clearly notice, that transform results are different from each other, therefore show promise for biometric applications.

Synchrosqueezing

Synchrosqueezing is a method for decreasing the smear within the CWT plot, by reassigning the signal energy in the frequency domain. The algorithm preserves the time resolution of the transform [37].

The algorithm is composed of the following steps:

1. Calculate the CWT of the target signal.
2. Extract the instantaneous frequencies of the CWT (denoted as Wf in the equation) using a phase transform, defined as:

$$\omega_f = \frac{\partial t W f(s,u)}{2\pi i W f(s,u)} \tag{4.8}$$

where s is:

$$s = \frac{fp}{f} \tag{4.9}$$

where fp is the peak frequency and f is the frequency. u is the translation of the wavelet.

3. "Squeeze" the CWT where the phase transformation is constant.

The result is a plot, with drastically reduced smearing and sharper details.

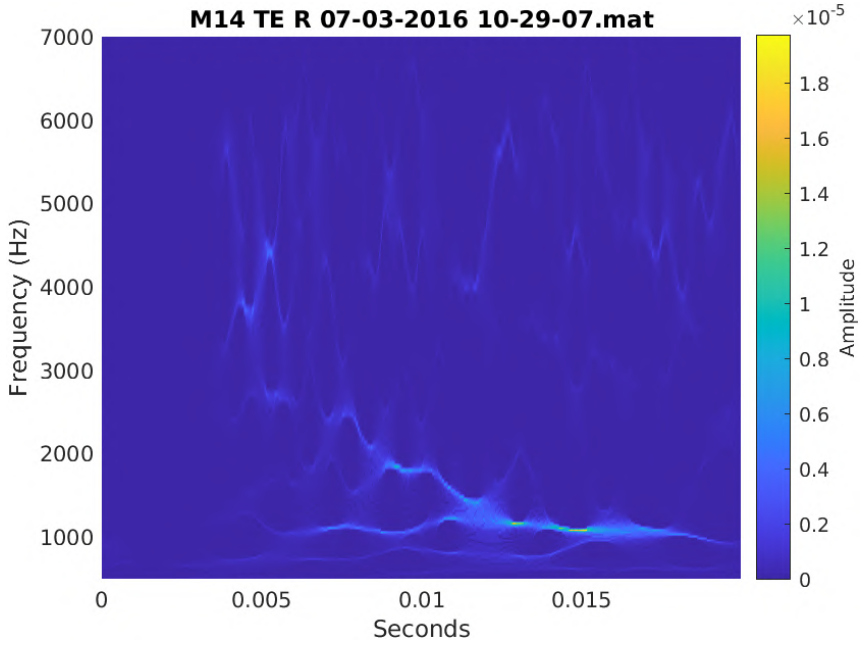


Figure 4.10: WSST generated from a subject's TEOAE, using the Morse wavelet, at 48000 Hz sampling frequency.

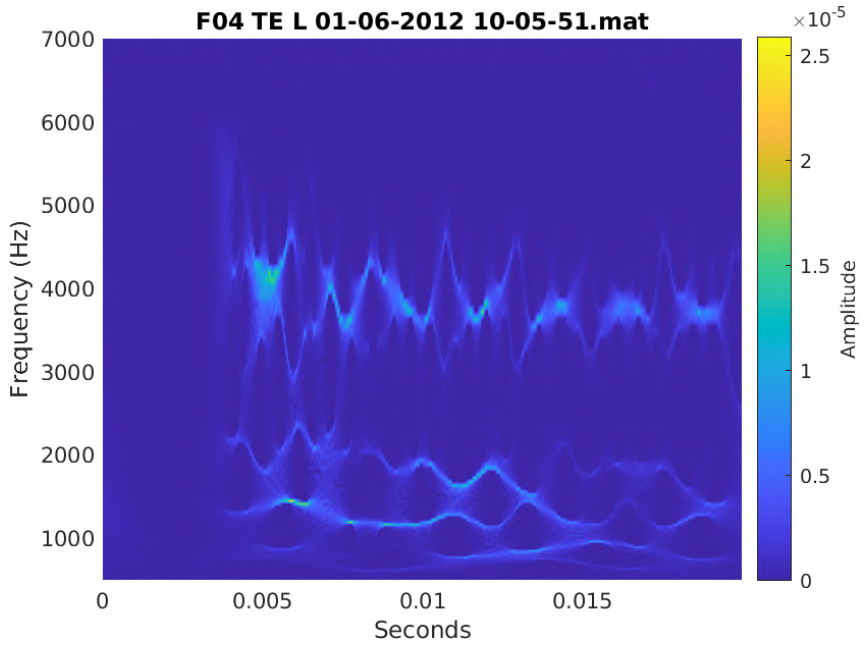


Figure 4.11: WSST generated from a subject's TEOAE, using the Morse wavelet, at 48000 Hz sampling frequency.

4.1.3 STFT

Short-time Fourier Transform (STFT) is a method to represent the frequency content of a signal over time. It is computed by sliding a window over the signal and calculating DFT within the window [38]. Matlab's implementation takes a couple of input arguments:

- x - input signal to be processed
- window - window type. Different window types will yield slightly different results depending on a specific case.
- noverlap - number of overlapping samples, when sliding the window across the signal.
- nfft - number of FFT bins.
- fs - sampling rate for STFT calculation.

After inputting a TEOAE recording, the resulting spectrogram will look similar to the figures 4.12 and 4.13

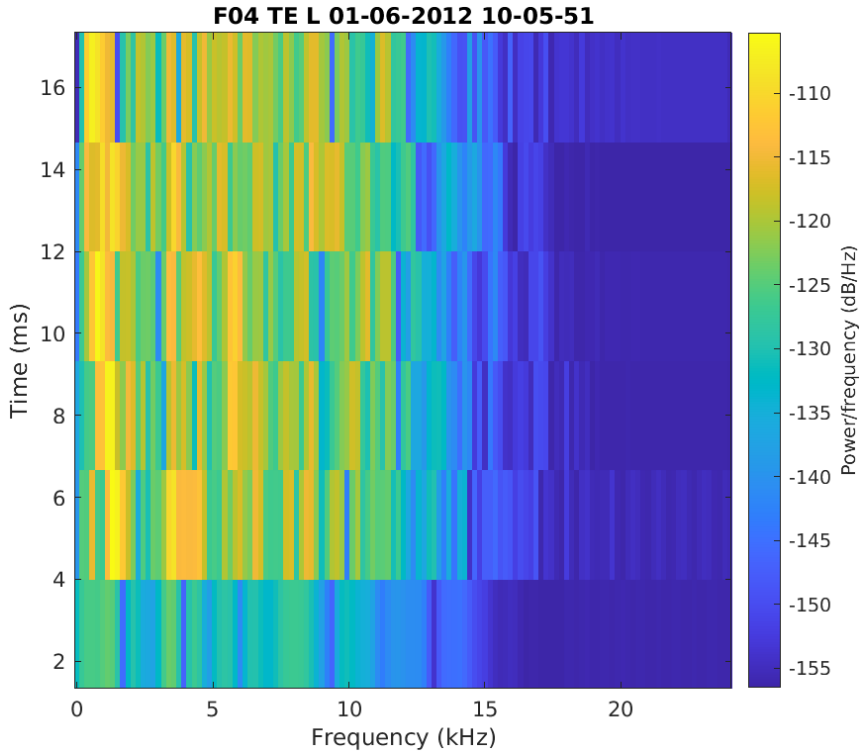


Figure 4.12: STFT spectrogram generated from one of TEOAE signals with a 256 sample wide hamming window, 128 samples of overlap, 256 FFT bins and 48000Khz sampling rate.

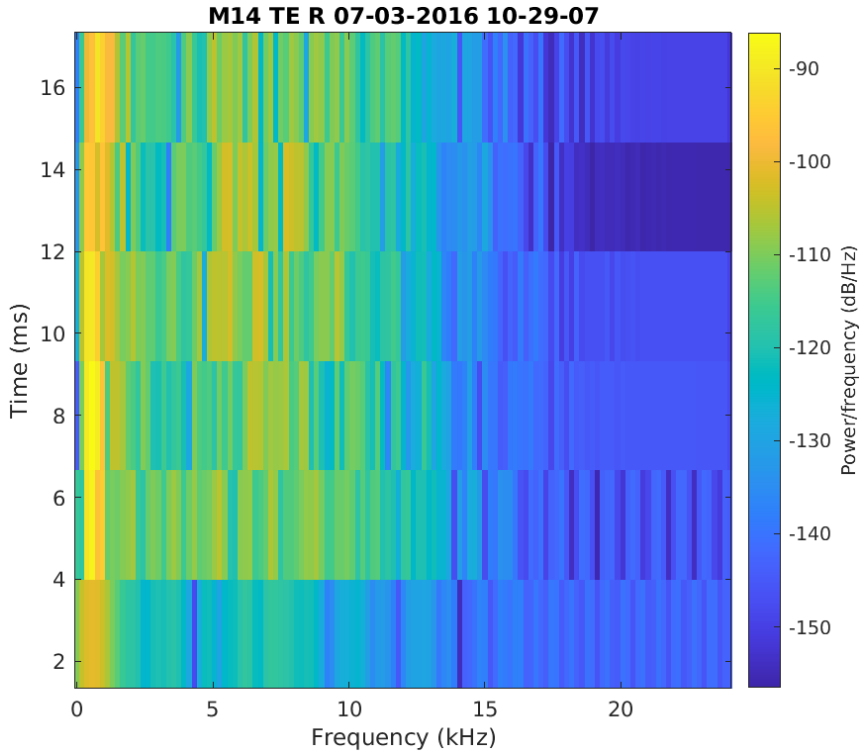


Figure 4.13: STFT spectrogram generated from one of TEOAE signals with a 256 sample wide hamming window, 128 samples of overlap, 256 FFT bins and 48000Khz sampling rate.

Chapter 5

Implementation

5.1 Hardware

This section will describe the hardware used.

Roland Quad-Capture Analog 2x2 Digital 2x2

This device is a sound card, developed by Roland, and is utilized for various audio applications. The main utilization of a device like this is to record and playback audio at a much higher quality, than what most systems are able to naturally. This particular model comes with 4 inputs, 2 analog, and 2 digital, as well as 4 outputs, 2 analog and 2 digital. The analog inputs support both XLR and TRS and are fitted with pre-amplifiers, to ensure the quality of the signal. The digital inputs are Coaxial and MIDI, with one of each. The digital output is identical to the input, with both coaxial and MIDI. The analog output consists of 2 TRS, as well as a headphone jack, though this is not intended to be utilized for recording purposes. The card also has a USB type B interface for interaction with computers. The device is built to have low-noise components and comes with sensitivity sliders for the analog inputs, as well as an auto sensitivity button. The interfaces can be seen below

Figure 5.1: Front interface [39]

Figure 5.2: Back interface [39]

ER-10C DPOAE Probe System

The ER-10C is a specialized device, developed by Etymōtic Research, and is exclusively meant for measuring DPOAEs. It consists of 2 major components, namely the probe itself, and the Pre-amplifier. The probe is made with a 6-foot shielded cable, to reduce noise, and is equipped with a microphone which has a sensitivity of -46 dB per $\frac{1V}{\mu Bar}$ [40]. The speakers in the probe each have a sensitivity of 72dB SPL at 1kHz. The pre-amplifier runs exclusively on 9v batteries, to isolate it from

Figure 5.4: Pictures of the modified ER-10C

the electrical grid (in an effort to decrease noise in the recorded OAE), and has a battery life of an estimated 35 hours. It has 2 analog inputs, and 1 analog output for the microphone, all of which are BNC cable ports. The amplifier has 3 settings. +0, +20, and +40 dB.
It should be noted that the Pre-amplifier is modified, to have a high pass filter for 300Hz and 600Hz, and this can be seen in the figure 5.4. This additional functionality has been disabled during calibration but used when taking measurements.

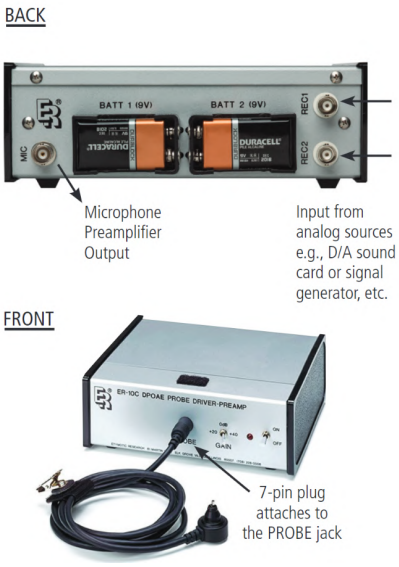


Figure 5.3: Pictures of the standard ER-10C [40]

5.1.1 Training hardware

Neural networks were trained on two hardware configurations:

- Lenovo ThinkPad P54 mobile workstation
- CLAUUDIA Strato VM with Nvidia T4 GPU

Lenovo ThinkPad P53 hardware specification

Lenovo ThinkPad P53 is a powerful, mobile workstation with a dedicated GPU equipped [41].
Its hardware specifications are as follows:

Component name	Component parameters
CPU	Intel Core i9-9880H @ 2.4 GHz 8c16t
GPU	Nvidia Quadro RTX4000 @ 1004 MHz, 8 GB VRAM
RAM	32 GB (2x16GB) @ PC4-25600 (3200 MT/s)
Storage	512 GB NVMe SSD

CLAUUDIA Strato VM

The VM used on CLAUUDIA is equipped with a dedicated Nvidia T4 GPU, specifically designed for AI-related workloads [42].
The VM hardware specifications are as follows:

Component name	Component parameters
CPU	AMD EPYC Rome @ 3 GHz 10c10t
GPU	Nvidia T4 Tensor Core GPU @ 1590 MHz, 16 GB VRAM
RAM	40GB
Storage	100 GB + 300 GB of virtualized storage

Both setups were used to train both neural networks, and the training results were reported to Weights and Biases [43] service for experiments tracking and visualisation.

5.2 Probe calibration

This section will be about the Calibration of the probe was done so that measurements could be taken safely, reliably and repeatable. Including the previously mentioned hardware, there will also be needed a reference syringe and B&K 4157 [44] that simulates the ear and a B&K type 4231 sound calibrator [45] that has a pressure of 1 Pa at 1 kHz and 94 dB. The calibration can be done with the calibration file (*calibration.m*) which is in the *OAEsystem-main* folder.

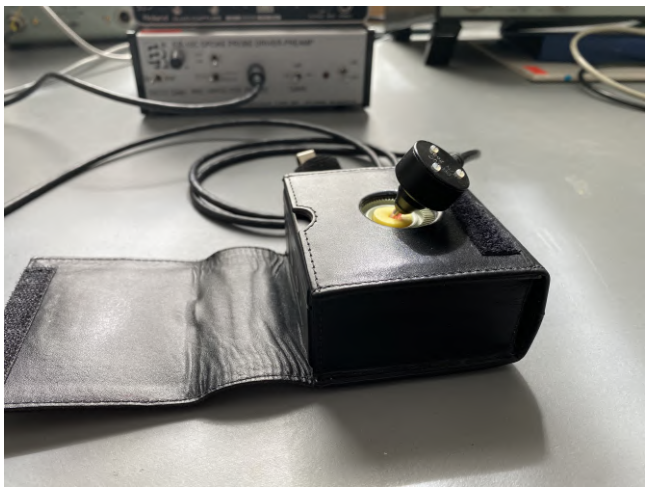


Figure 5.5: The probe which is the black earphone connected with the yellow tip to the calibrator

To start calibrating connect insert the yellow tip foam into the probe and then connect the probe connected to the ER-10C DPOAE probe driver, set the driver to +20 dB gain a fixed REC1 gain and the filter off. The probe drive is then connected to the Roland quad capture card, where REC1 is connected to output 1L, REC2 is connected to output 2R, and MIC is connected to the input of a multimeter or voltmeter. When everything is connected the probe can be inserted in the calibrator like in 5.5 and then can be turned on and the measured value on the voltmeter should be 500 mV_{RMS}, if that isn't the case the next steps should be scaled according to the error.

The second step in calibrating the probe is to connect MIC from the probe driver to the input of the Roland quad capture sound card. Then connect the probe to the B&K 4157 ear simulator and connect that to the input for a measuring amplifier and its output to the second input of the Roland quad capture sound card. The connection between the B&K 4157 ear simulator and the probe can be seen in figure 5.6 and the connection between the probe driver, sound card and measuring amplifier can be seen in figure 5.7.

Figure 5.6: The probe connected to the B&K 4157 ear simulator

Figure 5.7: The connection between the measuring amplifier, sound card, and probe driver

A $1V_{RMS}$ and 1 kHz tone can then be transmitted to the probe and the input sensitivity can then be adjusted to the point just before it peaks.

Then read out the digital values and calculate the RMS value with respect to the first second of the recorded data being skipped because of peaks in the beginning.

After the value is saved the transmission of the $1V_{RMS}$ and 1 kHz tone can be stopped and a 1kHz with a digital amplitude of 1 corresponding to a digital RMS amplitude of $\frac{1}{\sqrt{2}} = 0.707$ can be transmitted and the output value on the sound card can be measured.

The fifth step is to connect the sound card's input to its output and then run a Maximum Length Sequence (MLS) to get the impulse response and transfer function of the sound card.

With all these steps there should be saved values for microphone sensitivity, the microphone sensitivity coupler, the sound card input value, the sound card right and

left output value, and the speaker left and right value. The values saved by the program in this project were:

- Microphone sensitivity; mic_sens : 0.4500 V_{Pa}
- Microphone sensitivity coupler; MicSensCoupler: 0.0122
- Sound card in; scIN : [0.4140 $\frac{dig}{V}$, 0.41162$\frac{dig}{V}$]
- Sound card left out; scLOUT : 2.0000 $\frac{V}{dig}$
- Sound card right out; scROUT : 2.0500 $\frac{V}{dig}$
- Speaker left; speakerL : 0.2863 $\frac{Pa}{V}$
- speaker right; speakerR : 0.0790 $\frac{Pa}{V}$

The final step to be sure that the calibration was done correctly is to connect the probe to the syringe which can be seen in figure 5.8

Figure 5.8: The probe connected to the reference syringe

Then run the calibration check for the probe syringe which is the *cali_check_probeinsyringe* file in the *OAEsystem-main* folder, and with the reference syringe (*refinSyringe*) values being 81.4522 dB SPL for the left speaker and 80.8214 dB SPL for the right speaker get the measured dB SPL and the difference from the reference as output from the program.

5.3 Measurement Protocol

Additional data was measured using a program used in [13], implemented in Matlab. The software is capable of measuring both DPOAE and TEOAE. To be consistent with prior measurements, the same parameters were selected. Four measurements per person were taken, two per ear with probe refitting between measurements, for better variability in data. The measurements parameters are as follows:

- Sampling frequency of 48000Hz.
- 512 measurements, divided into two groups A and B (256 measurements per group)
- Measurement rejection threshold of 40 dB SPL.
- Band-pass filter with frequency range 500 Hz - 10 kHz.
- Time rejection of 3.8 ms to make sure, that the stimuli used to evoke TEOAE is not included.
- The spacing between stimuli is 20 ms.

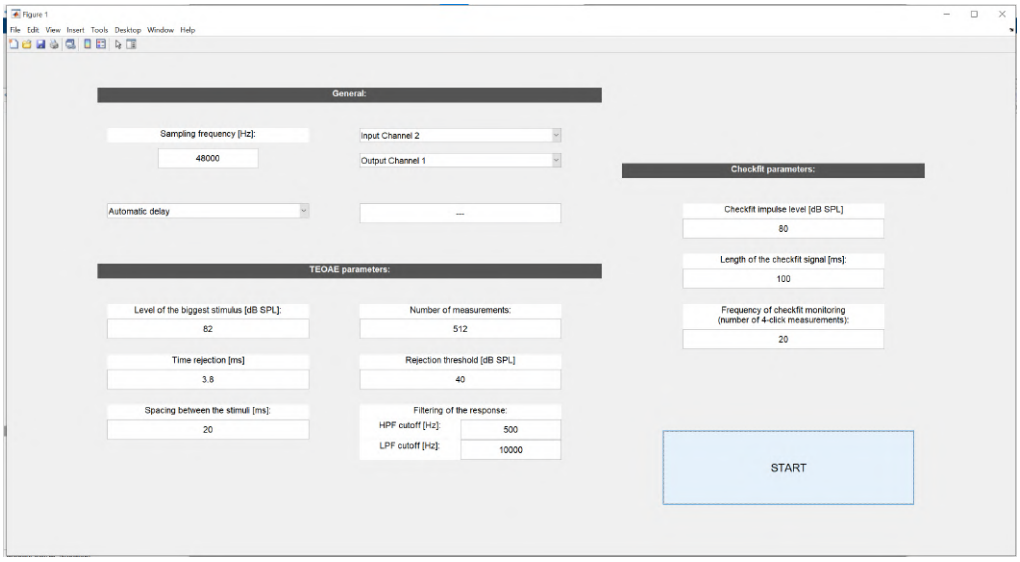


Figure 5.9: Screenshot of the parameter selection screen.

Before taking measurements, the so-called Check-fit procedure is performed. Its purpose is to make sure, that the probe is inserted correctly, by playing a click into the ear and listening back to the reflected sound. The threshold of received sound is set to 80 dB SPL, to be consistent with the previous measurements.

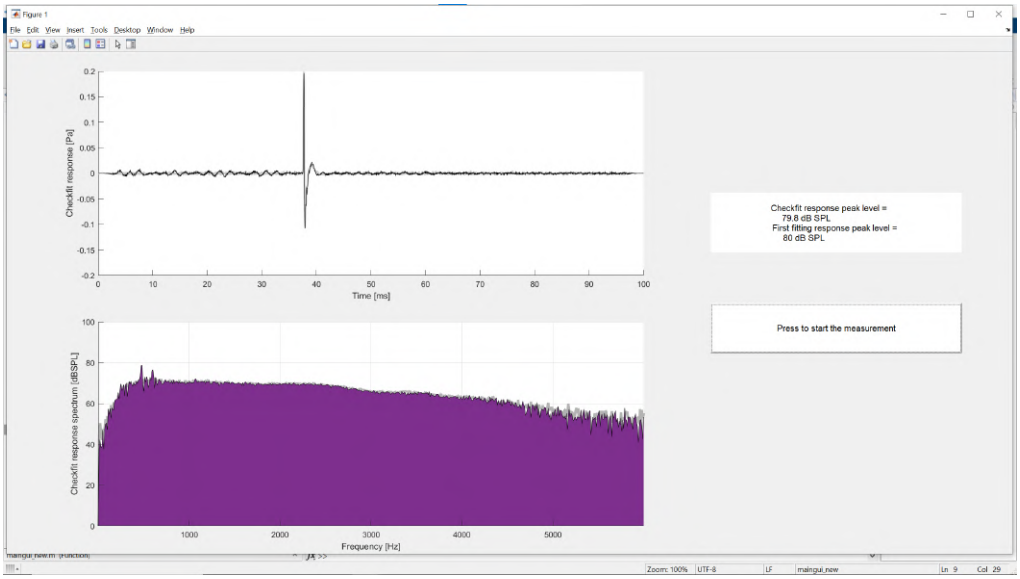


Figure 5.10: Screenshot of parameter check-fit procedure.

On the screenshot 5.10 the first plot represents received check-fit stimuli. The plot is saved to a file to be referenced during the next check-fit procedures. If the plot shape is vastly different in comparison to the saved one, then the probe is inserted wrong or the ear-tip might be clogged with earwax. The same applies to the check-fit response peak level - the level should be more-less around the first fitting level.
After making sure, that the probe is fitted correctly TEOAE measurement begins.

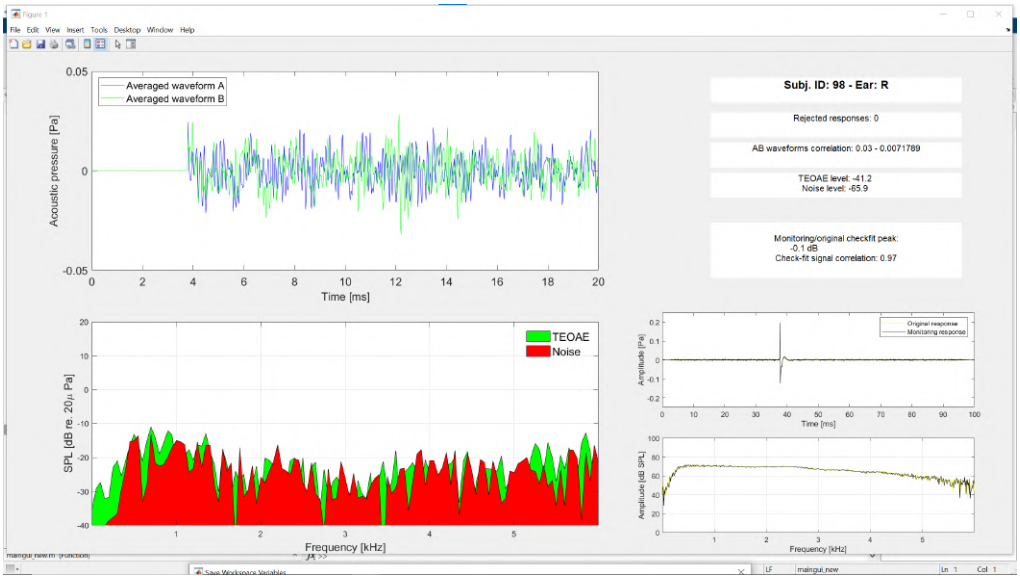


Figure 5.11: Screenshot of TEOAE measurement procedure.

Figure 5.11 represent the TEOAE measurement procedure. The first section plots the recorded TEOAE. The recordings are grouped into the mentioned A and B groups, and then their correlation is calculated. Poor correlation might indicate poor OAE or probe fitting. The second plot displays the frequency spectrum of both TEOAE and noise, with their magnitudes (in dB SPL). They are calculated using the FFT function. During the measurements, the response is monitored to make sure that the probe doesn't move too much. Changes in the monitoring response indicate that the probe moved too much during measurements and the measurement might need to be retaken.

Figure 5.12: Photo of measurement probe in a test mannequin's ear.

Figure 5.12 shows probe placement in a test mannequin. The foam probe expands in the ear canal to ensure a good fit.

5.3.1 Measurement Setup

Figure 5.13: Screenshot of measurement procedure on a test mannequin.

Figure 5.13 shows the overall measurement setup with the probe being fitted to a test mannequin. To achieve consistent results, there are 2 main components in the setup, namely the system itself, and the isolation of external factors. As such both are briefly described here.

System Loop

The system loop is intentionally kept simple, to reduce points of failure. The PC is connected to the sound card via the USB type B. The sound card runs 3 TRS to BNC cables, Specifically an analog input, and both analog outputs. These interface with both inputs and the output of the probe driver, which corresponds to both speakers and the microphone in the probe, the probe is connected to the driver, using only one cable for both input and output. The entire loop can be found in 5.14

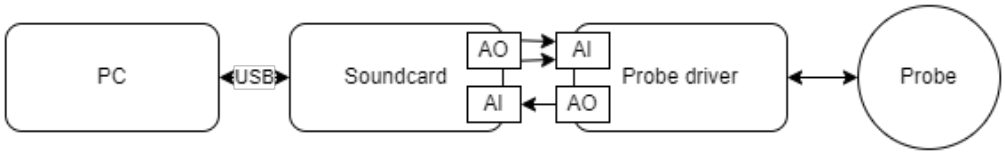


Figure 5.14: Simple illustration of the system loop

Factor Isolation

To isolate the data as much as possible all other stimuli, as well as sources of noise should be removed. To achieve this, it is recommended to conduct the experiment in an acoustically isolated environment. The environment should be plain with soft lighting, and the subject should be facing the wall, to reduce visual stimuli from observing the test conductor. No sounds should be made during the measurements. To avoid introducing noise, the probe cable, which is sensitive to external stimuli, should not move during measurements, and should especially not be rubbing against anything, as this will affect the readings greatly. The subject should be motionless, with steady breathing during measurements. It is also recommended to have the subject avoid swallowing during measurements, if possible.

5.3.2 OAE Dataset

Combining data provided by [13], data collected from AAU students, data collected on the test syringe and measurements taken on a test mannequin, a dataset was created. The dataset is divided into three - same fitting data (in_fitting_data), same person but not the same fitting data (out_of_fitting_data), and outsider data (outsider_data):

- in_fitting_data - measurements of 36 people/classes, included in the neural network training. The data is further divided into the train, validation and test directories.
- out_of_fitting_data - measurements of the same 36 people, not included in the training, used to test how well the trained classifiers generalize.
- outsider_data - measurements of 3 subjects (classes 37-39), test mannequin and test syringe not included in the training, used to check if the trained networks are able to reject unknown data, based on rejection threshold.

The same fitting data (in_fitting_data) is being processed in Matlab. Depending on the classifier type (either 1D CNN or 2D CNN), the measurements are split into train, and validation. Test directories either without transforming the measurements into time-frequency representations (1D CNN) or with such a transformation (2D CNN). The first 183 samples are being removed, due to the time rejection settings. The samples mentioned are zeroed out by the measurement software, so there is no point in keeping them. They hold no information and still occupy the memory. The sample values are being rescaled to range from -1 to 1. To simplify the implementation of the dataset class in PyTorch, all 512 measurements, that are contained within a single *.mat* file which is split into multiple files, where one file contains one measurement. The first 179 measurements are being copied to the test directory, the next 150 measurements are validation, and the rest are test.

To summarize file counts:

- train - 51552 measurements.
- val - 14698 measurements.
- test - 7488 measurements.

The data on which 1D CNN is being trained are being augmented by adding pre-recorded noise files at different SNR ratios, in an effort to create a more robust classifier. 4 files are being used, hosted on Freesound [46]:

- "crowd ext large dense street before parade in crowd Montreal" by kyles on Creative Commons 0 licence.
- "crowd int medium theater lobby quebecois voices7 lower ceiling Montreal, Canada" by kyles on Creative Commons 0 licence.
- "small cafe ambience" by Tomlija on Creative Commons CC-BY licence.
- "crowd int large basement reception room quebecois voices2 Montreal, Canada" by kyles on Creative Commons 0 licence.

Data processing

The data split is performed in Matlab because the CWT transform is built-in into one of Matlab's toolboxes. The only difference between Matlab scripts is that in the case of 2D CNN, the CWT transform is being calculated and the names of variables are different. The overall algorithm is the same.

```
mat_files = dir("/home/lab-user/datasets/project_data/in_fitting_data/*mat");
target_train_path = "/home/lab-user/datasets/project_data/time-domain/train";
target_test_path = "/home/lab-user/datasets/project_data/time-domain/test";
target_val_path = "/home/lab-user/datasets/project_data/time-domain/val";
for cur_file = 1:length(mat_files)
    path = mat_files(cur_file).name;
    fullpath = sprintf("/home/lab-user/datasets/project_data/in_fitting_data/%s",
↪   path)
    loaded_mat = load(fullpath);
    for i = 1:179
        loaded_teoae_A = loaded_mat.Data.A(:, i);
        loaded_teoae_B = loaded_mat.Data.B(:, i);
        cut_teoae_A = loaded_teoae_A(183:912);
        cut_teoae_B = loaded_teoae_B(183:912);
        mat_filename = sprintf('%s/%s %d A.mat', target_train_path,
↪   mat_files(cur_file).name, i)
```

```
15          mat_filename_B = sprintf('%s/%s %d B.mat', target_train_path,
    ↪  mat_files(cur_file).name, i)
16          save(mat_filename, "cut_teoae_A");
17          save(mat_filename_B, "cut_teoae_B");
18      end
19      for i = 180:230
20          loaded_teoae_A = loaded_mat.Data.A(:, i);
21          loaded_teoae_B = loaded_mat.Data.B(:, i);
22          cut_teoae_A = loaded_teoae_A(183:912);
23          cut_teoae_B = loaded_teoae_B(183:912);
24          mat_filename = sprintf('%s/%s %d A.mat', target_train_path,
    ↪  mat_files(cur_file).name, i)
25          mat_filename_B = sprintf('%s/%s %d B.mat', target_train_path,
    ↪  mat_files(cur_file).name, i)
26          save(mat_filename, "cut_teoae_A");
27          save(mat_filename_B, "cut_teoae_B");
28      end
29      for i = 231:256
30          loaded_teoae_A = loaded_mat.Data.A(:, i);
31          loaded_teoae_B = loaded_mat.Data.B(:, i);
32          cut_teoae_A = loaded_teoae_A(183:912);
33          cut_teoae_B = loaded_teoae_B(183:912);
34          mat_filename = sprintf('%s/%s %d A.mat', target_test_path,
    ↪  mat_files(cur_file).name, i)
35          mat_filename_B = sprintf('%s/%s %d B.mat', target_test_path,
    ↪  mat_files(cur_file).name, i)
36          save(mat_filename, "cut_teoae_A");
37          save(mat_filename_B, "cut_teoae_B");
38      end
39  end
```

Listing 1: Training loop implementation.

Listing 1 implements data split. First, the script detects all TEOAE files. Then loops over the list of files. Then it takes n samples specified in the loops, and removes the first 183 elements, as they are zeroed per measurement settings. Then the files are saved to the output directories.

5.4 Classifier implementation

The deep learning classifier was implemented in Python using the Pytorch library [47]. Two different Convolutional Neural Network (CNN) architectures were developed. A 1D-CNN was directly trained and used on the TEOAE time-domain signal, and a 2D-CNN was developed and trained on the data processed by the Continuous Wavelet Transform (CWT).

5.4.1 1D-CNN

The 1D CNN architecture has 4 layers of convolution that are followed by 2 layers of fully connected layers before being fed into the *LogSigmoid* layer which converts our outputs of the fully connected layer into the log-likelihoods for every class. As can be seen from figure 5.15, the 1st convolutional layer gets an input of 1x730, where 1 is the channel and 730 is the input signal and the output of the convolution is a matrix of 500x728.

The convolution is then activated by the Tanh function, which also activates negative values. The output is then inputted to a 1D *Batchnorm*, which normalizes the batch using the equation $y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$ before *MaxPool* is used to downsample and outputs a matrix of 500x242.

After this convolution, Tanh, and *MaxPool* block a drop layer is used which randomly zeroes some elements in the input with a probability that is given using samples from a Bernoulli distribution helping to generalise the model. After the first convolutional and drop layer, the 2nd, 3rd, and 4th convolutional layers do the same. The last layer has some different parameters for *MaxPool*, the parameters for these 4 layers is as follows:

- Conv1D, kernel size 3 and stride 3
- Tanh
- Batchborm1D
- MaxPool1D kernel size 3 and stride 3
- Drop layer with probability 0.5
- Conv1D, kernel size 3 and stride 3
- Tanh
- Batchborm1D
- MaxPool1D kernel size 3 and stride 3

- Drop layer with probability 0.5
- Conv1D, kernel size 3 and stride 3
- Tanh
- Batchborm1D
- MaxPool1D kernel size 3 and stride 3
- Drop layer with probability 0.5
- Conv1D, kernel size 3 and stride 3
- Tanh
- Batchborm1D
- MaxPool1D kernel size 5 and stride 1
- Drop layer with probability 0.5

The 4 convolutional layers are responsible for learning the filter coefficients, which will identify TEOAE features distinct to a specific class. This output is then flattened and used as input for the fully connected layer. The fully connected layer applies a linear transformation of $y = xA^T + b$, where A is the input, x is the weight, and b is the bias that is learned. The 2 fully connected layers downsample the 48000 features from the convolutional layers down to the number of classes that are trained on, where *LogSigmoid* is applied and the log probability is returned. *LogSigmoid* is used as it is faster and has better numerical properties.

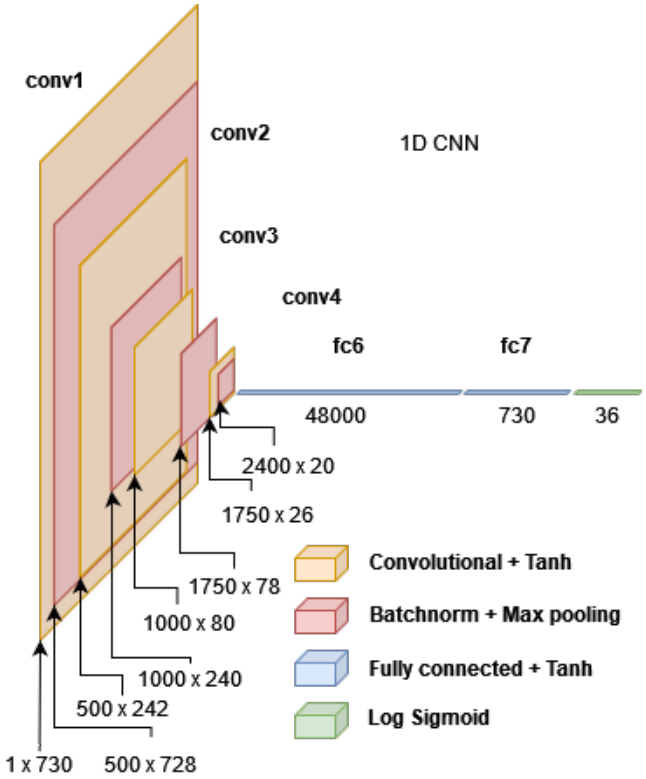


Figure 5.15: The architecture for the 1D CNN

5.4.2 2D CNN

Similarly to 1DCNN, the architecture consists of 4 convolution layers, but in this case, a previous implementation of 1D layers became their 2D variants. *Conv2D* layers are followed by *ReLU* activation functions. All but last layer include *MaxPool2D* layer. The architecture includes Dropout layers (at 0.6 probability) which deactivate neurons at random to combat the overfitting problem. To make a decision, the output from the last layer is flattened and fed into 2 fully-connected layers, to be fed into *LogSoftmax* to convert the fully-connected layers into log probabilities.

The layer's parameters are as follows:

- Conv2D, kernel size 5 and the padding set to 'same', to preserve input data size.
- BatchNorm2D with input of 10 features
- MaxPool2D with a kernel size of 5
- ReLU activation function

- Conv2D, kernel size 5 and the padding set to 'same', to preserve input data size.
- BatchNorm2D with input of 20 features
- MaxPool2D with a kernel size of 5
- ReLU activation function
- Conv2D, kernel size 5 and the padding set to 'same', to preserve input data size.
- BatchNorm2D with input of 40 features
- ReLU activation function
- Conv2D, kernel size 5 and the padding set to 'same', to preserve input data size.
- BatchNorm2D with input of 80 features
- ReLU activation function
- Conv2D, kernel size 5 and the padding set to 'same', to preserve input data size.
- BatchNorm2D with input of 160 features
- Dropout layer with 0.6 probability of disabling a neuron
- ReLU activation function
- Linear layer with 12160 input features and 12160 output features
- Linear layer with 12160 input features and 36 (number of classes) out features
- LogSoftmax layer to convert a linear layer into likelihoods.

Because 1D CNN architecture performs significantly better, is faster and uses less memory to train and use, the focus was put on 1D CNN networks.

5.4.3 Code implementation

Architecture implementation

Both CNN implementations consist of a class deriving from `nn.Module` class. Such a class implements necessary methods and attributes, making sure that code is consistent and describes necessary model attributes [48]. Several variants of the same base architectures were considered, with varying filter counts (`channel_size` attribute)

```
class OAEClassifierSmallerLayers2DCNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.dropout = Dropout(0.6)
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=10,   kernel_size=5,
        ↪  padding="same")
        self.conv2 = nn.Conv2d(in_channels=10,out_channels=20,  kernel_size=5,
        ↪  padding="same")
        self.conv3 = nn.Conv2d(in_channels=20,out_channels=40,  kernel_size=5,
        ↪  padding="same")
        self.conv4 = nn.Conv2d(in_channels=40,out_channels=80, kernel_size=5,
        ↪  padding="same")
        self.conv5 = nn.Conv2d(in_channels=80,out_channels=160, kernel_size=5,
        ↪  padding="same")
        self.maxpool1 = nn.MaxPool2d(kernel_size=5)
        self.maxpool2 = nn.MaxPool2d(kernel_size=5)
        self.batch_norm1 = nn.BatchNorm2d(10)
        self.batch_norm2 = nn.BatchNorm2d(20)
        self.batch_norm3 = nn.BatchNorm2d(40)
        self.batch_norm4 = nn.BatchNorm2d(80)
        self.batch_norm5 = nn.BatchNorm2d(160)
        self.relu = nn.ReLU()
        self.fc = nn.Linear(in_features=12160, out_features=12160)
        self.fc1 = nn.Linear(in_features=12160, out_features=36)
        self.logSoftmax = LogSoftmax(dim=1)

    def forward(self, x):
        x = self.conv1(x)
        x = self.batch_norm1(x)
        x = self.relu(x)
        x = self.maxpool1(x)

        x = self.conv2(x)
        x = self.batch_norm2(x)
        x = self.relu(x)
        x = self.maxpool2(x)

        x = self.conv3(x)
        x = self.batch_norm3(x)
        x = self.relu(x)

        x = self.conv4(x)
        x = self.batch_norm4(x)
        x = self.relu(x)
```

```
40
41          x = self.conv5(x)
42          x = self.batch_norm5(x)
43          x = self.relu(x)
44
45          x = flatten(x, 1)
46          x = self.dropout(x)
47          x = self.fc(x)
48          x = self.relu(x)
49          x = self.fc1(x)
50          x = self.logSoftmax(x)
51          return x
```

Listing 2: Code describing an implementation of a PyTorch model class. The 1D CNN counterparts will have a similar structure - just the layers will be different.

In the listing, 2 the `forward` method executes the architecture, by passing the input samples `x` (the batch, its size set by `batch_size` parameter in `DataLoader` class) to the layers. The `__init__` function initializes the class and defines the neural network architecture as class attributes.

Dataset implementation

PyTorch offers a standardized way to create datasets, by utilizing the `Dataset` (which serves as a 'container' for the dataset itself) and the `DataLoader` class (which loads the dataset to memory and creates the batches)

```
1
2   class OAEMatDataset(torch.utils.data.Dataset):
3       def __init__(self, dset_path, type: DatasetType):
4           self.dset_path = dset_path
5           self.dset_type = type
6           self.data, self.class_assigments = self.__load_dset()
7
8       def __getitem__(self, index):
9           return torch.tensor(self.data[index], dtype=torch.float).squeeze(0),
            ↪  torch.tensor(self.class_assigments[index])
10
11      def __len__(self):
12          return len(self.data)
```

Listing 3: Code describing a definition of a PyTorch `Dataset`

Listing 3 defines the `Dataset`, which implements initialization function `__init__`, `__getitem__` functions, which return OAE samples and its class assigments as an PyTorch tensor. The `__len__` returns dataset's sample number, necessary for `DataLoader` to function.

```
    def __load_dset(self):
        dset_path_obj = pathlib.Path(self.dset_path)

        if dset_path_obj.is_file():
            raise ValueError("Target dataset path should be a directory, not file.")

        if self.dset_type == DatasetType.DP:
            glob_pattern = "*DP*mat"
        elif self.dset_type == DatasetType.TE:
            glob_pattern = "*TE*mat"
        else:
            raise ValueError(f"Invalid dataset {self.dset_type} type. Either DP or
            ↪  TE are correct.")

        mat_files = dset_path_obj.glob(glob_pattern)
        class_assigments = []
        data = []
        for i, mat_file in enumerate(mat_files):
            split_filename = str(mat_file.stem).split(" ")
            try:
                loaded_mat = scipy.io.loadmat(str(mat_file))["output_teoae"][0]
                converted = [float(item) for item in loaded_mat]
                converted = converted / numpy.max(numpy.abs(converted))
            except KeyError as e:
                print("Invalid matfile, it should contain output_teoae key.")
                raise e
            class_number = int(split_filename[0][1:]) - 1
            class_assigments.append(class_number)
            data.append(converted)

        return data, class_assigments
```

Listing 4: Code describing an implementation of a PyTorch `Dataset`

Listing 4 loads the OAE Dataset samples. The method is able to load both DPOAE and TEOAE data. The method `__load_dset` first scans directory defined by `dset_path` in search for `*.mat` files. After getting the file list, the method iterates over the file list, loads Matlab's `mat` file containing a single OAE recording and creates a class

assignment based on the sample's filename.

Training loop

The training loop creates instances of classes mentioned in listing 2 and 3, but also the `Dataloader` and optimizer classes. The loop is also responsible for training and validation processes.

```
1  dset_train =
   ↪ OAEAugmentedMatDataset("/home/lab-user/datasets/project_data/time-domain/train",
2                                        type=utils.datasets.DatasetType.TE,
                                         ↪ noise_path="/home/lab-user/datasets/noises")
3
4  dset_val =
   ↪ OAEAugmentedMatDataset("/home/lab-user/datasets/project_data/time-domain/val",
5                                        type=utils.datasets.DatasetType.TE)
6  with open('config.yaml') as file:
7      sweep_configuration = yaml.load(file, Loader=yaml.FullLoader)
8  def main():
9      torch.manual_seed(2137)
10     run = wandb.init()
11     batch_size = int(wandb.config.batch_size)
12     epochs = int(wandb.config.epochs)
13     lr = float(wandb.config.lr)
14     train_loader = torch.utils.data.DataLoader(dset_train, batch_size=batch_size,
       ↪ shuffle=True, num_workers=16,
15                                                drop_last=True)
16     val_loader = torch.utils.data.DataLoader(dset_val, batch_size=batch_size,
       ↪ shuffle=True, num_workers=16,
17                                              drop_last=True)
18     device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
19     # device = "cpu"
20     oae_classifier = model.arch.MainOAEClassifierSigmoid().to(device)
21     loss_function = nn.CrossEntropyLoss().to(device)
22     optimizer = torch.optim.Adam(oae_classifier.parameters(), lr=lr,
       ↪ weight_decay=lr)
23     wandb.watch(oae_classifier)
```

Listing 5: Code initializing and instantiating necessary components for training to start.

Code example 5 creates global instances of `train` part and `val` part of the OAE-Dataset. Then the random seed is set, to ensure repeatability of training runs. The hyperparameters `batch_size`, `epochs` and `lr` are being loaded from `config.yaml`

file. Next, the PyTorch `Dataloader` classes are being created, which split the whole dataset into batches and shuffle the dataset [49], execution device is created (either CPU or a GPU), instances of the neural network model, loss function and optimizer are being created. Lastly, Weights and Biases library is instructed to monitor the training process [43].

```
for epoch in tqdm(range(0, epochs)):
    cur_loss = 0.0
    oae_classifier.train(True)
    for i, data in enumerate(train_loader, 0):
        optimizer.zero_grad()
        inputs, class_names = data
        inputs = inputs.to(device)
        class_names = class_names.to(device)
        outputs = oae_classifier(inputs)
        loss = loss_function(outputs, class_names)
        loss.backward()
        optimizer.step()
        cur_loss += loss.item()
    acc_train = 0
    for output, gt in zip(outputs, class_names):
        with open('train_prob.csv', 'a', newline='\n') as csvfile:
            test_writer = csv.writer(csvfile, delimiter=';')
            norm_outs = torch.exp(output.detach().cpu())
            listified = norm_outs.tolist()
            listified.append(gt.item())
            test_writer.writerow(listified)
        max_val = max(output.detach())
        classification = (output == max_val).nonzero(as_tuple=True)[0]
        if classification == gt:
            acc_train += 1
    print({"train_loss": cur_loss / len(train_loader), "train_acc": acc_train /
    ↪ len(class_names)})
    wandb.log({"train_loss": cur_loss / len(train_loader), "train_acc": acc_train /
    ↪ len(class_names), "Epoch": epoch})
```

Listing 6: Training loop implementation.

Listing 6 implements the training loop. First, the loss for an epoch is set to 0, then the model is set into training mode [50]. Looping over `train_loader`, we are getting batches created by `DataLoader` objects, restarting the gradients and setting execution devices for the neural network instance and the data itself. After all of the batches are processed, in the second loop we are saving model outputs, in order to calculate

the rejection threshold and check if the model is able to perform a classification. It is done by getting the maximum value among the model output and checking the class assignment of the value. Lastly, the training progress is being reported to Weights and Biases [43].

5.4.4 Training statistics

The training and validation are monitored by Weights and Biases service [43]. During run-time, train loss, train accuracy, validation loss, and validation accuracy metrics were logged to the service, so it's possible to monitor training progress in real-time.

What's important, is not the specific values of train and validation losses. It's the overall trend of the plots - it's important that the losses decrease until they converge (stops decreasing) at a value - that means that the network is correctly learning. If at some point train loss keeps decreasing, but the validation loss starts to rise significantly and the validation accuracy starts to drop, then the trained model is probably overfitting [51].

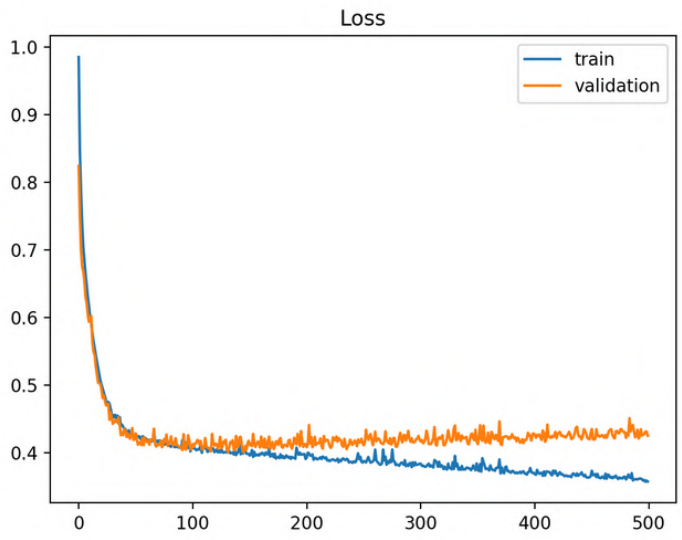


Figure 5.16: Example plot showing that the network is being overfitted [51].

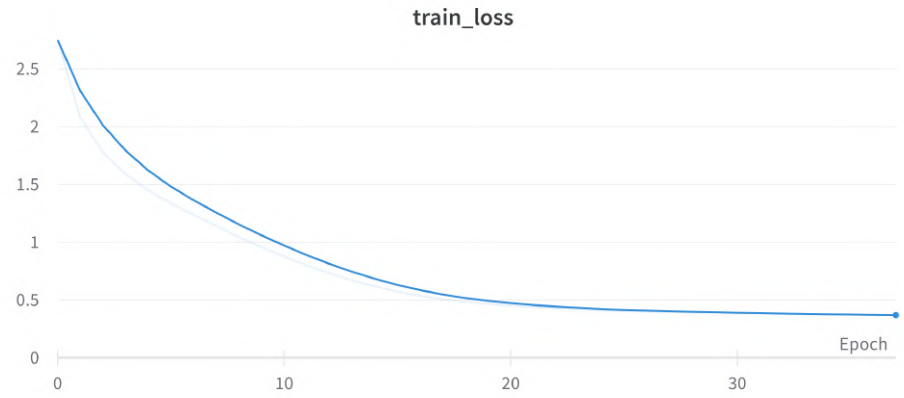


Figure 5.17: The train loss plot

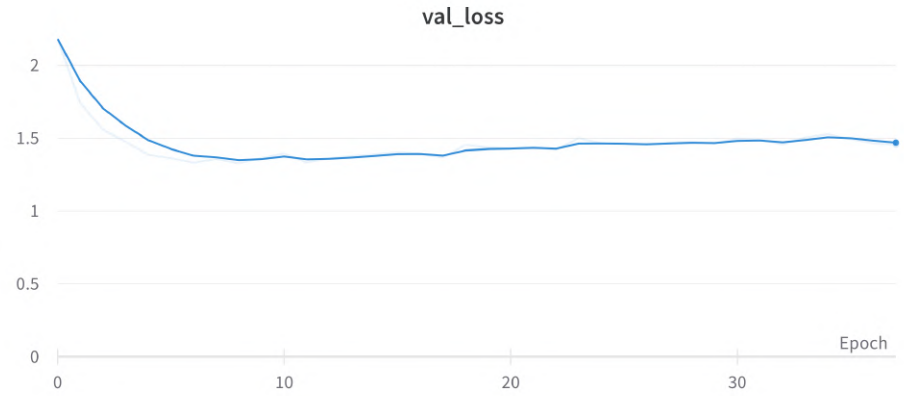


Figure 5.18: The validation loss plot

Figures 5.17 and 5.18 indicate that the network was trained successfully. The network converged around the 20th epoch. Around the 20th epoch, val loss did increase a bit. It took 16 hours to train for the 33rd epoch, which was the model used for testing.

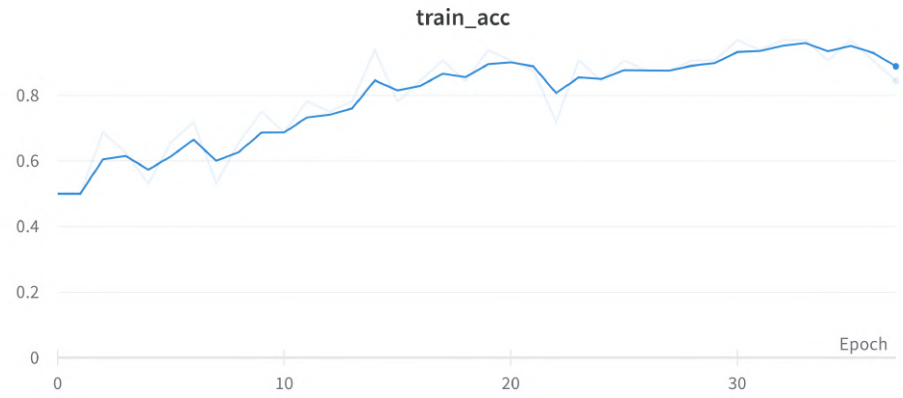


Figure 5.19: The train accuracy plot

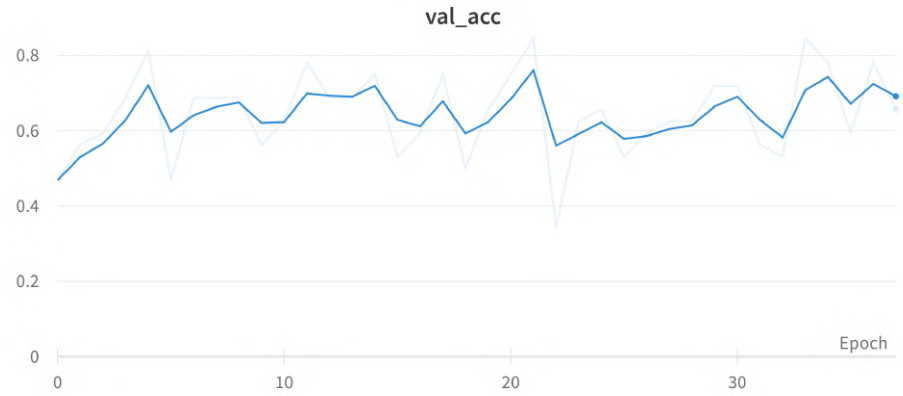


Figure 5.20: The validation accuracy plot

Figures 5.19 and 5.20 show that the network did learn specific class features and was able to perform classification. Accuracy during training reached around 80% and validation accuracy reached 75%.

5.4.5 Validation/testing loop

The validation and testing loop uses the same code as they are the same thing only distinguished by the fact that validation is used to get an early estimate of the performance of the model. The validation is also used to get the sensitivity threshold, which can be used for the test. In listing 7 the main function is defined for the program this initializes and instantiates the files that need to be evaluated, the device it has to be run on, and which model has to be run.

```
def main():
    test_dset = OAEAugmentedMatDatasetTest({dataset location}, DatasetType.TE)
    test_loader = torch.utils.data.DataLoader(test_dset, batch_size=512,
    ↪  shuffle=True, num_workers=16)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model_load = pathlib.Path("Saved models").glob("{model name and location}")
    best_model = ""
    tp = 0
```

Listing 7: Initializing and instantiating the necessary components for the validation and test loop

After initialization and instantiating the necessary components, the loop for the validation and test can be run, which can be seen in listing 8. This loop can go through multiple models if they instantiated, and are then set to evaluating mode as this loop is for validation/test. Another loop goes through the loaded files and inputs the samples into the model where after getting the output another loop goes through the output and appends the ground truth and confidence which is a value of 0 to 1 for every class into a Comma Separated Values (CSV) file.

```
    for loaded_model in model_load:
        oae_classifier = torch.load(str(loaded_model)).to(device)
        oae_classifier.eval()
        print(f"Loaded model: {loaded_model}")
        with torch.no_grad():
            for i, data in enumerate(test_loader, 0):
                val_acc = 0
                inputs, class_names = data
                inputs = inputs.to(device)
                class_names = class_names.to(device)
                outputs = oae_classifier(inputs)

                for output, gt in zip(outputs, class_names):
                    max_val = max(output.detach())
                    linear_likelihood = torch.exp(max_val)
                    classification = (output == max_val).nonzero(as_tuple=True)[0]
                    if classification == gt:
                        tp += 1
                    with open(f'{file name and save location}.csv', 'a',
                    ↪  newline='\n') as csvfile:
                        test_writer = csv.writer(csvfile, delimiter=',')
                        norm_outs = torch.exp(output.detach().cpu())
                        listified = []
```

```
23
24                          listified.extend(norm_outs.tolist())
25                          listified.append(gt.item())
26
27                          test_writer.writerow(listified)
28
29                      print(f"GT: {gt.item()}, Classification:
    ↪  {classification.item()}. Confidence: {linear_likelihood}")
```

Listing 8: The validation and testing loop

Class 0	**Class 1**	**Class 2**	**. . .**	**Class 33**	**Class 34**	**Class 35**	***Actual Class***
0.000	0.000	0.002	. . .	0.000	0.003	0.000	*15*
0.001	0.000	0.001	. . .	0.000	0.001	0.001	*26*
0.000	0.001	0.000	. . .	0.000	0.001	0.433	*22*
0.000	0.000	0.000	. . .	0.000	0.001	0.008	*14*
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0.011	0.000	0.000	. . .	0.035	0.001	0.001	*18*
0.000	0.002	0.000	. . .	0.001	0.000	0.001	*1*
0.000	0.002	0.001	. . .	0.001	0.000	0.000	*30*
0.005	0.000	0.000	. . .	0.000	0.000	0.000	*35*

Table 5.1: Snippet of the CSV file output from the validation.

5.5 Validation Analysis

The Validation data is input into the testing system in order to get the sensitivity threshold as explained in section 2.4.1.

5.5.1 Output of validation

The testing system outputs a CSV file containing the probabilities of each measurement being each class as per the sigmoid activation function described in section 2.3.2. This is represented by a value between 0 and 1. 0 being complete certainty that it's not a given class and 1 being certainty that it is. The validation set contains 14688 measurements with 36 numbers between 0 and 1 each, equating to 528768 values in total. A table of the output can be seen in table 5.1.

5.5.2 Sensitivity thresholding

Using the CSV file, the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN), of the validation data were derived as described in section 2.3.1 for each threshold between 0 and 1 with a resolution of 0.0005. The FAR and FRR were then calculated and plotted for these values together with the GAR and ACE and seen in figure 5.21.

The accuracy and F1-score were also compared depending on the threshold as seen in figure 5.22.

From these two graphs, the four different possible thresholds can be found, as described in section 2.4.1.

- **Equal Error Rate** (EER) where FAR equals FRR and ACE, is acquired with a **threshold of 0.0015**, where the error rate is about 0.14 for each parameter.

- **Lowest ACE** value where the mean of FAR and FRR is at its lowest, is acquired with a **threshold of 0.0025**, where the FAR is 0.10, FRR is 0.16 and the ACE is 0.13.
- **Highest F1-Score** where the harmonic mean is at its highest, is acquired with a **threshold of 0.07**, where the F1-Score is 0.50.
- **Intersection between GAR and FRR** where the false rejection rate is the most stable, is acquired with a **threshold of 0.525**, where the FRR and GAR are 0.50.

By comparing these thresholds with figure 5.21, figure 5.22 and figure 5.23, none of these would be able to reach all requirements from section 3.2. However choosing between those four, the one that seems the most reasonable is the Intersection between GAR and FRR with a threshold of 0.525 since the rest are below 50% confidence, which would be too low, so for the sensitivity threshold of the system **a threshold of 0.525 is chosen**. However, it would still be advantageous to look at how the other thresholds would affect the results of the tests.

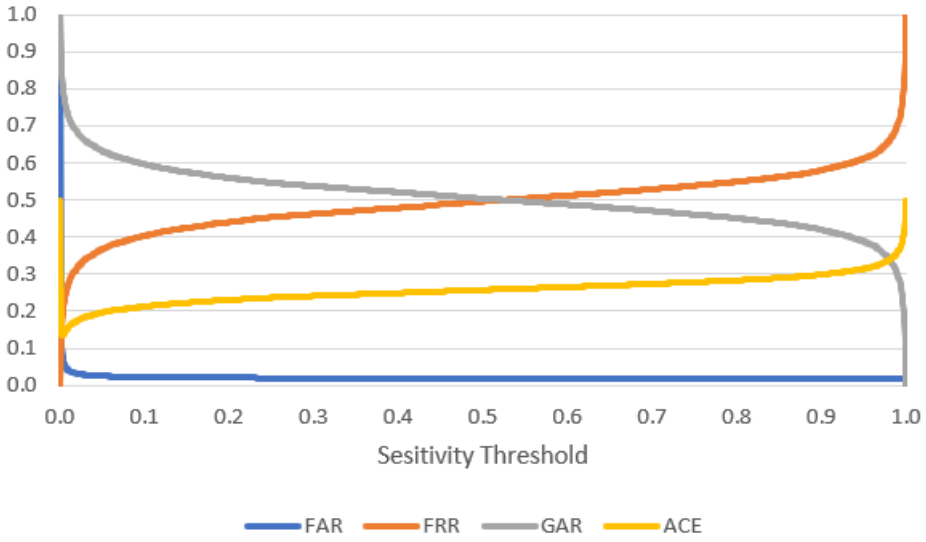


Figure 5.21: Graph comparing the False Acceptance Rate, False Rejection Rate, Average Classification Error and Genuine Acceptance Rate depending on the thresholds.

5.5.3 Analysing the threshold

Before the system and threshold are tested against the test data sets, some analysis of whether the threshold is viable can be made. Firstly the validation data can be inserted into a confusion matrix to see how the threshold has affected it. The

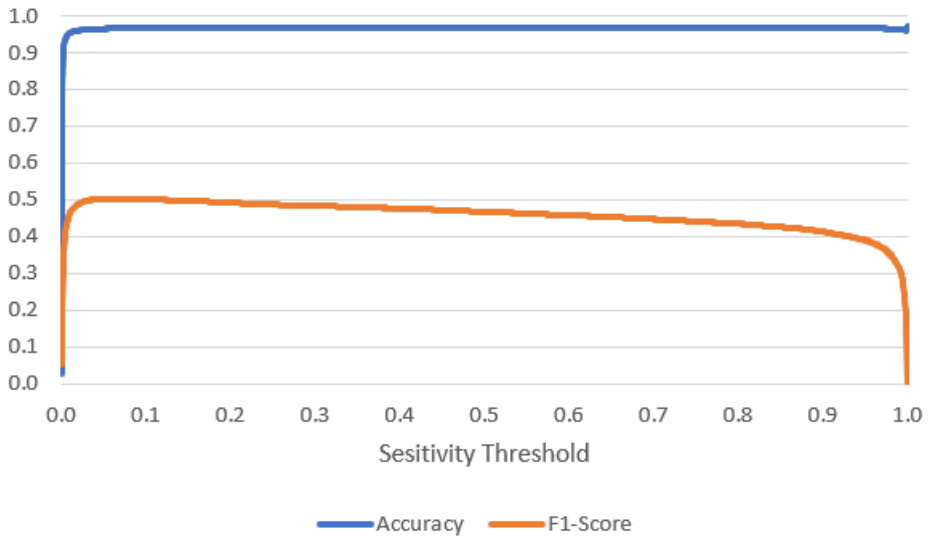


Figure 5.22: Graph comparing the Accuracy and F1-score of the validation depending on the thresholds.

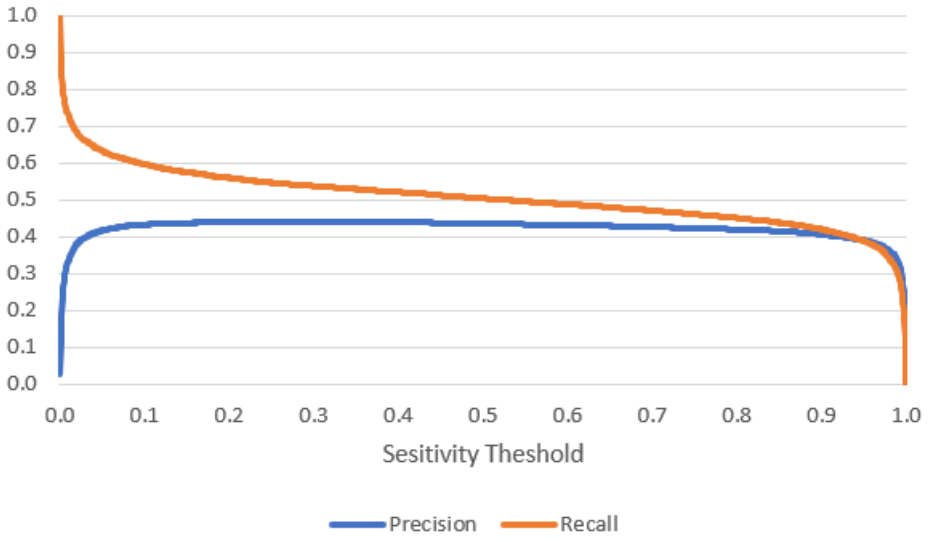


Figure 5.23: Graph comparing the Precision and Recall of the validation depending on the thresholds.

Ground Truth \ Prediction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	Outsider
0	255	0	0	1	0	3	0	0	1	0	0	0	2	1	2	0	0	1	0	0	2	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	14418
1	103	111	106	104	104	104	104	104	103	104	104	106	104	105	105	104	104	104	104	104	105	106	110	104	104	105	103	107	105	104	104	106	103	104	105	104	10318
2	5	3	19	0	1	2	1	1	3	1	10	1	1	1	1	1	0	1	1	1	1	2	4	2	1	2	1	3	1	3	1	1	1	1	2	1	14607
3	9	9	10	363	11	10	10	10	23	10	10	10	10	10	10	11	9	10	12	10	10	11	10	11	10	10	9	10	10	10	12	10	10	10	10	10	13958
4	0	0	0	0	330	0	23	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14308
5	6	3	3	3	3	273	3	3	4	4	3	4	4	3	3	3	3	3	4	5	3	4	4	4	4	3	3	4	3	4	3	3	3	4	4	3	14292
6	30	33	34	28	36	34	390	35	34	34	33	33	35	34	33	32	30	32	33	34	34	35	33	34	34	32	33	34	33	34	34	34	34	33	32	35	13133
7	0	0	0	0	3	0	15	388	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14282
8	3	4	4	14	5	5	3	3	81	3	3	3	4	4	3	3	3	3	4	3	5	5	8	3	3	6	4	3	3	5	4	4	3	3	4	3	14464
9	14	10	10	11	10	11	10	10	10	81	10	11	13	12	12	12	10	10	10	10	10	12	14	10	11	11	10	10	12	10	10	10	10	10	10	11	14230
10	4	2	2	2	2	4	2	2	3	2	252	3	3	2	4	2	2	2	2	2	3	2	2	2	2	2	2	4	2	2	2	2	2	2	2	5	14351
11	3	0	2	0	0	0	0	0	0	0	3	41	0	5	2	0	0	1	2	0	0	0	5	0	0	0	0	1	2	0	0	1	0	1	0	3	14616
12	16	13	15	13	15	20	15	14	15	15	17	15	102	15	17	16	14	16	15	19	20	15	15	16	15	15	14	15	15	16	15	15	17	18	14	15	14041
13	10	1	2	3	1	1	1	1	1	1	1	4	1	55	1	6	1	1	1	1	1	3	3	1	1	3	1	2	1	1	1	1	1	4	1	2	14567
14	6	6	8	6	6	9	7	6	6	6	9	6	21	7	179	6	5	6	7	9	6	6	6	6	8	7	6	7	6	6	7	7	9	6	6	8	14260
15	7	1	2	1	1	1	1	1	2	1	1	2	2	4	3	23	1	1	3	1	2	2	4	1	1	3	1	5	1	2	1	1	1	1	2	2	14599
16	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	403	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14281
17	3	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	332	1	0	0	0	2	0	0	1	0	0	1	0	0	0	0	0	0	1	14344
18	6	6	6	9	6	6	6	6	6	6	7	6	6	7	7	6	6	6	337	6	6	6	7	6	6	8	6	7	6	6	6	6	6	6	6	6	14131
19	5	5	5	5	5	6	5	5	5	5	5	5	5	5	5	5	5	5	5	297	5	5	5	5	12	5	8	5	5	5	5	5	10	5	5	5	14200
20	3	1	2	2	1	2	1	1	1	1	1	2	1	3	1	1	1	2	3	1	89	2	3	1	1	2	1	3	1	1	1	1	1	1	1	2	14546
21	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	0	1	1	1	1	229	3	1	1	1	1	1	1	1	1	1	1	1	1	3	14420
22	1	1	1	1	1	2	1	1	1	1	1	2	1	1	2	1	1	1	1	1	1	3	234	1	1	1	1	1	1	1	1	1	1	1	1	2	14413
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	386	0	0	0	0	0	0	0	0	0	0	0	0	14295
24	5	6	5	5	5	5	6	5	6	5	6	5	6	6	9	5	5	5	6	21	5	8	6	6	86	5	10	6	5	8	6	6	10	6	10	6	14372
25	7	4	5	4	4	7	4	4	4	4	4	4	4	4	5	6	3	5	4	4	4	4	8	4	4	81	4	6	4	6	4	5	5	7	4	4	14444
26	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0	0	0	0	2	0	387	0	0	1	0	0	1	0	0	0	14291
27	12	12	13	10	11	14	12	11	13	13	12	13	13	13	14	13	11	11	12	13	11	14	12	12	13	16	11	132	12	14	13	12	14	13	11	14	14118
28	1	0	1	0	0	1	0	0	0	0	0	2	0	2	0	0	0	0	1	0	1	5	1	0	0	0	0	1	3	0	0	1	0	1	1	2	14664
29	4	1	1	1	1	2	1	1	1	1	1	1	1	1	3	1	1	1	1	1	1	1	1	2	1	1	1	3	1	325	1	3	2	1	1	1	14316
30	10	7	7	8	7	8	7	6	7	7	8	8	7	7	7	7	7	7	8	7	8	9	8	7	7	7	7	9	7	7	292	7	7	9	7	8	14135
31	8	8	8	9	8	8	9	8	8	9	9	9	8	9	11	8	7	8	8	13	8	8	10	8	10	10	8	9	8	12	9	31	10	10	8	9	14347
32	6	6	6	6	6	10	6	6	6	6	6	6	6	6	7	6	6	6	6	19	6	6	6	6	11	7	9	6	6	6	6	6	174	6	13	6	14270
33	6	4	3	3	3	7	3	3	4	3	4	5	4	3	3	3	3	3	3	3	5	3	4	3	3	3	3	3	3	5	3	4	3	38	4	3	14525
34	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	1	0	1	0	0	0	0	0	0	0	2	0	310	0	14368
35	3	3	4	3	3	3	3	3	3	3	4	4	3	3	4	3	2	4	3	3	3	4	5	3	3	4	3	4	3	4	3	3	3	3	3	235	14338

Figure 5.24: Confusion Matrix of the validation data set with a threshold of 0.525. Each value is between 0 and 408 with the outsiders being the total amount of every value below the threshold for that given ground truth.

Confusion matrix can be seen in figure 5.24.

The confusion matrix shows that the main problems lie with a few classes. Firstly class 1 contains measurements which are classified equally as all of the classes with around 100% confidence each. This means the class causes a lot of False Acceptances. In addition, class 28 only has 3 out of 408 true positives, meaning that it causes a lot of False Rejections. This is an indication that the data set has some bad measurements. In addition, this threshold grants the following metrics:

- *Precision:* 0.4344
- *Recall:* 0.5000
- *Accuracy:* 0.9991
- *F1-Score:* 0.4649
- *FAR:* 0.0005
- *FRR:* 0.5000

The FAR is a lot lower than the required 0.01 and the FRR is a lot higher than the required 0.05. However, if multiple attempts were to be done, these values could

become closer to each other through cumulative probabilities where:

$$FalseAcceptanceProbability = 1 - (1 - FAR)^{Attempts}$$

$$FalseRejectionProbability = FRR^{Attempts}$$

Here the False Acceptance Probability (FAP) is the probability that non of the attempts will have any false acceptances, and the False Rejection Probability(FRP) is the probability that all of the attempts will be false rejections. These can be seen calculated and plotted in figure 5.25.

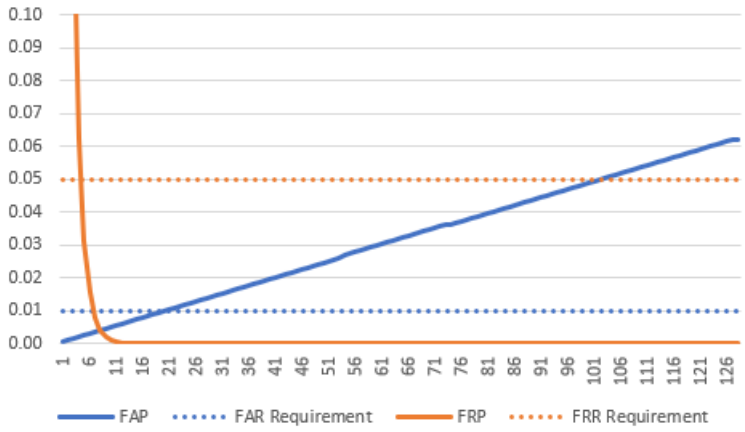


Figure 5.25: Plot showing the difference in False Acceptance Probability and False Rejection Probability depending on the number of attempts. The x-axis is the attempt amounts and the y-axis is the probability.

The figure shows that as long as between 5 and 19 attempts are tried, both the estimated FAR and FRR are between the required values.

Chapter 6

Testing

6.1 Test 1 - Same fitting test

For this test, the *in_fitting_data* is input into the testing loop to test the performance of the system. The loop outputs a CSV file with the confidence of each of the 36 classes for each of the 7488 measurements. For this test, each class was tested 208 times each.

6.1.1 Confusion Matrix

The results is converted into the multi class confusion matrix seen in figure 6.1.

		Prediction																																	
	GAR	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	Outsider
Ground Truth	0	108	0	0	0	0	0	0	0	2	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7373
	1	56	55	55	54	54	54	54	54	54	54	60	54	54	56	54	54	54	54	54	54	55	54	55	54	54	54	54	54	54	54	54	55	54	55	54	55	5527
	2	2	3	10	1	0	0	0	0	0	1	3	1	1	1	2	0	0	0	0	1	1	1	0	0	1	0	0	1	0	1	0	0	0	0	0	2	7455
	3	1	1	1	167	1	1	1	1	5	1	1	1	1	2	1	3	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	7278
	4	0	0	0	0	155	0	51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7282
	5	1	1	2	1	1	141	2	1	1	2	1	1	2	3	1	2	1	1	3	3	2	2	2	1	2	1	1	2	1	2	2	2	1	1	1	1	7293
	6	13	13	13	12	13	15	192	13	14	13	13	13	13	14	13	13	13	13	13	14	13	13	13	13	14	13	13	14	13	13	13	13	13	13	13	14	6828
	7	0	0	0	0	1	0	4	203	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7280
	8	2	2	2	23	2	2	2	2	52	2	2	2	2	2	2	3	2	2	2	2	2	2	2	2	2	3	2	2	2	2	2	2	2	2	3	2	7342
	9	10	9	9	9	9	9	9	9	9	41	9	9	9	10	9	10	9	10	9	9	9	11	9	9	9	9	9	9	9	11	10	9	9	9	9	9	7123
	10	1	0	0	0	0	0	0	0	0	0	117	0	3	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	7363
	11	2	0	0	0	0	0	0	0	0	0	1	21	0	3	0	0	0	0	1	0	0	0	3	0	0	0	0	0	0	0	0	0	1	1	3	7452
	12	8	8	8	8	8	11	8	8	8	8	8	8	53	8	9	8	8	8	9	8	8	8	8	8	8	9	8	8	8	8	8	8	9	8	8	8	7148
	13	0	0	0	0	0	0	0	0	1	0	1	1	0	27	0	0	0	0	0	1	0	0	4	0	0	1	0	1	1	0	1	0	0	1	0	0	7448
	14	3	4	4	3	3	5	4	3	3	4	6	3	6	4	108	3	3	3	3	6	3	3	4	3	4	6	3	4	3	5	3	3	4	3	3	3	7250
	15	0	0	0	0	0	0	0	0	0	0	1	0	0	2	1	11	0	0	1	0	1	0	2	0	0	0	0	0	2	0	0	0	0	0	0	0	7467
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	205	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7283
	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	168	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	7319
	18	2	2	2	2	2	2	2	2	2	2	3	2	2	2	2	2	2	2	145	2	2	2	3	2	2	3	2	2	2	2	2	2	2	2	2	2	7270
	19	7	7	7	7	7	8	7	7	7	7	7	7	7	7	9	7	7	7	7	139	7	8	7	8	12	7	7	7	7	7	7	9	9	7	7	7	7090
	20	0	1	0	0	0	1	0	0	1	0	2	1	1	0	0	0	0	0	0	1	0	47	0	0	0	0	0	0	0	0	0	0	1	0	1	1	7430
	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	2	0	0	0	0	0	0	0	0	0	0	0	0	7366
	22	1	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	113	1	1	1	1	0	0	1	0	0	0	1	1	0	7359
	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	197	0	2	0	0	0	0	0	0	0	0	0	7286
	24	1	1	1	1	1	1	1	1	1	2	1	1	3	1	1	1	1	1	1	9	1	1	1	1	39	1	3	1	1	1	1	2	4	1	1	1	7397
	25	8	8	8	9	8	9	8	8	10	8	8	8	8	8	11	10	8	8	8	8	8	8	10	8	8	40	8	8	8	8	8	8	8	9	10	9	7153
	26	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	199	0	0	0	0	0	0	0	0	0	7286
	27	9	9	9	9	9	10	9	9	9	10	10	10	9	9	9	9	9	9	10	9	9	9	9	9	9	9	9	57	9	9	9	9	9	9	9	9	7111
	28	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	7479
	29	2	1	1	1	1	1	1	1	1	1	2	1	1	1	1	2	1	1	2	1	1	1	1	1	1	1	1	1	1	166	1	1	1	1	1	1	7283
	30	2	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	150	1	1	3	1	2	7297
	31	1	1	1	1	1	2	1	1	2	2	1	1	1	1	1	1	1	1	1	4	1	1	2	1	1	1	1	3	2	1	1	7	1	4	1	2	7432
	32	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	11	8	8	9	9	11	9	8	8	8	8	8	8	70	8	11	8	7126
	33	1	1	1	1	1	3	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	21	3	3	7423
	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	143	0	7341
	35	2	2	2	3	2	2	2	2	2	3	2	2	2	3	2	2	2	2	2	2	2	3	4	2	2	2	2	2	2	2	2	2	2	2	3	110	7301

Figure 6.1: Confusion Matrix of the *In_Fitting* data set with a threshold of 0.525. Each value is between 0 and 208 with the outsiders being the total amount of every value below the threshold for that given ground truth.

Ideally, the values in the green main diagonal in the matrix would be 208 and the outsiders 7280. The rest should be 0.

6.1.2 Performance Metrics

The confusion matrix is then converted to the metrics from the requirements which can be seen in table 6.1.

For this, the lowest values are:

- Lowest single-class precision is class 28 with 0

Class	*Precision*	*Recall*	*Accuracy*	*F1-Score*	*FAR*	*FRR*
0	0.4303	0.5192	0.9991	0.4706	0.0005	0.4808
1	0.3957	0.2644	0.9991	0.3170	0.0003	0.7356
2	0.0685	0.0481	0.9988	0.0565	0.0005	0.9519
3	0.5186	0.8029	0.9993	0.6302	0.0006	0.1971
4	0.5272	0.7452	0.9993	0.6175	0.0005	0.2548
5	0.4896	0.6779	0.9992	0.5685	0.0005	0.3221
6	0.5217	0.9231	0.9993	0.6667	0.0007	0.0769
7	0.6060	0.9760	0.9995	0.7477	0.0005	0.0240
8	0.2667	0.2500	0.9989	0.2581	0.0005	0.7500
9	0.2398	0.1971	0.9989	0.2164	0.0005	0.8029
10	0.4349	0.5625	0.9991	0.4906	0.0006	0.4375
11	0.1313	0.1010	0.9988	0.1141	0.0005	0.8990
12	0.2775	0.2548	0.9989	0.2657	0.0005	0.7452
13	0.1534	0.1298	0.9988	0.1406	0.0006	0.8702
14	0.4372	0.5192	0.9991	0.4747	0.0005	0.4808
15	0.0728	0.0529	0.9987	0.0613	0.0005	0.9471
16	0.6065	0.9856	0.9995	0.7509	0.0005	0.0144
17	0.5508	0.8077	0.9993	0.6550	0.0005	0.1923
18	0.5142	0.6971	0.9993	0.5918	0.0005	0.3029
19	0.4793	0.6683	0.9992	0.5582	0.0006	0.3317
20	0.2541	0.2260	0.9989	0.2392	0.0005	0.7740
21	0.4598	0.5769	0.9992	0.5117	0.0005	0.4231
22	0.4124	0.5433	0.9991	0.4689	0.0006	0.4567
23	0.5934	0.9471	0.9995	0.7296	0.0005	0.0529
24	0.2131	0.1875	0.9988	0.1995	0.0005	0.8125
25	0.2273	0.1923	0.9989	0.2083	0.0005	0.8077
26	0.5940	0.9567	0.9995	0.7330	0.0005	0.0433
27	0.3032	0.2740	0.9990	0.2879	0.0005	0.7260
28	0.0000	0.0000	0.9987	Div/0	0.0005	1.0000
29	0.5461	0.7981	0.9993	0.6484	0.0005	0.2019
30	0.5282	0.7212	0.9993	0.6098	0.0005	0.2788
31	0.0490	0.0337	0.9987	0.0399	0.0005	0.9663
32	0.3415	0.3365	0.9990	0.3390	0.0005	0.6635
33	0.1296	0.1010	0.9988	0.1135	0.0005	0.8990
34	0.4983	0.6875	0.9992	0.5778	0.0005	0.3125
35	0.4331	0.5288	0.9991	0.4762	0.0005	0.4712
Average	**0.3696**	**0.4804**	**0.9991**	**0.4238**	**0.0005**	**0.5196**

Table 6.1: Performance metrics from in-fitting test.

- Lowest single-class Recall is class 28 with 0
- Lowest single-class Accuracy is class 28 with 0.9987
- Lowest F1-Score is class 1 with 0.0399

6.1.3 Cumulative Probability Modification

Using the same method as for validation, the FAR and FRR can be converted to False Acceptance Probability and False Rejection Probability by calculating the cumulative probabilities depending on the attempts.

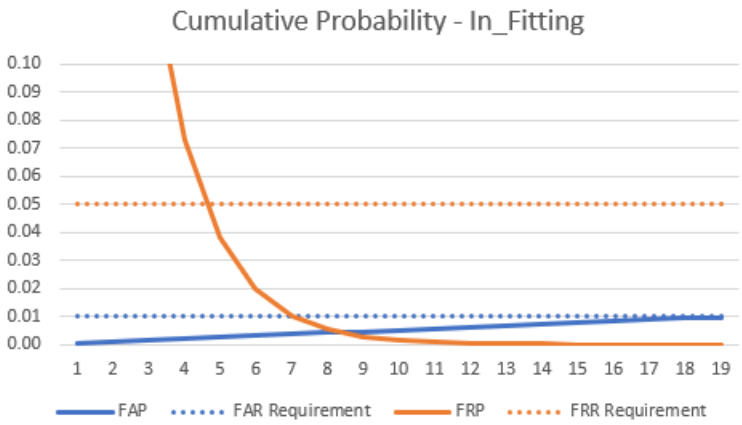


Figure 6.2: Plot showing the difference in False Acceptance Probability and False Rejection Probability depending on the number of attempts for the in fitting test. The x-axis is the attempt amounts and the y-axis is the probability.

The graph shows the False Rejection Probability becomes lower than the FRR requirement after 5 attempts without the False Acceptance Probability rising above the FAR requirement until after 19 attempts.

6.2 Test 2 - Other fitting test

To test whether the model is biased towards measurements from the same fitting, a test using measurements from fittings that the model was not trained on, but still from the same individuals is conducted. Only a few of the individuals had more than the four fittings the model were trained on, so the distribution of ground truth classes is a lot more unbalanced. The distribution of measurements are:

- Class 0: 8448 measurements
- Class 1: 1721 measurements

- Class 2: 19456 measurements
- Class 3: 11520 measurements
- Class 4: 2048 measurements
- Class 6: 1024 measurements
- Class 7: 1024 measurements
- Class 8: 512 measurements
- Class 9: 2048 measurements
- Class 10: 2048 measurements
- Class 11: 2560 measurements
- Class 13: 5632 measurements
- Class 15: 4096 measurements
- Class 16: 2048 measurements
- Class 17: 2048 measurements

The rest of the classes has no measurements.

6.2.1 Confusion Matrix

The results of the other-fitting test is then input into the confusion matrix in figure 6.3.

Ground Truth \ Prediction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	Outsider
0	484	65	68	74	61	83	68	62	72	67	70	70	70	80	68	68	64	68	70	68	74	67	75	67	68	69	62	69	66	67	67	65	67	70	67	68	301240
1	20	22	20	18	17	23	20	19	22	20	21	21	23	21	20	23	19	19	20	19	19	18	21	20	20	21	20	20	21	19	20	20	20	20	21	20	61229
2	552	578	618	474	529	567	546	527	557	553	556	545	573	581	584	564	524	555	558	566	561	542	556	551	548	656	541	572	546	549	548	555	562	546	542	557	680377
3	16	16	16	2215	22	16	16	16	29	16	16	16	17	23	18	16	14	16	18	16	16	21	23	16	16	17	16	17	17	16	16	16	16	17	16	16	411900
4	8	9	9	45	256	8	8	8	43	8	8	8	9	9	9	10	8	9	8	9	8	8	8	8	10	9	8	9	8	9	8	8	8	8	8	8	73106
6	4	3	3	29	7	3	3	3	17	3	3	3	3	10	4	4	3	3	4	3	3	3	3	3	3	6	4	4	3	3	3	3	3	3	4	3	36695
7	13	14	13	20	13	13	13	13	23	13	13	13	15	25	13	14	13	14	14	13	13	13	13	13	13	15	14	13	13	13	13	13	13	14	13	14	36356
8	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4	2	0	0	0	2	0	0	0	0	0	0	0	0	1	0	18421
9	13	10	11	12	10	11	10	10	11	18	12	11	15	11	12	13	10	10	11	11	11	10	10	10	13	12	10	14	10	10	10	10	10	10	10	10	73326
10	12	10	11	11	9	13	11	10	19	11	24	11	13	13	14	12	10	9	14	11	12	11	12	11	11	12	11	12	11	12	12	11	11	11	11	11	73298
11	5	2	2	2	1	3	1	1	2	3	4	19	4	6	6	2	1	3	1	2	5	2	6	1	1	5	1	1	1	4	2	2	2	1	4	4	92048
13	115	125	113	82	112	115	112	111	112	116	117	116	116	138	118	117	111	116	114	116	115	123	124	112	115	119	113	115	125	115	117	114	114	128	118	115	198578
15	241	247	239	163	225	233	231	225	237	237	232	232	241	236	228	255	215	228	241	239	232	228	245	238	232	245	235	225	234	224	234	241	243	224	222	243	139086
16	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	145	0	0	12	0	1	0	2	3	0	3	0	0	3	0	0	0	0	3	0	73554
17	4	3	3	4	3	6	3	3	4	3	4	5	3	3	4	3	3	3	89	3	4	4	14	3	3	4	3	4	3	3	3	3	3	4	3	4	73507

Figure 6.3: Confusion Matrix of the *Out_of_Fitting* data set with a threshold of 0.525. Each value varies with the outsiders being the total amount of every value below the threshold for that given ground truth.

Ideally all values, on this table should be 0, however it still functional, so ling each row is somewhat evenly distributed.

6.2.2 Performance Metrics

Class	*Precision*	*Recall*	*Accuracy*	*F1-Score*	*FAR*	*FRR*
0	0.3253	0.0573	0.9962	0.0974	0.0004	0.9427
1	0.0199	0.0128	0.9988	0.0156	0.0005	0.9872
2	0.5488	0.0318	0.9919	0.0601	0.0002	0.9682
3	0.7034	0.1923	0.9957	0.3020	0.0004	0.8077
4	0.2024	0.1250	0.9988	0.1545	0.0004	0.8750
6	0.0029	0.0029	0.9991	0.0029	0.0004	0.9971
7	0.0129	0.0127	0.9992	0.0128	0.0004	0.9873
8	0.0009	0.0020	0.9993	0.0012	0.0005	0.9980
9	0.0169	0.0088	0.9987	0.0116	0.0004	0.9912
10	0.0222	0.0117	0.9987	0.0153	0.0004	0.9883
11	0.0178	0.0074	0.9985	0.0105	0.0004	0.9926
13	0.1194	0.0245	0.9973	0.0407	0.0004	0.9755
15	0.2316	0.0623	0.9980	0.0981	0.0004	0.9377
16	0.1272	0.0708	0.9988	0.0910	0.0004	0.9292
17	0.0028	0.0015	0.9987	0.0019	0.0004	0.9985
Total	**0.2218**	**0.0636**	**0.9979**	**0.0989**	**0.0004**	**0.9364**

For this, the lowest values are:

- Lowest single-class precision is class 8 with 0.0009
- Lowest single-class Recall is class 17 with 0.0015
- Lowest single-class Accuracy is class 2 with 0.9919
- Lowest F1-Score is class 8 with 0.0012

While accurracy looks really good, the F1 score reveals some severe performance issues to take note of.

6.2.3 Cumulative Probability Modification

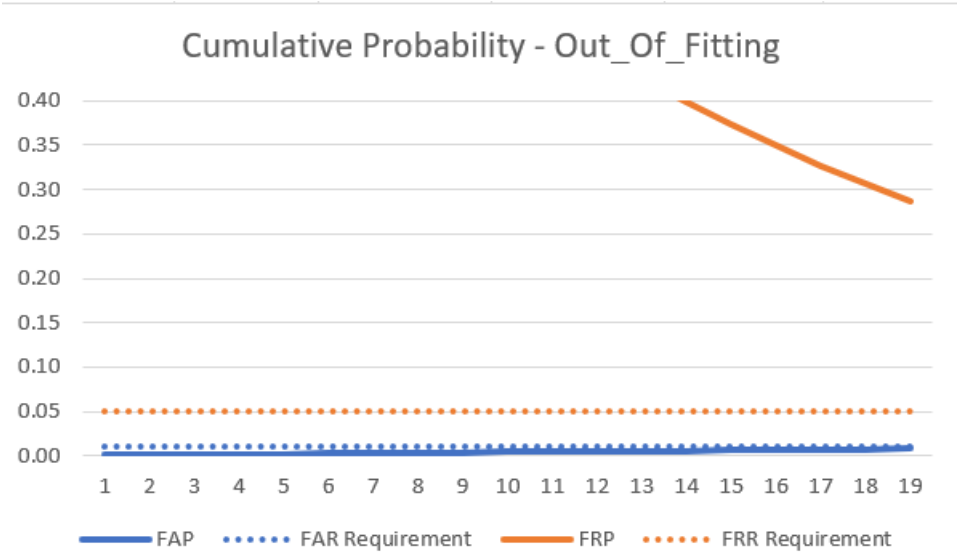


Figure 6.4: Plot showing the difference in False Acceptance Probability and False Rejection Probability depending on the number of attempts for the other fitting test. The x-axis is the attempt amounts and the y-axis is the probability.

The graph shows promise, as the FRP is very far from it's threshold. The FAP is creeping up on it's own threshold, making it hard to push it past 20 attempts

6.3 Test 3 - Outsider test

6.3.1 Confusion Matrix

		Prediction																																		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	Outsider
Ground Truth	36	28	27	27	27	27	52	27	27	29	28	31	27	31	27	37	29	29	27	28	202	29	28	30	38	50	32	27	29	28	34	27	30	34	39	31	31	72444
	37	164	147	148	158	143	189	153	145	151	148	159	156	172	165	160	153	151	154	160	201	151	151	163	153	169	159	147	155	151	162	153	153	160	163	165	162	141762
	38	0	0	0	78	1150	0	37	5	8	0	0	0	0	0	0	0	0	0	0	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	44798
	97	11	3	0	1	0	2	0	0	2	2	2	11	1	56	0	1	0	1	0	0	4	6	23	0	0	2	0	1	8	1	0	0	0	6	2	7	73575
	98	3	1	0	0	0	2	0	0	0	2	1	15	0	29	1	0	0	0	0	1	2	18	26	0	0	1	0	1	8	2	1	5	0	19	2	8	73580

Figure 6.5: Confusion Matrix of the *Outsider_Fitting* data set with a threshold of 0.525. Each value varies with the outsiders being the total amount of every value below the threshold for that given ground truth, User 36, 37 and 38 are humans, where User 97 and 98 are a syringe and a mannequin.

It is curious how human 36 is clearly predicted to be 4. It is reassuring that both the mannequin and syringe are clearly identified to not be part of the data set.

6.3.2 Performance Metrics

Class	*Accuracy*	*FAR*
36	0.9826	0.0174
37	0.9614	0.0386
38	0.9722	0.0278
Total	**0.9691**	**0.0309**

Class	*Accuracy*	*FAR*
97	0.9979	0.0021
98	0.9980	0.0020
Total	**0.9980**	**0.0020**

While these numbers look mighty fine, the FAR is way too high for both humans and objects. Ideally, they should be in the realm of 1/100th of a percent.

6.3.3 Cumulative Probability Modification

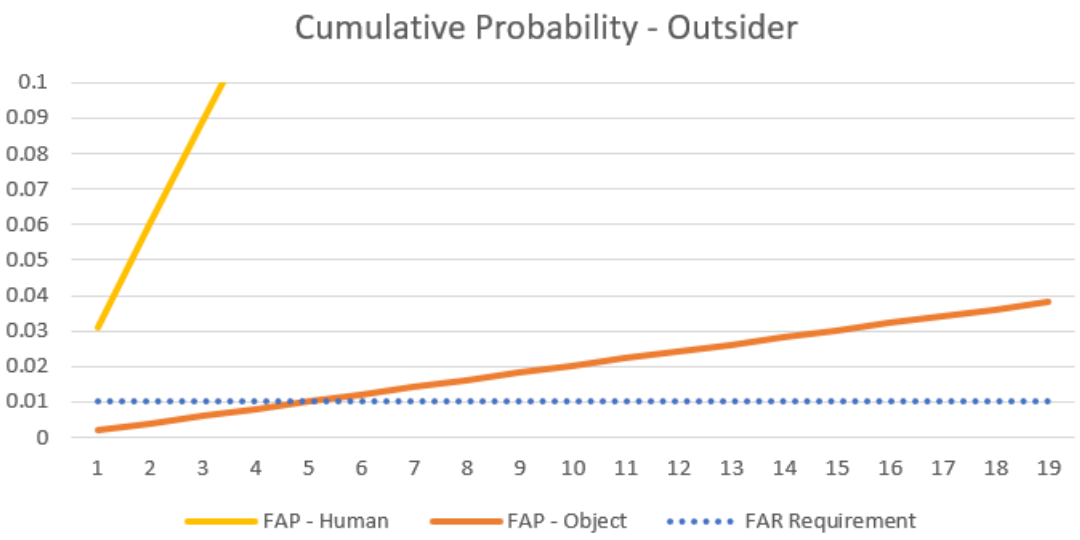


Figure 6.6: Plot showing the difference in False Acceptance Probability depending on the number of attempts for the outsider fitting tests. The x-axis is the attempt amounts and the y-axis is the probability.

As we can see, the graph shows how the FAP for humans, already far excedes the requirement, and the FAP for objects very quickly starts to do so as well.

6.4 Test 4 - Time and Memory measurement

For the last test, each measurement was run through the code and the time was measured. The slowest measurement took 0.1 seconds to be tested.

In addition, the memory required for each line of the test code was measured with a line profiler and had a peak of 1.688 GiB, where the model itself accounts for 1.389 GiB.

Chapter 7

Results and Discussion

7.1 Requirement overview

This project was made in an attempt to succeed at 8 different requirements from section 3.2. This section looks at each one to determine if it succeeded and why. For the requirements, only test 1 and test 4 is taken into account, since the rest are very unbalanced and unsystematic. As such they might have a very different margin of error. Instead, tests 2 and 3 be judged individually.

Single-Class Precision

The Single-Class Precision was a requirement that no class was allowed to have a precision of less than 0.75. The test contained classes with results less than a precision of 0.75, even going as low as 0.00, as some classes did not have any True Positives at all, like Class 28. However, even if the outlier classes were removed it is unlikely that the requirement would be met.

Single-Class Recall

Like the Single-Class Precision, the Single-Class Recall was a requirement that no class was allowed to have a Recall less than 0.5. The test contained classes with results less than a recall of 0.5, even going as low as 0.00, as some classes did not have any True Positives at all, like Class 28. However, unlike the precision, the high average recall for the in-fitting test, of 0.48, suggests that if the outliers and noisy files were removed or ignored, the requirement would have been met.

Single-Class Accuracy

The Single-Class Accuracy was a requirement that no class was allowed to have an Accuracy of less than 0.65. The test did not contain any class with an Accuracy below 0.998 which is incredibly high. However, that is due to the fact that accuracy works poorly when the Positive:Negative ratio is unbalanced, and since there are 36 times as many negatives as positives, this makes the accuracy unreliable.

Single-Class F1-Score

The Single-Class F1-score was a requirement that no class was allowed to have an F1-score less than 0.65. The test contained classes with results less than a recall of 0.65, even going as low as an error due to dividing by 0, as some classes did not have any True Positives at all, like Class 28. Only 7 of the 36 classes exceeded an F1-score of 0.65.

False Acceptance Rate

The False Acceptance Rate was a requirement that the system should not have a false acceptance rate of 0.01, since most use cases of biometrics require a low FAR. Since the system has a false acceptance rate of about 0.0005 this is more than enough. In addition, No single class exceeds 0.001 either. This metric could be higher by doing multiple attempts that would lower the False Rejection Rate.

False Rejection Rate

The False Rejection Rate shows how often an authorised user can expect to be rejected. The requirement was that the False Rejection Rate should be below 0.05. However, the system has an FRR of 0.52 instead. So then times as much. However, it is possible with between 5 and 19 attempts to have the probability of being a False Rejection, as well as the probability of Having a False Acceptance be below the requirements.

Identification Time

The identification time can only be estimated since it would be made up of 3 parts: The measuring, the processing and the validation. The measuring takes 20 milliseconds to create and measure the sounds of TEOAE. The validation takes less than 100 milliseconds which leaves 1.88 seconds for the processing, which is unlikely. So the Identification Time requirement of less than 2 seconds is highly likely.

Memory Use

Every iteration of the test code used about 1.688 GiB, at its peaks, which is 69% higher than the requirement, however, the model itself only used about 1.389 GiB. However, it is still slightly more memory intensive than required. However, it would be possible to lower the memory usage by lowering the precision of the floating points from double precision (float64) to single precision (float32) or lower.

7.1.1 Requirement Table

The requirements from section 3.2 consisted of the values seen in table 7.1, these are then judged based on the previous comments on whether the requirement is met or not.

Functional Requirements		**Success?:**
Single-Class Precision:	>0.75	No
Single-Class Recall:	>0.5	No
Single-Class Accuracy:	>0.65	Yes
Single-Class F1-score:	>0.65	No
False Acceptance Rate:	<0.01	Yes
False Rejection Rate:	<0.05	No
Time spent for 95% of the identifications:	<2 sek	Yes
Memory used for 95% of the identifications:	<1 GiB	No

Table 7.1: Requirement list for the solution

5 out of 8 of the requirements are not met. However, the False Reject Rate would be within the requirements if we did 4 more attempts, which is acceptable since the time constraint still allows for up to 16 tests before it becomes nearly 2 seconds.

The memory requirement would be possible to reach by lowering the floating point precision from double precision to anything lower.

The Precision and Recall would require being more selective in terms of the data used and how much correlation the individuals' measurements have, and the F1-score would improve as a result as well if they do.

7.2 Fitting Bias and Outsider detection

The results of test 2 show a very high False Rejection Rate. This could be due to the fact that each class is only trained on 4 different fittings, so they become biased towards those. This could be solved by expanding the data set to include more fittings per individual. An ideal amount of fittings would be 20 since that would allow for the 70/20/10 split to be done without any overlap of the fittings, for each ear.

The result of the outsider test showed a low False Acceptance Rate of the objects, of around 0.002 however for the human outsiders, the False Acceptance rate is higher than the requirement, but class 38 is also clearly predicted as Class 4, which might be because an individual has been measured both i the original data-set and update, or just a mislabeling of the data.

7.3 Additional comments

One thing to note about the results is also that the training and testing is done with a single 20-millisecond measurement per test. This is the absolute minimum

for measuring the TEOAEs and means the test and training data is very noisy. The main reasoning behind this attempt is to test the possibilities of using the extremum of the data. For future attempts, some mean filtering would be ideal to test how much that influences both the bias and the precision of the classification.

Chapter 8

Conclusion

This report tried to solve the following problem statement:

Given a personal device meets the technical constraints measuring TEOAEs and the user has healthy ears, is it possible to create a TEOAE-based deep learning model to identify and authenticate a user for the device within a practical time frame and memory amount?

And while the requirement stated in section 3.2 was not all met. The balance between False Acceptance Rate and False Rejection Rate as well as the measurement time of the data, verifies most of the parts of the problem analysis. The solution does however require some optimization in order to also be able to use a practical memory amount. Ultimately, it proves the validity of OAEs for biometric applications

Bibliography

[1] Google, *Biometrics- google trends*, https://trends.google.com/trends/explore?date=all&q=%2Fm%2F01qjv4, 2010.

[2] M. Corporation, *Learn about windows hello and set it up*, https://support.microsoft.com/en-us/windows/learn-about-windows-hello-and-set-it-up-dae28983-8242-bb2a-d3d1-87c9d265a5f0, Accessed: May 24th, 2023.

[3] D. Goodin, "Here's how long it takes new bruteprint attack to unlock 10 different smartphones," 2023. [Online]. Available: https://arstechnica.com/information-technology/2023/05/hackers-can-brute-force-fingerprint-authentication-of-android-devices/.

[4] A. Makan, "Structures of the ear and hearing," 2019. [Online]. Available: https://www.theartofhearing.co.uk/post/structure-of-the-ear.

[5] B. C. J. Moore, *An introduction of the Psychology of Hearing*, 6. ed. Brill Academic Publishers, 2013.

[6] J. E. Hawkins, "Human ear," 2023. [Online]. Available: https://www.britannica.com/science/ear.

[7] "Stimulated acoustic emissions from within the human auditory system," *The Journal of the Acoustical Society of America*, vol. 64, no. 5, pp. 1386–1391, 1978.

[8] M. J. Penner, "An estimate of the prevalence of tinnitus caused by spontaneous otoacoustic emissions," *Archives of Otolaryngology–Head Neck Surgery*, vol. 116, no. 4, pp. 418–423, Apr. 1990, ISSN: 0886-4470. DOI: 10.1001/archotol.1990.01870040040010. eprint: https://jamanetwork.com/journals/jamaotolaryngology/articlepdf/618619/archotol_116_4_010.pdf. [Online]. Available: https://doi.org/10.1001/archotol.1990.01870040040010.

[9] R. Probst, B. L. Lonsbury-Martin, and G. K. Martin, "A review of otoacoustic emissions," *The Journal of the Acoustical Society of America*, vol. 89, no. 5, pp. 2027–2067, May 1991, ISSN: 0001-4966. DOI: 10.1121/1.400897. eprint: https://pubs.aip.org/asa/jasa/article-pdf/89/5/2027/9431715/2027_1_online.pdf. [Online]. Available: https://doi.org/10.1121/1.400897.

[10] "Dedication," in *Office Practice of Neurology (Second Edition)*, M. A. Samuels and S. K. Feske, Eds., Second Edition, Philadelphia: Churchill Livingstone, 2003, p. v, ISBN: 978-0-443-06557-6. DOI: https://doi.org/10.1016/B978-044306557-6.50002-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780443065576500027.

[11] M. J. Penner, "Audible and Annoying Spontaneous Otoacoustic Emissions: A Case Study," *Archives of Otolaryngology–Head Neck Surgery*, vol. 114, no. 2, pp. 150–153, Feb. 1988, ISSN: 0886-4470. DOI: 10.1001/archotol.1988.01860140048019. eprint: https://jamanetwork.com/journals/jamaotolaryngology/articlepdf/615004/archotol_114_2_019.pdf. [Online]. Available: https://doi.org/10.1001/archotol.1988.01860140048019.

[12] M. Sliwinska-kowalska, "Chapter 19 - hearing," in *Occupational Neurology*, ser. Handbook of Clinical Neurology, M. Lotti and M. L. Bleecker, Eds., vol. 131, Elsevier, 2015, pp. 341–363. DOI: https://doi.org/10.1016/B978-0-444-62627-1.00018-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444626271000184.

[13] R. Ordoñez, D. Hammershøi, C. Borg, and J. Voetmann, "A pilot study of changes in otoacoustic emissions after exposure to live music," Cited by: 3, 2012, 11 – 19. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84883316973&partnerID=40&md5=f89654fc33322d8cef7f5ee9823768f4.

[14] M. A. Ueberfuhr, H. Fehlberg, S. S. Goodman, and R. H. Withnell, "A DPOAE assessment of outer hair cell integrity in ears with age-related hearing loss," *Hear Res*, vol. 332, pp. 137–150, 2016.

[15] D. H. Keefe, M. P. Feeney, L. L. Hunter, and D. F. Fitzpatrick, "Comparisons of transient evoked otoacoustic emissions using chirp and click stimuli," *J Acoust Soc Am*, vol. 140, no. 3, p. 1949, 2016.

[16] A. Anwar, 2021. [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-regression-analysis-in-machine-learning-8a828b491bbf.

[17] prakharr0y, *Probability density estimation amp; maximum likelihood estimation*, 2021. [Online]. Available: https://www.geeksforgeeks.org/probability-density-estimation-maximum-likelihood-estimation/.

[18] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html.

[19] P. Team, *Torchmetrics documentation: Accuracy metric*, 2021. [Online]. Available: https://torchmetrics.readthedocs.io/en/stable/classification/accuracy.html.

[20] IBM, *Convolutional neural networks*, https://www.ibm.com/topics/convolutional-neural-networks, Accessed on May 24, 2023, 2023.

[21] E. Fathi and B. Maleki Shoja, "Chapter 9 - deep neural networks for natural language processing," in *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, ser. Handbook of Statistics, V. N. Gudivada and C. Rao, Eds., vol. 38, Elsevier, 2018, ch. 4. DOI: https://doi.org/10.1016/bs.host.2018.07.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016971611830021X.

[22] Information Technology Laboratory - NIST, *Computer security resource center - biometrics glossary*, definition of Biometrics. [Online]. Available: https://csrc.nist.gov/glossary/term/biometrics.

[23] S. Boonkrong, "Biometric authentication," in *Authentication and Access Control: Practical Cryptography Methods and Tools*. Berkeley, CA: Apress, 2021, pp. 107–132, ISBN: 978-1-4842-6570-3. DOI: 10.1007/978-1-4842-6570-3_5. [Online]. Available: https://doi.org/10.1007/978-1-4842-6570-3_5.

[24] C. Yuan, Q. Cui, X. Sun, Q. J. Wu, and S. Wu, "Chapter five - fingerprint liveness detection using an improved cnn with the spatial pyramid pooling structure," in *AI and Cloud Computing*, ser. Advances in Computers, A. R. Hurson and S. Wu, Eds., vol. 120, Elsevier, 2021, pp. 157–193. DOI: https://doi.org/10.1016/bs.adcom.2020.10.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245820300863.

[25] "An effective method of fingerprint classification combined with afis," eng, in *EMBEDDED AND UBIQUITOUS COMPUTING - EUC 2005*, ser. Lecture Notes in Computer Science, vol. 3824, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1107–1122, ISBN: 9783540308072.

[26] D. Maltoni, "Fingerprint recognition, overview," eng, in *Encyclopedia of Biometrics*, Boston, MA: Springer US, 2015, pp. 664–668, ISBN: 9781489974877.

[27] F. Liu, *Advanced Fingerprint Recognition: From 3D Shape to Ridge Detail*, eng, 1st ed. 2020. Singapore: Springer Singapore, 2020, ISBN: 981-15-4128-0.

[28] S. C. Dass, "Individuality of fingerprints, a review," eng, in *Encyclopedia of Biometrics*, Boston, MA: Springer US, 2015, pp. 924–940, ISBN: 9781489974877.

[29] A. Kholodenko, S. Huckemann, and H. Munk, *Uses of quadratic differentials in fingerprints recognition*, Jun. 2013.

[30] N. J. Grabham, M. A. Swabey, P. Chambers, *et al.*, "An evaluation of otoacoustic emissions as a biometric," eng, *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 174–183, 2013, ISSN: 1556-6013.

[31] M. A. Swabey, S. P. Beeby, A. D. Brown, and J. E. Chad, "Using otoacoustic emissions as a biometric," in *Biometric Authentication*, D. Zhang and A. K. Jain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 600–606, ISBN: 978-3-540-25948-0.

[32] N. J. Grabham, M. A. Swabey, P. Chambers, *et al.*, "An evaluation of otoacoustic emissions as a biometric," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 174–183, 2013. DOI: 10.1109/TIFS.2012.2228854.

[33] L. Wang, G. Sun, Y. Wang, J. Ma, X. Zhao, and R. Liang, "Afexplorer: Visual analysis and interactive selection of audio features," *Visual Informatics*, vol. 6, no. 1, pp. 47–55, 2022, ISSN: 2468-502X. DOI: https://doi.org/10.1016/j.visinf.2022.02.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468502X22000110.

[34] T. Giannakopoulos and A. Pikrakis, "Chapter 4 - audio features," in *Introduction to Audio Analysis*, T. Giannakopoulos and A. Pikrakis, Eds., Oxford: Academic Press, 2014, pp. 59–103, ISBN: 978-0-08-099388-1. DOI: https://doi.org/10.1016/B978-0-08-099388-1.00004-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080993881000042.

[35] T. M. Inc., "Continuous wavelet transform and scale-based analysis," 2023. [Online]. Available: https://se.mathworks.com/help/wavelet/gs/continuous-wavelet-transform-and-scale-based-analysis.html.

[36] T. M. Inc., "Practical introduction to time-frequency analysis using the continuous wavelet transform," 2023. [Online]. Available: https://se.mathworks.com/help/wavelet/ug/practical-introduction-to-time-frequency-analysis-using-the-continuous-wavelet-transform.html.

[37] T. M. Inc., "Wavelet synchrosqueezing," 2023. [Online]. Available: https://se.mathworks.com/help/wavelet/ug/practical-introduction-to-time-frequency-analysis-using-the-continuous-wavelet-transform.html.

[38] T. M. Inc., "Spectrogram computation with signal processing toolbox," 2023. [Online]. Available: https://se.mathworks.com/help/signal/ug/spectrogram-computation-with-signal-processing-toolbox.html.

[39] Roland, *Quad-capture usb 2.0 interface*, Quad capture documentation. [Online]. Available: https://www.roland.com/us/products/quad-capture/features/.

[40] Etymōtic Research, *Er-10c lo noise dpoae probe system*, ER-10C documentation. [Online]. Available: https://manualzz.com/doc/28438828/er-10c-lo-noiseâĎć-dpoae-probe-system.

[41] Lenovo, "Thinkpad p53," 2023. [Online]. Available: https://www.lenovo.com/us/en/p/laptops/thinkpad/thinkpadp/p53/22ws2wpwp53.

[42] Nvidia Corporation, "Nvidia t4 flexible design, extraordinary performance," 2023. [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-t4/.

[43] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: https://www.wandb.com/.

[44] Brüel & Kjær Sound & Vibration, *Ear and mouth simulators*, https://www.bksv.com/en/transducers/simulators/ear-mouth-simulators/4157, Accessed: May 24th, 2023.

[45] Brüel & Kjær Sound & Vibration, *Sound calibrator type 4231*, https://www.bksv.com/en/transducers/acoustic/calibrators/sound-calibrator-4231, Accessed: May 24th, 2023.

[46] "Freesound," in S. X. Font F Roma G, Ed., 2013, ISBN: 978-0-443-06557-6. DOI: http://dx.doi.org/10.1145/2502081.2502245. [Online]. Available: http://hdl.handle.net/10230/45254.

[47] A. Paszke, S. Gross, F. Massa, *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: 1912.01703 [cs.LG].

[48] 2023. [Online]. Available: https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html.

[49] 2023. [Online]. Available: https://pytorch.org/docs/stable/data.html.

[50] [Online]. Available: https://pytorch.org/tutorials/beginner/introyt/trainingyt.html.

[51] J. Brownlee, "How to use learning curves to diagnose machine learning model performance," 2023. [Online]. Available: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.