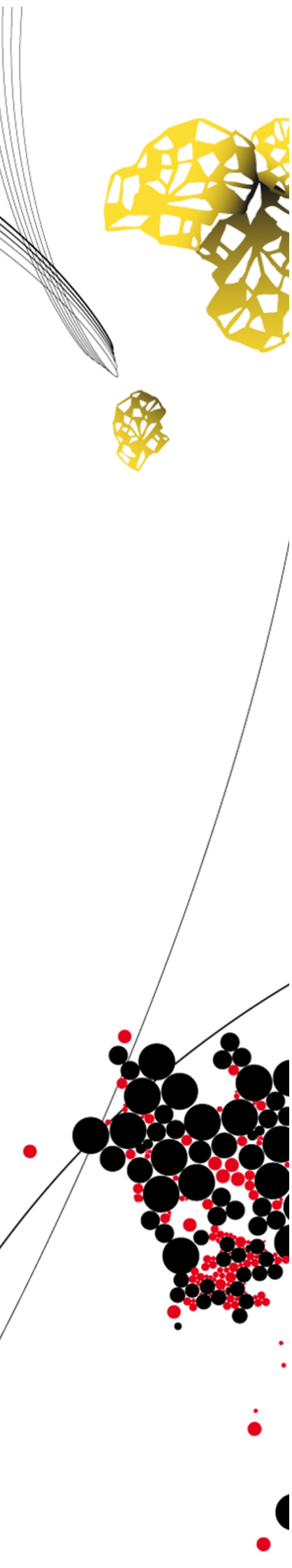

UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

An audio based feature detector for shavers using Artificial Intelligence

Wessel H. Nijhuis

M.Sc. Thesis
March 2022

Supervisors:
ir. E. Molenkamp
dr. ir. N. Alachiotis
dr. D.V. Le Viet Duc

Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Acknowledgements

The Thesis was made possible with the available internship position at Philips Drachten. Martin Sibum introduced me in a first interview to the available assignments at the company. Out of the assignments we ended up with the assignment written in this report, devised and created by Andries Bron. I'd like to thank him by the guidance in weekly meetings towards the goal of the assignment to create a contribution to the product.

Also, i'd like to thank dr. ir. Nikolaos Alachiotis by guide me through the Thesis in by-weekly meetings to correct and advised me on how to write the report and give me use full tips on how to create the system and especially on how to measure the system.

Finally, i would like to thank ir. E. Molenkamp and dr. D.V. Le Viet Duc for being part of the graduation committee and providing feedback on the final report.

Summary

Current techniques to explore unit recognition in personal care embedded devices have failed, have an accuracy which is too low or are too expensive in production cost (From personal communication with employees at Philips Drachten, 2021). Previous examples are unit recognition by measuring the motor current, using a NFC communication or an electrical pin connection with a specific slot. The next alternative which will be explored is to mount a microphone inside the body of the shaver and use the audio, combined with a machine learning algorithm, to detect which unit is connected.

The aim of the paper is to discover whether units can be recognized using sound and machine learning on personal care embedded devices, e.g. unit recognition. There are different ways to prepare raw audio for a machine learning network, each with its own type of machine learning network. In this paper there will be investigated which method is the most appropriate to prepare the audio for the machine learning network. By this method is also investigated which type of machine learning network is appropriate to detect the specific units. The validation of how well an unit is detected or labelled is expressed in terms of accuracy, precision and recall.

The machine learning method which is chosen is to train and predict a feed-forward artificial neural network with audio. The output layer consist out of number of neurons equal to the number of units. The input layer depends on the output of preparing the audio. The audio samples are normalised in multiple ways and the output is the input of the neural network.

The best result is achieved in combination with a Finite Impulse Response filter before the audio is filtered. The overall accuracy which is achieved is 91.1% for this combination. The precision values vary between 79.2% and 100%. The recall values vary between 81.3% and 97.5%. The combination consist out of the filter with a supervised feed-forward Artificial Neural Network. The results are measured for the situation when device with an unit has no load. The first results for the model while using the product looks promising but must be worked out further on.

Contents

List of acronyms

AI	Artificial Intelligence
ML	Machine Learning
NN	Neural Network
DL	Deep Learning
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
DBN	Deep Belief Network
MKL	Multiple Kernal Learning
SVM	Support-vector Machine
MFCC	Mel-frequency Cepstral Coefficient
PCA	Principal Component Analysis
LSTM	Long-Short Term Memory
MPW	Minimum Phase Whitening
autoML	automated Machine Learning
FFT	Fast Fourier transform
STFT	Short-time Fourier transform
NFC	Near Field Communication
FIR	Finite Impulse Response

Chapter 1

Introduction

1.1 Motivation

In this paper, the possibilities of detecting labelled features out of sound are discovered on personal care embedded devices. Labelled features can be unit recognition, beard density, the state (wear) of the units or a classic ”Ouch”. Units can be different kind of shaver-heads, trimmers or brushes. The aim of having this information about which unit is attached or having information about the density of the beard, is to adjust motor controller parameters. Every unit requires other parameters to control a shaver-head, trimmer or brush. Besides the motor controller parameters, the mobile application can be informed about the type and state of the units. The mobile application can give at his turn more specific information to the user and some tips to improve the use of the device. At the moment the user uses those tips, the device knows what is changed and can learn again if the tip was useful and evaluates again the state of the device. Also, the application can give feedback if an unit is worn out and it needs to be replaced. If an unit is significant worn out, the user isn't making nice sounds like ”Ouch” or the device fell on the floor, the controller parameters can be changed in such way that it prevents possible danger.

At the moment, there are some techniques to detect the units. The first technique is with Near Field Communication (NFC) but this is quit expensive in parts and production cost (From personal communication with employees at Philips Drachten, 2021). This is because both the unit attached to the device, which will wear out, and the device itself, needs to be implemented within the electronic architecture. Also, this technique is not appropriate to detect the other features.

Also, an implementation by measuring the motor current to identify units was not successful (From personal communication with employees at Philips Drachten, 2021). This implementation is made because the motor current was already been measured. But, the accuracy of these measurements between the different units is unreliable or satisfactory to conclude correct labels for the units.

Besides electrical solutions, a mechanical solution was also considered. By having different connectors, if a certain pen is attached in the specific slot, you know which unit is attached. But, this solution relative expensive and never reached the production state (From

personal communication with employees at Philips Drachten, 2021).

An alternative is defined by implementing only microphones into the device which can cooperate with the micro-controller. The production cost of the material against the NFC example is negligible. Especially, at the unit side where it is not essential and necessary to adjust the design with additional hardware and electronic components inside the units. To realize the functionality, Artificial Intelligence (AI) is going to be researched, especially the Machine Learning (ML) field of AI. This field is very upcoming the recent years because there is a lot more data available for certain subjects [1] [2]. Because of the abundant data which is available now, many more algorithms are used and tested for the specific subjects [1]. This improves the quality and accuracy of the existing algorithms. By improving and testing multiple subjects and different algorithms, much more information is gathered on how to implement such algorithm in a specific application. In the continuation of the paper, different methods of AI are discovered and researched to explore the method which is suitable for this application. The application will be defined as using a AI method to recognize the units on a specific embedded device. The results will be measured in terms of accuracy, recall and precision to define the robustness and meaningfulness of the approach with AI.

1.2 Goal and research questions

There are many techniques and methods in the field of ML. Every method is more suitable for a specific application. In chapter 2 are some methods described which will be researched further on to investigate which is suitable for the approach. There are many methods in the field of ML to detect features. The goal is to investigate which method is suitable to detect the different sound features in the device. Features can be unit recognition at a single speed, unit recognition at both speeds, unit recognition while using the product or the status of wear. The following main question can be asked:

- Which features can be recognized or detected with sound inside a shaver using ML approaches?

There are many methods and different kind of algorithms to implement such a recognizing system. There are methods specific for images, weather forecast or other predicting applications and also for audio examples. To answer the main question, the following question can be asked first:

- Which method of ML is suitable to detect features out of audio?

Before the audio data is fed into the ML method or algorithm, the data needs to be prepared and filtered so the method can make use of it. As there will be many ways to prepare the audio for a particular method, another question can be asked:

- How to make raw audio a suitable input for the ML method?

After detecting the features out of the audio, the features needs to be measured against the analytical, electrical or mechanical, methods which are available. To make it more specific, the last question which can be asked is:

- Which features can be detected by the method in terms of accuracy, precision and recall?

1.3 Report organization

The structure of the report is build up in the following way. Chapter 2 describes the background information about the techniques of the project. There are comparable solutions described in chapter 3. But, with a complete different setting as the proposed solutions. Chapter 4 describes the approach, setup, architecture and realization of the project. The results are described in chapter 5. Different situations are measured in this section. The complete project is concluded in chapter 6.

Chapter 2

Background

In this section is some background information provided to help interpreting the way of understanding on how AI is build up, especially the field of ML. There is some background information about the used libraries for the Neural Network (NN) and on how to create, train and predict a NN.

2.1 Artificial intelligence

To achieve a basic understanding of what artificial intelligence is and which part is interesting for the research, a basic overview of the field is made in fig. 2.1. A common misunderstanding is that most of the terms in the figure are used interchangeably. When people mean AI, they mean deep learning or the other way around. But, some of them are based on the other.

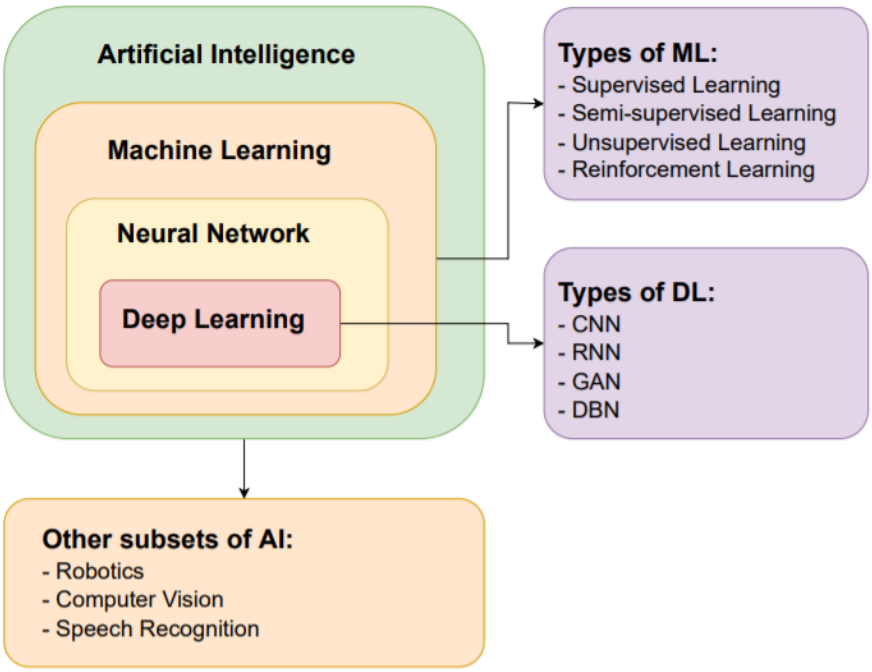

Figure 2.1: Overview of a subset of AI

Also, the fact that AI is only a self-learning machine, but there are other subsets of AI. A couple examples of subsets of AI is shown in the figure, namely, robotics, computer vision and speech recognition. These subjects are not specific self learning but make decisions based on the inputs. This brings us to a definition of AI, defined by McCarthy [3], as the science and engineering of making intelligent machines, where most of these machines contain a programmable algorithm to understand and mimic human, animal or other biological intelligence. Where artificial is the part to mimic the biological intelligence in to (programmable) machines and intelligence is the ability of computational resource of the machine to achieve the biological intelligence.

2.1.1 Machine Learning

The research is based on the sub-field ML of AI. There are different types in ML, namely, supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning. In supervised learning the is data already labelled. For example, recognize the cat and dogs in the picture. The machine is trained with given data to define what is a cat or a dog in the picture. In unsupervised learning, the data is not labelled and the machine is trying to find patterns on its own and defines the features. With this method, hidden features can be discovered which is hard for a human brain to discover, like malware detection. Semi-supervised learning is in between both of the types. There are some labelled features but also new features which can be discovered. But, there is a desired output where the discovered feature on must fulfill. Reinforcement learning is based on the way humans learn, by trying things and get response out of that with a positive or negative reward. If it makes a mistake it needs to be corrected, otherwise a positive response is needed if it works well.

2.1.2 Neural Network

A sub-set of ML is a NN, or as so-called an Artificial Neural Network (ANN). An example of a simple feed-forward NN is shown in fig. 2.2. A NN can be based on any type of ML, depending on the requirements a machine must execute. A NN consist out of a input layer, one or multiple hidden layers and an output layer. Each layer consist out of neurons, the colored circles. Each neuron has is own weights and biases, this can be negative or positive weight or bias depending on the connection between the neurons. After the calculation in the neuron, an activation function, for example a sigmoid function, is used to re-scale the answer between 0 and 1. For the sigmoid function, large numbers are set close to 1 and small numbers set close to 0. The equation for a neuron in the hidden layer of fig. 2.2 is

$$h = (I1 * w1 + bias) + (I2 * w2 + bias) \tag{2.1}$$

$$H1 = \frac{1}{1 + e^{-h}}, \tag{2.2}$$

where $H1$ is the output of the neuron, $I1$ and $I2$ are the outputs of neuron 1 and 2 of the input layer, $w1$ and $w2$ are, respectively, the corresponding weights. Also, at both terms is

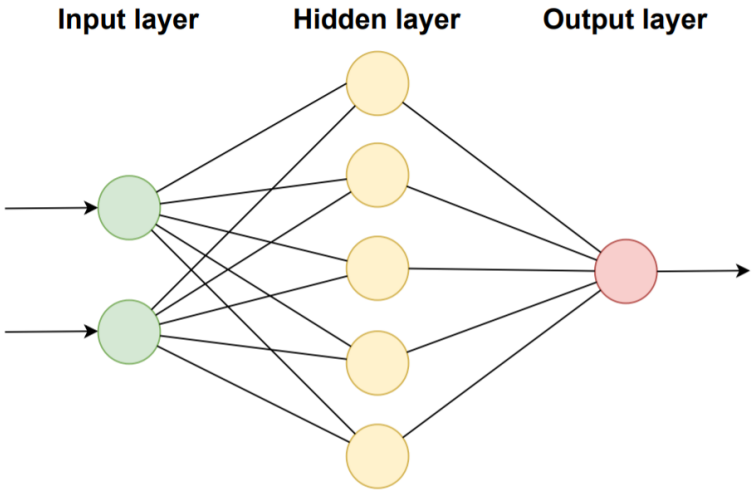

Figure 2.2: Overview of a simple Neural Network

a bias added. h is used to make it more readable, the answer of the summation is set to the sigmoid function. Every neuron is based on this equation. Therefore, we can set the equation for the output neuron to

$$o = \sum_{n=1}^{n=5} H_n * w_n + bias_n \tag{2.3}$$

$$O1 = \frac{1}{1 + e^{-o}}, \tag{2.4}$$

where H_n is the output of the hidden layer, w_n are the corresponding weights and $bias_n$ are the corresponding biases. The answer is again put in the sigmoid function with a temporarily variable o to make it readable. $O1$ is the resulting answer of the output layer. Where fig. 2.2 is an example, it is also possible to have more input neurons and multiple output neurons. More inputs as it can have more variables where the answer depends on, more outputs as it can detect more than one feature.

In case of a supervised network, weights are adjusted and corrected every epoch so it will learn the right features. The network is executed every epoch and if the answer is not correct with the real valid answer, weights are a bit adjusted. Mostly, it is programmed to stop after a number of epochs, but the rule is that if then number of epochs increases, the accuracy increases as well. After the training, the weights are trained in such way that it can find the features out of the inputs with a certain accuracy. It will never be 100% true or 0% false. Thereby there can be set a certain threshold. For example, above 90% is the result valid and is the feature detected.

Unsupervised works technical the same except that it is not corrected by a true or false answer (unlabeled). By giving multiple similar inputs, the network extracts features which are repeatedly given to the network. The weights are adjusted by the found features. After the training, the network can find the features in the new input, but thus without a label. Also, the network can detect irregularities if the input is differences from the trained input. So, new features which is not classified before. An example of malware detection.

Semi-supervised is in between both supervised and unsupervised. There are some labelled features but the network discovers also new features. Reinforcement learning is based on a reward structure by taking actions. The weights will also be adjusted in the way the system is corrected.

2.1.3 Deep Learning

Based on NN is Deep Learning (DL). A DL is called a DL if has at least 3 hidden layers. An example of a DL is shown in fig. 2.3. As shown the network looks like a NN but then

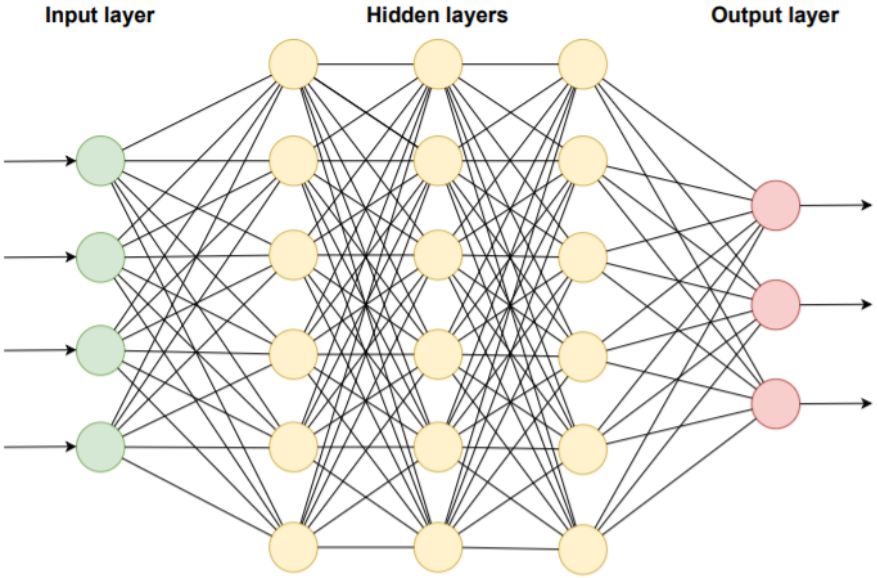

Figure 2.3: Overview of a simple Deep Learning network

with at least 3 hidden layers. The number of inputs and outputs can be any number if it is at least 1. Again, every neuron is connected to each neuron in the next layer. Some types of a DL network are, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Generative Adversarial Network (GAN) and Deep Belief Network (DBN). Each type has its own specific application, like extracting features out of images with a CNN. Notice, that fig. 2.3 shows an example of a feed-forward network. Depending on the type of the network, extra connections between the neurons can be implemented.

2.2 Keras & AutoKeras

Chollet et al. [4] presents a library, named Keras, to design, create, train, test and predict an neural network. This library is based on the presented library of Google Brain team called Tensorflow. Chauhan et al. [5] present an ANN example using the Tensorflow framework with promising results to set up a network in a easy way. Keras is developed to do fast experimentation in a simple, flexible and powerfull way, to go from idea to a solid result in a fast way [4].

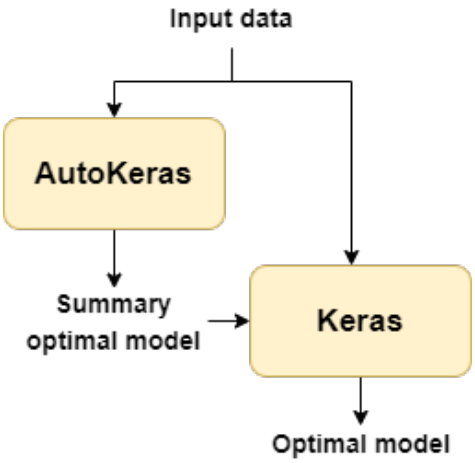

Figure 2.4: Overview of the working combination between AutoKeras and Keras

Based on Keras is the presented work of Jin et al. [6] to develop an automated Machine Learning (autoML) algorithm for Keras, named AutoKeras. AutoKeras creates with the given data multiple neural networks and returns the most optimal model. The returned optimal model can also be trained with the provided data with AutoKeras. With the case of infinity amount of (computation) time, there can be tried an infinity number of models. Thereby, there are some restrictions given by the user such as max trials and number of epochs to limit the computation time. As AutoKeras is an efficient autoML algorithm, the number of trials can be set such low number that the computation time is acceptable for this purpose. AutoKeras is used to find a summary of the optimal architecture for the given data. The summary is used to create a neural network architecture in Keras to train, test and predict different models with different given data. An small overview is shown in fig. 2.4. The choice to do the last step in Keras is made because in Keras only the created model is trained, instead of trying different models, which will save computation time. The summary of the models both with Keras and AutoKeras is the same because in the in Keras created model is the most optimal one and AutoKeras returns the most optimal model.

2.3 Shuffle samples

To make the training, validation and test data representative, the data need to be shuffled before the training and between each epoch. This reduces the variance and prevents over-fitting of the model. The model is trained on the characteristics of the samples and not trained to be a specific model for the training data. Otherwise, if new data appears, there is a large chance that it doesn't fit into the model. Bengio [7] stated that the model converged faster if the training data is shuffled every epoch. The first shuffle round before the training is needed because the data is ordered by class and label. Shuffling after each epoch is needed to prevent data-ordering effect learning. If the data is not shuffled after each epoch, the gradient descent is pointing to the same direction between sample N and N-1 if sample

N is always offered after sample N-1. This result that the gradient descent of a sample N is making use of the bias of sample N-1 and doesn't make an independent change for its own biases, so learning the order of the samples. Shuffling after each epoch means that the gradient descent is changing for every point in an other direction [7]. This is because the sample has another label every epoch at the same point of training in the epoch. Now it updates its values to the direction of the label instead of the order of data. The model is then less over-fit and predicts a more accurate result if the model is fed with new data.

Chapter 3

Literature

In this section are six comparable solutions presented by other authors. All with different approaches to reach a certain accuracy. Besides that, one section describes an alternative to do a FFT analysis using a neural network. There is also a proposal to de-noise the signal.

3.1 Comparable solutions

Other comparable studies are done to recognize environmental sounds by a NN. Different methods in the field of ML are proposed, all with different ways of preparing audio samples for the network. Chandu et al. [8] presents a method describing the use of a CNN network to identify types of bird sounds. The audio sample is prepared by removing the silence or background noise in a sample by thresholding every frame. With this method, only the most part of the audio frame are bird sounds. The audio sample is converted into a spectogram after concatenating the frames without the silence. The length of the samples remain between the 5 and 20 seconds. The spectogram is used to train and test the CNN implemented on a GPU. The proposal reached a maximum accuracy of 97%. A real-time application on a smartphone app reached an accuracy of 91% and Chandu et al. [8] stated that it can be improved if the data is send to a cloud server.

Bold et al. [9] present another method to identify types of bird sounds. The authors present multiple methods in combination with a CNN. The best results are achieved in combination with Multiple Kernal Learning (MKL). The authors [9] realised this by extracting deep neural features based on the activation values of an inner layer of CNN. An accuracy of 78.15% [9] is achieved with a CNN plus an MKL with a ρ equal to 8. This is a bit of an improvement compared to the original method where a CNN is combined with Support-vector Machine (SVM). The last combination reached an maximum accuracy of 75.74% [9]. The length of the samples are 10 seconds.

Agarwal et al. [10] present a method of classifying glass breaking, gunshots and smoke alarms. This method is also realised by setting audio samples into a spectogram and this is used to train the CNN. The authors [10] reached an accuracy of 90%, a loss of 0.30, a precision of 0.857 and a recall of 0.84 based on a test set. The training set has a length between 1.05 and 1.39 minutes. The test set is between 0.33 and 0.5 minutes.

Avanzato et al. [11] present a method of classifying the intensity of the rain using a CNN. The rain is falling on a piece of material with a diameter of 20 cm and that is recorded. Ceramic, plastic, aluminium and steel are the 4 piece of materials which are tested. The audio signal is analysed by a Mel-frequency Cepstral Coefficient (MFCC) method and fed to the CNN. Avanzato et al. [11] reached an accuracy between 65% and 93% where the best results are achieved for the materials ceramic and plastic.

Khunarsal et al. [12] present different methods to identify environmental sounds using a feed-forwarded neural network and a k-nearest neighbour network. Different methods are presented but the methods with a spectogram analysis give the best results. Khunarsal et al. [12] compared the two networks in accuracy. There are 20 different environmental sound identified. The best results are achieved with a spectogram in combination with a linear prediction coefficient and a matching persuit with an accuracy of 94.98% with a k-nearest neighbour network. The feed-forwarded neural network achieved an accuracy of 85.66% for a 1 second sample. Increase the sample time to 6 seconds and the accuracy increases to 90.57%.

Abdoli et al. [13] present a method with a one-dimensional CNN to identify environmental sounds. The small size of a 1D CNN compared to a 2D CNN makes it more suiteble to implement it inside a mobile or embedded device. The audio is directly used to the network and initialised with a gammatone filterbanks. Abdoli et al. [13] reached a mean accuracy of 89% for this setup with an audio length of at least one second.

Sabab et al. [14] presents a comparison between a MFCC, a Principal Component Analysis (PCA) and other reduction techniques in combination with a 1D CNN. The experiment was executed by speech recognition of the Bangla language. The accuracy of the PCA in combination with a 1D CNN is 94.58%. As the accuracy is higher compared to the other presented reduction methods, the accuracy in combination with a MFCC, which is 97.26%, is more accurate. The difference in accuracy is even higher if it is in combination with a Long-Short Term Memory (LSTM). Namely, 83.12% for the PCA and 93.83% for the MFCC.

Dash et al. [15] present a method to reduce the training time with similarly accuracy. The purpose is investigated by using brain signals of speech. The baseline CNN has an accuracy of 95% and a training time of 24.61 hours. A proposal with a PCA with 50 components reaches an accuracy of 93.89% and a training time of 2.02 hours.

Besides using a CNN with a spectogram, an ANN is used where the first input layer uses every frequency component of the FFT analysis of the audio sample. Tangkawanit et al. [16] present a feed-forwarded ANN recognizing the type of gun shots. To prevent an unlimited source of frequency components, the frequency components are set into bins with a specific bin size. The frequency components inside the bins are averaged. Also, only the interesting part (0-2000Hz [16]) is set to the ANN. After averaging, the bins are normalized between 0 and 1 and feed to the input layer of the ANN. The results are 100% accurate for no noise and 30dB SNR and decreases to 25% for 5dB SNR.

To increase the accuracy for the higher noise regions, a continuation of the research is done. Tangkawanit et al. [17] presents an example where the feature extraction step is fur-

ther investigated. Especially the bin size of the frequency components is further investigated. The accuracy is measured for different bin sizes to measure which is best. Tangkawanit et al. [17] presents that the maximum accuracy is increased to 100% for a SNR of 10dB, 98% for 5dB and 92% for a 0dB SNR.

Rios-Gutierrez et al. [18] present an example of classifying heart beats between healthy and unhealthy patients using an ANN. The authors [18] claim an accuracy of 85 $\pm$ 7.4% and 95 $\pm$ 6.8% sensitivity in an ideal simulated situation. This drops down to 48.5% accuracy by recording a new sample by a patient. The authors [18] suggest that the possible issue is the background noise of the patient and the environment. A suggestion for a solution is made by adding Gaussian noise to the training samples to train the network with simulated background noise, but no results are presented.

In table 3.1 is a summary of the proposed solutions. The table presents an overview of the type of network which is used and the achieved accuracy. In some cases are multiple accuracy's presented. A presented worst case situation and best case situation.

Authors	Purpose	Type of NN	Accuracy
Chandu et al. [8]	Bird classification	CNN	91% - 97%
Bold et al. [9]	Bird classification	CNN-MKL	78.15%
Agarwal et al. [10]	Glass breaking, gunshot and smoke-alarm classification	CNN	90%
Avanzato et al. [11]	Rain classification	CNN	93%
Khunarsal et al. [12]	Environmental sounds	k-NN & ANN	94.98% & 90.57%
Abdoli et al. [13]	Environmental sounds	CNN	89%
Sabab et al. [14]	Recognition Bangla language	MFCC-CNN & PCA-CNN	97.26% & 94.58%
Dash et al. [15]	Brain speech recognition	PCA-CNN	93.89%
Tangkawanit et al. [16]	Type of gunshot classification	ANN	25% - 100%
Tangkawanit et al. [17]	Improvement of the gunshot classification	ANN	92% - 100%
Rios-Gutierrez et al. [18]	Heartbeat classification	ANN	48.5% - 85 $\pm$ 7.4%

Table 3.1: Overview of the related work

3.2 Neural Network alternative for FFT

One of the sub questions is on how to prepare the audio data for the ML method. Given is that most studies use data in the frequency domain, the data need to be converted from audio to frequency, either with short samples or with a complete audio.

Velik [19] and Keerthipala et al. [20] propose an idea on how to convert an audio signal into FFT components using a neural network. Both papers present a working result of using a neural network to do the conversion. Where Velik [19] concludes that the neural network

implementation on discrete time has exactly the same result as doing a classic FFT analyse of a sampled signal. As the results are good, the condition to achieve good performance is that there is the ability of parallel computation. Also, the proposed method is designed in Matlab and the conclusion is based on simulation results, not a real implementation.

Keerthipala et al. [20] present also a proposal of doing a FFT analyses with a neural network. The results are good, 95% to 100% accuracy of all the components. But, as it said by Keerthipala et al. [20], a neural network can never give always 100% accuracy. It concludes that it is an approximate alternative for a FFT analyses. Also, only the odd components are simulated.

3.3 De-noise a signal using a FIR filter

Hossin et al. [21] present different windows in combination with a low-pass FIR filter to de-noise the input signal. As all the filters reduce the noise over the signal, the proposed low-pass FIR filter has the best result. The proposed window is compared to the Gaussian, Tukey, Kaiser and the proposed method by Rakshit et al. [22]. The SciPy library proposed by Virtanen et al. [23] will be used to implement the filter. The function *firwin* is used to create a FIR filter. Out of the proposed windows is only the Gaussian, Tukey and Kaiser window available. The Tukey window is chosen because the overall results are comparable with the proposed method [21] and it didn't require a non-given parameter such as β in the Kaiser window.

Boche et al. [24] compared a Wiener filter which is interpreted as a cascade whitening- and estimation filter with a FIR filter. Boche et al. [24] conclude that the Wiener filter can estimate the FIR filter but it never improves the smoothness of a FIR filter. Also, the Wiener filter can be unstable if the spectral densities are continuous but not smooth enough.

Jeong [25] present a method for adaptive noise canceling. The presented method with a Minimum Phase Whitening (MPW) with in cascade an adaptive FIR filter. Jeong [25] concludes the noise is reduced and that there is less fluctuation in a desired signal.

Chapter 4

Methodology

4.1 Proposed method

The proposed method is partially based on research findings reviewed in section 3.1. There are basically two solutions proposed, one with a CNN solution and one with a feed-forward ANN. As both results have good accuracy, the ANN solution proposed by Tangkawanit et al. [16] [17] is more accurate. Also, an implementation of an ANN is simpler and lower in resources. The CNN implementation proposed by Chandu et al. [8] uses parallel computation where an ANN is also suitable for sequential processing. The proposal by Bold et al. [9] is not that accurate but have a short sample time compared to the other proposals. Agarwal et al. [10] present higher accuracy but have a long sample time compared to other proposals. Rios-Gutierrez et al. [18] have a low accuracy in a random environment, but the difference between the labels is small. For this reason, a method with a feed-forward ANN is proposed, with a number of layers and neurons defined in section 4.2. The input layer consist out a number of frequency components divided in a number of bins, comparable with the method by Tangkawanit et al. [17].

To prepare the input layer of the ANN, the audio data need to be converted to the frequency domain. In section 3.2 is a method proposed to realize this with a NN. As the results has an accuracy between 95% and 100%, it is never the 100% accuracy of a classic FFT. Also, the necessary resources isn't an improvement which is needed to implement the system on the device. This way, the decision is made to implement the classic FFT. Before calculating a FFT, the solution proposed by Chandu et al. [8] to remove noise will be applied and measured if it benefits to the accuracy. After the FFT, the signal needs to be normalised and divided into bins as proposed by Tangkawanit et al. [16] [17].

One approach will be realised by applying a FIR filter before the FFT realised by the finding in section 3.3. In this case there will be no noise reduction in the normalisation step as it must realised by the filter. The complete realisation is explained in section 4.2.3.

Before the audio can be converted to the frequency domain, the length of an audio sample need to be considered to detect a specific feature. Consider the complete sample at once in the frequency domain or use Short-time Fourier transform (STFT) and feed the network with smaller samples. The length of the sample is determined in section 4.2.2.

The shaver can run at two different motor speeds and speed and sound during using the product can vary. Different approaches and situations are trained, tested and predicted to measure the results in a confusion matrix. There will be situations where the train set is based on recordings of one motor speed, two motor speeds, multiple motor speeds where the speeds varies around the two motor speeds and situations where some units are recorded while using the product. All the situations are presented in chapter 5.

The important aspects where a feature must be measured against, is in terms of accuracy, recall and precision. This will be realised with hand of a confusion matrix, explained in chapter 5. The findings of every situation, case and approach will be measured and compared to other situations if it improves, decreases or doesn't change the result at all. There are basically two approaches, one with removing the noise under a certain threshold, in both training, testing and prediction samples, which can be a threshold of 0 and the noise isn't removed and one with a FIR filter. The first approach is detailed described in section 4.2.2 and the second approach in section 4.2.3. The main flow of both approaches is given in fig. 4.1. The main flow of both situations is comparable except that in the right situation there is a FIR filter applied. The normalisation step is also different in both approaches which will be explained in section 4.2.3. The main flow is that a audio sample is sampled and if the FIR filter is applied, de-noised by the FIR filter. Hereafter, the the audio sample is applied by a

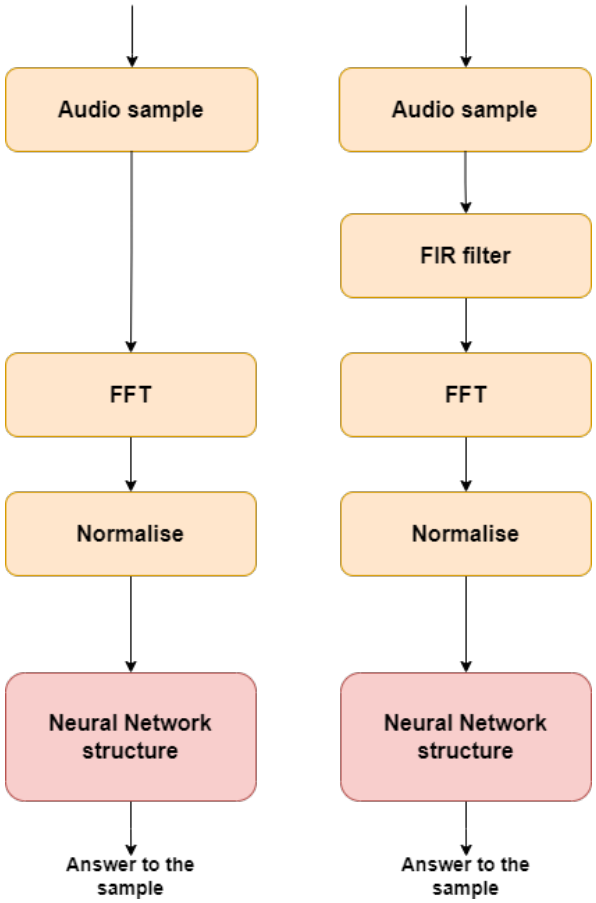

Figure 4.1: An approach without FIR filter (left) and an approach with FIR filter (right)

FFT. The response is normalised and applied to the Neural Network structure.

4.2 Architecture

4.2.1 Hardware

To show an impression of the physical setup of the shaver a real image in shown in fig. 4.2. This is the top view of the shaver without the external body and with a USB microphone

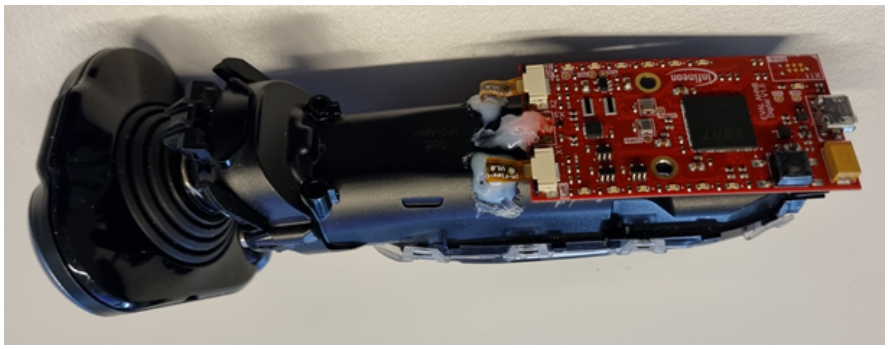

Figure 4.2: Real image of the shaver, top view

board of Infineon mounted on top. In fig. 4.3 is an overview of the important components which is related to the research. The two microphones are mounted behind the motor related to the shaver head to shorten the cable length to the PCB and reducing the complexity in terms of space inside the shaver. The sound and vibrations are recorded behind the motor and sampled audio is read through a USB cable by a Python PC program. The approach of how the audio is handled in python is explained in section 4.2.2.

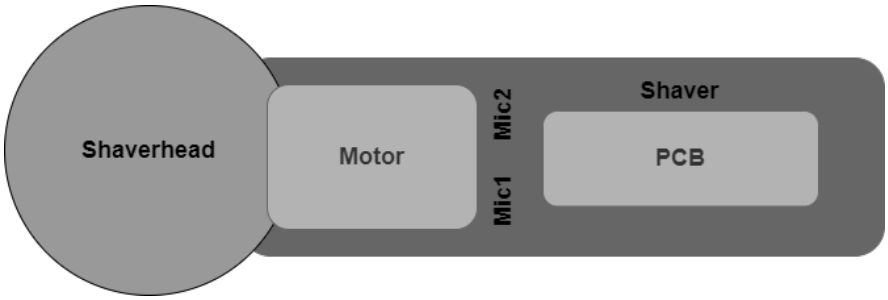

Figure 4.3: Schematic overview of the shaver, top view

4.2.2 Software de-noising with thresholding

An overview of how the approach is elaborated is shown in fig. 4.4. On the left side is the step by step approach on how the network is trained and tested, starting with the left top initialization arrow. On the right side is the step by step approach on a prediction by the network of a new sample, starting with the right top initialization arrow. The initialization on the left side starts with reading one large sample of 1250 seconds from the database.

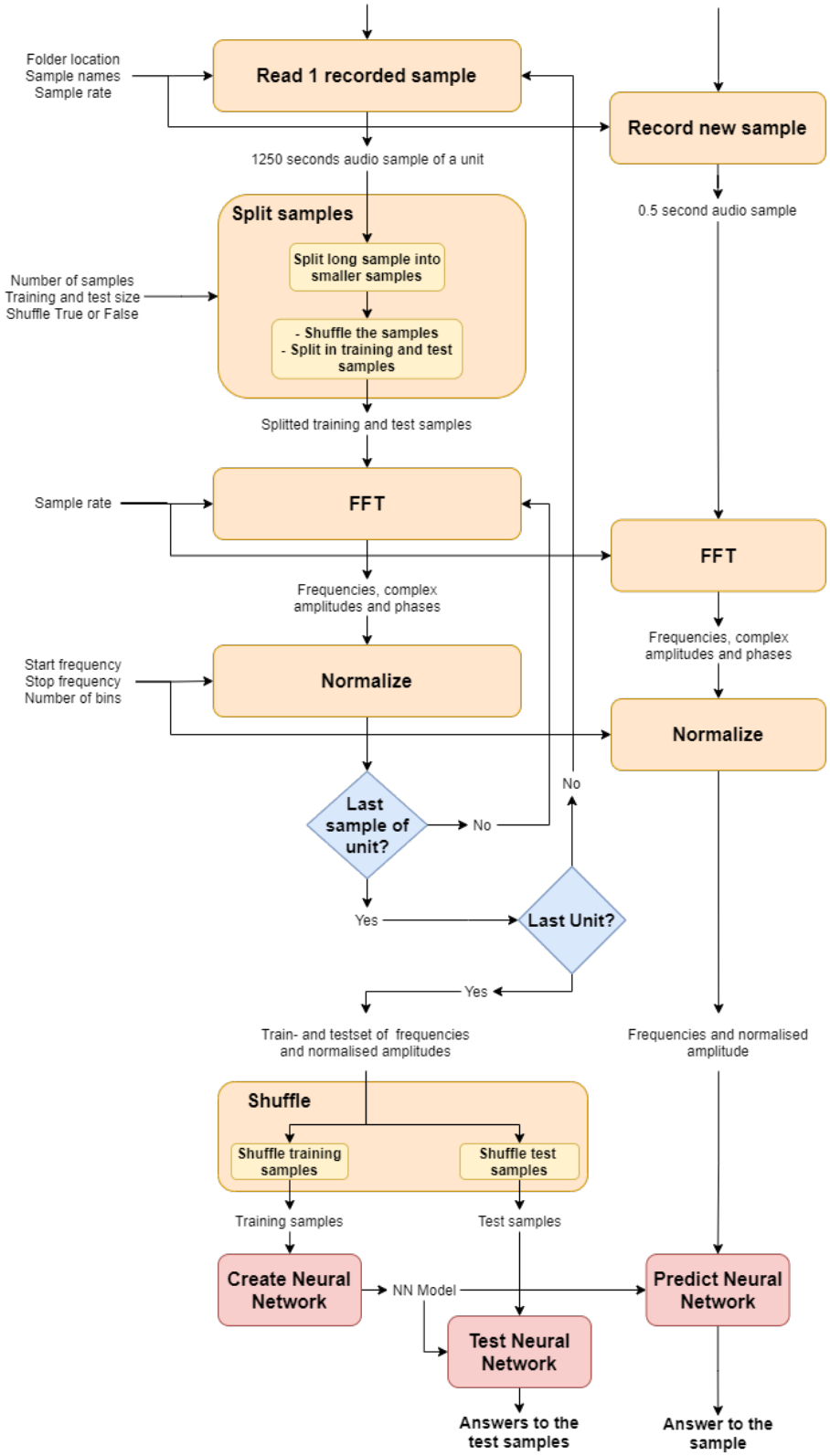

Figure 4.4: Overall architecture

Hereafter, the sample is split into smaller samples. The length of the sample is defined in section 4.2.2. Given the length of the sample, the number of samples is defined as

$\frac{1250}{sampletime} = number\ of\ samples$. Then the samples are shuffled to create randomness between training and test set every training. The samples are split 90% for the training set and 10% for the test set. Now every training and test sample is calculated towards the frequency domain using the FFT function of the numpy library of Harris et al. [26]. The complex frequency amplitudes returned by the FFT function and normalised. The normalization steps are explained in section 4.2.2. This is executed for every sample and every selected unit. Now there is a training and test set ordered by unit. This must be shuffled to randomize the samples and get rid off the order in the units. The training samples are fed to function of creating a neural network architecture, in the way as explained in section 2.2. The created model is together with the test samples tested on accuracy and losses. As this test samples are recorded in the same session as the training set, the accuracy and losses aren't representative for at a long term estimation. Therefore there is a setup made to predict a new sample with the neural network, shown on the right side. The sample is recorded with the same length, calculated to the frequency domain and normalised with the same parameters and then fed to the neural network. These setup is used to measure the results in chapter 5.

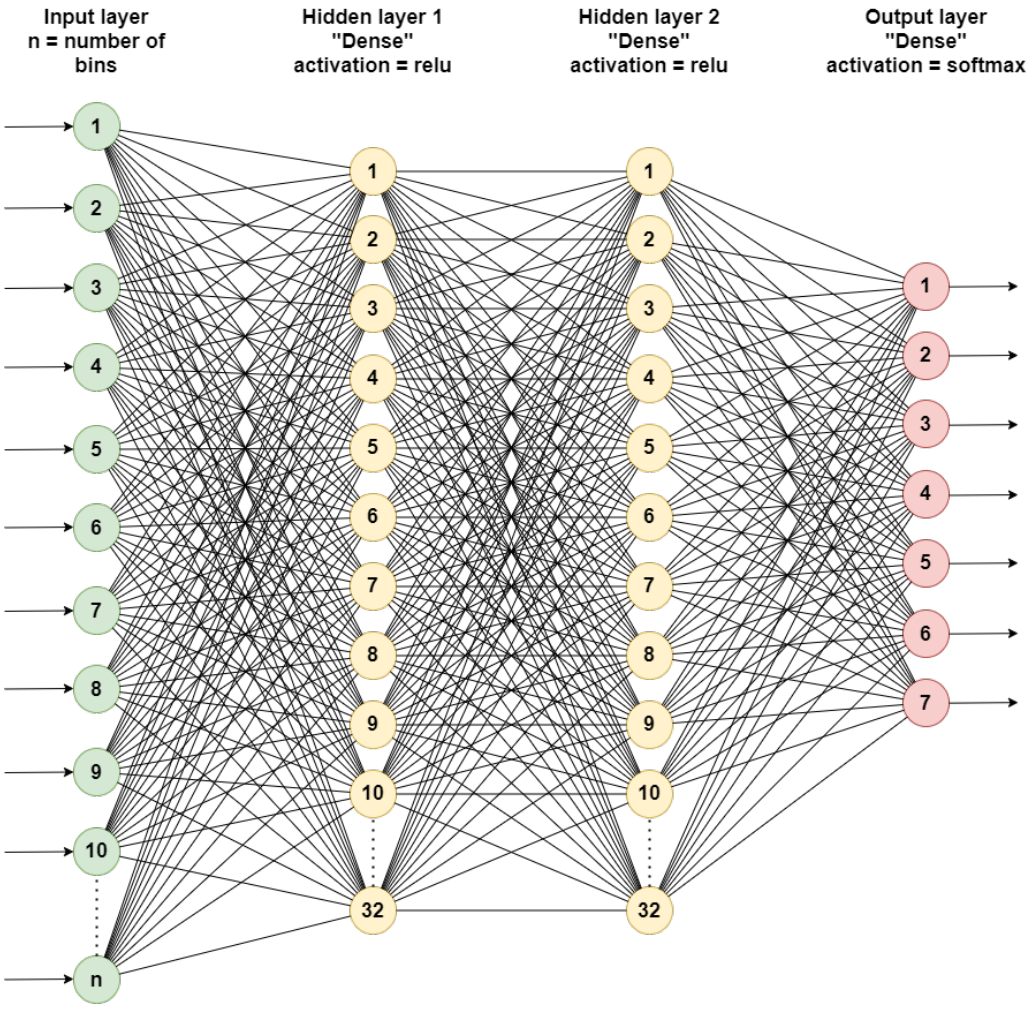

Figure 4.5: Optimal architecture of the Artificial Neural Network

The architecture of the neural network, which is the optimal model returned by AutoKeras, is shown in fig. 4.5. The architecture is the most optimal one for every set of data, only the size of the input layer will differ depending on the number of bins equal to *n*. There are two hidden *Dense* layers with a size of 32 neurons. Both layers have an *relu* activation on the output. The output layer is a *Dense* layer with a *softmax* output activation and a size of 7 neurons. There are 5 units, one label without an unit and also *Silence* is labeled. Every label corresponds to one neuron in the output layer. The numbers inside the neurons indicate the number of neurons.

Normalisation

The normalisation step is calculated before the sample will be fed to the feed-forward ANN. In the normalisation step are some parameters which need be analysed so it can be set to correct values. These values are the interesting frequency part of the FFT, sample rate, sample time, noise threshold and bin size. In fig. 4.6 is an overview of the normalization steps.

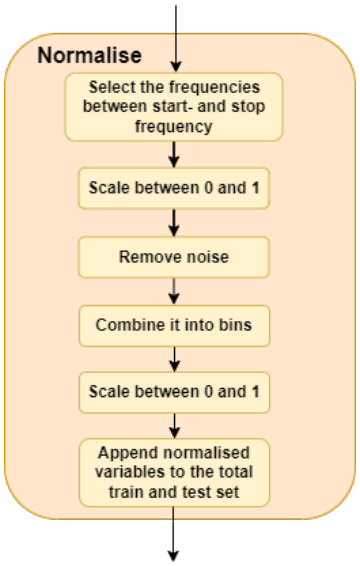

Figure 4.6: Architecture of normalising

First the interested frequencies are selected. Then the amplitude vectors are scaled between 0 and 1. Now the noise can be easier removed without knowing which unit is on the shaver head. Then the frequency components are added together and again scaled between 0 and 1. Every sample will be appended with all the samples together with an index. Hereafter, will be explained how every unknown value is determined which is needed to do the normalization steps. In fig. 4.7 is an overview of the frequency components when the motor is at normal speed. The frequency components are complex. The vector length of the complex values is used to include also the phase in the data and analyse. Different units are shown in the figure, together with a sample when no unit is attached. Besides the units, silence is also shown. This is to prevent that the network will predict an unit at the

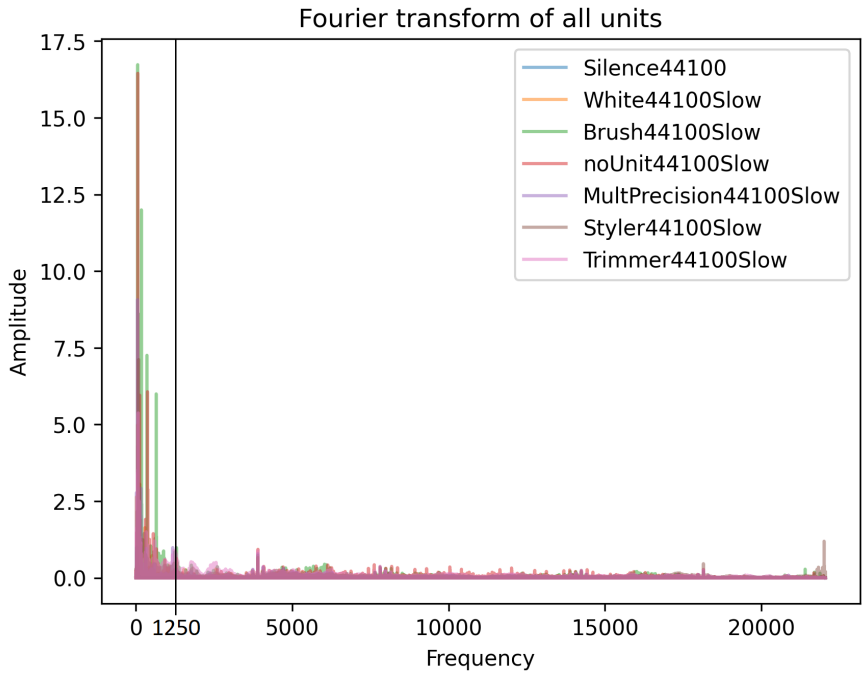

Figure 4.7: Amplitudes of each frequency component at low motor speed

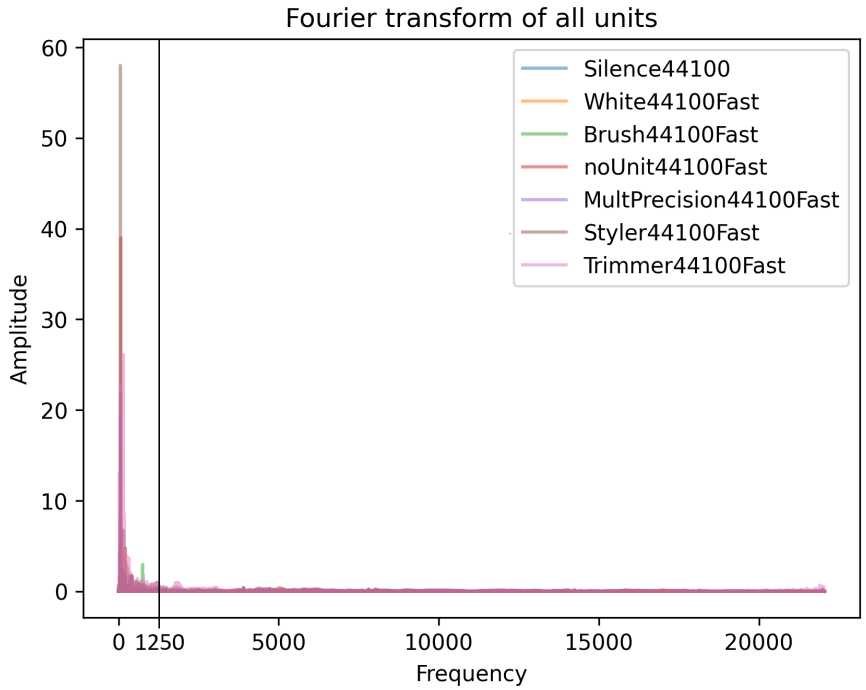

Figure 4.8: Amplitudes of each frequency component at high motor speed

moment that there are no vibrations or sound from the device itself. The sample rate is set to 44100 samples per second to discover every frequency a human ear can hear. Figure 4.8 shows the overview of the frequency components when the motor is at high speed. As can seen in the figures, the highest amplitudes are in the lower frequency range and thus the

vibrations and sounds of the different units are in the lower frequencies, below the indicated 1250 Hertz. Figure 4.9 and fig. 4.10 are shown to analyse the interesting lower frequencies

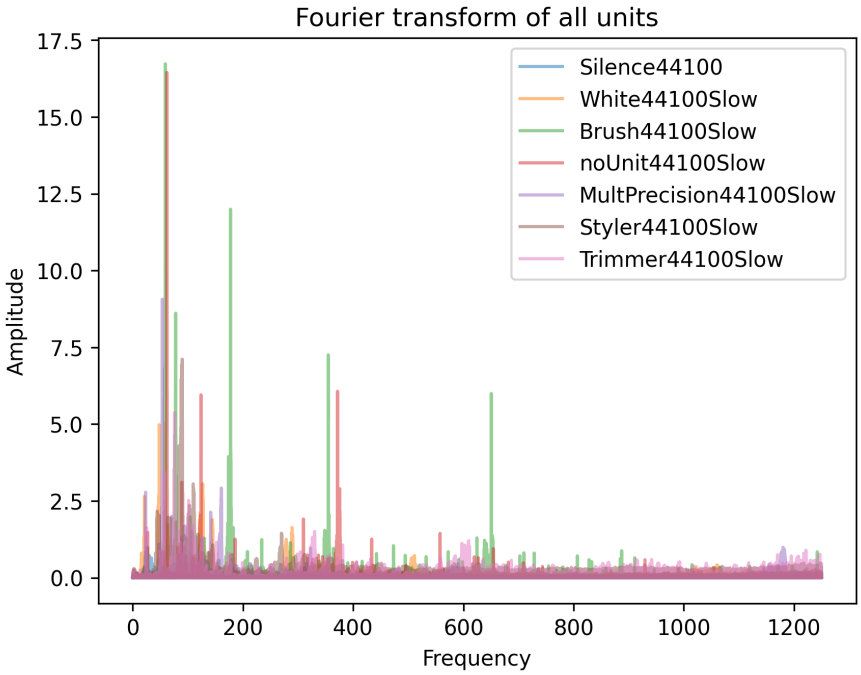

Figure 4.9: Amplitudes between 0Hz and 1250Hz at low motor speed

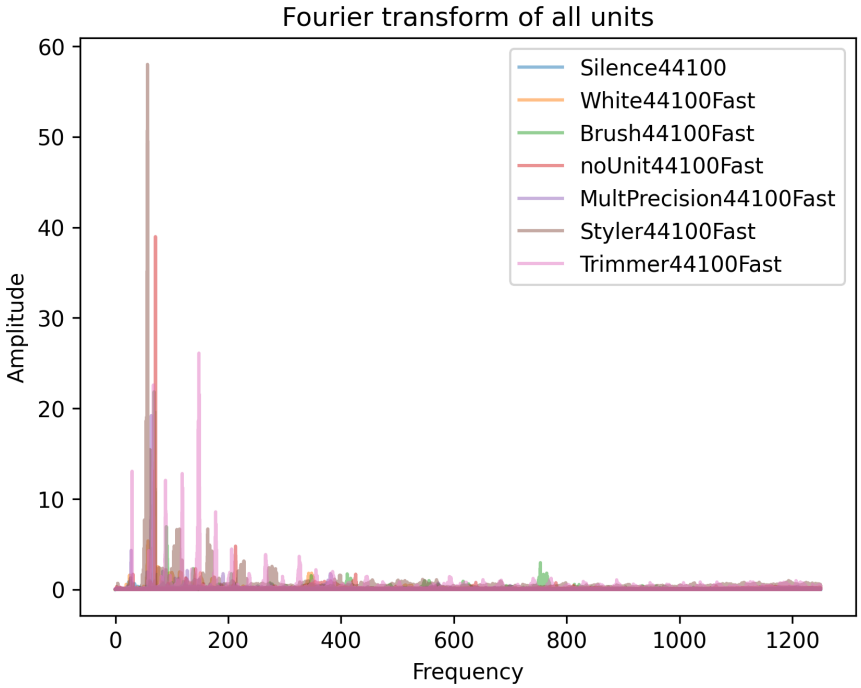

Figure 4.10: Amplitudes between 0Hz and 1250Hz at high motor speed

of the FFT. The frequency range is set between 0Hz and 1250Hz. When the motor runs at

normal speed, the interesting information is below 700 hertz. The *Brush* has a peak between the 600 and 700 hertz which contains information about the vibrations for this unit. When the motor is at high speed, the lower frequency peaks are relative higher comparing to normal speed. The highest peak of the *Brush* is moved between 700 and 800 hertz. The lowest frequency in where information is available is if the shaver is turned off, denoted as *Silence*. This label is trained to identify at the moment the shaver is turned off and it doesn't predict an unit instead. The lowest frequency where the information starts is approximate 10Hz. The highest frequency is 800Hz. According to the Nyquist sample theorem must the sample frequency be at least 2 times the highest frequency. To explore this, there is in chapter 5 besides the 44100 samples per second also a trade-off with a sample rate of 2000 samples per second.

The lowest frequency in the bandwidth is needed to determine the minimum sample time. Namely, the signal with the largest period is the signal with the lowest frequency. This is determined by $\frac{1}{10Hz} = 0.1$ second. In theorem it is possible to determine the lowest frequency with one period. But, to make the analyse more smooth and stable, a larger length of the sample is needed to measure the lowest frequency period at least multiple periods. It is also important that the network is trained with same or shorter sample periods compared to the length of a new sample fed to the trained network. This is defined as $Sample_{training} \leq Sample_{predict}$. This is needed to train the network with the same or more irregularities in the signals which also prevents over-fitting. If the new sample has a shorter length than the network is trained on, there is chance that the new sample has some irregularities where the network isn't covered for. The sample time is also influenced by the FFT analysis. The formula $\frac{1}{2*f} = sampletime$ determines the sample time depending on the accuracy in frequencies, denoted as *f*. To have a descent accuracy depicting the frequencies, the accuracy of the fft analysis is set to 1 Hz. This means a minimum sample time of $\frac{1}{2*1} = 0.5$ seconds.
The $Sample_{training}$ is determined on 0.2 seconds to cover at least two periods of the lowest frequency. The $Sample_{training}$ is determined on 0.5 seconds to reach an accuracy of 1 Hz. To cover both requirements, the sample time is set to 0.5 seconds.

In the figures 4.7 to 4.10 is shown a lot of noise over the complete spectrum for all the units. This is different for every sample and can be left out by setting a threshold. Everything below a certain threshold can considered as noise and contains no information related to the unit. This prevents that the model can be possible trained, tested and predicted with the wrong information. By setting the noise to 0 for the training, test and prediction samples, the information is equal for every unit and the algorithm will be trained on the vibration characteristics of the unit. As defined before, the information is between 10 and 800 hertz. To define at which level the frequencies are noise, different threshold levels are tested to see the difference. The units are first scaled between 0 and 1 to treat every unit equally with the same threshold value. The original scaled Fourier transforms are shown in fig. 4.11 and fig. 4.12.

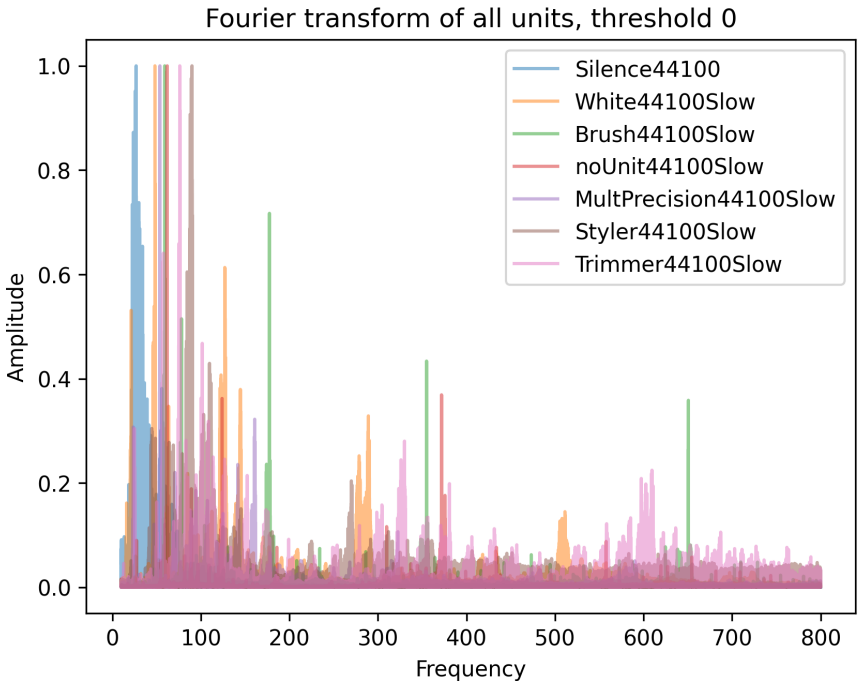

Figure 4.11: Scaled amplitudes of the frequencies between 10 and 800Hz at slow speed

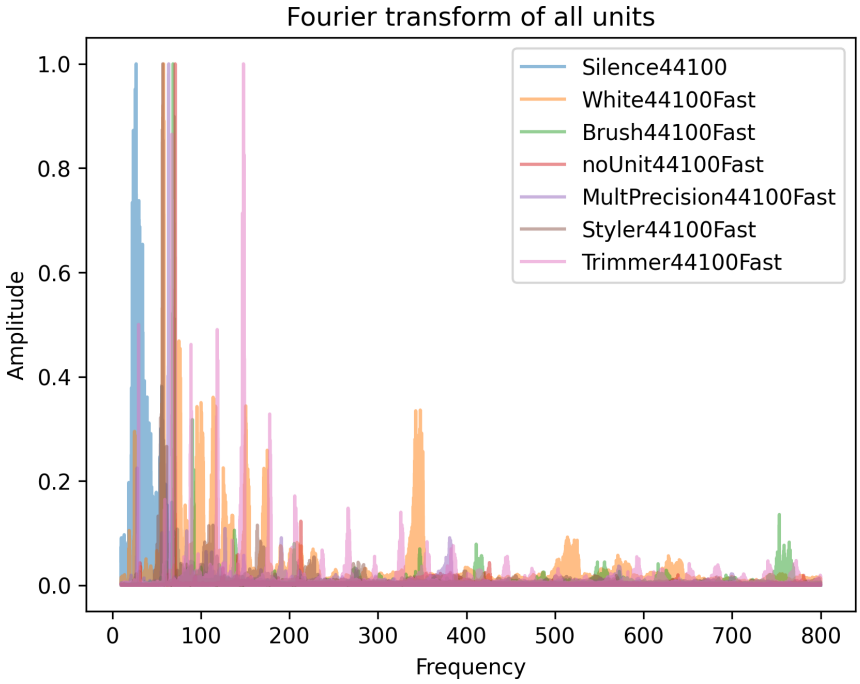

Figure 4.12: Scaled amplitudes of the frequencies between 10 and 800Hz at high speed

Figure 4.13 shows the Fourier transform with noise reduction with different threshold values at slow speed. Only slow speed is considered because high speed is less sensitive to noise. A threshold below 0.06 is needed to keep the important data. A trade-off in accuracy with the use of a confusion matrix is made with different threshold level trained models in

chapter 5 to find the best option.

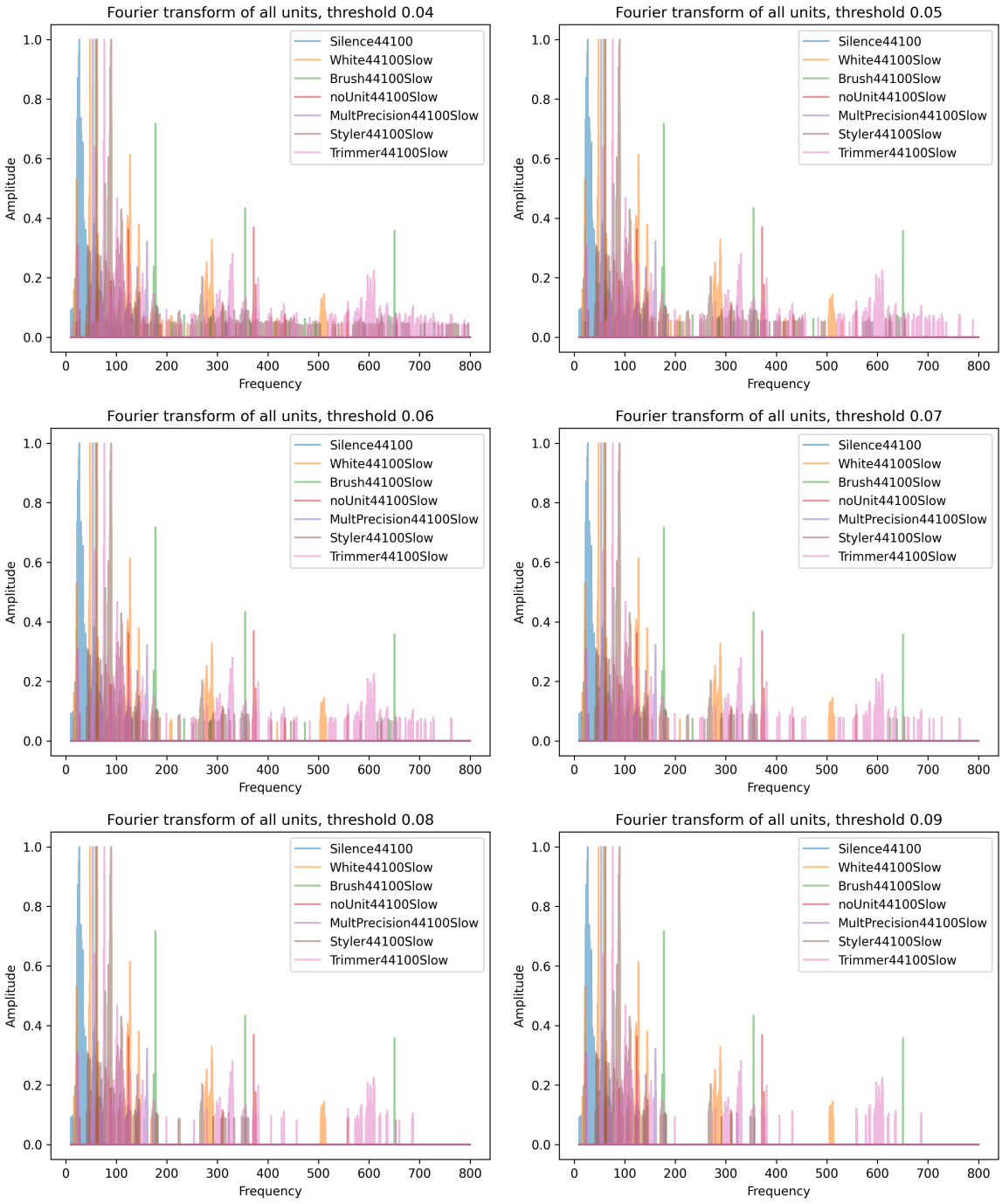

Figure 4.13: Noise filtered out at levels 0.04 to 0.09

To reduce the input layer of the ANN and compensate for small motor changes, frequencies are combined to bins. For example, 10 to 14 hertz are combined in to 1 bin, 14 to 18 hertz are combined and so on. After combining the frequencies into bins, the bins are scaled linearly between 0 and 1 to prepare it for the input layer of the ANN. To show the influence of the noise, the figures are shown with and without the noise with different threshold levels. The result without noise filtering is shown in fig. 4.14. Noise filtering at a level of 0.03 is shown in fig. 4.15.

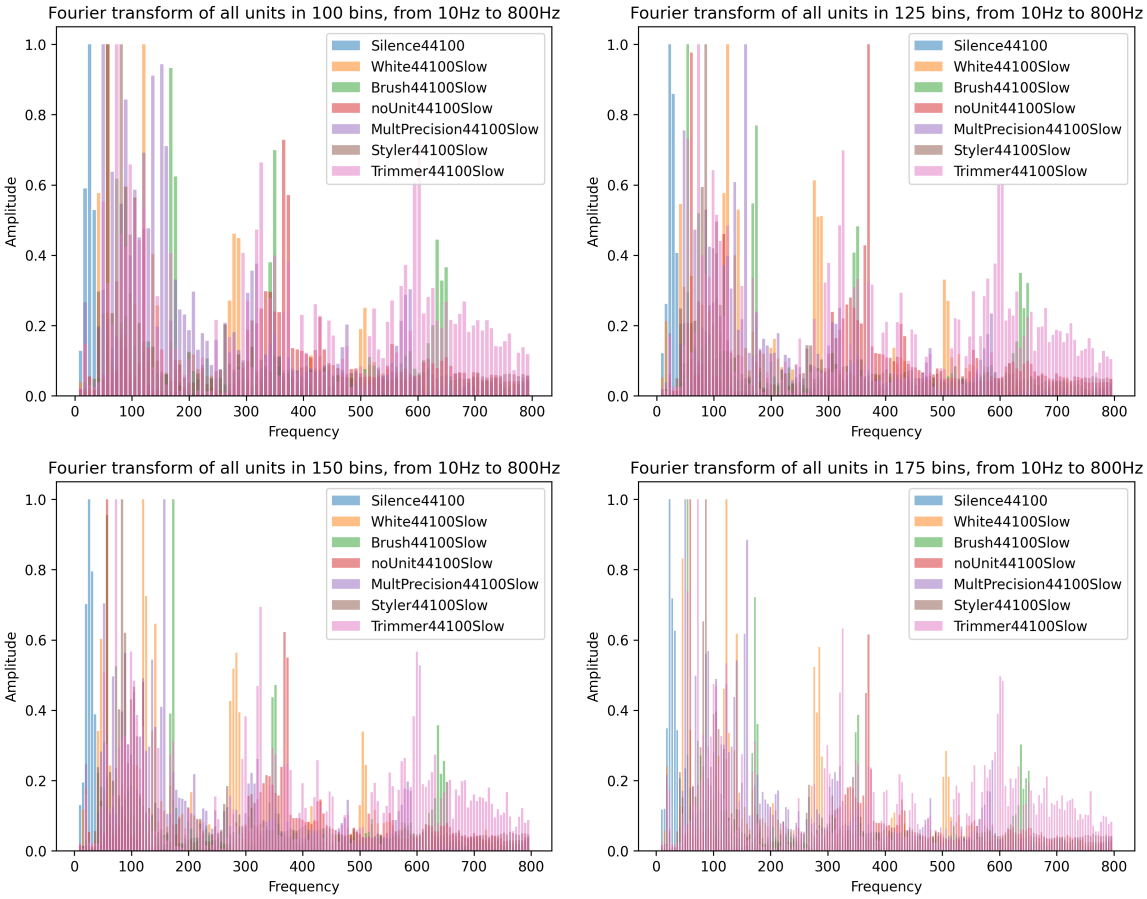

Figure 4.14: Different bin sizes with no noise filtering

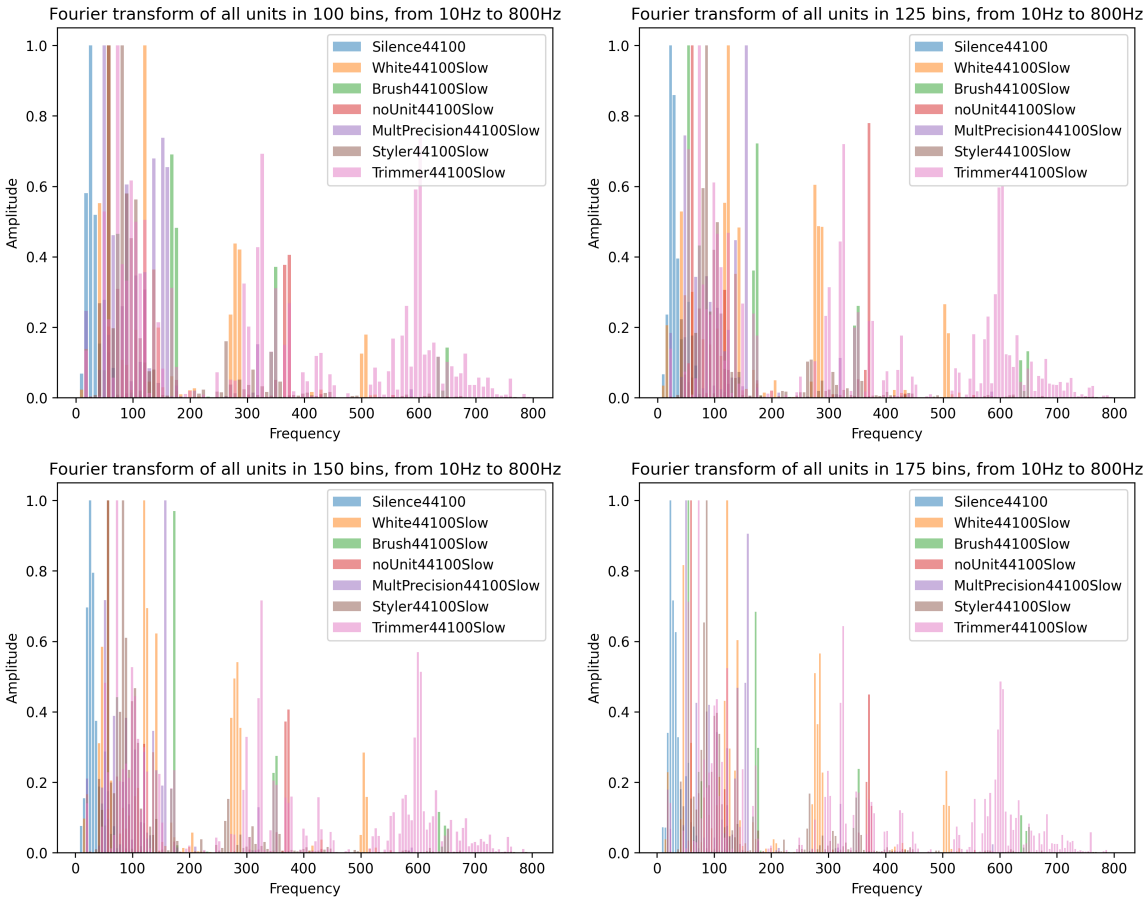

Figure 4.15: Different bin sizes with noise filtering at 0.03

In fig. 4.16 is an overview of different bins with a noise level at 0.06. Most of the fundamental frequencies are in a different bin at a bin size bigger then 150 bins. Besides the fundamental frequency, other dominant frequencies related to an unit can bring a contribution to the result. It is not completely depending on the fundamental frequency. As where there is an overlap in the fundamental frequencies between the units, there are enough dominant frequencies to distinguish between the units. By increasing the threshold value of the noise reduction, there can be more distinguished in the dominant frequencies between the units. Now the input layer of the ANN is kept as small as possible without causes of mistakes predicted by the ANN.

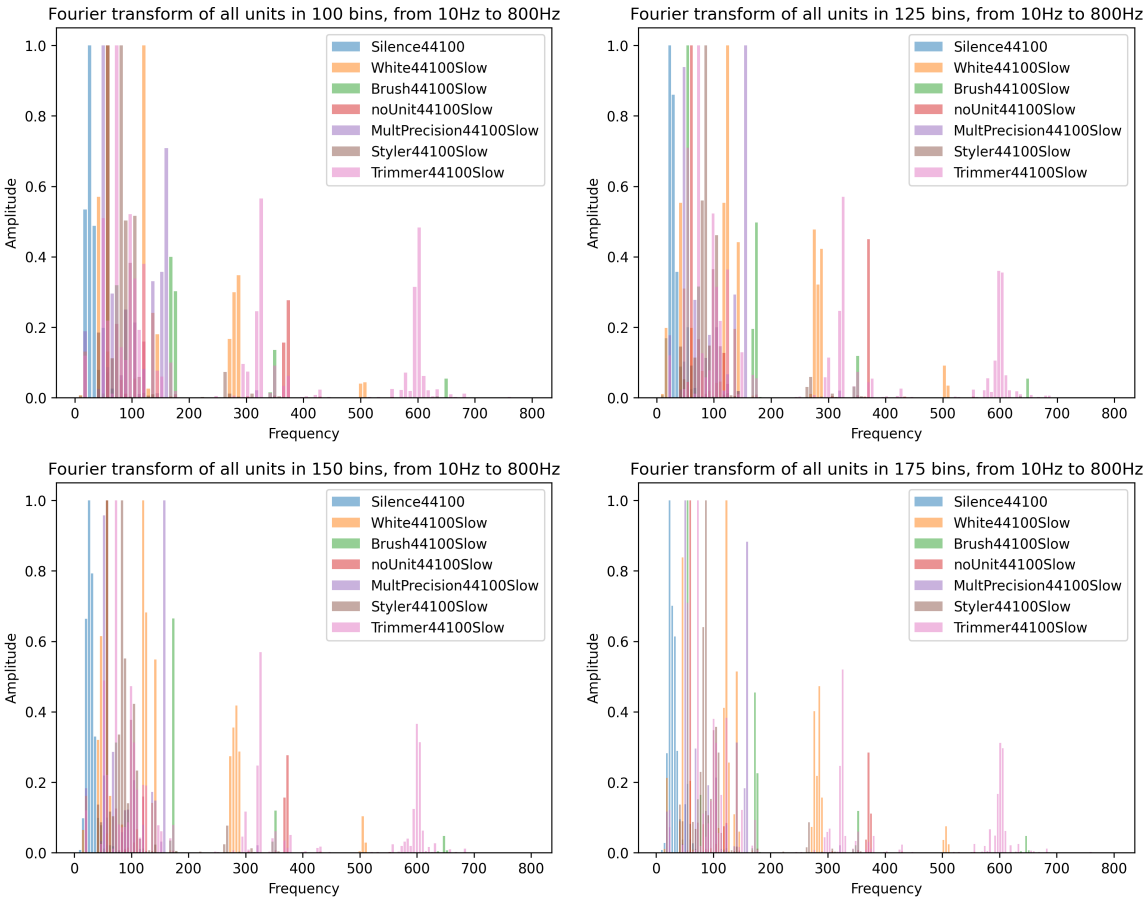

Figure 4.16: Different bin sizes with noise filtering at 0.06

The important parameters normalisation

In the beginning there need be determined five different variables, namely, the interesting frequencies, the sample rate, the sample time, noise threshold value and the bin size. The interesting frequencies are set between 10 and 800 Hertz. The sample time is set to 0.5 seconds. The bin size is determined at 150 bins. The sample rate is first set to 44100 samples per second, in section 5.3 is a trade-off made to investigate if a sample rate of 2000 samples per second has comparable results. In section 5.4 is a conclusion made of every case if the noise threshold value has a positive influence to the result.

4.2.3 Software de-noising with FIR filter

Another method which will be investigate is that the audio is de-noised by a FIR filter. A complete overview is given in fig. 4.17.

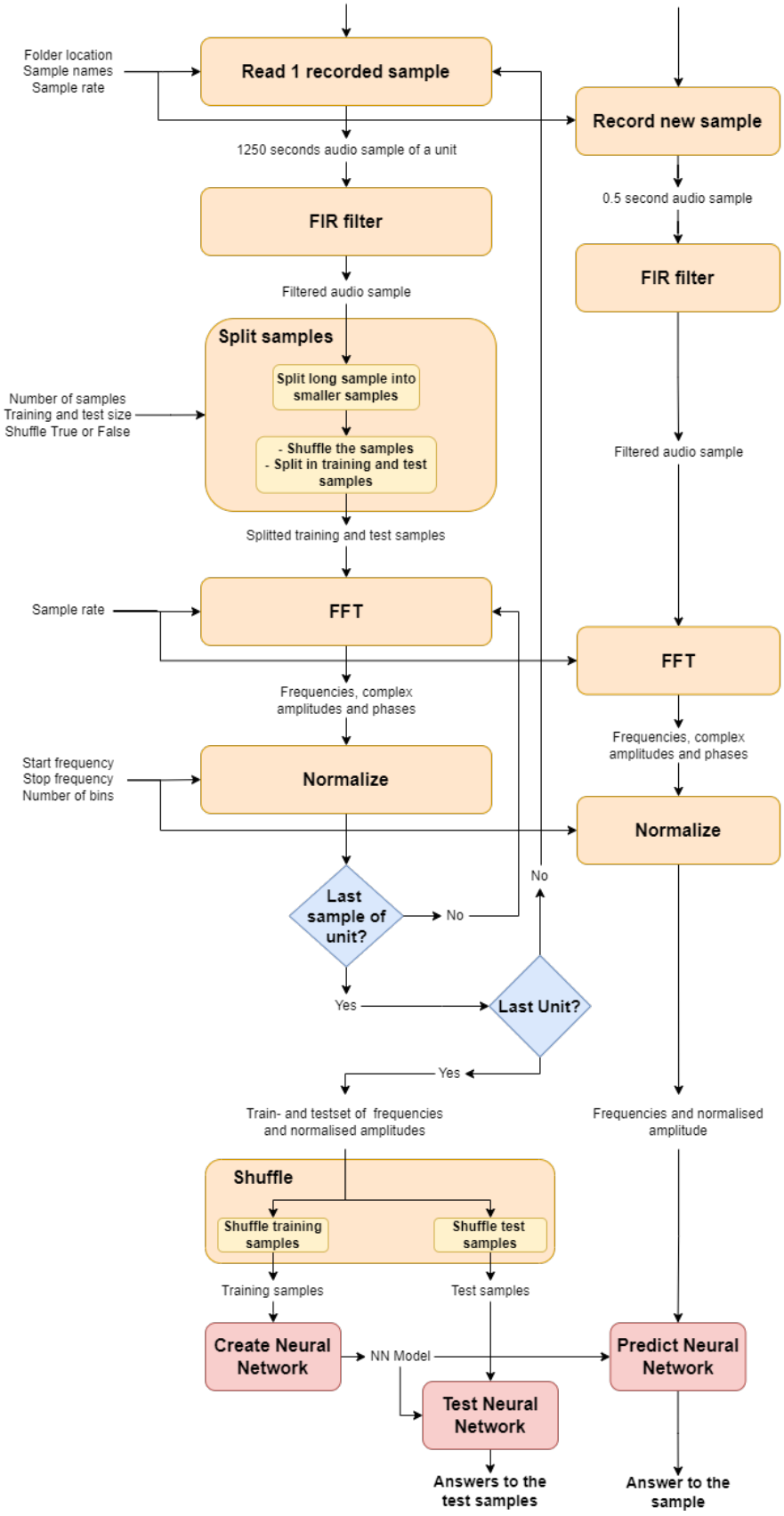

Figure 4.17: Overall architecture with a FIR filter

The difference between fig. 4.4 is that the audio sample is now filtered by a FIR filter. The filter has a cut-off frequency of 1000 Hz to have a save margin to the 800 Hz used frequencies. The samples are sampled at a sample frequency of 3000 samples per second to create a save playing margin for the interesting frequencies and the cut-off frequency. The filter is set to 23 delaying taps with a Tukey window. The first 22 delays are corrupted and therefore the prediction samples are set to $0.5seconds + (22/2)/samplerate$. The filter returns only the not corrupted part of the signal which result that the prediction sample is still 0.5 seconds. Other parameters will be the same as the determined in section 4.2.2. The normalisation step is also simplified. An overview is given in fig. 4.18. The steps to remove the noise with a threshold value is removed. The implementation with the FIR

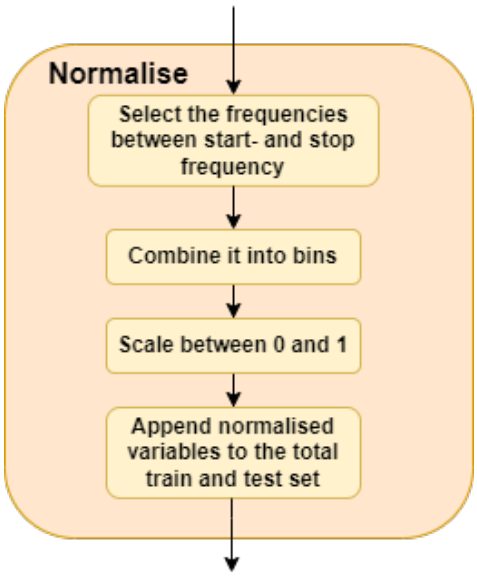

Figure 4.18: Architecture of normalising with FIR filter

filter is realised with the same architecture and setup as where the training samples are sampled with different speeds around the setpoint. For example, the setpoint of one speed is 4300 rounds per minute. The recording is made with setpoints at 4250, 4290, 4300, 4310, 4350 rounds per minute. The results for this case is presented in section 5.5. Also, at a sample rate of 3000 samples per second. To measure the influence of the filter, the case is measured against the case in section 5.5. Because different units of the same type have little differences, the training set recordings is extended with multiple units of the same type and concatenated to each other. The training set differs now from 40 to 60 minutes of an unit with more variation inside a training set. A new estimation of AutoKeras resulted in a complete new architecture. The first hidden layer is grown from 32 to 256 neurons, the second hidden layer is grown from 32 to 128 neurons. Also, each layer is extended with a *BatchNormalization()* and 1 or 2 *Dropout(0.2)* layers in each hidden layer. The *BatchNormalization()* layer normalises the input of the layer. The rate of 0.2 means that neurons during the training are randomly frozen by the rate 0.2, so 20% of the neurons in the layer [4]. The *Dropout(0.2)* doesn't function for testing and predicting and is only used in

the training phase to prevent over-fitting. An overview of the optimal architecture is shown in fig. 4.19.

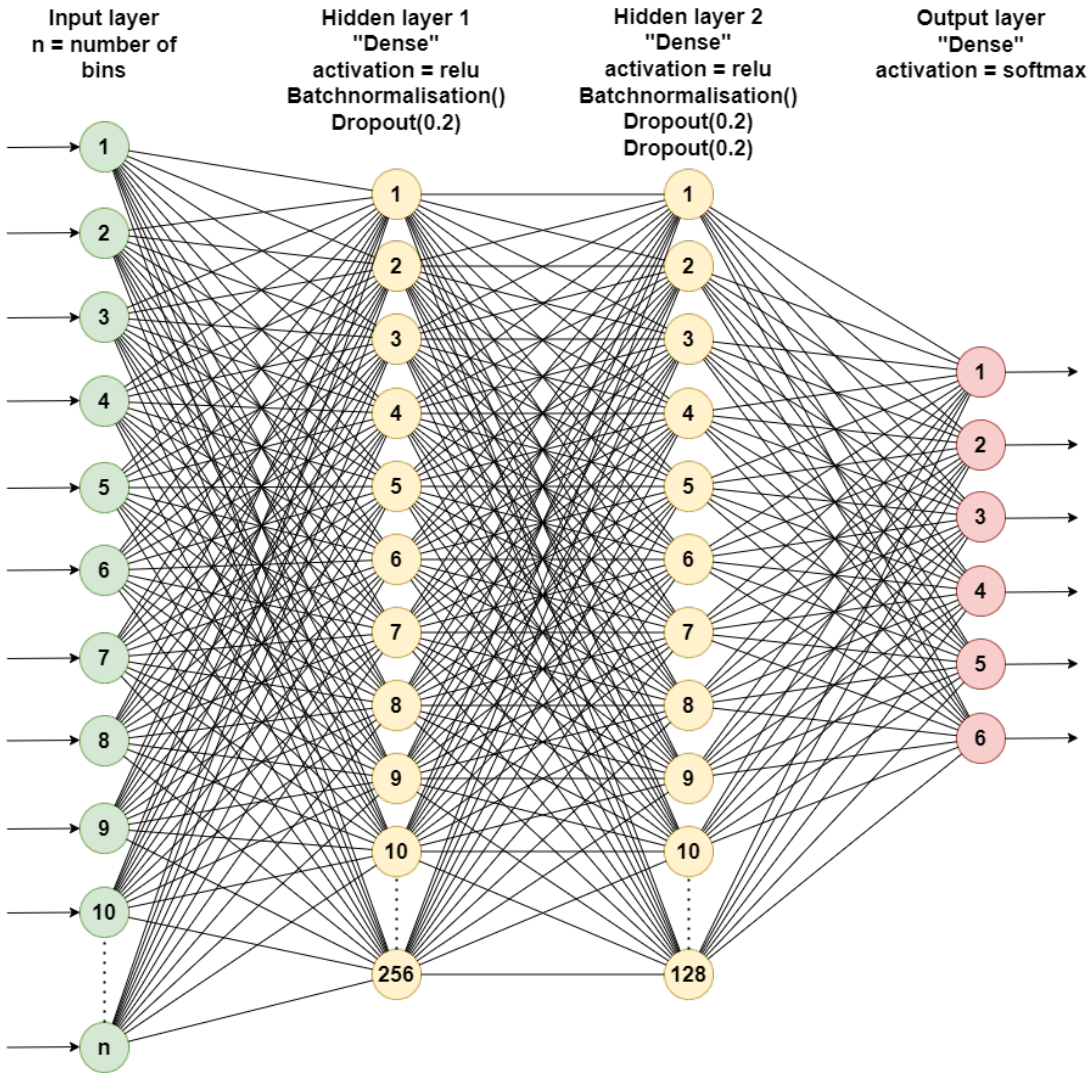

Figure 4.19: Optimal architecture of the Artificial Neural Network

4.3 Important adjustable parameters of the system

In section 4.2.2 is a summary of the parameters in the normalisation step. There are also some variable parameters in the neural network architecture and there are a couple new parameters introduced in section 4.2.3. Some of them are determined and some of them can be adjusted to measure the result. An overview of all the parameters are in table 4.1. The *Parameter* value is the type of variable which can be adjusted. *Section* indicates in which subject it is determined or used. *Value* indicates what the determined value is or what the range is of what it can be.

Parameter	Section	Value
Training and test size	NN Architecture	0.9 and 0.1
Number of epochs	NN Architecture	2500
Number of training samples per unit	NN Architecture	$\geq$2160
Sample time	NN Architecture	0.5 seconds
Sample rate	Normalisation	2000-44100
Interesting frequency band	Normalisation	10 to 800 Hertz
Number of bins	Normalisation	150
Sample time	Normalisation	0.5 seconds
Threshold	Normalisation	0, 0.02, 0.04 & 0.06
Number of taps	FIR	23
Cut-off frequency	FIR	1000 Hertz

Table 4.1: Overview all the parameters which can influence the result

Chapter 5

Results

In this section the results are presented with the help of confusion matrices. These matrices show the accuracy, precision and recall values. Accuracy is calculated as follows $Accuracy = \frac{TP}{Total} * 100\%$, precision is calculated as $Precision = \frac{TP}{TP+predictedFP} * 100\%$ and recall is calculated as $Recall = \frac{TP}{TP+actualFP} * 100\%$. The confusion matrices indicates the precision and recall values of each unit, the accuracy is of the overall model. The accuracy of the system per matrix is highlighted in green, the precision values are highlighted in yellow and the recall values are highlighted in lime. The prediction samples are recordings of 40 seconds divided in to 80 samples, this results in samples of 0.5 seconds each. There is amount of time in the order of a couple weeks between the training and test set and the prediction set. This results that it covers more likely a real-life situation. The units will heat up after a couple minutes while using the product and it produces a different audio spectrum. The environment can differ in practice and will influence the result.

The training and test set is divided in 90% training samples and 10% test samples. The test set gives an accuracy of above 99.5% and a loss below 0.4 in the situations section 5.1, section 5.2 and section 5.3. The situation in section 5.5 gives an accuracy above 99.5% and a loss below 1.6 for the test set. The prediction samples are labelled if it has a confidence level $\geq$ 99.5%, otherwise it is not labelled. In table 5.1 is an overview of the abbreviations shown in the confusion matrices. The results for the prediction set is presented for the influence of background noise in section 5.1. The results at high speed are measured in

Abbreviations	Unit
Si	Silence
Wh	White
MP	MultPrecision
nU	noUnit
Br	Brush
St	Styler
Tr	Trimmer
nL	no label

Table 5.1: Overview of the unit abbreviations

section 5.2. The results where both speeds are mixed and the influence of a lower sample rate are measured in section 5.3.

5.1 Influence of background noise

In the first case is the influence of background noise measured. Two situations with and without background noise are measured at four (0, 0.02, 0.04 and 0.06) different threshold levels. The influence of background noise is measured at slow speed at a sample rate of 44100. It is measured at slow speed because the amplitudes are lower and so the background noise has more influence at slow speed. In the first situation is there less environment noise and the hand is covered around the shaver to mute the environment noise. An overview of the accuracy results is in table 5.2. The accuracy is between 99.3% for a noise threshold value

Situation	Threshold training	Threshold prediction	Accuracy
No background noise	0	0	99.8%
No background noise	0.02	0.02	99.3%
No background noise	0.04	0.04	99.6%
No background noise	0.06	0.06	100%

Table 5.2: Accuracy's without background noise

of 0.02 and 100% for a threshold value of 0.06. Values 0 and 0.04 have an accuracy in between. The best result is shown in a confusion matrix in fig. 5.1, the complete results are in appendix A.1. For this situation, the threshold value has a low influence on the result. In every situation is the accuracy $\geq 99.3\%$.

		Predicted								
		Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Actual	Si	80	0	0	0	0	0	0	0	100%
	Wh	0	80	0	0	0	0	0	0	100%
	MP	0	0	80	0	0	0	0	0	100%
	nU	0	0	0	80	0	0	0	0	100%
	Br	0	0	0	0	80	0	0	0	100%
	St	0	0	0	0	0	80	0	0	100%
	Tr	0	0	0	0	0	0	80	0	100%
	Precision	100%	100%	100%	100%	100%	100%	100%		100%

Figure 5.1: No background noise at threshold level 0.06

The second situation is in a random situation where there can be background noises and no covering filter of the hand. An overview of the accuracy results is in table 5.3. The accuracy drops between the 84.8% for a noise threshold value of 0.02 and 88.9% for a threshold value of 0.06. The best result is shown in a confusion matrix in fig. 5.2, the complete results are in appendix A.2. For this situation, a threshold value of 0.06 has a positive influence

Situation	Threshold training	Threshold prediction	Accuracy
Background noise	0	0	85.9%
Background noise	0.02	0.02	84.1%
Background noise	0.04	0.04	84.8%
Background noise	0.06	0.06	88.9%

Table 5.3: Accuracy's with background noise

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	74	0	0	2	0	10	0	4	92.5%
Wh	0	46	0	0	0	7	0	27	57.5%
MP	0	0	77	0	0	0	0	3	96.3%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	13	64	0	0	3	80%
St	0	0	0	0	0	77	0	3	96.3%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	100%	100%	84.2%	100%	81.9%	100%		88.9%

Figure 5.2: Background noise at threshold level 0.06

on the overall accuracy compared to the situations without a threshold. But, if the individual values of the units are compared to each other, the recall values for the *White* shaver head drops from 100% to 57.5% and for the *MultPrecision* shaver head increases from 52.5% to 96.3% compared to the first situation. All other units are also slightly increased in the recall value. An overview between the precision values of different threshold levels is shown in fig. 5.3. The precision values are the highest for a threshold value of 0.02. The threshold values 0 and 0.06 are comparable. The biggest difference is percentage point of 7.5 for the *Styler* between the threshold values of 0 and 0.02. Where the precision value is high, the false positive predicted unit is mostly predicted as a false negative, so not labelled. That is the reason that the recall value and thus the accuracy isn't increased by a higher threshold value, comparing the threshold values 0 and 0.02. The threshold of 0.06 has comparable precision values but a higher accuracy compared to a threshold value of 0. If both the situations are compared to each other, it is clear to see that background noise influence the result in accuracy, recall and precision. Because the test shaver has no isolation possibilities to mute background noises, the next cases are measured with background noises to give a better view of a real-time implementation.

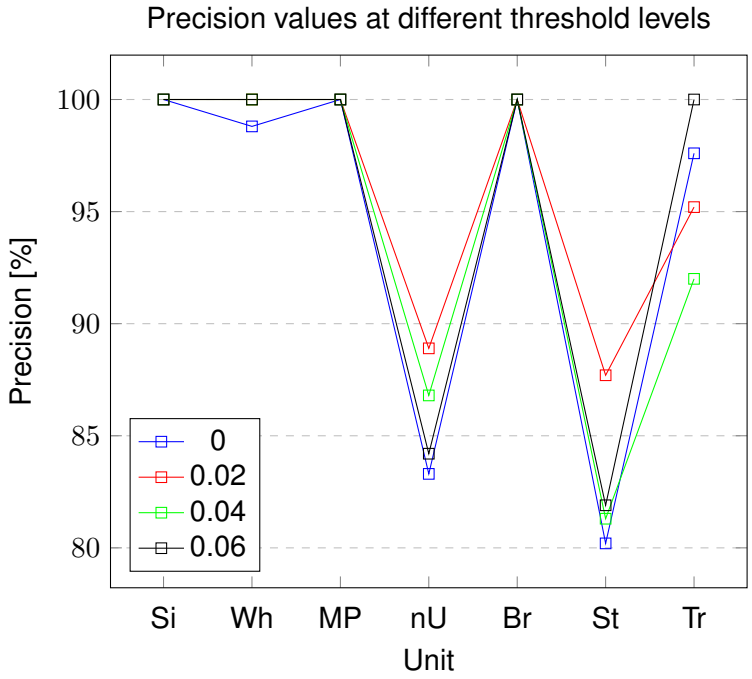

Figure 5.3: Precision values of slow speed at threshold levels 0, 0.02, 0.04 and 0.06, with background noise

5.2 High speed measurement with background noise

The second case is where the shaver is running on high speed. The measurement is measured with background noise and at a sample rate of 44100. An overview of the accuracy results is in table 5.4. The accuracy is 91.3% for a noise threshold value of 0 and 75.9% for a threshold value of 0.06. The best result is shown in a confusion matrix in fig. 5.4, the complete results are in appendix A.3. It is clear to see that the threshold value has not a positive

Situation	Threshold training	Threshold prediction	Accuracy
High speed	0	0	91.3%
High speed	0.02	0.02	89.1%
High speed	0.04	0.04	82.7%
High speed	0.06	0.06	75.9%

Table 5.4: Accuracy's at high speed with background noise

influence on the accuracy when the motor is running at high speed. In a high speed situation is the network less sensitive to noise and the amplitude of the fundamental frequency much bigger. This means that at the moment of setting a threshold value to remove the noise, important data of the unit is already filtered out. An overview between the precision values of different threshold levels is shown in fig. 5.5. Besides the *Brush*, it is clear to see that the threshold values doesn't have a positive result on the precision values. The difference in the

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	79	0	0	0	0	0	0	1	98.8%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	0	74	0	0	6	92.5%
St	0	0	0	10	0	48	0	22	60%
Tr	0	0	0	0	1	0	70	9	87.5%
Precision	100%	100%	100%	88.9%	98.7%	100%	100%		91.3%

Figure 5.4: High speed with background noise at threshold level 0

Brush is 1.3 percent point, which is a small difference.

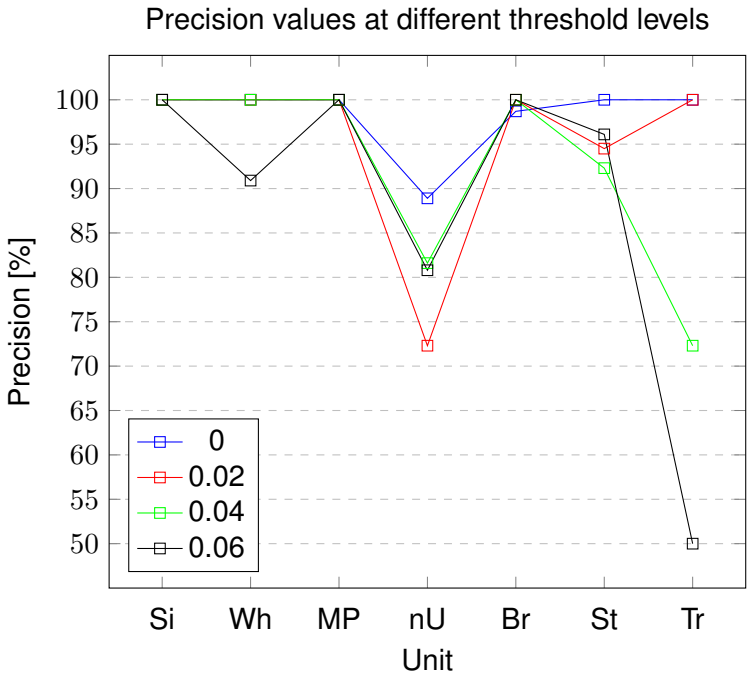

Figure 5.5: Precision values of high speed at threshold levels 0, 0.02, 0.04 and 0.06, with background noise

5.3 Sample rate influence at both speeds

The third case is where both speeds are mixed together. The shaver contains a button change the speed at normal and a higher speed. In practice, an user can switch between the speeds for every unit several times while using the product. Besides the measurement while both speeds are mixed, the influence of lower sample rate at 2000 samples per second is measured. The training set is the training set of slow and high speed added together and then shuffled before the setup is split in a train and test set. The test set to predict the model

Situation	Threshold training	Threshold prediction	Accuracy
Both speeds	0	0	78.9%
Both speeds	0.02	0.02	74.6%
Both speeds	0.04	0.04	70%
Both speeds	0.06	0.06	68.6%

Table 5.5: Accuracy's at both speeds with background noise

for the confusion matrix is a new sample where the shaver is running at slow and high speed randomly, approximate fifty-fifty between slow and high speed. At the moment of switching between the speeds, there is a possibility that the network detects silence instead of the unit. The reason for that is that the motor stops running a small amount of time at the moment of pressing the speed button. The motor speed is switched between the 2 and 4 times every sample. An overview of the accuracy for the first situation is in table 5.5. The accuracy is 68.6% for a noise threshold value of 0.06 and 78.9% for a threshold value of 0. The best result is shown in a confusion matrix in fig. 5.6, the complete results are in appendix A.4.

		Predicted								
		Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Actual	Si	79	0	0	1	0	0	0	0	98.8%
	Wh	1	67	0	0	0	9	0	3	83.8%
	MP	2	0	53	1	0	3	0	21	66.3%
	nU	2	0	1	77	0	0	0	0	96.3%
	Br	1	0	1	32	26	0	11	9	32.5%
	St	1	0	0	0	0	65	5	9	81.3%
	Tr	0	0	1	0	0	0	75	4	93.8%
	Precision	91.9%	100%	94.6%	69.4%	100%	84.4%	82.4%		78.9%

Figure 5.6: Both speeds with background noise at threshold level 0

The threshold has not a positive influence on the accuracy. The *MultPrecision* and *Brush* units is not predicted well, which is also the case in situation 2 in section 5.1. Where the *MultPrecision* is mostly not labelled, is the *Brush* mostly predicted that there is no unit is attached. The FFT spectrum's of the *Brush* and when there is no unit attached is close to each other. The load of the *Brush* is slightly higher and it doesn't produces many vibrations on its own. The *Styler* is not predicted well at high speed situations which result in a lower

Situation	Threshold training	Threshold prediction	Accuracy
Both speeds at sample rate 2000	0	0	79.3%
Both speeds at sample rate 2000	0.02	0.02	66.9%
Both speeds at sample rate 2000	0.04	0.04	67.9%
Both speeds at sample rate 2000	0.06	0.06	61.6%

Table 5.6: Accuracy's at both speeds at a sample rate of 2000

recall value at mixed speed. The different precision values between the higher threshold values are not relevant as the accuracy values of the higher threshold values are lower. The second situation is the same setup measured with a sample rate of 2000 samples per second, as described in section 4.2.2. An overview of the accuracy results is in table 5.6.

The accuracy is between 61.6% for a noise threshold value of 0.06 and 79.3% for a threshold value of 0. The best result is shown in in a confusion matrix in fig. 5.7, the complete results are in appendix A.5. The overall result for a threshold value of 0 is comparable with

		Predicted								
		Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Actual	Si	80	0	0	0	0	0	0	0	100%
	Wh	13	55	1	0	0	0	0	11	68.8%
	MP	9	5	49	0	0	0	0	17	61.3%
	nU	0	0	13	52	10	0	0	5	65%
	Br	0	0	0	0	77	0	0	3	96.3%
	St	5	0	3	0	0	54	11	7	67.5%
	Tr	2	0	1	0	0	0	77	0	96.3%
	Precision	73.4%	91.7%	73.1%	100%	88.5%	100%	87.5%		79.3%

Figure 5.7: Both speeds at threshold level 0 with a sample rate of 2000 samples per second

the situation a sample rate of 44100. There is a difference in recall and precision values. In fig. 5.8 is an overview of the recall values between the situation with a sample rate of 44100 and 2000. Only, the recall values for the *Brush* and when then there is no unit attached are switched. Switched in a way that the *Brush* is well predicted in situation 4 (sample rate

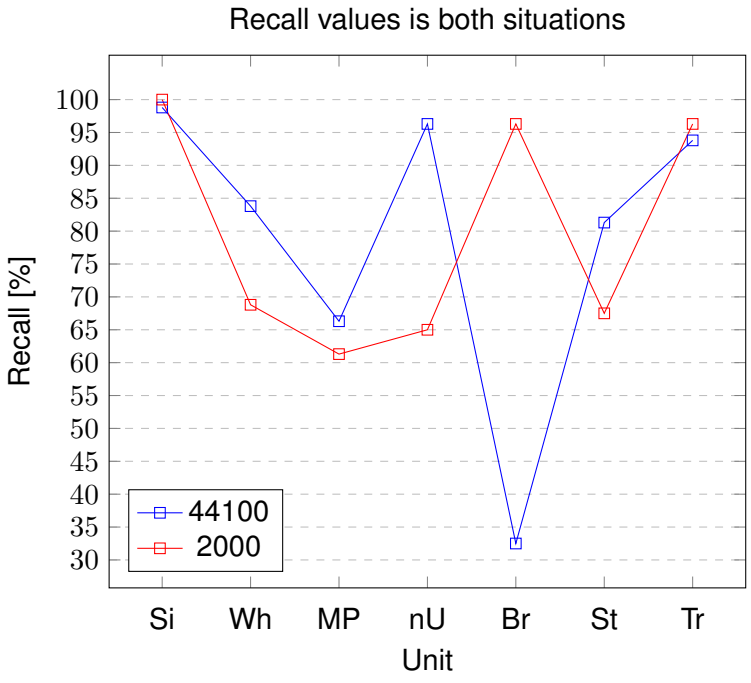

Figure 5.8: Recall values in the situations with 44100 and 2000 samples per second

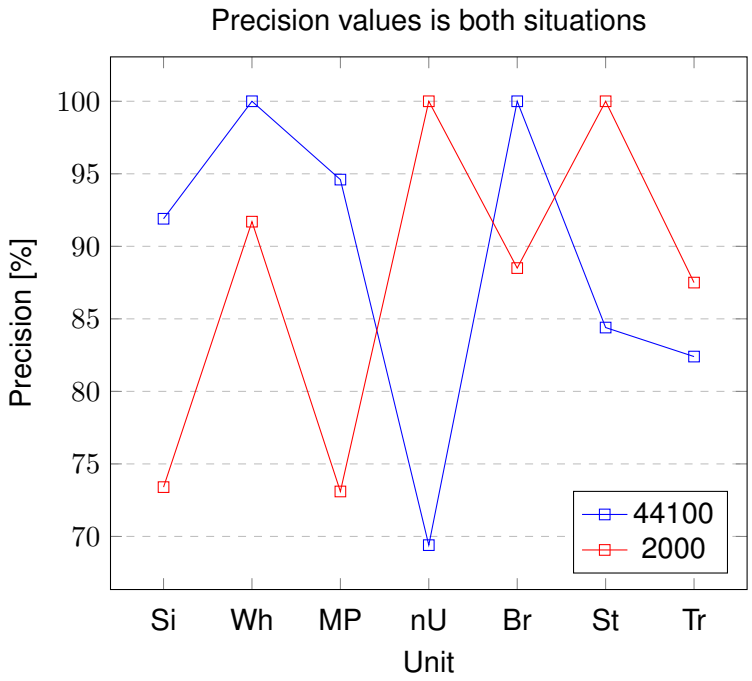

Figure 5.9: Precision values in the situations with 44100 and 2000 samples per second

44100) and when there is no unit attached is well predicted in situation 5 (sample rate 2000). The rest of the units are comparable which result in a comparable accuracy. This structure comes back in the precision values shown in fig. 5.9. The *Brush* and when there is no unit attached is again switched. Another relation between the two is that when one unit is well predicted, so a high recall value, the precision value is low. This is because the other unit is false predicted as the unit with a high recall value. The cause of this is the close FFT analysis of both units which is close to each other in the fundamental and dominant frequencies. Overall is the system is comparable as at a sample rate of 44100 and the system can predict at same level at a lower sample rate.

5.4 Influence of noise

An overview of the accuracy results of every situation and cases is given in table 5.7. There are 5 different situations trained and predicted. Every situation is trained on different threshold levels for noise to measure the influence. The threshold value where the model is trained on is the same value as where the model has been predicted. It is clear to see that it has low influence to the situations at slow speed. For the situations at high and mixed speed is the accuracy even lower compared to the situations at a threshold value of 0.

A continuation for this step is further worked out. A system at mixed speed is trained without a noise threshold value, so a value of 0. The prediction is realised with different noise threshold values. The results of this system has not a positive influence on the accuracy.

A version where Gaussian noise is added to training samples is also realised. Different models are tested and predicted from a standard deviation of 1 to a standard deviation of

Situation	Threshold training	Threshold prediction	Accuracy
No background noise	0	0	99.8%
No background noise	0.02	0.02	99.3%
No background noise	0.04	0.04	99.6%
No background noise	0.06	0.06	100%
Background noise	0	0	85.9%
Background noise	0.02	0.02	84.1%
Background noise	0.04	0.04	84.8%
Background noise	0.06	0.06	88.9%
High speed	0	0	91.3%
High speed	0.02	0.02	89.1%
High speed	0.04	0.04	82.7%
High speed	0.06	0.06	75.9%
Both speeds	0	0	78.9%
Both speeds	0.02	0.02	74.6%
Both speeds	0.04	0.04	70%
Both speeds	0.06	0.06	68.6%
Both speeds at sample rate 2000	0	0	79.3%
Both speeds at sample rate 2000	0.02	0.02	66.9%
Both speeds at sample rate 2000	0.04	0.04	67.9%
Both speeds at sample rate 2000	0.06	0.06	61.6%

Table 5.7: Summary of the accuracy's in every situation

200. Step-by-step is the standard deviation increased with a mean defined at 0. The results of the systems has also not a positive influence on the accuracy.

5.5 Situation with training samples recorded around motor setpoint

In the previous situations are many false positives predicted in a situation where the fundamental frequency is in the same bin or frequency range of a dominant frequency of the average training set of the false predicted unit. To improve the accuracy for this situations, a training set is recorded with different speeds around the setpoint of the controller. For example, the setpoint of one speed is 4300 rounds per minute. The recording is made with setpoints at 4250, 4290, 4300, 4310, 4350 rounds per minute. All mixed in one recording of 20 minutes, which is the same length of recording in the previous situations. For some units is a concatenation realised with another recording of a different unit but which must be labelled the same. For example, two different *MultPrecision* units have both a recording of 20 minutes which is concatenated to one sample of 40 minutes. Both units are also recorded inside the prediction set which result in a prediction set of 2 times 40 seconds. An overview

of the recordings of the train/test and prediction set of each unit is in table 5.8. The sam-

Unit	Number of units	Training/test set	Prediction set
White	3	60 minutes	120 seconds
MultPrecision	2	40 minutes	80 seconds
noUnit	1	20 minutes	40 seconds
Brush	2	40 minutes	80 seconds
Styler	2	40 minutes	80 seconds
Trimmer	2	40 minutes	80 seconds

Table 5.8: Summary of the training/test and prediction set

ple rate is set to 3000 to create extra space in case the important frequency information is higher and there is more space to create a FIR filter to de-noise the signal. In this situation is *Silence* not trained with the situation. At the moment that the motor is not running, the motor current is 0mA which is the same situation at the moment of *Silence*. By removing *Silence* from the model, the overall accuracy is lower because of the high recall value of *Silence*. As concluded in section 5.4, a threshold value of 0 has the best performance. Therefore, a new situation, without a removing noise with a threshold system, is trained and tested with a set concluded in table 5.8 and shown in a confusion matrix in fig. 5.10. The overall accuracy is

		Predicted							
		Wh	MP	nU	Br	St	Tr	nL	**Recall**
Actual	Wh	185	49	0	0	0	0	8	77.1%
	MP	34	116	0	0	0	0	10	72.5%
	nU	1	0	79	0	0	0	0	98.8%
	Br	1	0	0	143	0	0	16	89.4%
	St	3	0	1	0	152	0	5	95%
	Tr	0	0	0	14	0	125	21	78.3%
	Precision	82.6%	54%	98.8%	91.1%	100%	100%		83.3%

Figure 5.10: Both speeds with training variation around the setpoint

increased from 79.3% in the previous situation in section 5.3 to 83.3% in the new situation. The same recordings are used to train, test and predict a network in combination with a FIR filter as explained in section 4.2.3. The results are shown in fig. 5.11.

The overall accuracy is increased from 83.3% in situation without a FIR filter to 91.1% in the new situation. The biggest differences in recall values is in the *MultPrecision* and *Trimmer* units. In one of the two recording of the *Trimmer* a truck passed by in the end of the recording. In the situation without the FIR filter was the unit often not labelled. The unit is often correct labelled in the situation with the FIR filter. A graph is shown in fig. 5.12 to compare the precision values between the situation in section 5.3, the situation without a FIR filter and the situation with a FIR filter. Besides that the accuracy is significant increased in the situation with a FIR filter, the precision values are also improved. Also, the values are more stable and in 5 of the 6 situations $>$90%. Notice that the precision values with a

Actual \ Predicted	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Wh	195	37	0	0	0	0	8	81.3%
MP	6	145	0	0	0	0	9	90.6%
nU	0	1	78	0	0	0	1	97.5%
Br	0	0	1	149	0	0	10	93%
St	1	0	0	0	154	0	5	96.3%
Tr	0	0	0	0	0	154	6	96.3%
Precision	96.5%	79.2%	98.7%	100%	100%	100%		91.1%

Figure 5.11: Both speeds with training variation around the setpoint with a FIR filter

varying setpoint without a FIR filter decreased for two units, is comparable for three units and is improved for one unit compared two the situation in section 5.3.

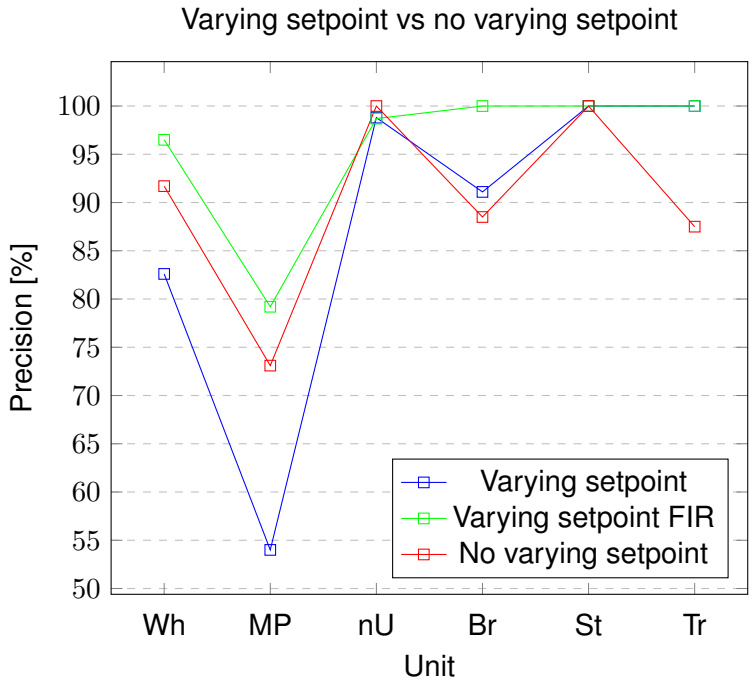

Figure 5.12: Varying setpoint precision values vs no varying setpoint in section 5.3 at a sample rate of 2000

5.6 Combined units and embedded system performance

The execution and measurements in previous situations are evaluated at PC level. To give an impression of the memory and execution performance on a comparable embedded device, the models are evaluated on a Nucleo-F401RE developing board of ST. The processor is set to a clock frequency of 64MHz. The clock frequency and processor are the speed and architecture which is the same or comparable with the current high-end shaver handles. The

STM32CubeMX environment contains a *STM32Cube.AI* extension to include, measure and evaluate a Keras model with a *.h5* extension. The previous results are using the same model format. The extension has the support to analyse and generate C++ code which can be included as a library in the project, created in the *IAR EW* environment. The developing board is programmed with the generated C++ code in the *IAR EW* environment and evaluated with the *STM32CubeMX* application.

The measurements are realised with a small adjustment in the dataset. Both the *White* and *MultPrecision* shaver head are combined to same label, namely, a *Shaverhead*. Also, *Brush* and *noUnit* are combined to 1 label, namely, *BrushOrLess*. The last adjustment is that there is added a new unit, which is named a *Bodyshaver*. The overall average accuracy performance of all the units for different models is shown in fig. 5.13. The complete

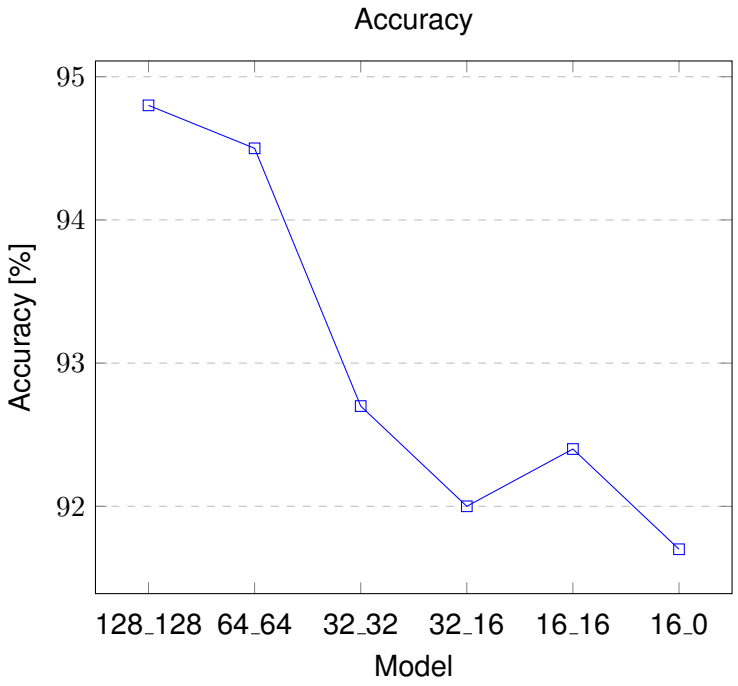

Figure 5.13: Overall accuracy of the different models

results presented in confusion matrices are shown in appendix A.6. The models vary in the size of the hidden layers, starting with two hidden layers of 128 neurons each. Hereafter, the performance with two hidden layers of 64 neurons each. This continues until a network with one hidden layer of 16 neurons. The training, testing and prediction set is the same for every model. As shown in the figure, the accuracy drops at the moment the number of neurons decrease. The different models are shown, so there can be a balance consideration between accuracy, run-time and flash memory. In fig. 5.14 is an overview of the run-time of the different models. Only the run-time of the models, so without the time of filtering, FFT and normalisation. The run-time from the model of the previous results, indicated with 256_128, is added to compare those results. As expected, the run-time decreases if the model becomes smaller. It starts from approximate 11.03 milliseconds to approximate 0.43 milliseconds. Compared to the sample time of 0.5 seconds, which will be stored in a buffer,

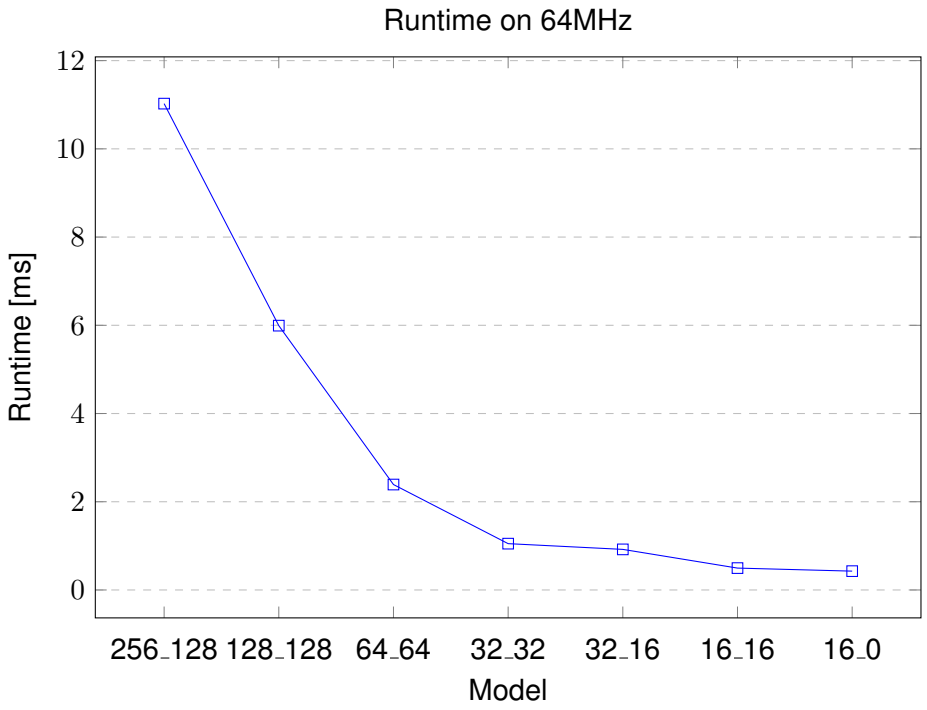

Figure 5.14: Run-time of the different models

there is in all situations enough time to calculate the answer. Other, non-determined requirements so far, can influence the definition of a maximum run-time. In fig. 5.15 is overview of the necessary flash memory of the different models. Like the run-time, the flash memory

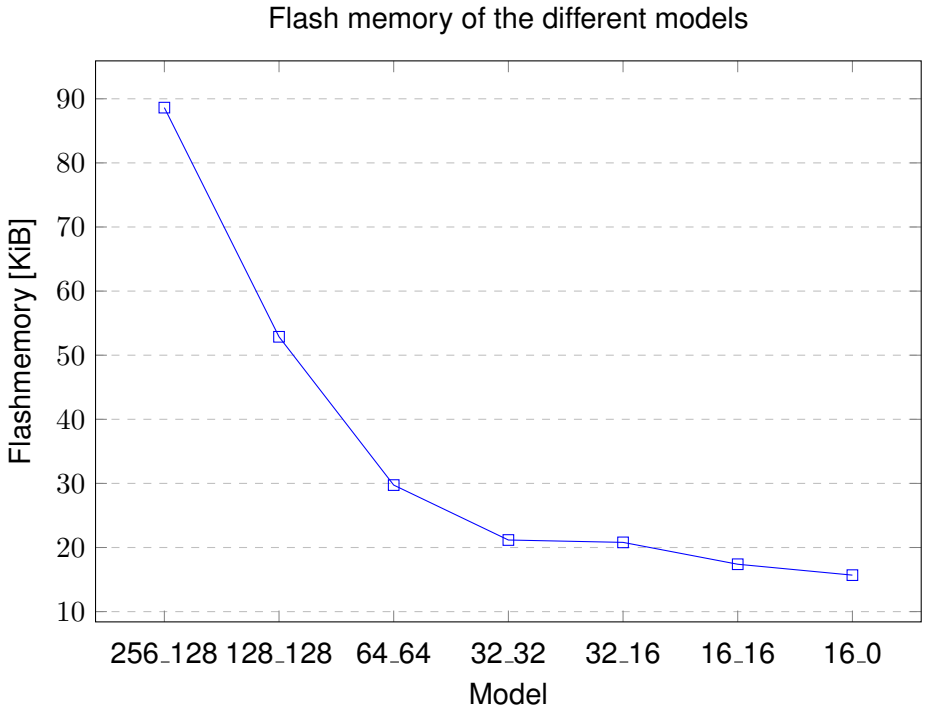

Figure 5.15: Flash memory of the different models

decreases at the moment the model is smaller. It has a size of 88.62 kilobytes for the largest model and it shrinks down to 16 kilobytes for the smallest model. As the board has a flash memory of 512 kilobytes, all the models will easily fit including all additional and current software. As the options for the shaver controllers have between 512 and 2048 kilobytes of flash memory, it will also fit in the shaver controllers.

Overall, the necessary memory and run-time for all the models is, respectively, small and fast enough to implement it on the high-end shaver controllers. There could be chosen to implement a smaller model to save memory space en decrease the run-time of the model but that will reduce the accuracy of the system.

Chapter 6

Conclusions and recommendations

This chapter presents answers and concludes the main- and sub-questions asked in chapter 1. In section 6.2 is a discussion over what can be improved next and which could be done better.

6.1 Conclusions

Before the main research question can be answered, the sub-questions need to be answered first. The first sub-question which is asked is:

- Which method of ML is suitable to detect features out of audio?

In section 3.1 are different solutions presented with different types of ML algorithms. The two main types of NN which is presented is a CNN and an ANN. The solution which is more suitable for this purpose was the solution with a feed-forward ANN. As presented in the chapter 5, different datasets and normalisation steps are tried to measure the result of the ANN. Labelled units need to be recognized by the network and otherwise the unit must not be labelled. The final results show an overall accuracy of 91.1%. This means that there are samples which are true or false negative or false positive predicted. As stated, the network must predict the sample true positive if it is the unit and otherwise a true negative if it isn't the unit. The precision values indicates the rate between true and false positives for unit, which isn't always 100% and with the lowest value of 79.2% in the network with the highest accuracy. Hereby, there can be stated that the network predicts an unit as a true positive, is with a prediction score between 79.2% and 100%.

The network is not retraining the network by prediction samples. The method which is used and is suitable to detect features or units is a supervised feed-forwarded ANN. The network consist out of an input layer, two hidden layers and one output layer. The raw samples are made suitable for the input layer of the ANN which is asked by the second sub-question:

- How to make raw audio a suitable input for the ML method?

The raw audio samples are filtered and normalised before the sample is fed to the ANN. In section 3.3 is a method proposed to reduce noise in the audio with a FIR filter which

improved the accuracy, precision and recall values. The information of the audio for an ANN is in the frequency domain. Therefore, the sample is calculated by a FFT. The response is normalised by selecting only the interesting frequencies, combining frequencies into bins and scale the complex amplitudes between 0 and 1, as presented in section 4.2.3. The system need to be measured which state to the next sub-question:

- Which features can be detected by the method in terms of accuracy, precision and recall?

There are six labelled features measured in terms of accuracy, precision and recall values. Namely, the *White* shaverhead, the *MultPrecision* shaverhead, a *Styler*, a *Trimmer*, a *Brush* and when there is no unit attached. There is an overall accuracy achieved of 91.1% in a real situation. There is a precision value achieved of over 96.5% for 5 out of 6 units. Only, the *MultPrecision* unit achieved a precision value of 79.2%. The recall values vary between 81.3% and 97.5%. The 81.3% is for the *White* shaverhead. The false positives are mostly a *MultPrecision* shaverhead which are both shaverheads. The main question was stated as:

- Which features can be recognized or detected with sound inside a shaver using ML approaches?

The features which are recognized by the proposed method with a FIR filter are 6 labelled units, e.g. unit recognition. The overall combination of all units are recognized with an accuracy of 91.1%. The *White* unit is recognized with a recall value of 81.3%. The *MultPrecision* unit is recognized with a recall value of 90.6%. The *Brush* unit is recognized with a recall value of 93%. The other units are recognized with a recall value $\geq$ 96.3%.

The model is evaluated on the feasibility on a embedded controller which has the same architecture and speed as the controllers in the shavers. A flash memory of 88.62 kilobytes will fit inside the controller. A run-time of 11.03 milliseconds will easily fit for the calculation for the 0.5 second sample which takes 0.5 seconds to record in the buffer.

To conclude, unit recognition based on sound with a supervised feed-forwarded artificial neural network can achieve an overall accuracy of 91.1% with the presented method. A filter to reduce noise prevents that units not will be detected in the case of a noisy background. Normalisation limited the size of the input layer in a fixed size and compensates for small motor changes. Shuffling of the training samples in the normalisation step, shuffling of the training samples between each epoch and the dropout layers prevents over-fitting of the model.

6.2 Recommendations

If the research will be repeated in the future, attention to noisy, no noisy and echo environments should be considered. It could be normalised, trained within the network or both options should be considered. The train, test and prediction set should be recorded in the different environments with the proposed methods to solve the issue.

The microphones can be placed closer towards the units inside the shaverhead. This can result that the vibrations of units are clearer and noise has less influence to the result. Besides that, the environment can be isolated a bit further that the vibrations has more influence to result instead of the noise.

The method of filtering the audio to reduce the noise can also be more explored. A different digital filter or an integrated hardware filter can be explored to measure the influence towards the result.

The measurements are realised with no load, in other words, the product is not used while shaving, brushing, styling or trimming. The first result while brushing looks very promising ($\geq 99\%$) but other units are not explored yet. Unit recognition while using the unit can be explored further on. A recommendation would be to train the model with samples while using the product in different situations.

So far, six units are trained and measured by the model. The system can be extended by for example a nose trimmer or a body shaver. The two types of shaverheads can be combined to one as it are both shaverheads. Also, the system can be further explored for other personal care embedded devices.

The sensitivity can be explored further for the case if the product is mass produced or the unit will wear out. Every device, microphone and unit can have a little tolerance which result in a different frequency spectrum. A recommendation is to record and train as many units and devices to prevent that a model will not recognize an unit, but is not explored yet.

Bibliography

[1] I. Sarker, "Machine learning: Algorithms, real-world aplications and reasearch directions," in *SN Computer Science*, 2021, accessed: 2022-2-23. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x

[2] A. Khan, K. Chui, and D. Peraković, "Future scope of ai and machine learning in 2022," 2021, accessed: 2022-2-23. [Online]. Available: https://insights2techinfo.com/future-scope-of-machine-learning-and-ai-in-2022

[3] J. McCarthy, "What is artificial intelligence?" Computer Science Department, Stanford University, 11 2004, accessed: 2021-10-14. [Online]. Available: https://homes.di.unimi.it/borghese/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf

[4] F. Chollet *et al.*, "Keras," https://keras.io, 2015, accessed: 2022-2-23.

[5] N. Chauhan, A. K. Bhatt, R. K. Dwivedi, and R. Belwal, "Accuracy testing of data classification using tensor flow a python framework in ann designing," in *2018 International Conference on System Modeling Advancement in Research Trends (SMART)*, 2018, pp. 44–48.

[6] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1946–1956.

[7] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *CoRR*, vol. abs/1206.5533, 2012, accessed: 2022-2-23. [Online]. Available: http://arxiv.org/abs/1206.5533

[8] B. Chandu, A. Munikoti, K. S. Murthy, G. Murthy V., and C. Nagaraj, "Automated bird species identification using audio signal processing and neural networks," in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, 2020, pp. 1–5.

[9] N. Bold, C. Zhang, and T. Akashi, "Bird species classification with audio-visual data using cnn and multiple kernel learning," in *2019 International Conference on Cyberworlds (CW)*, 2019, pp. 85–88.

[10] S. Agarwal, K. Khatter, and D. Relan, “Security threat sounds classification using neural network,” in *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2021, pp. 690–694.

[11] R. Avanzato, F. Beritelli, F. Di Franco, and V. F. Puglisi, “A convolutional neural networks approach to audio classification for rainfall estimation,” in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, 2019, pp. 285–289.

[12] P. Khunarsal, C. Lursinsap, and T. Raicharoen, “Very short time environmental sound classification based on spectrogram pattern matching,” *Information Sciences*, vol. 243, pp. 57–74, 2013, accessed: 2022-2-23. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025513003113

[13] S. Abdoli, P. Cardinal, and A. Lameiras Koerich, “End-to-end environmental sound classification using a 1d convolutional neural network,” *Expert Systems with Applications*, vol. 136, pp. 252–263, 2019, accessed: 2022-2-23. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419304403

[14] M. N. Sabab, M. A. R. Chowdhury, S. M. M. I. Nirjhor, and J. Uddin, “Bangla speech recognition using 1d-cnn and lstm with different dimension reduction techniques,” in *Emerging Technologies in Computing*, M. H. Miraz, P. S. Excell, A. Ware, S. Soomro, and M. Ali, Eds. Cham: Springer International Publishing, 2020, pp. 158–169.

[15] D. Dash, P. Ferrari, D. Heitzman, and J. Wang, “Decoding speech from single trial meg signals using convolutional neural networks and transfer learning,” in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2019, pp. 5531–5535.

[16] S. Tangkawanit, C. Pinthong, and S. Kanprachar, “Development of gunfire sound classification system with a smartphone using ann,” in *2018 International Conference on Digital Arts, Media and Technology (ICDAMT)*, 2018, pp. 168–172.

[17] S. Tangkawanit and S. Kanprachar, “Spectral vector design for gunfire sound classification system with a smartphone using ann,” in *2018 21st International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2018, pp. 421–426.

[18] F. Rios-Gutierrez, R. Alba-Flores, and S. Strunic, “Recognition and classification of cardiac murmurs using ann and segmentation,” in *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, 2012, pp. 219–223.

[19] R. Velik, “Discrete fourier transform computation using neural networks,” in *2008 International Conference on Computational Intelligence and Security*, vol. 1, 2008, pp. 120–123.

[20] W. Keerthipala, L. T. Chong, and T. C. Leong, “Artificial neural network model for analysis of power system harmonics,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 2, 1995, pp. 905–910 vol.2.

[21] M. A. Hossin, M. Shil, V. Thanh, and N. T. Son, “An adjustable window-based fir filter and its application in audio signal de-noising,” in *2018 3rd International Conference on Robotics and Automation Engineering (ICRAE)*, 2018, pp. 248–252.

[22] M. Shil, H. Rakshit, and H. Ullah, “An adjustable window function to design an fir filter,” in *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, 2017, pp. 1–5.

[23] P. Virtanen *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.

[24] H. Boche and V. Pohl, “Spectral factorization, whitening- and estimation filter - stability, smoothness properties and fir approximation behavior,” in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, 2005, pp. 1701–1705.

[25] J. Jeong, “Effect of minimum phase whitening filter in adaptive beamforming structure on fluctuating acoustic signal,” in *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 1162–1165.

[26] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, accessed: 2022-2-23. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

Appendix A

Confusion matrix results in different situations

In the sections below is an extensive overview of all the confusion matrices which are not shown in chapter 5.

A.1 Situation at slow speed without background noise

In figures A.1 to A.4 is an overview of the four confusion matrices in the situation without background noises, presented in section 5.1. The matrices give a detailed overview of the test results in this situation.

		Predicted								
		Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Actual	Si	79	0	0	0	0	0	0	1	98.8%
	Wh	0	80	0	0	0	0	0	0	100%
	MP	0	0	80	0	0	0	0	0	100%
	nU	0	0	0	80	0	0	0	0	100%
	Br	0	0	0	0	80	0	0	0	100%
	St	0	0	0	0	0	80	0	0	100%
	Tr	0	0	0	0	0	0	80	0	100%
	Precision	100%	100%	100%	100%	100%	100%	100%		99.8%

Figure A.1: Ideal situation at slow speed, threshold 0

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	80	0	0	0	0	0	0	0	100%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	0	80	0	0	0	100%
St	0	0	0	0	0	76	0	4	95%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	100%	100%	100%	100%	100%	100%		99.3%

Figure A.2: Ideal situation at slow speed, threshold 0.02

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	80	0	0	0	0	0	0	0	100%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	0	80	0	0	0	100%
St	0	0	0	0	0	78	0	2	97.5%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	100%	100%	100%	100%	100%	100%		99.6%

Figure A.3: Ideal situation at slow speed, threshold 0.04

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	80	0	0	0	0	0	0	0	100%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	0	80	0	0	0	100%
St	0	0	0	0	0	80	0	0	100%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	100%	100%	100%	100%	100%	100%		100%

Figure A.4: Ideal situation at slow speed, threshold 0.06

A.2 Situation at slow speed with background noise

In figures A.5 to A.8 is an overview of the four confusion matrices in the situation background noises, presented in section 5.1. The matrices give a detailed overview of the test results in this situation.

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	71	0	0	0	0	6	0	3	88.8%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	42	0	0	11	0	27	52.5%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	16	59	0	0	5	73.8%
St	0	1	0	0	0	69	2	8	86.3%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	98.8%	100%	83.3%	100%	80.2%	97.6%		85.9%

Figure A.5: Random situation at slow speed, threshold 0

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	64	0	0	1	0	9	0	6	80%
Wh	0	66	0	0	0	0	0	14	82.5%
MP	0	0	49	0	0	1	0	30	61.3%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	9	61	0	3	7	76.3%
St	0	0	0	0	0	71	1	8	88.8%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	100%	100%	88.9%	100%	87.7%	95.2%		84.1%

Figure A.6: Random situation at slow speed, threshold 0.02

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	60	0	0	1	0	14	0	5	75%
Wh	0	57	0	0	0	3	0	20	71.3%
MP	0	0	77	0	0	0	0	3	96.3%
nU	0	0	0	79	0	0	0	1	98.8%
Br	0	0	0	11	48	0	5	16	60%
St	0	0	0	0	0	74	2	4	92.5%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	100%	100%	86.8%	100%	81.3%	92%		84.8%

Figure A.7: Random situation at slow speed, threshold 0.04

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	74	0	0	2	0	10	0	4	92.5%
Wh	0	46	0	0	0	7	0	27	57.5%
MP	0	0	77	0	0	0	0	3	96.3%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	13	64	0	0	3	80%
St	0	0	0	0	0	77	0	3	96.3%
Tr	0	0	0	0	0	0	80	0	100%
Precision	100%	100%	100%	84.2%	100%	81.9%	100%		88.9%

Figure A.8: Random situation at slow speed, threshold 0.06

A.3 Situation at high speed

In figures A.9 to A.12 is an overview of the four confusion matrices in this situation, presented in section 5.2. The matrices give a detailed overview of the test results in this situation.

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	79	0	0	0	0	0	0	1	98.8%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	0	74	0	0	6	92.5%
St	0	0	0	10	0	48	0	22	60%
Tr	0	0	0	0	1	0	70	9	87.5%
Precision	100%	100%	100%	88.9%	98.7%	100%	100%		91.3%

Figure A.9: Random situation at high speed, threshold 0

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	69	0	0	0	0	3	0	8	86.3%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	7	65	0	0	8	81.3%
St	0	0	0	23	0	52	0	5	65%
Tr	0	0	0	0	0	0	73	7	91.3%
Precision	100%	100%	100%	72.3%	100%	94.5%	100%		89.1%

Figure A.10: Random situation at high speed, threshold 0.02

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	76	0	0	0	0	2	0	2	95%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	18	55	0	0	7	68.8%
St	0	4	0	0	0	24	26	26	30%
Tr	0	0	0	0	0	0	68	12	85%
Precision	100%	100%	100%	81.6%	100%	92.3%	72.3%		82.7%

Figure A.11: Random situation at high speed, threshold 0.04

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	76	0	0	0	0	2	0	2	95%
Wh	0	80	0	0	0	0	0	0	100%
MP	0	0	80	0	0	0	0	0	100%
nU	0	0	0	80	0	0	0	0	100%
Br	0	0	0	19	57	0	0	4	71.3%
St	0	8	0	0	0	49	3	20	61.3%
Tr	0	0	0	0	0	0	3	77	3.8%
Precision	100%	90.9%	100%	80.8%	100%	96.1%	50%		75.9%

Figure A.12: Random situation at high speed, threshold 0.06

A.4 Both speeds

In figures A.13 to A.16 is an overview of the four confusion matrices in this situation, presented in section 5.3. The matrices give a detailed overview of the test results in this situation.

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	79	0	0	1	0	0	0	0	98.8%
Wh	1	67	0	0	0	9	0	3	83.8%
MP	2	0	53	1	0	3	0	21	66.3%
nU	2	0	1	77	0	0	0	0	96.3%
Br	1	0	1	32	26	0	11	9	32.5%
St	1	0	0	0	0	65	5	9	81.3%
Tr	0	0	1	0	0	0	75	4	93.8%
Precision	91.9%	100%	94.6%	69.4%	100%	84.4%	82.4%		78.9%

Figure A.13: Random situation at both speeds, threshold 0

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	79	0	0	1	0	0	0	0	98.8%
Wh	1	69	0	0	0	7	0	3	86.3%
MP	2	0	54	0	0	10	0	14	67.5%
nU	2	0	1	76	0	0	0	1	95%
Br	1	0	0	33	0	0	37	9	0%
St	1	0	0	0	0	67	5	7	83.8%
Tr	2	0	0	0	0	0	73	5	91.3%
Precision	89.8%	100%	98.2%	69.1%	–	79.8%	63.5%		74.6%

Figure A.14: Random situation at both speeds, threshold 0.02

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	80	0	0	0	0	0	0	0	100%
Wh	1	40	0	0	4	26	0	9	50%
MP	0	0	45	2	0	21	0	12	56.3%
nU	2	0	1	77	0	0	0	0	96.3%
Br	0	0	2	52	13	0	2	11	16.3%
St	5	0	0	0	0	68	4	3	85%
Tr	2	0	5	0	0	0	69	4	86.3%
Precision	88.9%	100%	84.9%	58.8%	76.5%	59.1%	92.1%		70%

Figure A.15: Random situation at both speeds, threshold 0.04

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	80	0	0	0	0	0	0	0	100%
Wh	1	64	0	0	0	7	0	8	80%
MP	2	0	9	17	0	41	0	11	11.3%
nU	3	0	0	77	0	0	0	0	96.3%
Br	0	12	0	45	16	0	0	7	20%
St	0	2	2	0	0	65	4	7	81.3%
Tr	4	0	0	0	0	0	73	3	91.3%
Precision	88.9%	82.1%	81.8%	55.4%	100%	57.5%	94.8%		68.6%

Figure A.16: Random situation at both speeds, threshold 0.06

A.5 Both speeds at a sample rate of 2000 samples per second

In figures A.17 to A.20 is an overview of the four confusion matrices in this situation, presented in section 5.3. The matrices give a detailed overview of the test results in this situation.

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	80	0	0	0	0	0	0	0	100%
Wh	13	55	1	0	0	0	0	11	68.8%
MP	9	5	49	0	0	0	0	17	61.3%
nU	0	0	13	52	10	0	0	5	65%
Br	0	0	0	0	77	0	0	3	96.3%
St	5	0	3	0	0	54	11	7	67.5%
Tr	2	0	1	0	0	0	77	0	96.3%
Precision	73.4%	91.7%	73.1%	100%	88.5%	100%	87.5%		79.3%

Figure A.17: Random situation at both speeds at sample rate 2000, threshold 0

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	65	0	0	0	10	0	0	5	81.3%
Wh	20	32	6	4	0	4	0	14	40%
MP	39	0	28	7	0	0	0	6	35%
nU	2	0	0	57	18	0	0	3	71.3%
Br	0	1	0	0	77	0	0	2	96.3%
St	7	3	2	0	0	40	15	13	50%
Tr	3	0	0	0	0	0	76	1	95%
Precision	47.8%	88.9%	77.8%	83.8%	73.3%	90.9%	83.5%		66.9%

Figure A.18: Random situation at both speeds at sample rate 2000, threshold 0.02

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	79	0	0	0	1	0	0	0	98.8%
Wh	28	18	0	0	0	17	0	17	21.9%
MP	37	0	39	0	0	0	0	4	48.8%
nU	2	0	0	52	24	0	0	2	65%
Br	1	0	0	0	77	0	0	2	96.3%
St	3	0	0	0	12	39	20	6	48.8%
Tr	3	0	0	0	0	0	76	1	95%
Precision	51.6%	100%	100%	100%	67.5%	69.6%	79.2%		67.9%

Figure A.19: Random situation at both speeds at sample rate 2000, threshold 0.04

Actual \ Predicted	Si	Wh	MP	nU	Br	St	Tr	nL	**Recall**
Si	75	0	0	0	3	0	0	2	93.8%
Wh	12	21	0	3	0	25	0	19	26.3%
MP	15	1	2	41	0	2	0	19	0.03%
nU	2	0	0	52	12	0	11	3	65%
Br	0	1	0	0	79	0	0	0	98.8%
St	10	1	0	0	0	43	13	13	53.8%
Tr	2	0	0	0	0	4	73	1	91.3%
Precision	64.7%	87.5%	100%	54.2%	84%	58.1%	75.3%		61.6%

Figure A.20: Random situation at both speeds at sample rate 2000, threshold 0.06

A.6 Combined units and embedded system performance

In figures A.21 to A.26 is an overview of the four confusion matrices in this situation, presented in section 5.6. The matrices give a detailed overview of the test results in this situation.

Actual \ Predicted	SH	Br	St	Tr	BS	nL	**Recall**
SH	222	10	0	0	0	8	92.5%
Br	1	239	0	0	0	0	99.6%
St	1	0	137	0	19	3	85.6%
Tr	1	0	0	159	0	0	99.4%
BS	4	0	0	0	153	3	95.6%
Precision	96.9%	96%	100%	100%	89%		94.8%

Figure A.21: Situation 128 128

Actual \ Predicted	SH	Br	St	Tr	BS	nL	**Recall**
SH	218	11	0	0	0	11	90.8%
Br	1	239	0	0	0	0	99.6%
St	0	0	133	0	22	5	83.1%
Tr	0	0	0	159	0	1	99.4%
BS	2	0	0	0	158	0	98.8%
Precision	98.6%	95.6%	100%	100%	87.8%		94.5%

Figure A.22: Situation 64 64

Actual \ Predicted	SH	Br	St	Tr	BS	nL	**Recall**
SH	214	14	0	0	0	12	89.2%
Br	2	237	0	1	0	0	98.8%
St	4	0	127	0	24	5	79.4%
Tr	0	0	0	160	0	0	100%
BS	7	0	0	0	152	1	95%
Precision	94.3%	94.4%	100%	99.4%	86.4%		92.7%

Figure A.23: Situation 32 32

Actual \ Predicted	SH	Br	St	Tr	BS	nL	**Recall**
SH	221	14	0	0	0	5	92.1%
Br	0	238	0	1	0	1	99.2%
St	0	0	125	0	30	5	78.1%
Tr	0	0	0	159	0	1	99.4%
BS	15	0	1	0	140	4	87.5%
Precision	93.6%	94.4%	100%	99.4%	82.4%		92%

Figure A.24: Situation 32 16

Actual \ Predicted	SH	Br	St	Tr	BS	nL	**Recall**
SH	215	17	0	3	0	5	89.6%
Br	0	232	0	5	0	3	96.7%
St	1	0	127	0	26	6	79.4%
Tr	4	0	0	155	0	1	96.9%
BS	1	0	0	1	158	0	98.8%
Precision	97.3%	93.2%	100%	94.5%	85.9%		92.4%

Figure A.25: Situation 16 16

Actual \ Predicted	SH	Br	St	Tr	BS	nL	**Recall**
SH	217	3	0	2	2	5	90.4%
Br	4	231	0	2	2	1	96.3%
St	0	0	122	0	27	11	76.3%
Tr	2	0	0	158	0	0	98.8%
BS	7	0	0	1	152	0	95%
Precision	94.3%	96.3%	100%	96.9%	83.1%		91.7%

Figure A.26: Situation 16 0