Unsupervised screening of hearing status: automatic evaluation of the probe placement

Björn De Vogelaere

Supervisors: Prof. dr. ir. Dick Botteldooren, Dr. Annelies Bockstael
Counsellor: Vincent Nadon

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de ingenieurswetenschappen: computerwetenschappen

Department of Information Technology
Chairman: Prof. dr. ir. Daniël De Zutter
Faculty of Engineering and Architecture
Academic year 2014-2015
Unsupervised screening of hearing status: automatic evaluation of the probe placement

Björn De Vogelaere

Supervisors: Prof. dr. ir. Dick Botteldooren, Dr. Annelies Bockstael
Counsellor: Vincent Nadon

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de ingenieurswetenschappen: computerwetenschappen

Department of Information Technology
Chairman: Prof. dr. ir. Daniël De Zutter
Faculty of Engineering and Architecture
Academic year 2014-2015
Preface

This thesis concludes my five year lasting trip as a student at Ghent University. It was certainly a voyage worth remembering, which would not have been as interesting without the people that supported and accompanied me during the years. Therefore I would like to write a special word of thanks.

First of all I would like to thank my supervisors, Prof. Dr. Ir. Dick Botteldooren and Dr. Annelies Bockstael, for offering me this subject, for giving me the opportunity to put the knowledge I’ve acquired over the years into practice and for providing support when necessary. I would also like to thank my counsellor Vincent Nadon for the advice on writing my thesis and for the support on the electrotechnical aspects of this research.

Furthermore, I would like to thank all participating experts for spending their time on the labelling sessions.

I would also like to thank all of the professors and assistants who guided me through the courses and who provided me with new insights and useful knowledge.

Finally I would like to thank Marlies, Inge, my parents and my fellow students. Thanks for your support over the years, for making my time as a student enjoyable and for understanding that this work required a lot of effort and time.

“Vooruit nu naar de laatste slag,
O ruiterlust, in vroege dag”
Ruiterslied - Emiel Vereecken, naar het Duits, G. Herwegh - Tübinger Burschenschaft 1835

Thank you,

Björn De Vogelaere
Permission to loan

“The author gives permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use. In the case of any other use, the copyright terms have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master dissertation.”

Bjørn De Vogelaere, 22nd May 2015

Toelating tot bruikleen

“De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de bepalingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef.”

Björn De Vogelaere, 22 mei 2015
Unsupervised screening of hearing status: automatic evaluation of the probe placement

by

Björn DE VOGELAERE

Master’s dissertation submitted in order to obtain the academic degree of Master of Science in Computer Science Engineering

Academic year 2014-2015

Supervisors: Prof. Dr. Ir. D. BOTTELDOOREN, Dr. A. BOCKSTAEL

Counsellor: V. NADON

Faculty of Engineering and Architecture

Ghent University

Department of Information Technology

Chairman: Prof. Dr. Ir. D. DE ZUTTER

Summary

Otoacoustic emissions (OAEs) are widely used in clinical applications to monitor the status of the outer hair cells of the inner ear. OAEs are low-level signals produced by non-linear activities taking place in the cochlea. To make sure that these signals are registered accurately, it is of utmost importance that the probe used to measure these signals is placed correctly. Usually an expert supervisor interprets feedback a software system gives to act accordingly. This is however not possible if the system should be deployed in the field, e.g. in industry where no experts are present. In this thesis two domains are explored to evaluate the probe fit automatically, namely machine learning and fuzzy modelling. While machine learning allows to learn automatically from data, fuzzy modelling is based on expert knowledge, therefore suited to support an expert. All constructed models are evaluated using both expert feedback and OAE measurement data. Fuzzy models based on expert knowledge and literature prove to be able to evaluate the checkfits in an intuitive and effective manner. An Android application is developed which integrates a fuzzy model that allows online evaluation.

Keywords

otoacoustic emissions, hearing, unsupervised, machine learning, fuzzy logic
Unsupervised screening of hearing status: automatic evaluation of the probe placement

Björn De Vogelaere

Supervisor(s): prof. dr. ir. Dick Botteldooren, prof. dr. Annelies Bockstael, Vincent Nadon

Abstract—The status of our hearing is an important aspect that controls our daily routines. The measurement of otoacoustic emissions (OAEs) is widely used to screen the status of the outer hair cells of the inner ear because of its speed and non-invasiveness. As the measured OAE signals are low-level it is important that the probe used to measure them is placed correctly. Usually, the placement is performed by an expert. In this work a novel system is proposed to check the probe fit automatically without the need of any supervision. Two paths are explored to achieve such an expert system: machine learning and fuzzy modelling. While machine learning allows to learn automatically from data, fuzzy modelling is based on expert knowledge, therefore suited to support an expert. A mobile application is proposed to make the system portable and usable on the field, without the need of a supervising expert.

Keywords— otoacoustic emissions, hearing, unsupervised, machine learning, fuzzy logic

I. INTRODUCTION

OTOACOUSTIC emissions (OAEs) are widely used in a clinical context to screen the status of the outer hair cells of the inner ear. OAEs can be seen as low-level sound waves originating from nonlinear activities of the outer hair cells. The outer hair cells in the inner ear serve as acoustical pre-amplifiers which make it possible to perceive low-level sounds. This causes energy to propagate backwards to the middle ear. By placing a sensitive microphone in the outer ear, the resulting air pressure fluctuations can be recorded in the form of low-level signals, which are known as OAEs.

As these signals are low-level, it is important that the probe used to measure them is placed correctly. It has been shown in previous research that a bad probe fit can lead to large deviations on the final OAE measurements [1], [2]. There are two types of OAEs that are often used in a clinical context: Transient Evoked Otoacoustic Emissions (TEOAEs) and Distortion Product Otoacoustic Emissions (DPOAEs). They are measured using different techniques and arise from different mechanisms [3], but a bad probe fit has a negative effect on both. When placing the probe, the following steps are followed. First, a suitable probe tip is placed over the measuring probe. Then, the probe is placed in the ear canal. After the probe is placed, a stimulus is sent through a speaker in the probe in the form of rectangular pulses. The impulse response is then recorded with a microphone in the probe. Subsequently the supervising expert analyses the impulse response and the frequency response, i.e. the impulse response in the frequency domain, to see if the probe needs refitting.

It would be useful to have a system that is able to evaluate the probe fit automatically and eventually evaluate the actual OAE signal measurement. This system could for instance be deployed at a construction site, where there is a lot of environmental noise, so that construction workers would be able to monitor their hearing over the day. This way additional precautions could be taken if necessary. In this thesis, possible solutions for the first part, namely the automatic evaluation of the probe fit, are explored.

Two types of expert models were considered: models created with machine learning and models based on fuzzy logic. While machine learning models are constructed based on data (for instance evaluation scores given by experts), fuzzy models originate from literature and expert knowledge. In other words, machine learning models discover knowledge by processing data, while fuzzy models are based on knowledge that is already present. This makes it easier for experts to interpret fuzzy model outputs, as it is based on their knowledge. Additionally, a hybrid system based on fuzzy logic and machine learning, namely an Adaptive Neuro-Fuzzy Inference system, was tested.

In the end, a mobile application was created that allows to record analog input via the headset port and to analyse the recorded signal in real time. This application makes it therefore possible to evaluate impulse responses on the fly.

II. METHODS

A. Data collection and preparation

For this thesis, checkit data samples were extracted from data files present in an existing database. This database contains data written by the ILO-system, a system for clinical OAE analysis and data management created by Otodynamics Ltd. [4]. Each checkit data sample consists of the impulse response (length 5 ms), sampled at 25600 Hz. An example of an impulse response is shown in Figure 1.

![Figure 1](image)

Fig. 1. Example of an impulse response. (RA = ringing amplitude, P1A = first peak amplitude, P2A = second peak amplitude)

Each data sample was extended with the frequency response by taking a Fast Fourier transform. Thereafter, features were
extracted. These features represent characteristics of the signal which are fed to the expert system. Table I shows the list of extracted features.

Two forms of checkit labels were considered. The first one is the opinion of an expert, while the other one is objective OAE measurement data related, namely the deviation on a TEOAE measurement and the deviation on the noise level. In [2] 18 experts were asked to label 48 checkits by means of a Q distribution, as shown in Figure 2. For this work the numerical expert labels were averaged over all experts and rounded to obtain a dataset of 48 averaged labels. A second dataset was created in [2] containing checkits and matching deviations on both noise and TEOAE signal level. These deviations were obtained by measuring the OAEs of 34 subjects first in optimal conditions, i.e. with a normal fit. Thereafter, measurements were made in suboptimal conditions, by for instance using a probe tip that is too small or inserting the probe not deep enough.

![Figure 2](image)

Fig. 2. Example of a Q distribution [2]. Each column has a fixed amount of slots and represents a numerical label.

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-nearest neighbours</td>
<td>classification and regression</td>
</tr>
<tr>
<td>Decision tree</td>
<td>classification and regression</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>classification</td>
</tr>
<tr>
<td>Gaussian SVM</td>
<td>classification</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>classification</td>
</tr>
<tr>
<td>Linear regression</td>
<td>regression</td>
</tr>
</tbody>
</table>

Table II
Trained models.

A Java application was developed which had two purposes. The first one was to collect additional checkit evaluation scores from experts. The second one was to evaluate several fuzzy models already created based on literature. Seven experts labelled 50 randomly drawn samples each. The overlapping samples were averaged, resulting in 255 different samples. Four different models were evaluated. Each labelling session consisted of two parts per sample. First, the experts were asked to attribute a score in the 0-100 interval to a given checkit. They were also asked whether or not they would refit the probe. Then, a second screen was shown with model parameter feedback and an overall reference score. For instance, the model indicated the quality of the frequency spectrum with ‘good’, ‘medium’ or ‘bad’. Experts had the opportunity to modify their original score and to indicate if they agreed with the model parameter feedback.

B. Machine learning

Several classification models were trained on binary label data and several regression models were trained on the evaluation scores. The trained models are shown in Table II. Two sets of features were considered. The first one contained only features mentioned in literature, being the first nine features in Table I. The second set contained additional features relating to the frequency spectrum, such as spectral flatness, kurtosis and skewness of the frequency spectrum (in overlapping bands). For both feature sets a feature selection method was used to reduce the dimensionality, thereby avoiding problems caused by the curse of dimensionality [5]. For the expert feature space, a simple heuristic was used: add the feature that decreases the error the most, until adding a feature does not decrease the error any more. For the second one, random forests were used, as done in [6]. A random forest is a collection of decision trees trained on subsets of the training data. Therefore, the samples which are not used for training, also called ‘Out-Of-Bag’ samples, can be used to check the importance of a specific feature.

C. Fuzzy modelling

Five fuzzy models of the Mamdani-type were created, based on literature and expert feedback. The models were developed consecutively, the next model coping with issues of the previous model. The actual data from the labelling application were analysed in the end. Two promising models were then further optimized in Matlab ©2014 using neuro-fuzzy optimization.
D. Android application

An Android application was developed to test the final result. For this purpose the fuzzy module created for the Java application was reused. A synthetic signal simulating a perfect fit, an ILO stimulus recorded with a Head And Torso Simulator and an actual in-ear recording were sent through a TRRS cable from a laptop to a smartphone running the application. For the construction of the last signal and the recording of the resulting impulse response a DPOAE measurement system [7], [8] was used. The recorded signals were preprocessed by applying a low-pass and high-pass filter to put the focus on the [0,6000] Hz interval.

III. RESULTS AND DISCUSSION

A. Expert data test

The expert label set from [2] was converted to three different binary label sets. The set was split up in two with a certain decision boundary, being -1 for the first dataset, 0 for the second dataset and 1 for the last dataset. This means that for the first dataset all samples with a label '-3' or '-2' were labelled as 'refit needed'. The labels are shown in the first row in Figure 2. For each model the area under the curve (AUC) was computed [9]. This is a number between 0 and 1 and depicts how well the model separates bad fits from good fits. The results for all models (machine learning and fuzzy logic models) are shown in Table III. It is clear from this table that the first two sets can be separated well by all models, with the linear regression model scoring the highest. The third column shows that it is harder for models to separate the third dataset. This is because label '0' matches 'neutral', meaning that this is a vague label where most of the experts can disagree with each other. Nevertheless, linear regression still achieves an AUC of 92.96 %, which could suggest that the experts did not use many conditions for the labelling.

B. OAE data test

Three tests were performed on the OAE data. First, the data samples were split up in two sets: 'refit needed' and 'no refit needed'. This was constructed by taking the output from the classification models as is (0 or 1), while rounding and inverting regression/fuzzy model output. Subsequently, the distributions of both signal and noise deviation were analysed. A one-way ANOVA showed that for each model the mean of noise deviation was significantly higher for the 'refit' group (p < 0.05), except for the KNN classification model trained on the expert feature set. The interquartile range was both for noise and signal deviation larger for the 'refit' group, for each model. An example of a signal deviation box plot for both 'refit needed' samples and 'no refit needed' samples is shown in Figure 3. The samples were classified using the logistic regression model (using expert features). It is clear that most of the 'no refit needed' samples are within the [-2, 2] dB range. An example of a noise deviation box plot is shown in Figure 4. The mean of the noise of the 'refit needed' samples differs significantly from 0, following from a one-sample t-test with reference value 0. Furthermore, the interquartile range is clearly smaller for the 'no refit needed' samples. The groups resulting from other model predictions behaved similar for both signal and noise deviation.

Second, the effect of a moving acceptance threshold was analysed. Two things can be evaluated by moving the threshold. First of all, softening the threshold should allow more bad samples to be accepted, therefore resulting in a larger deviation. If this is not the case, i.e. if worse samples do not result in larger deviation, this could indicate that the model does not score the fits accurately. If the deviation increases moreover uniformly, this would indicate that the score is a good indication of the

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC1</th>
<th>AUC2</th>
<th>AUC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>92.84</td>
<td>92.31</td>
<td>70.37</td>
</tr>
<tr>
<td>GaussSVM</td>
<td>97.84</td>
<td>97.06</td>
<td>88.7</td>
</tr>
<tr>
<td>LinSVM</td>
<td>97.03</td>
<td>99.32</td>
<td>78.15</td>
</tr>
<tr>
<td>LogReg</td>
<td>89.19</td>
<td>92.53</td>
<td>76.3</td>
</tr>
<tr>
<td>KNN</td>
<td>96.49</td>
<td>97.74</td>
<td>78.33</td>
</tr>
<tr>
<td>DTALL</td>
<td>92.84</td>
<td>92.31</td>
<td>70.37</td>
</tr>
<tr>
<td>GaussSVMALL</td>
<td>97.03</td>
<td>99.32</td>
<td>86.85</td>
</tr>
<tr>
<td>KNNALL</td>
<td>96.76</td>
<td>98.64</td>
<td>84.81</td>
</tr>
<tr>
<td>LinSVMALL</td>
<td>97.3</td>
<td>99.77</td>
<td>89.63</td>
</tr>
<tr>
<td>LogRegALL</td>
<td>97.3</td>
<td>99.32</td>
<td>90.74</td>
</tr>
<tr>
<td>DTReg</td>
<td>97.3</td>
<td>98.19</td>
<td>84.63</td>
</tr>
<tr>
<td>KNNReg</td>
<td>97.57</td>
<td>97.51</td>
<td>84.63</td>
</tr>
<tr>
<td>LinReg</td>
<td>97.3</td>
<td>100</td>
<td>92.96</td>
</tr>
<tr>
<td>DTRegALL</td>
<td>97.03</td>
<td>98.19</td>
<td>84.26</td>
</tr>
<tr>
<td>KNNRegALL</td>
<td>96.76</td>
<td>95.02</td>
<td>71.11</td>
</tr>
<tr>
<td>LinRegALL</td>
<td>97.13</td>
<td>96.83</td>
<td>74.19</td>
</tr>
<tr>
<td>1 Mamdani</td>
<td>98.38</td>
<td>96.95</td>
<td>85.19</td>
</tr>
<tr>
<td>2 Mamdani</td>
<td>98.38</td>
<td>95.81</td>
<td>81.76</td>
</tr>
<tr>
<td>3 Mamdani</td>
<td>97.84</td>
<td>97.06</td>
<td>81.94</td>
</tr>
<tr>
<td>3 ANFIS</td>
<td>93.24</td>
<td>95.93</td>
<td>82.78</td>
</tr>
<tr>
<td>4 Mamdani</td>
<td>97.30</td>
<td>95.70</td>
<td>83.89</td>
</tr>
<tr>
<td>5 Mamdani</td>
<td>97.30</td>
<td>97.96</td>
<td>84.91</td>
</tr>
<tr>
<td>5 ANFIS</td>
<td>98.65</td>
<td>92.76</td>
<td>76.30</td>
</tr>
</tbody>
</table>
quality. This would mean that quality score and deviation are more or less inversely proportional. Per threshold the collection of accepted samples, i.e. the set of samples classified/scored as ‘no refit needed’, was considered. The interquartile range of the signal deviations for those samples was plotted. This was done based on the argument that only the accepted checkfits are of interest, as only the ones that are accepted will be followed by an actual OAE measurement. To obtain scored samples from classification models, the classification scores were used instead of the binary output. For the regression and fuzzy models the usual (continuous) output could be used. The plots showing the most differences between models are shown in Figures 5 and 6. Figure 5 shows that for Mamdani models 3 and 5 the interquartile range decreases monotonically as expected. For the optimized models 3 and 5 the interquartile range seems to decrease more uniformly. However, for model 5 it also goes up again in the end, which is not wanted. This could be due to noise (not enough samples to represent a good idea of the deviation), but as the interquartile range was only plotted when the amount of accepted samples was higher than 50, the probability of noise is rather low. Figure 6 shows that a decision tree does not have a uniform decrease. This allows less flexibility when choosing a good acceptance threshold.

Finally, the set of OAE samples was labelled with 1 or 0 according to the deviation on the signal. If the total signal deviation was more than 2.5 dB or the signal deviation in one octave band was more than 3 dB the sample was labelled with 1, otherwise 0. This was done based on the findings in [3]. The recall and precision of bad fits were computed for each model. The recall is defined as the number of actual bad fits correctly recognized divided by the total amount of actual bad fits. The precision is defined as the number of actual bad fits correctly recognized divided by the total amount of fits classified as ‘bad’. Furthermore, recall and precision curves were plotted by moving the labelling threshold. Figures 7 and 8 show the curves for regression models and three Mamdani models. Logistic regression and KNN for classification produced similar curves. It could be concluded from these figures that the performance of a given model depends on the recall. Some models produce curves that have a uniform transition therefore allowing more choice in decision boundaries, while other models have a better recall of bad fits for a certain threshold. For instance, Mamdani model 1 has a good precision for a recall of 87.65 %, but does not have a threshold for which the recall is in the [70, 85] % range. Overall, Mamdani models had the best recall-precision trade-off, considering a recall of 70 % and up.

IV. ANDROID APPLICATION

Mamdani model 5 was used for testing. The model could be evaluated on the fly. The audio signal recorded with phone via the headset port had fluctuations and more ringing, which indicates that additional measures need to be taken to offer an ILO-like display of the impulse and frequency response. Figure 9 shows the evaluation of the ILO-stimulus recording.
V. CONCLUSION

It can be concluded that Mamdani models based on expert knowledge are capable of evaluating probe fits rather well. The only reason why some machine learning models would be preferred is because of the fast evaluation. Mamdani models have a simple and intuitive structure and are therefore more suited for classification, e.g. for accepting and rejecting fits than for providing a quality score. Therefore, it could be useful to collect scores given by several experts for the same checkfits. This would allow to train models on more accurate scores, as they would be averaged over all experts therefore averaging out outlier scores. This way a good model could be constructed capable of providing a quality score proportional to the expected deviation on the OAE measurement. In the future, fuzzy modelling and machine learning could be used to evaluate the actual OAE signal measurements and suggest a new measurement or probe fit if needed by looking at both the checkfit, noise measurements and OAE signal measurements.

REFERENCES

Screening van het gehoor zonder supervisie: automatische evaluatie van de probeplaatsing

Björn De Vogelaere

Begeleider(s): prof. dr. ir. Dick Botteldooren, prof. dr. Annelies Bockstael, Vincent Nadon

Abstract—De status van het gehoor heeft een belangrijke invloed op onze dagelijkse routinetjes. Het meten van oto-akoestische emissies (OAEs) wordt dikwijls gebruikt bij de screening van de uitwendige haarcellen van het binnenoor omwille van de snelheid en niet-invasiviteit. Aangezien de opgenomen OAE signalen een lage intensiteit hebben is het belangrijk dat de meet-probe correct geplaatst is. Doorgaans wordt een meting uitgevoerd door een expert. In dit onderzoek wordt een nieuw systeem voorgesteld dat toelaat om de probeplaatsing automatisch te evalueren, zonder de nood aan een superviserende expert. Twee domeinen worden daarbij verkend: machinaal leren en vage logica. Machinaal leren laat toe om systemen te ontwikkelen gebaseerd op data. Vage logica is gebaseerd op literatuur en algemene kennis van experts, en is daarom geschikt om experts te ondersteunen. Een mobiele applicatie wordt voorgesteld om het systeem draagbaar en bruikbaar te maken in een niet-klinische setting.

Trefwoorden—oto-akoestische emissies, gehoor, probeplaatsing, supervisie, machinaal leren, vage logica

I. INTRODUCTIE

OTO-AKOESTISCHE emissies (OAEs) worden dikwijls gebruikt in een klinische context voor de screening van de uitwendige haarcellen in het binnenoor. OAEs kunnen gezien worden als geluidsgolven van een lage intensiteit die hun oorsprong vinden in niet-lineaire activiteiten van de uitwendige haarcellen. De uitwendige haarcellen in het binnenoor dienen als akoestische actuators die het mogelijk maken om geluiden van lage intensiteit waar te nemen. Dit zorgt ervoor dat energie terug gepropageerd wordt naar het middenoor. Door het plaatsen van een gevoelige microfoon in het oorkanaal kunnen de resulterende fluctuaties in luchtdruk geregistreerd worden in de vorm van laagintense geluiden, namelijk OAEs. Aangezien deze signalen laagintens zijn, is het van uiterst belang dat de meet-probe correct geplaatst is. In vorig onderzoek werd aangetoond dat een slechte plaatsing tot grote afwijkingen kan leiden in de uiteindelijke OAE-metingen [1], [2]. Er zijn twee types OAEs die dikwijls gebruikt worden in een klinische context: transient geëvoekte oto-akoestische emissies (TEOAEs) en distortieproduct oto-akoestische emissies (DPOAEs). Deze worden gemeten op verschillende wijze en worden veroorzaakt door verschillende mechanismen [3], maar een slechte plaatsing van de meet-probe heeft een negatief effect op beide. Voor het plaatsen van de probe worden de volgende stappen gevolgd. Doorgaans wordt een geschikte dop (tip) geplaatst over de meet-probe. Vervolgens wordt de probe in de gehoororgang geplaatst. Nadat de probe geplaatst is wordt een stimulus gestuurd door een speaker in de probe. Deze stimulus bestaat uit opeenvolgende rectangulaire pulsen. De impulsrespons wordt vervolgens opgenomen met een gevoelige microfoon die zich net als de speaker in de probe bevindt. Vervolgens analyseert de superviserende expert de impulsresponsen en de frequentieresponsen, i.e. de impulsrespons in het frequentiespectrum, om te bepalen of de probe al dan niet opnieuw geplaatst moet worden.

Het zou nuttig zijn om over een systeem te beschikken dat in staat is om de probeplaatsing automatisch te evalueren en vervolgens de uiteindelijke OAE-metingen. Dit systeem zou bijvoorbeeld kunnen gebruikt worden op een werfplaats, waar veel omgevingsgeluid aanwezig kan zijn, zodanig dat bouwvakkers de status van het gehoor zouden kunnen volgen over een werkdag. Op die manier kunnen er voorzorgsmaatregelen genomen worden indien dit nodig zou blijken. In dit onderzoek worden er mogelijke oplossingen voorgesteld voor het eerste deel, namelijk de automatische evaluatie van een probeplaatsing. Twee types expert modellen werden beschouwd: modellen gebaseerd op machinaal leren en modellen gebaseerd op vage logica. Terwijl modellen gebaseerd op machinaal leren geconstrueerd worden op basis van data (bijvoorbeeld scores gegeven door experts) baseren vage modellen zich op algemene kennis die er reeds is. Dit laatste laat bijvoorbeeld toe dat resultaten gegeven door het model geïnterpreteerd kunnen worden door een expert. Daarnaast werd een hybride systeem gebaseerd op vage logica en machinaal leren gerealiseerd. Uiteindelijk werd een mobiele applicatie ontwikkeld die toelaat om analooge input op te nemen via de headset poort en om die input vervolgens te analyseren in reële tijd. Deze applicatie maakt het daardoor mogelijk om impulsresponsen te evalueren terwijl de probe wordt gepast.

II. METHODE

A. Data verzameling en verwerking

Voor dit onderzoek werden er probeplaatsing-gegevens geëxtraheerd uit databestanden opgeslagen in een bestaande datataban. Deze databank bevat data geschreven door het ILO-systeem, een systeem voor klinische OAE analyse en databeheer ontwikkeld door Otodynamics Ltd. [4]. Ieder gegevensrecord van kortweg 'checkit' bevat een impulsrespons (lengte 5 ms), be-monsterd aan 25600 Hz. Figuur 1 toont een voorbeeld van een impulsrespons.

Ieder gegevensrecord werd aangevuld met de frequentieresponsen door het nemen van een Fourier-transformatie. Daarna werden er features geëxtraheerd. Deze features bepaalden eigenschappen van het signaal, die aan het systeem werden aangeboden. Tabel I toont de lijst van gebruikte features.

Twee types gegevenslabels werden beschouwd. De eerste is de mening van de expert. De tweede is objectieve OAE-data geïntegreerd, namelijk de afwijking van een TEOAE-meting en de afwijking op het ruisoniveau. In [2] werd aan 18 experts gevraagd om 48 gegevensrecords te labellen door middel van een Q dis-
Fig. 1. Voorbeeld van een impulserespons. (RA = ringing amplitude, P1A = eerste piek amplitude, P2A = tweede piek amplitude)

**Feature**
- Amplitude van ringing (I)
- Asymmetrie van de stimulus (I)
- Lengte van de ringing (I)
- Maximum amplitude (I)
- Gemiddelde amplitude P1 en P2 (I)
- Ringing amplitude / maximum amplitude (I)
- Ringing amplitude / gemiddelde amplitude (I)
- Heeft een dip van meer dan 3 dB in 2000 Hz venster (F)
- Maximum dip in 2000 Hz venster (F)
- Heeft een pick van meer dan 3 dB in 2000 Hz venster (F)
- Spectrale vlakheid (banden) (F)
- Kurtosis (banden) (F)
- Scheefheid (banden) (F)
- Standardafwijking (banden) (F)
- Gemiddelde (banden) (F)

**Tabel 1**
<table>
<thead>
<tr>
<th>Feature</th>
<th>(I)</th>
<th>(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geëxtraheerde features. Features in het tweede deel werden niet gebruikt voor vage modellen. (I): impulserespons features, (F) frequentierespons features</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Voor dit onderzoek werden de resultaten uitgevoerd onder alle experts en afgerond zodanig dat een set van 48 labels werd gekozen. Een tweede dataset werd gecreëerd in [2] bevattende de checkfits en overeenkomstige deviaties op zowel het TEOAE signaal niveau als op het ruis niveau. Deze afwijkingen werden gekozen door het meten van OAEs van 34 subjecten in optimale omstandigheden, i.e. voorafgegaan door een quasi perfecte probeplaatsing. Daarna werden OAE metingen uitgevoerd in suboptimale omstandigheden, door bijvoorbeeld een probe dop te gebruiken die te klein is.

Voor dit onderzoek werd een Java-applicatie ontwikkeld met twee bedoelingen. Ten eerste werden er bijkomende checkfit scores verzameld, gegeven door experts. Ten tweede konden deze resultaten uitgevoerd onder alle experts en afgerond zodanig dat een set van 48 labels werd gekozen. Een tweede dataset werd gecreëerd in [2] bevattende de checkfits en overeenkomstige deviaties op zowel het TEOAE signaal niveau als op het ruis niveau. Deze afwijkingen werden gekozen door het meten van OAEs van 34 subjecten in optimale omstandigheden, i.e. voorafgegaan door een quasi perfecte probeplaatsing. Daarna werden OAE metingen uitgevoerd in suboptimale omstandigheden, door bijvoorbeeld een probe dop te gebruiken die te klein is.

Voor dit onderzoek werd een Java-applicatie ontwikkeld met twee bedoelingen. Ten eerste werden er bijkomende checkfit scores verzameld, gegeven door experts. Ten tweede konden deze resultaten uitgevoerd onder alle experts en afgerond zodanig dat een set van 48 labels werd gekozen. Een tweede dataset werd gecreëerd in [2] bevattende de checkfits en overeenkomstige deviaties op zowel het TEOAE signaal niveau als op het ruis niveau. Deze afwijkingen werden gekozen door het meten van OAEs van 34 subjecten in optimale omstandigheden, i.e. voorafgegaan door een quasi perfecte probeplaatsing. Daarna werden OAE metingen uitgevoerd in suboptimale omstandigheden, door bijvoorbeeld een probe dop te gebruiken die te klein is.

<table>
<thead>
<tr>
<th>-3</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>13</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>42</td>
<td>16</td>
<td>6</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>54</td>
<td>1</td>
<td>32</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>41</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>20</td>
<td>36</td>
<td>25</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>9</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>46</td>
<td>40</td>
<td>33</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>18</td>
<td>22</td>
<td>27</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>47</td>
<td>5</td>
<td>38</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Voor dit onderzoek werden de resultaten uitgevoerd onder alle experts en afgerond zodanig dat een set van 48 labels werd gekozen. Een tweede dataset werd gecreëerd in [2] bevattende de checkfits en overeenkomstige deviaties op zowel het TEOAE signaal niveau als op het ruis niveau. Deze afwijkingen werden gekozen door het meten van OAEs van 34 subjecten in optimale omstandigheden, i.e. voorafgegaan door een quasi perfecte probeplaatsing. Daarna werden OAE metingen uitgevoerd in suboptimale omstandigheden, door bijvoorbeeld een probe dop te gebruiken die te klein is.

Voor dit onderzoek werden de resultaten uitgevoerd onder alle experts en afgerond zodanig dat een set van 48 labels werd gekozen. Een tweede dataset werd gecreëerd in [2] bevattende de checkfits en overeenkomstige deviaties op zowel het TEOAE signaal niveau als op het ruis niveau. Deze afwijkingen werden gekozen door het meten van OAEs van 34 subjecten in optimale omstandigheden, i.e. voorafgegaan door een quasi perfecte probeplaatsing. Daarna werden OAE metingen uitgevoerd in suboptimale omstandigheden, door bijvoorbeeld een probe dop te gebruiken die te klein is.

Voor dit onderzoek werden de resultaten uitgevoerd onder alle experts en afgerond zodanig dat een set van 48 labels werd gekozen. Een tweede dataset werd gecreëerd in [2] bevattende de checkfits en overeenkomstige deviaties op zowel het TEOAE signaal niveau als op het ruis niveau. Deze afwijkingen werden gekozen door het meten van OAEs van 34 subjecten in optimale omstandigheden, i.e. voorafgegaan door een quasi perfecte probeplaatsing. Daarna werden OAE metingen uitgevoerd in suboptimale omstandigheden, door bijvoorbeeld een probe dop te gebruiken die te klein is.

**B. Machinaal leren**


**C. Vage modellen**

Vijf vage modellen van het Mamdani-type werden getraind gebaseerd op literatuur en feedback van experts. De modellen werden achtereenvolgens ontwikkeld, waarbij een volgend model telkens de problemen van het vorige model trachtte op te lossen. De uiteindelijke data verzameld met de Java-applicatie
werd geanalyseerd op het einde. Twee beloftevolle modellen werden vervolgens verder geoptimaliseerd in Matlab® 2014 gebruikmakende van neuro-fuzzy optimalisatie.

D. Android applicatie

Een Android applicatie werd ontwikkeld om het uiteindelijke resultaat te evalueren. Daarvoor werd de module verantwoordelijk voor de vage modellen gecreëerd voor de Java-applicatie opnieuw gebruikt. Een synthetisch signaal dat een goede impuls-respons modelleerde, een ‘Head And Torso Simulator’-opname van de stimulus gebruikt in het ILO-systeem en een opname van een eigen signaal werden verstuurd door een TRRS kabel van een laptop naar een smartphone waarop de applicatie werd geïnstalleerd. Voor het genereren van het laatste signaal en het opnemen van de resulterende impulserespns werd een DPOAE-meetsysteem gebruikt [7, 8]. De opgenomen signalen werden door een laag- en hoogdoorlaat filter gestuurd om de focus te leggen op het [0,6000] Hz frequentie-interval.

### RESULTATEN EN DISCUSSIE

De expert label set van [2] werd geconverteerd naar drie verschillende binair sets. De set werd opgesplitst in twee subsets d.m.v. een bepaalde drempelwaarde, zijnde -1 voor de eerste dataset, 0 voor de tweede dataset en 1 voor de laatste dataset. Dit wil zeggen dat in de eerste dataset alle checkfits die oorspronkelijk een label '-3' of '-2' hadden gekregen werden beschouwd als zijnde 'slecht'. Voor ieder model werd de 'area under the curve' (AUC) berekend [9]. Dit is een waarde tussen 0 en 1 die beschrijft hoe goed een bepaald model de slechte en goede checkfits kan onderscheiden. De resultaten werden getoond in Tabel III. Het is duidelijk dat uit deze tabel kan afgeleid worden dat de eerste twee sets goed kunnen worden benaderd door vrijwel alle modellen. Het lineaire regressiemodel scoort daarbij het hoogst. De derde kolom toont dat het moeilijker is om vrijwel alle 'geen herplaatsing nodig'-checkfits zich binnen het [-2, 2] dB interval bevinden. Een voorbeeld van een box plot voor de signaalafwijking wordt getoond in Figuur 3. De checkfits werden geclassificeerd gebruikmakende van logistische regressie (expert features). Het is duidelijk dat vrijwel alle 'geen herplaatsing nodig'-checkfits zich verschilt significant van 0: dit volgt uit een one-sample t-test met referentie waardeloos 0. Daarmee wordt de lineaire regressiemodel nog steeds een AUC van 92.96 % behalen, wat suggerereert dat de experts niet veel criteria gebruikten tijdens het labelen.

#### A. OAE data test

Er werden drie tests uitgevoerd op de OAE data. Ten eerste werden gegevensrecords verdeeld over twee verzamelingen: 'herplaatsing nodig' en 'geen herplaatsing nodig'. Dit werd gedaan door het nemen van de voorspelling van de classificatiemo-
den, wat dan zou moeten resulteren in meer deviatie. Indien dit niet het geval is, kan dit een aanwijzing zijn dat het model de checkfits niet correct scoort. Als bovendien de afwijking uniform stijgt, kan dit erop wijzen dat de score een goede maat is voor de kwaliteit van de checkfit. Dit zou betekenen dat de score en de afwijking omgekeerd evenredig zijn. Per drempel werd de verzameling van geaccepteerde checkfits beschouwd (i.e. de ‘geen herplaatsing nodig’-checkfits). De interkwartielafstand van de signaalafwijkingen voor die checkfits werd geplot. Om een continue score te verkrijgen van classificatimodellen werden de classificatiescores gebruikt. Bij de regressiemodellen en vage modellen kon de gewoonlijke output worden gebruikt. De plots die het meeste verschil tussen modellen aantonen worden getoond in Figuren 5 en 6.

Figuur 5 toont dat voor de Mamdani-modellen de interkwartielafstand monotoon daalt zoals verwacht. De geoptimaliseerde modellen (3 en 5) hebben een curve die meer uniform daalt. De curve stijgt echter weer op het einde, wat niet gewenst is. Dit zou kunnen veroorzaakt worden door ruis (te weinig samples), maar aangezien er enkel geplot werd indien de resterende verzameling meer dan 50 checkfits bedroeg, is deze kans echter klein. Figuur 6 toont dat een beslissingsboom geen uniforme daling in deviatie veroorzaakt. Dit zorgt voor minder flexibiliteit bij het kiezen van een acceptatiedrempel.

Ten slotte werden de checkfits gelabeld met 1 of 0 volgens de afwijking op het signaal. Indien de totale signaalafwijking meer dan 2,5 dB bedroeg of de signaalafwijking in een octaafband meer dan 3 dB bedroeg werd de checkfit gelabeld met 1. Dit werd gedaan gebaseerd op de bevindingen in [3]. De recall en precision (met label 1 de positieve klasse) werden berekend voor ieder model. De recall is gedefinieerd als het aantal herkende werkelijke slechte checkfits gedeeld door het totaal aantal werkelijke slechte checkfits. De precision is gedefinieerd als het aantal herkende werkelijke slechte checkfits gedeeld door het totaal aantal checkfits voorspeld als slecht. Daarnaast werden recall en precision curves geplot, door het verplaatsen van de acceptatiedrempel. Figuren 7 en 8 tonen de curves voor regressiemodellen en drie Mamdani modellen. Logistische regressie en KNN voor classificatie produceren gelijkaardige curves. Uit deze figuren kon worden afgeleid dat de perfor-
mante van een gegeven model afhangt van de vereiste recall. Sommige modellen produceren curves die een betere uniforme overgang hebben, waardoor er meer keuze is in drempelwaarden, terwijl andere modellen een betere recall hebben voor een bepaalde drempelwaarde. Mamdani model 1 heeft bijvoorbeeld een goede precision voor een recall van 87.65 %, maar heeft geen drempelwaarde waarbij de recall in het [70,85] % interval ligt. Over het algemeen hadden Mamdani modellen de beste recall-precision trade-off indien een recall van minstens 70 % nodig is.

![Fig. 7. Effect of the placement of the acceptance threshold on 'slechte plaatsing' recall and precision (regressiemodels).](image1)

![Fig. 8. Effect of the placement of the acceptance threshold on 'slechte plaatsing' recall and precision (Mamdani fuzzy models 1.2 and 4).](image2)

**IV. ANDROID APPLICATIE**

Mamdani model 5 werd gebruikt bij het testen van de applicatie. Het model kon online geëvalueerd worden. De smartphone ervaarde fluctuaties met betrekking tot het ontvangen audiosignaal en meer ringing, wat erop wijst dat bijkomende maatregelen nodig zijn om een resultaat te bekomen dat in staat is om een score te geven die evenredig is met de verwachte deviatie op de uiteindelijke OAE meting. In de toekomst zouden vage modellen en modellen gebaseerd op machinaal leren gebruikt kunnen worden om metingen van OAE te evalueren en een nieuwe meting of probeplaatsing te suggereren indien nodig door te kijken naar de checkfit, nuisieveu’s en OAE signaalmetingen.

**V. CONCLUSIE**

Er kan geconcludeerd worden dat Mamdani modellen gebaseerd op algemene kennis in staat zijn om probeplaatsingen vrij accuraat te evalueren. De enige reden waarom modellen gebaseerd op machinaal leren zouden worden verkozen is omwille van de snelle evaluatie. Omwille van de eenvoud van Mamdani modellen zijn deze meer geschikt voor checkfit classificatie, bijvoorbeeld voor het accepteren of weigeren van checkfits, dan voor het numerieke scoren van checkfits. Het zou nuttig zijn om meer scores te verzamelen gegeven door verscheidene experts voor dezelfde checkfits. Dit zou toelaten om modellen te trainen op meer accurate scores, aangezien deze zouden kunnen uitgebreid worden over alle experts, waarbij uitschieters vermeden zouden worden. Op die manier kan dan een model worden gekomen dat in staat is om een score te geven die evenredig is met de verwachte deviatie op de uiteindelijke OAE meting. In de toekomst zouden vage modellen en modellen gebaseerd op machinaal leren gebruikt kunnen worden om metingen van OAE te evalueren en een nieuwe meting of probeplaatsing te suggereren indien nodig door te kijken naar de checkfit, nuisieveu’s en OAE signaalmetingen.

**REFERENTIES**


Contents

Preface i

Summary iii

List of Figures xvii

List of Tables xx

Acronyms xxi

1 Introduction 1

1.1 Motivation 1

1.2 Problem definition 1

1.3 Objective 2

1.4 Method 2

2 Literature review 4

2.1 OAEs 4

2.2 Checking the probe placement 6

2.3 Machine learning 8

2.3.1 The steps of supervised machine learning 9

2.3.2 Models 10

2.3.3 Additional notes on machine learning 16

2.4 Fuzzy logic 17

2.4.1 General structure of a fuzzy model 17

2.4.2 Mamdani and Sugeno fuzzy models 20

2.4.3 Neuro-fuzzy modelling 21
## Methodology

### 3.1 Problem definition

### 3.2 Data preparation
- **3.2.1 Data selection**
- **3.2.2 Data preprocessing**
- **3.2.3 Data transformation**

### 3.3 Machine learning
- **3.3.1 Model selection**
- **3.3.2 Feature space dimension reduction**
- **3.3.3 Validation**

### 3.4 Fuzzy modelling
- **3.4.1 Mamdani models**
- **3.4.2 Neuro-fuzzy modelling**

### 3.5 Model tests and overview

### 3.6 Developing an Android application for automatic probefit evaluation
- **3.6.1 Test signals**
- **3.6.2 Application flow**

## Results

### 4.1 Machine learning models

### 4.2 Fuzzy models
- **4.2.1 Labelling application data analysis**
- **4.2.2 Mamdani models evaluated on new expert data**
- **4.2.3 Neuro-fuzzy results**

### 4.3 Model tests
- **4.3.1 Expert data evaluation**
- **4.3.2 OAE measurements**

### 4.4 Android application

## Discussion

### 5.1 Data collection

### 5.2 Machine learning

### 5.3 Fuzzy logic
CONTENTS

5.3.1 Fuzzy model evaluation data .............................................. 72
5.3.2 Neuro-fuzzy optimization ................................................. 73
5.4 Model evaluation ................................................................. 74
  5.4.1 Evaluation on expert data ................................................ 74
  5.4.2 Evaluation on OAE data .................................................. 74
  5.4.3 Overall evaluation discussion ......................................... 76
5.5 Android application .............................................................. 76
5.6 Future work .......................................................... 77

6 Conclusion ................................................................. 79

A Fuzzy logic application using java ............................................. 81
  A.1 Global architecture ......................................................... 81
  A.2 Fuzzy module architecture .............................................. 82

B Android application ............................................................. 84

C Algorithms ............................................................................. 85
  C.1 Extraction of impulse response features ............................... 85
  C.2 Extraction of frequency response features ........................... 85

Bibliography ................................................................. 90
List of Figures

2.1 Schematic drawing of a system to assess evoked otoacoustic emissions in the presence of external noise. [1] ................................. 5

2.3 Left: the DPOAE spectrum relates to f1 and f2 as: \( f_{dp} = 2 \cdot f_1 - f_2 \). Right: DP-gram with the response for specific frequencies. [2] ................................. 6

2.4 Different placements of the probe cause different impulse responses. A shows a normal probe placement, B shows a leaky fit and C shows a bad rubber tip fitting. Figure extracted from [3] ................................. 7

2.5 Features used by expert to determine the quality of the probe checkfit ................................. 8

2.6 ROC curve examples. ................................................. 12

2.7 Example of using a decision tree for heartbeat classification. ................................. 13

2.8 Random forest and multiple experts ................................. 14

2.9 Maximal margin classification ................................. 15

2.10 Schematic overview of a fuzzy model ................................. 18

2.11 A set of three membership functions for the variable ‘service’. A value of 5 results in a Y-value of 0 for both the membership functions ‘poor’ and ‘excellent’, and a Y-value of 1 for the membership function ‘good’. ................................. 19

2.12 Firing the rule ‘if service is good OR food is delicious, then tip is average’. Average_MF(X) represents the Y value given by the membership function ‘average’ for the value X. ................................. 20

2.13 The outputs of the rules are aggregated using the max function. Subsequently the centroid is computed (red line). ................................. 20

2.14 An example of an ANFIS structure. ................................. 21
3.1 An example of a Q distribution. Each sample is assigned to a column. Each column has a fixed amount of slots. Each column represents a label or score, given by the value in the upper row. .................................................. 25

3.2 The distribution of averaged expert labels. .................................................. 25

3.3 The labelling phase in the Java application .................................................. 27

3.4 Labelling app scores distribution. Each bar represents an interval of 10 %. .... 28

3.5 Fuzzy model evaluation in a Java application ............................................. 29

3.6 Difference between using global or local extrema to identify stimulus peaks. . . 31

3.7 The S-shape (= stimulus) of the impulse response followed by ringing. ....... 32

3.8 Difference in result between using global or local extrema to identify stimulus peaks: using local extrema a much higher amplitude of the ringing is discovered (which is equal to the global maximum in this example). ........................ 32

3.9 Expanding input intervals by stretching the lower and upper membership functions. 41

3.10 Constructed models overview and test methodology. .............................. 43

3.11 Recording of impulse responses and sending to a smartphone. (Cav. = cavity of the HATS ear coupler, Imp. resp. = impulse response) .......................... 45

3.12 Signals used for application evaluation. .................................................. 46

3.13 Application flow. ................................................................................. 47

4.1 Example of a good fit. With an epsilon of 0.005 Pa, the ringinglength parameter has a value of 2.11 ms. With an epsilon of 0.015 Pa, the ringinglength parameter has a value of 1.25 ms. .......................... 50

4.2 Mamdani models tested on collected expert scores. ............................... 53

4.3 Training of the ANFIS models. The checking option was used to prevent over-fitting. 54

4.4 Comparing the predictions of the optimized model with the training data. ...... 54

4.5 ROC curves for different models given the expert label set with decision boundary 1. ................................................................. 58

4.6 Boxplots showing the signal deviations grouped by model output. ............ 60

4.7 Boxplots showing the noise deviations grouped by model output. ............ 60

4.8 Effect on the interquartile range of the signal deviation by moving the acceptance threshold (classification models). ........................................... 61

4.9 Effect on the interquartile range of the signal deviation by moving the acceptance threshold (regression models). ........................................... 62
4.10 Effect on the interquartile range of the signal deviation by moving the acceptance threshold (fuzzy models). ........................................ 62
4.11 Effect of moving the refit threshold on the model scores. ....................... 65
4.12 Recall and precision curves for regression models (Linear Regression and Decision Tree). ........................................... 66
4.13 Effect of moving the threshold (model 3 and model 5). .......................... 67
4.14 Effect of moving the threshold (model 1 and model 2). ......................... 68
4.15 Application checkfit evaluation results. ‘Amplitude of the response’ is calculated using the ampratio feature (I8), while ‘amplitude of stimulus’ is calculated using the mean amplitude of the stimulus peaks (I6). ......................... 69
4.16 The HATS recording of the impulse response shown in the ILO system. .... 69
A.1 The packages and their dependencies .............................................. 82
A.2 Extending the Jfuzzylite library. ................................................. 83
B.1 Simplified package diagram and class diagram for the levelmeter package. External packages are omitted here. .............................................. 84
List of Tables

3.1 Table of participants ............................................................. 26
3.2 Evaluated classification techniques. ........................................ 35
3.3 Evaluated regression techniques. ........................................... 35
3.4 Nine expert features derived from literature [4, 3, 5]. ............... 36
3.5 Constructed Mamdani models. Models 1-4 were evaluated by at least one expert. Model 5 was created in the end. ........................................... 39
3.6 Display names of parameters which were shown in the labelling application. ........ 40
3.7 Amount of rules per model. ..................................................... 40
3.8 Evaluated model per expert ..................................................... 40
3.9 Model changes compared to previous version .......................... 42
4.1 Final machine learning models with cross-validation error. Models having an ‘ALL’-suffix are trained on the extended feature set. ‘DT’ stands for ‘decision tree’. 49
4.2 Percentage of agreement per parameter. ................................... 51
4.3 Fuzzy model evaluation. The last column shows the mean difference of the scores that were modified after system feedback. ................................. 51
4.4 AUC [%] for each model, given the expert labels. The best option for each column is indicated in bold. ......................................................... 57
4.5 Difference in interquartile range of ‘refit’ and ‘no refit needed’ group for both signal and noise deviation [dB]. ................................. 59
4.6 Confusion data given the model output (rounded output for regression models) and OAE data. FP = False Positives, FN = False Negatives, TP = True Positives, TN = True Negatives, with the positive class being the ‘refit needed’ class. ........ 64
Acronyms

**ANFIS** adaptive-network-based fuzzy inference system. 21, 24, 27, 55, 56, 61, 71, 75–77

**AUC** area under the curve. 11, 37, 48, 55, 74

**DPOAEs** distortion product otoacoustic emissions. 4, 6, 7

**DSP** digital signal processor. 76, 77

**EOAEs** evoked otoacoustic emissions. 1, 4

**FFT** Fast Fourier Transform. 28

**KNN** K-nearest neighbours. 56, 61, 63, 72

**MSE** Mean Squared Error. 10, 37, 48, 72, 73

**OAEs** otoacoustic emissions. 1, 4, 6, 78

**OHCs** outer hair cells. 1, 4

**OOB** out-of-bag. 13, 36

**ROC** receiver operating characteristic. 11, 55

**SOAEs** spontaneous otoacoustic emissions. 4

**SVM** support vector machine. 13, 14, 55, 61

**TEOAES** transient evoked otoacoustic emissions. 4–6
Chapter 1

Introduction

1.1 Motivation

Our hearing helps us with many tasks during our daily routines. However, the auditory system proves to be a sensitive part of the body. According to [6] over 360 million people worldwide have disabling hearing loss (over 30dB for children, 40dB for adults), half of whom could have avoided it using primary protection. Therefore it is important to be able to screen the hearing status thoroughly and efficiently. Nowadays, such hearing tests are done by experts (audiologists, doctors . . . ) in a clinical setting. It would be less cumbersome if it were possible to perform tests of the same quality automatically and in a non-clinical setting, for example in noisy industrial environments [7, 8]. This would provide industrial workers useful information on their hearing status which enables them to take precautions if necessary.

1.2 Problem definition

The measurement of otoacoustic emissions (OAEs) is used to screen the status of the outer hair cells of the inner ear because of its speed and non-invasiveness and is widely used in hearing screening. OAEs can be described as sound waves produced by the outer hair cells (OHCs) in the inner ear. There are different types of OAEs and some of them can be evoked using a stimulus. These OAEs are called evoked otoacoustic emissions (EOAEs). The activity of the OHCs can then be registered as follows. A probe containing a microphone and one or more speakers (depending on the type of EOAE that is recorded) is placed in the ear canal. A stimulus is sent through the speaker(s) and the resulting OAEs are recorded by the microphone. As the resulting OAEs have a low amplitude, it is of utmost importance that the probe used to measure them is
placed correctly. It has been shown that the reliability of the hearing test will be low if the probe does not fit well \cite{9,5}. Nowadays, there exist numerous software applications that support the user in evaluating the probe fit (this evaluation is also called a ‘checkfit’). However, the user has to be familiar with the feedback the system gives to be able to evaluate the actual hearing test correctly. Usually, before the actual OAE measurement, a click stimulus is sent through the ear canal and the frequency response and/or the impulse response is analysed. Often the evaluation of such responses is determined by experience and by learned rules \cite{5}, which can be found in handbooks and manuals \cite{3,4}.

In other words, it is advised to let an expert perform the hearing test. As mentioned above, it would be useful to have an open system which permits to check up on the hearing status without the need of an audiologist or other expert on the field. Such a system should evaluate the probe fit as if it were an expert, but most of all it should reflect the consequences of the fit. If it gives a probe fit a good evaluation, thus suggesting that a refit is not needed, the OAE measurement itself should be accurate and should not deviate much from a measurement taken in optimal conditions. Furthermore, it would be useful for experts to get additional information on the fit that they can interpret. This would prevent them from missing important aspects of the fit and would give them a reference.

### 1.3 Objective

The main objective of this thesis is to develop an application that can evaluate the probe fit automatically, without the help of any experts. Additionally, the application could also support experts with their decision making during the probe checkfit. The optimal solution should be able to reject probe fits that would have a significant (unwanted) influence on the final measurement.

### 1.4 Method

To achieve the objective described above, the following steps are followed. First, a literature review is given on OAE screening in general and especially on the fitting of the measurement probe. In the literature review two domains to achieve an expert model are explored, namely machine learning and fuzzy modelling. In the next chapter it is explained how sample checkfit
scores were collected given by experts, and how the acquired data was processed. The process of model construction and evaluation is explained. Finally, the chapter is concluded by a section on the construction of an Android application that allows automatic evaluation of the probe fit. The results of the model tests are given in Chapter 4. In Chapter 5 some aspects of the followed procedure and the results of the model tests are discussed. Future research possibilities are also presented. Chapter 6 concludes this thesis.
Chapter 2

Literature review

2.1 OAEs

Otoacoustic emissions or OAEs are low-level sound waves produced by the OHCs in the inner ear. The latter ones serve as acoustical pre-amplifiers that contribute to sound sensitivity and discrimination [2]. They can be seen as actuators which amplify low-level sounds while leaving high-level sounds as they are. This ‘pre-amplification’ causes a non-linear process taking place in the cochlea which results in energy propagated backwards towards the middle ear [10]. By placing a sensitive microphone in the outer ear, one can record the resulting air pressure fluctuations in the form of low-level signals [11].

There are two main types of OAEs [10]: spontaneous otoacoustic emissions (SOAEs) and EOAEs. The former are present in about 40 to 60 % of normal hearing individuals [3],[4] and therefore not used in clinical applications. Two subclasses of EOAEs are widely used in medicine: transient evoked otoacoustic emissions (TEOAEs) and distortion product otoacoustic emissions (DPOAEs). They are both favoured mainly because of the limited test-time and non-invasiveness [10, 12, 13, 4]. They are however measured by applying different techniques (see below) and arise from fundamentally different mechanisms [10].

Measurement of OAEs

The general setup for EOAEs measurement is shown in Figure 2.1. It should be noted that the noise reduction system can differ or can even be absent in some settings. While TEOAEs are evoked with a click or toneburst, DPOAEs are evoked with two pure tones [4].
2.1 OAEs

Figure 2.1: Schematic drawing of a system to assess evoked otoacoustic emissions in the presence of external noise. [1]

(a) TEOAE response waveform.  
(b) Barplot per frequency band.

TEOAEs

TEOAEs are measured by sending clicks or a toneburst through the ear canal. TEOAE responses can be split up in frequency bands to give a frequency specific indication of the cochlear status [2]. TEOAEs are usually only measured for the 1-4 kHz band, as above 4 kHz many clinically normal-hearing adult ears give weak responses. In young ears however the responses can be significant up to 6-7 kHz [2].

A sample of the response waveform can be seen in Figure 2.2a. The result can also be plotted per frequency band. This is shown in Figure 2.2b. The noise amplitude is shown in red while the energy amplitude is shown in blue.
2.2 Checking the probe placement

DPOAEs

DPOAEs are measured by sending two pure tones via two different loudspeakers through the ear canal. They can be measured in the 1-(6 up to 10) kHz interval, depending on manufacturer of the software [14]. As opposed to the measurement of TEOAEs, DPOAEs are measured per frequency band. The result of the response can be plotted in a DP-gram (see Figure 2.3): this shows both the noise and signal level.

2.2 Checking the probe placement

As mentioned above, OAEs are low-level signals. In order to be able to register these signals accurately, several measurement conditions must be met. First of all, the noise level needs to be kept at a minimum. This is usually not a problem when the measurement is done in an audiometric test booth. At environments with a significant amount of noise, noise reduction is needed. [15] discusses a noise reduction technique which can be used in such environments.

Furthermore, attention is needed during the probe placement [9, 4, 5]. Each time a test is performed a probe tip has to be chosen that fits the ear of the subject. Subsequently, once a good probe tip has been found, the executor of the test has to assure that the probe is inserted correctly and that nothing is blocking the probe.

Then, a so-called ‘checkfit test’ is performed. Usually an impulse (a click) is sent through a speaker in the probe and the impulse response is captured by a microphone in the probe. The expert can then analyse the impulse response in both time domain and frequency domain. Figure

Figure 2.3: Left: the DPOAE spectrum relates to f1 and f2 as: \( f_{dp} = 2 \cdot f_1 - f_2 \). Right: DP-gram with the response for specific frequencies. [2]
2.2 Checking the probe placement

2.4 shows several possible outputs for different placements of the probe. It must be noted that for DPOAEs the intensity levels for both speakers should be equal during the checkfit test, e.g. a canal can be obstructed by earwax.

![Figure 2.4: Different placements of the probe cause different impulse responses. A shows a normal probe placement, B shows a leaky fit and C shows a bad rubber tip fitting. Figure extracted from [3]](image)

In [4] the main important factors of a good impulse response for the checkfit are described. First of all, the positive and negative deflection of the stimulus should be symmetrical. Furthermore, the ringing, i.e. the signal after the stimulus, should be minimal. Additionally, the amplitude of the impulse response should have the desired level. This depends on the subject and the OAE device. Finally, the frequency spectrum should have a smooth rounded form.

[5] concluded that experts mainly look at 3 aspects. First of all, the symmetry of the positive and negative deflection. Secondly, the shape of the frequency response and thirdly the ringing. A broad frequency spectrum seemed to have a rather negative effect on the final OAE measurements. When suboptimal test conditions were compared to the baseline condition, the conditions with a broader spectrum than the baseline condition leaded to higher noise levels. Additionally, the degree to which experts refit a probe differs and can depend on the subject. The attributes described above are depicted in Figure 2.5.

In [3, 4] no numerical definitions are given. In [5] a spectrum is considered to be smooth if there are no dips of more than 5 dB in an interval of 1 kHz, ringing is considered to be minimal when it is lower than 0.1 Pa and shorter than 3 ms and the stimulus is symmetrical if the difference between the positive and negative deflection of the stimulus is lower than 0.1 Pa. However, the
2.3 Machine learning

In this section several techniques are described to achieve an expert system by deriving a classification and scoring strategy based on labelled data. These techniques belong to the domain of machine learning, a “Field of study that gives computers the ability to learn without being explicitly programmed” [16].

The objective of this thesis is to create a program that is able to perform the tasks an expert

Figure 2.5: Features used by expert to determine the quality of the probe checkfit

experts queried in [5] may have used own numerical decision boundaries.

(a) Impulse response with features.

(b) Frequency response with dip feature.
2.3 Machine learning

does when executing a hearing test: checking the impulse and frequency response during the checkfit phase (and eventually evaluate the measurement results). It is possible to program all of the instructions explicitly, together with some rules that need to be checked at certain moments in time. However, as been mentioned above, not all of these rules adhere to a standard. Some rules used are subjective and depend on the experience or background a certain individual has. For instance, some experts can have a different definition of a smooth frequency spectrum. This is where the ‘learning without being explicitly programmed’ can help. Machine learning permits to create a program that is able to identify rules or patterns by itself by processing data. There are several classes of machine learning. When both input data and output data are available, the ‘learning of the model’ is called ‘supervised learning’.

In machine learning, a computer program learns to execute a certain task by feeding it information. Here this task is evaluating probe placements and the information are checkfit data together with the expert evaluation of that checkfit data. The performance of the system can then be tested by feeding the system unseen checkfit data and letting it evaluate it. The outcome can then be compared to the expert evaluation of the unseen expert data. It can also be compared to deviations on OAE measurements: if a checkfit is evaluated as ‘good’, the OAE measurement should be accurate.

Cases where the model assigns a class to data samples are called classification problems, while cases where the model assigns a continuous output value to data samples are called regression problems. Both types are handled in this thesis. The first type of model can be used to output a class, e.g. ‘refit or proceed’, while the second type of model can be used to output a score.

2.3.1 The steps of supervised machine learning

Supervised learning is usually performed in four steps: data preparation, model selection, model validation and model testing [17, 18]. First, each input sample is transformed into a feature vector. A feature represents a certain characteristic of the input sample. Usually, features are extracted that explain the output data the most, this means that with those features it is possible to predict the output accurately. The set of data samples is split in a training set and a test set. The test set is used after the whole process to test the final model. Optionally, a validation set can be constructed, which contains samples other than the ones in test and training set. Usually this is not used if the amount of data samples is low. Other validation methods, such
as cross-validation, do not require a separate validation set \cite{19, 20}. Subsequently, a model is selected. This model can have several parameters. To know which parameters are best suited for the task, validation is performed. When the optimal parameters for the selected model have been found, the model can be tested on the test set.

### 2.3.2 Models

There exists a wide range of machine learning models. In this subsection a short description of the most used algorithms is given.

#### Linear regression

A simple technique that is often used for regression is linear regression \cite{19}. The model computes a continuous output given some feature vector \(\mathbf{x} = x_1 \ldots x_n\):

\[
y(\mathbf{x}) = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n
\]  

To find the weight \(w_i\) an error function is optimized, usually this is the Mean Squared Error (MSE):

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{Y}_i - Y_i)
\]

With \(\hat{Y}_i\) the predicted value and \(Y_i\) the known (correct) value for sample \(i\). Sometimes it is useful to construct additional features. For instance, all products of pairs of distinct features could be included. Then the final vector would be \(\mathbf{x} = x_1 \ldots x_n, x_1 \cdot x_2, x_1 \cdot x_3 \ldots x_{n-1} \cdot x_n\). This example is called the feature vector with interactions. Linear regression is a simple model that requires almost no memory (only the weights \(w_i\) need to be saved) and it can be evaluated very fast (one equation).

#### Logistic regression

Logistic regression is a binary classification algorithm. This algorithm is based on maximizing the likelihood function \(p(t|w)\) with \(t\) the training labels and \(w\) the set of parameters that the model uses \cite{19}. The output of this model gives the probability \(p(C_1|\phi_n)\), which is the probability that the class/label will be the ‘positive class’ \((C_1)\), given the feature vector \(\phi_n\) from a certain sample.
This property makes it possible to easily compute the receiver operating characteristic (ROC). The probability $p(C_1|\phi_n)$ is given by $\sigma(w^T\phi)$, with $\sigma(.)$ the sigmoid function. This function maps the input argument to the $[0-1]$ interval. The calculation time is therefore negligible for a small amount of features.

The ROC curve plots the false positive rate versus the true positive rate and is drawn by increasing a classification threshold for the class probability or score. The lower the threshold, the more samples that will be classified as positive. The true positive rate is estimated as the amount of positives correctly classified divided by the total amount of positives, while the false positive rate is estimated as the amount of negatives incorrectly classified divided by the total negatives \[21\]. Often the area under the curve (AUC) is used as a testing measure. An optimal classifier will have an AUC of 1, this means that there is a certain threshold for which all of the samples are correctly predicted (all positives correctly classified without any false negatives). An AUC of 0.5 means that the classifier produces random output. Figure 2.6a shows the ideal ROC-curve. A more realistic example is shown in Figure 2.6b. Furthermore, several methods exist to use logistic regression in a multiclass problem \[19\]. Sometimes the model gets biased too much by the training data and does not generalise well anymore. This is called over-fitting. A ‘regularization’ parameter can be added to avoid this. The higher this parameter, the more noise that will be added to the data and the less the model will over-fit. Of course, the more noise, the lower the accuracy of the model. A good trade-off needs to be searched by means of validation.

**Decision trees**

A decision tree is a directed connected graph without cycles. Each node represents a condition to be checked and each edge represents an outcome of such a condition. Each leaf node represents an output. One node of the graph does not have any edges pointing to it. This is called the root node: this node represents the first condition to be tested. Decision trees can be used for many purposes, especially for medical applications \[22\]. First of all, they are easy to use: no computations need to be made, only conditions need to be tested. They can thus be interpreted by a medical expert (i.e. a medical expert can easily see why a certain output is given). Figure 2.7a shows an example of a decision tree which can be used by experts to classify heartbeats \[23\]. Furthermore, they can be easily implemented using if-then conditions and can be evaluated fast. Finally, they are able to deal with nominal values. Decision trees can be used both for
2.3 Machine learning

classification and regression: this means the output can be discrete or continuous. There are however some drawbacks to this technique [24]. First of all a decision tree is susceptible to over-fitting. Therefore, additional measures need to be taken, e.g. pruning of the tree. Furthermore the model is not stable. A slight change in one input parameter may result in a totally different outcome. This happens near a decision boundary: two different paths are taken for two similar values at different sides of that boundary.

**Random forests**

To cope with the limitations of a single decision tree, a collection or ‘bag’ of decision trees could be used. Each tree in the collection is trained on different subsets of the data. These subsets are obtained by drawing samples with replacement from the original data set (bootstrapping). Furthermore, during training for each tree node the feature that is used for the condition is chosen from a subset of the features (size $\sqrt{N}$ with N the amount of features) [26]. The collection of trees can be compared to a group of medical experts: each expert has an own background and experience. A patient can ask several experts for their diagnosis. The final result can be derived from the collection of diagnoses [27]. Random forests are widely used in medicine: e.g. for cancer diagnosis [28], heart problem detection [29], heartbeat classification [25] and other important applications [30]. Random forests can also be used for variable ranking [26, 31]. As each tree

(a) Example of an ideal ROC curve (AUC = 1)  
(b) Example of a realistic ROC curve (AUC ≈ 0.9)

Figure 2.6: ROC curve examples.
2.3 Machine learning

(a) Example of a decision tree used to classify heartbeats. Source: [23]

(b) Features of the heartbeat waveform. Source: [25]

Figure 2.7: Example of using a decision tree for heartbeat classification.

uses a subset of training samples, the remainder, called out-of-bag (OOB) samples, can be used to test the importance of each variable. This is done as follows. For every tree the OOB samples are tested and the number of correct predictions is counted. Then, one feature is chosen and the values for that features are randomly permuted among the OOB samples. The number of correct predictions is again counted. A score for the selected feature is obtained by subtracting the last number of the first number for each tree and averaging the results [26].

Support vector machine

A more advanced classification technique than logistic regression is the support vector machine (SVM). This technique tries to find the optimal decision boundary that satisfies the training samples [19]. This is only possible if the data samples are linearly separable. A simplified example is given in Figure 2.9. Sometimes, it is not possible to find such an optimal decision boundary, and thus certain ‘slack variables’ need to be taken into account. The more slack variables that are permitted, the more the model will be regularized (more detail will be lost), the less slack variables, the higher the probability on over-fitting. The amount of permitted slack variables can be controlled with the parameter ‘box constraint’. The lower this parameter, the more slack variables allowed, and thus the more regularization [19].

SVMs are frequently used because of the compatibility with kernel functions. A kernel function
permits to model higher dimensions (even infinite dimensionality). For instance, instead of features $x_1$ and $x_2$ the kernel function $(x^Tz)^2$ gives the feature mapping $(x_1^2, \sqrt{2}x_1x_2, x_2^2)$ and the kernel $\exp(-||x - z||^2/2\sigma^2)$ gives a feature mapping of infinite dimensionality (due to the exponential function with Taylor series expansion) [19]. It is sometimes possible that a kernel itself has a regularization parameter. For instance, the Gaussian kernel has a size parameter. The bigger the kernel, the more regularization and the less probability on over-fitting. [32, 33] use SVMs for heartbeat classification.

**K-nearest neighbours (KNN)**

This technique does not require a training phase. However, it requires all of the training samples to remain in the memory. Output computation is done as follows: given a new data sample, calculate the distance to all of the training samples (given a certain distance measure). Take the K closest samples and perform majority voting/weighted majority voting if the output is discrete, or compute the average/weighted average if the output should be continuous. The parameter K can be considered as a regularization parameter. The smaller K, the less neighbours that will determine the output and the higher the probability that there will be over-fitting (e.g. due to outliers/noise/bad samples). The higher K however, the more detail that will be lost and the worse the final prediction. Thus, a certain trade-off needs to be found. This technique has been successfully applied to classify heartbeats [34].
2.3 Machine learning

(a) An example of linearly separable samples in a two-dimensional space.

(b) Different solutions for a linear decision boundary exist.

(c) The optimal decision boundary. Samples on the margin are called support vectors.

(d) Sometimes it is not possible to have a linear separation. Some slack variables (grey) are allowed.

Figure 2.9: Maximal margin classification
2.3 Machine learning

2.3.3 Additional notes on machine learning

Machine learning techniques can acquire an accurate prediction power. Sometimes they can even outperform experts. However, as everything is learned from data these techniques will not work if data quantity and/or quality is not sufficient. In general, when using data, following conditions need to be met [35]:

- Data is complete.
  It is assumed that the training data give a good representation of the real world. In machine learning: when a model exploits this assumption too much, the model over-fits. That is why certain measures need to be taken (e.g. regularization, tree pruning).

- Data is consistent.
  It is assumed that the same input data sample is always given the same label. This will not always be the case as different experts can give different labels, an expert can make a mistake, ...

- Data is correct. (‘accuracy’ of data)
  The data samples that are used need to conform to the real-world facts or values. The data should contain as few errors as possible.

- Data is not out of date. (‘timeliness’ of data)

- Data is relevant.

For the specific problem handled in this thesis, it can not always be assumed that the data is complete, consistent and accurate. Collecting enough labelled data requires several experts with different backgrounds to label a significant amount of data samples. Moreover, the same expert can label differently under different circumstances (e.g. when time is of the issue). This could make data inconsistent if the same samples are evaluated a second time. Furthermore, a score for a certain checkfit given by one single expert might not be very accurate, e.g. it is hard to score a checkfit of medium quality.

Finally, most of the techniques mentioned above are a ‘black box’, i.e. it is not clear how a certain outcome is obtained. If the expert system is to be used by experienced people, such a black box model will not be satisfactory. Experts want to know why a certain decision has been made, and statements as ‘most of the trees decided to accept the fit’, ‘the five closest neighbours voted yes’ will not make those decisions very understandable. On the other hand, decision trees
(which can also be constructed using machine learning) are intuitive. However, ‘hard’ decision boundaries are used, it is rather hard to construct a decent decision tree that can be used in general (as different experts use different decision boundaries or even look at different features) and decision trees tend to become big and complex if several conditions need to be considered at the same time (instead of testing one condition at a time).

2.4 Fuzzy logic

In Section 2.3, several methods to achieve a good expert model by means of labelled data were described. In 2.3.3 some challenges and disadvantages of machine learning techniques were discussed. Fuzzy logic counters some of these disadvantages, namely the strict decision boundaries of the decision trees and the non intuitiveness of the other techniques. First of all fuzzy logic tries to loosen the strictness of decision boundaries by using a continuous transition to another class (i.e. instead of good-bad, detailed gradations are used, such as ‘good 1, good 0.8, good 0.5, bad 0.4, bad 0.6, bad 1’; note that bad and good can overlap). Secondly it takes into account the strategies of experts by means of rules. Finally, changes to the decision boundaries can be made without needing to change the whole structure (the rules themselves can remain the same, only some parameter values need to change). Fuzzy logic can also be combined with machine learning to learn parameters from the data.

2.4.1 General structure of a fuzzy model

A fuzzy model consists of four main parts [36]:

- Knowledge base: consists of a rule base and a database.
- Fuzzification interface: transforms crisp input values into fuzzy values.
- Decision-making unit: performs inference operations on the rules.
- Defuzzification interface: aggregates fuzzy results into one or more crisp outputs.

The ‘crisp values’ represent numerical values not associated with linguistic terms, e.g. a numerical score. Figure 2.10 gives a schematic overview of these modules. Each module is discussed in more detail in the next paragraphs. It should be noted that there are different types of fuzzy inference systems (see Subsection 2.4.2) which implement the modules in a specific manner.
The modules will be illustrated with an example of the Mamdani-type, the most intuitive and commonly used fuzzy inference type.

**Knowledge base**

The knowledge base consists of a database and a rule base. The database contains both the membership functions used by the fuzzification interface to transform crisp input values into fuzzy values and the membership functions used by the defuzzification interface to create crisp output values. An example of a set of membership functions is shown in Figure 2.11. The use of these membership functions is explained in the next paragraph. The rule base consists of one or more rules. These rules all have an if-then layout, for example ‘if the service in the restaurant is good and the quality of the food is medium, then tip is medium’. The part before ‘then’ is called the antecedent, while the other part is called the consequent. Optionally, a weight can be given to each rule, to signify the relative importance.

**Fuzzification interface**

In this module crisp data values are transformed to fuzzy values. Suppose a tipping system needs to be created. A data sample consists of a pair of scores: one for the quality of the food, another one for the quality of the service. Considering the quality of the service, it can be said that there are three categories: poor, good and excellent. A score of five means that the service was good. A score of six means also that the service is good, however, it is better than a score
Figure 2.11: A set of three membership functions for the variable ‘service’. A value of 5 results in a Y-value of 0 for both the membership functions ‘poor’ and ‘excellent’, and a Y-value of 1 for the membership function ‘good’.

of five. This is reflected in the fuzzy values for the membership function ‘excellent’: five will result in a lower fuzzy value than six. In short, after fuzzification the degree to which each part of the antecedent is satisfied for each rule is known [37].

Decision-making unit

The decision-making unit uses the fuzzy input values together with the rules present in the rule base to compute fuzzy output sets. Such a set can be seen as a modified output membership function. First, the antecedent of a rule is evaluated. This results in one output number. This number is then used by the implication to determine a fuzzy output set. In Mamdani systems, there are several functions that can be used for the rule inference, the most commonly used functions are the maximum function to represent the a fuzzy ‘or’ in rules and the minimum function to represent a fuzzy ‘and’ and a fuzzy implication.

Suppose the rule ‘if service is good OR food is delicious, then tip is average’ needs to be evaluated, and the input for service is 6.74 and for food is 7.77, as given by the red lines in Figure 2.12. The membership function ‘good’ for the variable service is shown on the left, the membership function ‘delicious’ for the variable food is shown in the middle. The maximum output of the two membership functions is taken to truncate the output membership function ‘average’ [37]. The output is in fact a new function or set of (X,Y)-values.
2.4 Fuzzy logic

2.4.2 Mamdani and Sugeno fuzzy models

There are many types of fuzzy inference systems, but the two most commonly used are Mamdani and Sugeno. While Mamdani is often considered to be the most intuitive, Sugeno is computationally efficient and can be easily optimized by means of machine learning (see Section 2.4.3) [37, 38].

The main difference between the two is the implementation of the decision-making module and the defuzzification interface. In Sugeno modelling, instead of using the min-operator (or another one used for implication in Mamdani) on a rule output, the rule output is used as a weight. This
2.4 Fuzzy logic

Figure 2.14: An example of an ANFIS structure.

weight is multiplied by either a constant or a variable that is a linear combination of inputs. Now there are no output membership functions but constants and/or linear equations (i.e. linear combinations of inputs). The final result is given by [37]:

\[
\frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}
\]

(2.3)

with N the number of rules, \(w_i\) the output of each rule and \(z_i\) the output constant or linear combination of inputs for the rule.

2.4.3 Neuro-fuzzy modelling

A classic fuzzy expert model is completely based on the experience of experts. A machine learning model is completely based on data. It is possible to combine them to obtain a model which is more accurate. A widely used example is the adaptive-network-based fuzzy inference system (ANFIS). This model tries to optimize the parameters of the membership functions, while keeping the initial structure of the model. This is not possible for the Mamdani systems, but is widely used for Sugeno models.

First, the structure is modelled by a feedforward neural network. This is shown in Figure 2.14. Then, the whole set of training samples is propagated towards the end of the network, the error between the predicted outcome is propagated back to the start of the network. The parameters from the membership function are adapted to minimize this error. It is very important to use validation and train on enough data samples to avoid over-fitting [37]. Each forward-backward pass of the whole training set is called an epoch [36].
Neuro-fuzzy modelling is also widely used in medicine: e.g. heart-beat recognition [39] and medical image classification [40]. Often a Mamdani model is constructed first, as this is easily constructed with expert knowledge. Then a Sugeno equivalent is constructed, which can be further optimized using the neuro-fuzzy technique.
Chapter 3

Methodology

In order to find a good strategy for automatic checkfit evaluation, both machine learning and fuzzy logic were explored. In this chapter it is explained how the research objective was tackled. First, the problem is defined and the data preparation is explained. Then, the construction of machine learning models is described. Subsequently, the process of fuzzy model construction and neuro-fuzzy optimization is discussed. Then, an overview is given of all models and tests. Finally, the development of a checkfit Android application is described.

3.1 Problem definition

The first objective is to have an application that is able to classify or score probe fits. The resulting class or score should reflect an expert’s opinion and it should be an identifier of fits that will cause large deviations on OAE signal measurements (large deviations could lead to false conclusions). Therefore both expert data and objective OAE measurements should be taken into account. Additionally, information on the impulse and frequency response could be provided, in a way that the application could also support experts in their decision making. The second objective is to create such an expert support system.

Machine learning models can be trained on two types of expert data: class labels and real numbers. Models trained on data of the first type are classification models and models trained on data of the second type are called regression models. Neuro-fuzzy optimization also uses data of the second type. In this thesis the class labels considered are binary: ‘refit needed’ or ‘no refit needed’. The real numbers are scores between 0 and 100. Fuzzy models can in general provide additional feedback using the fuzzy membership functions. For each variable the membership
function with the highest Y-value can be given. For example, if a certain parameter has a Y-value of 0.2 for the membership function ‘bad’, 0.5 for ‘medium’ and 0.1 for ‘good’, the system can indicate that the value for that variable is ‘medium’.

3.2 Data preparation

To build a model with machine learning, it is important to extract, process and transform useful data. The same goes for fuzzy models, since they require input in a numerical format. Additionally, ANFIS models require data to train on.

3.2.1 Data selection

Checkfit data sample

Checkfit data samples were extracted from data files present in an existing database. This database contained data written by the ILO-system, a system for clinical OAE analysis and data management created by Otodynamics Ltd. [41]. Each checkfit data sample consists of the impulse response (length 5 ms), sampled at 25600 Hz (i.e. 128 samples for 5 ms).

Datasets from previous work

In [5] two datasets were constructed. The first dataset contains 48 impulse responses and their expert labels. Eighteen experts were asked to label all of the 48 impulse responses by means of a Q distribution [42]. Figure 3.1 gives an example of a Q distribution.

For the construction of the second dataset, TEOAE measurements were performed on 34 normal hearing subjects for nine different conditions. One condition was considered to be the baseline, i.e. the ideal condition. The other ones were suboptimal and thus there was a certain difference in OAE signal level and noise level compared to the baseline condition. Each measurement was preceded by a checkfit test. This resulted in a dataset consisting of the impulse response and deviations (= absolute differences) on both noise and OAE signal level.

In this thesis, the first set is used as a test set to see if constructed models conform to an expert’s opinion. The dataset is not used to train a model on as it does not satisfy the conditions mentioned in 2.3.3. The amount of samples is low and can not be used as a general example of the real world: one can not assume that enough impulse response possibilities are present. Furthermore, this dataset was built for the identification of important features, such as ‘sym-
Figure 3.1: An example of a Q distribution. Each sample is assigned to a column. Each column has a fixed amount of slots. Each column represents a label or score, given by the value in the upper row.

Figure 3.2: The distribution of averaged expert labels.

metry of stimulus’ and ‘length of ringing’, not for the training of a model. The numerical labels (ranging from -3 to 3, see Figure 3.1) of the 18 experts were averaged. There turned out to be one duplicate with the same averaged label. Therefore this sample was removed from the test set. The distribution of the averaged labels is shown in Figure 3.2.

The second dataset is also used as a test set, rather than a training set. This choice was made to prevent over-fitting on the data. Only 255 samples were available and a large portion was needed for the testing (to have a general idea of the deviation), which would leave out too few samples for training. Based on these data samples a binary label set was constructed. The data samples were split into two groups by using the following condition. A sample was considered to be a bad one if it had a total signal deviation of more than 2.5 dB or a signal deviation of
Table 3.1: Table of participants

<table>
<thead>
<tr>
<th>Expert</th>
<th>Graduation Year</th>
<th>Profession</th>
<th>Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2015</td>
<td>Master’s Student</td>
<td>2 times a week</td>
</tr>
<tr>
<td>2</td>
<td>1998</td>
<td>Audiologist</td>
<td>Daily</td>
</tr>
<tr>
<td>3</td>
<td>2011</td>
<td>Audiologist</td>
<td>10 times a week</td>
</tr>
<tr>
<td>4</td>
<td>2002</td>
<td>Clinical Audiologist</td>
<td>/</td>
</tr>
<tr>
<td>5</td>
<td>2010</td>
<td>Audiologist</td>
<td>20 times a week</td>
</tr>
<tr>
<td>6</td>
<td>2015</td>
<td>Master’s Student</td>
<td>Internship</td>
</tr>
<tr>
<td>7</td>
<td>2005</td>
<td>Postdoc Audiology</td>
<td>10 times a month</td>
</tr>
<tr>
<td>8[*]</td>
<td>2011</td>
<td>Audiologist</td>
<td>/</td>
</tr>
</tbody>
</table>

For this thesis a Java application was created to serve two purposes: data collection and fuzzy model evaluation. Information on the architecture of this application can be found in Appendix A. The application was developed in NetBeans IDE 8.0. Table 3.1 gives an overview of the participants who used the application. Each participant labelled 50 samples randomly drawn from a dataset consisting of the 255 samples mentioned above and 264 other samples extracted from an ILO database. The data from the last participant (marked with [*]) were collected in the end and were used only for the neuro-fuzzy model training. The samples were drawn randomly each session to have random overlap.

The labelling itself comprised of two questions. Two graphs were shown, namely the impulse and frequency response, similar to the ones shown in the ILO system (same axes bounds and units). Participants were told that the fits were from adults and that time was not an issue here. This was done based on the argument that potential users of the final application will be able to take their time and want the measurement to be as accurate as possible. This means that it is better to refit when in doubt.

Two questions were asked:

- Which score would you give this probe fit?
3.2 Data preparation

Figure 3.3: The labelling phase in the Java application

- Would you refit this probe?

Figure 3.3 shows the interface that was used. In the end, two labels were obtained: an integer score in the range [0-100], and a boolean ‘refit’ true or false.

As the data samples were drawn randomly, some overlap was present. The final tuple set (label, score) contained 266 tuples obtained by averaging the overlapping tuples. 128 samples out of 266 are from the second dataset described in 3.2.1. The distribution of the scores is given by Figure 3.4. 159 out of 266 samples were labelled as ‘refit needed’. In the end, one new expert labelled 50 data samples. This dataset was only used as a checking dataset for the ANFIS models.

Each expert evaluated a Mamdani fuzzy model via the labelling application. After the scoring form shown in Figure 3.5, a second form was shown were experts could see additional information on the impulse and frequency response. The amount of parameters and the classification of the parameter values depended on the model itself. Additionally, a system reference score was given by the model. The experts were asked if they agreed with the parameter classification. Furthermore, the opportunity was given to modify the original score and refit label that was
3.2 Data preparation

Figure 3.4: Labelling app scores distribution. Each bar represents an interval of 10%.

given in the previous phase. A sample form is shown in Figure 3.5. After the labelling session each expert was asked an opinion about the overall system performance. This information was then used to create a next model (hence five different models).

3.2.2 Data preprocessing

The frequency response was retrieved by performing the Fast Fourier Transform (FFT) on the impulse response. It was then transformed to a dB scale using the formula:

\[ Y' = 20 \cdot \log_{10}(Y) + C \]  

With C a constant. Here a constant of 85 dB was used so that the output would be the same as the output in the ILO system. Only the range [0,6] kHz was considered, as this is the output shown in the ILO system.

3.2.3 Data transformation

After the data is preprocessed, the data can be transformed into a feature vector or set of features. A feature represents a characteristic of the input data sample. Such a characteristic can be used to predict the final output. Therefore features are often called ‘predictor variables’. In this subsection the used features are given and the extraction/computation method is described.
3.2 Data preparation

Figure 3.5: Fuzzy model evaluation in a Java application
The features annotated with I.[number] are impulse response features and the features annotated with F.[number] are frequency response features. The frequency spectrum covers a range of [0,12600] Hz. For the frequency response features mentioned below, the range [600, 5000] Hz was considered. In the ILO-system, only the [0,6000] range is plotted.

I1. First and second peak index
While this is not a feature that is used for analysis, it is required to compute other features such as the difference of peak amplitude and the amplitude of the ringing. Two methods were compared. The first one is simply looking for the minimum and maximum value of the complete signal. The other method uses local extrema. A detailed description can be found in Appendix C, Algorithm 3.

Using local extrema gives better results as shown by Figure 3.6. There are however two assumptions made in the second method:

- The stimulus (here it is referred to as the ‘S-shape’) starts at the first sample s where $|s| > \epsilon$
- The two local optima that are found after the start of the S-shape represent the two peaks

Before the actual stimulus, the signal can go slightly down or up. Therefore a good $\epsilon$ needs to be chosen to skip his part. In this research a value of 0.025 Pa was used.

I2. Amplitude of ringing
The ringing starts after the actual stimulus or S-shape. A formal description of the algorithm to detect the start of the ringing is given in Appendix C, Algorithm 2. Figure 3.7 shows the S-shape in blue. A high amplitude of ringing ($> 0.1$ Pa) indicates a bad probe fit. Figure 3.8 shows the difference between using global and local extrema to locate the start of the ringing.

I3. Absolute difference of first and second peak
After finding the indices of the S-shape extrema, this feature can be computed with

$$\delta_{\text{peak}} = ||y(\text{peak}_1)| - |y(\text{peak}_2)||$$

This feature depicts the asymmetry of the stimulus, which is an important characteristic of the impulse response [5, 4].
3.2 Data preparation

I4. Maximum amplitude
The maximum amplitude of the whole signal (max abs(y)). If the maximum amplitude of the impulse response is too low (< 0.2 Pa) or too high (> 0.4 Pa), the probability on a bad fit is high [5]. It should be noted however that the required stimulus intensity level may vary among OAE devices [4].

I5. Maximum amplitude of S-shape
This feature is strongly related to the previous one, however only the S-shape is considered instead of the whole signal. For the scenario given in Figure 3.6 this would not make any difference, as the maximum would still be the minimum peak. However, if the global maximum would be the positive peak, this feature would have a different value than the previous one.

I6. Mean amplitude S-shape
This is another parameter that could represent the intensity of the impulse response, like I4 and I5. Here, the average of the amplitude of the two stimulus peaks is calculated.

I7. Length of ringing/length of the impulse response
The length of the ringing is the time it takes for the impulse response to fade out. This means...
3.2 Data preparation

Figure 3.7: The S-shape (= stimulus) of the impulse response followed by ringing.

Figure 3.8: Difference in result between using global or local extrema to identify stimulus peaks: using local extrema a much higher amplitude of the ringing is discovered (which is equal to the global maximum in this example).
that the length of the S-shape is included in this feature. Therefore, it would be more intuitive to say ‘length of the impulse response’. However, as ‘length of the ringing’ is used by experts, this is the term that is used in the rest of this thesis. When the impulse response is smaller than a certain constant $\epsilon$, the ringing is assumed to have stopped. If this length is $> 3$ ms, the probability is high that a refit is needed [5, 3]. An $\epsilon$ of 0.015 Pa was used for model testing, after it became clear during the labelling sessions that 0.005 Pa was too low.

**I8. Amplitude ratio**

As the absolute value of the ringing often depends on the amplitude of the S-shape, another feature to indicate the amplitude of the ringing can be used. This is given by the ringing amplitude divided by the maximum amplitude. Alternatively, the mean amplitude of the S-shape can be used as a divisor. In what follows, the maximum amplitude is assumed, unless stated otherwise (then it is referred to as I8bis).

**F1. Dips**

A dip is detected by looking in a window for a fall and rise of a certain amount in dB. Algorithm 4 in Appendix C shows how dips are detected. In this work, a value of 3 dB was used as a threshold and a window of 2 kHz.

**F2. Maximum dip**

This feature represents the biggest dip in a certain search window. For this thesis a search window of 2 kHz was chosen.

**F3. Peaks**

A similar algorithm as used for detecting dips was used for peak detection.

**F4. Max Peak**

A similar algorithm as used for detecting the maximum dip was used for the detection of the biggest peak.

**F5. Spectral flatness**
The spectral flatness is defined by:

\[
\frac{\text{geometric mean}}{\text{arithmetic mean}} = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{1}{N} \sum_{n=0}^{N-1} x(n)}
\]

This feature can be used for a certain range of the frequency spectrum, to see whether or not the spectrum is flat enough.

**F6. Bin distribution**

To have an idea of how the samples are distributed on the dB axis, the range between the maximum and minimum value can be divided into a certain number of bins. For each bin the number of samples that fall into the bin range are counted.

**F7-11. Band deviations/means/skewness/kurtosis/spectral flatness**

The considered range of [600-5000] Hz can be divided into several bands with overlap. Per band the standard deviation can be computed. In this thesis, we used bands of 1 kHz with 0.5 kHz overlap, and bands of 0.5 kHz with 0.25 kHz overlap. A similar technique can be used to obtain the mean/skewness/kurtosis/spectral flatness per band. The skewness is a statistical measure for asymmetry. The kurtosis is a statistical measure for peakedness (a high kurtosis indicates the presence of a pronounced peak).

### 3.3 Machine learning

In the following section the method to train and select machine learning models is explained.

#### 3.3.1 Model selection

The machine learning models were constructed and tested in Matlab® 2015. Table 3.2 lists the models that were trained and tested for the classification problem and Table 3.3 lists the models that were trained and tested for the regression problem. The classification models were trained on refit-no refit labels while the regression models were trained on scores. Each technique has several parameters which were tested using validation. Most of these parameters were discussed in 2.3.
Table 3.2: Evaluated classification techniques.

<table>
<thead>
<tr>
<th>Model number</th>
<th>Type</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Decision tree</td>
<td>maximum number of splits in each node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>minimum training samples per leaf</td>
</tr>
<tr>
<td>2</td>
<td>Linear SVM</td>
<td>box constraint</td>
</tr>
<tr>
<td>3</td>
<td>Logistic regression</td>
<td>regularization</td>
</tr>
<tr>
<td>4</td>
<td>KNN</td>
<td>number of neighbours</td>
</tr>
<tr>
<td></td>
<td></td>
<td>distance weight</td>
</tr>
<tr>
<td>5</td>
<td>Gaussian SVM</td>
<td>box constraint</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kernel size</td>
</tr>
</tbody>
</table>

Table 3.3: Evaluated regression techniques.

<table>
<thead>
<tr>
<th>Model number</th>
<th>Type</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linear regression</td>
<td>mode</td>
</tr>
<tr>
<td>2</td>
<td>Decision tree</td>
<td>maximum number of splits in each node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>split criterion</td>
</tr>
<tr>
<td>3</td>
<td>KNN</td>
<td>number of neighbours</td>
</tr>
<tr>
<td></td>
<td></td>
<td>distance weight</td>
</tr>
</tbody>
</table>

### 3.3.2 Feature space dimension reduction

It is important to keep the feature dimensionality as low as possible for two main reasons. First of all many techniques suffer the so-called ‘curse of dimensionality’ when using a large feature space (i.e. a large amount of different features per data sample), especially when the training set is rather limited [43]. One can say that high dimensional spaces are not intuitive and therefore complex to deal with. This results in over-fitting to the training data. The feature space becomes very sparse which makes it easy to find some decision boundary and to score high on the training data. However, this decision boundary will not generalize well [44] and the model will score badly on test data. On the other hand, the less features that are used, the less computation time needed, both for training as for predicting.

Two feature spaces were considered: one containing nine expert features listed in 3.4 and one containing all features described above. The first feature set was taken into account as it is
3.3 Machine learning

<table>
<thead>
<tr>
<th>Feature</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplitude ratio with max (I8)</td>
<td>Robust representation of ringing amplitude.</td>
</tr>
<tr>
<td>Amplitude ratio with mean (I8bis)</td>
<td>Robust representation of ringing amplitude.</td>
</tr>
<tr>
<td>Mean amplitude S-shape (I6)</td>
<td>Idea of the amplitude of the stimulus.</td>
</tr>
<tr>
<td>Difference in peak amplitude (I3)</td>
<td>Asymmetry of the sinus shape.</td>
</tr>
<tr>
<td>Ringing length (I7)</td>
<td>Length of ringing.</td>
</tr>
<tr>
<td>Ringing amplitude (I2)</td>
<td>Absolute value of ringing amplitude.</td>
</tr>
<tr>
<td>Dips (F1)</td>
<td>Smooth curve will not have dips.</td>
</tr>
<tr>
<td>Maximum dip (F2)</td>
<td>Large dip is worse than a small dip.</td>
</tr>
<tr>
<td>Maximum amplitude (I4)</td>
<td>Idea of the amplitude of stimulus.</td>
</tr>
</tbody>
</table>

Table 3.4: Nine expert features derived from literature [4, 3, 5].

common practice to look at the domain knowledge first [45]. For this feature space, features can be selected in an incremental manner: a feature is added that gives the best score until adding a feature does not improve the score. For the other feature space this would take too long and therefore another technique is used, namely the feature ranking method described in 2.3. More specifically, the technique described in [46] was used. A random forest is trained, while the OOB samples are used to rank the features. The OOB prediction error, i.e. the error of the forest tested with the OOB samples, is saved. Subsequently, 80 % of the highest ranked features are used to train a new random forest with. The prediction error is again computed. This is done until no features are left. In the end, the feature set that leads to the lowest error is kept. 1000 trees were used per forest. For models trained on all features, the same final feature set was used, namely the outcome of the feature selection with random forests. During the model validation five different feature sets were used, one for each fold.

3.3.3 Validation

When training a model with specific values for the model parameters, it is important to validate the model by using a validation set. One can not validate on test data as this could lead to overfitting [19, 20]. When the amount of available data samples is scarce, as it is here, an alternative method of validation can be used, called ‘k-fold cross-validation’. The training data set is split up in k parts. Subsequently, the validation procedure is done as described in Algorithm 1. In this thesis five folds were used, as this splits the training set in 80 % training data and 20 %
test data, a ratio that is used often [47]. It is important that dimension reduction is performed in the inner loop, as for each iteration a new training set is used and otherwise the selection of features would be influenced by the samples that are used as test samples. Therefore, the feature selection method described above was performed for each fold, resulting in five different feature sets. The resulting mean error depicts the error for a certain technique with a specific parameter value set $S$. For classification, the AUC was used (see 2.3.2) as error measure, while for regression the MSE was used. The same partition of the data was used for all models.

Data: number of folds $k$, training set, model parameter set $S$

Result: mean error

1. Divide training set in $k$ parts;
2. $\text{sum} = 0;$
3. $\text{for } i = 1 \text{ to } i = k \text{ do}$
   4. Test set = part $k$;
   5. Training set = all parts except part $k$;
   6. $M = \text{model trained on training set with parameter set } S;$
   7. $P = \text{predict test set with } M;$
   8. $E = \text{prediction error } P \text{ compared to test set labels};$
   9. $\text{sum} = \text{mean} + E;$
10. $\text{end}$
11. mean error = $\text{sum} / k;$

Algorithm 1: Performing k-fold cross-validation

### 3.4 Fuzzy modelling

#### 3.4.1 Mamdani models

Four Mamdani models were evaluated using the labelling application. A fifth one was created in the end. Table 3.5 lists the input and output variables for each model. As the parameters described here are not intuitive to experts, the parameter names were replaced by more familiar terms: these names were then shown on the screen. This mapping is shown in Table 3.6. It can be noted that several technical parameters were tested to represent the same expert parameter, for example the flatness of the frequency response. The variable ‘has dips’ is simply 1 if the amount of dips (F1) > 0 and 0 otherwise, thus having a binary output. Table 3.7 lists the
amount of rules per model. Table 3.8 lists the experts together with the model they evaluated. The Mamdani models were constructed and evaluated consecutively. Each model copes with remarks of experts who evaluated the previous model, with model 1 being the initial pilot model. Table 3.9 lists the improvements for each model, given the remarks of the previous ones. Model 5 represents the final optimization of model 4 and was not tested using the labelling application. It should be noted that the actual data analysis was done in the end, no new models were constructed based on numerical data.

3.4.2 Neuro-fuzzy modelling

Mamdani to Sugeno conversion

To be able to optimize a Mamdani model, a Mamdani to Sugeno conversion was used, namely the mam2sug built-in Matlab function. This conversion outputs a Sugeno model that has constant membership output functions: the center of area value for a given output membership function is used as constant [37]. This can change to linear membership output functions using the neuro-fuzzy optimization technique. As the optimization technique requires the membership output functions to not be shared between rules, each membership function that was shared was split up in multiple membership functions with the same parameters.

Training

Data samples collected from all experts except the last one were used for the training of the model, using the backpropagation method together with a least squares type of method in Matlab®2014. This hybrid method prevents getting stuck in local optima [36]. The number of epochs, i.e. the number of forward and backward passes was set on 50. The error tolerance was set on 0 so that the training would only stop after reaching the desired number of epochs. This was chosen based on the argument that it was unknown how the error would behave [37]. The training phase can shrink input intervals for some parameters. However, when tested on new data, unseen samples can fall outside of the input range. Therefore the input intervals were extended to their original size, while leaving the parameters of the membership functions intact (except the outer ends to cover the whole input area). This process is illustrated in Figure 3.9.
### Table 3.5: Constructed Mamdani models. Models 1-4 were evaluated by at least one expert. Model 5 was created in the end.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter</th>
<th>Membership functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Symmetry (I3)</td>
<td>bad, medium, good (gauss)</td>
</tr>
<tr>
<td></td>
<td>Ampratio (I8)</td>
<td>bad, medium, good (gauss)</td>
</tr>
<tr>
<td></td>
<td>Has dips (F1)</td>
<td>bad, good (trap)</td>
</tr>
<tr>
<td></td>
<td>Length of ringing (I7)</td>
<td>bad, good (gauss)</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>bad, medium, good (gauss)</td>
</tr>
<tr>
<td>Model 2</td>
<td>Symmetry (I3)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Ampratio (I8)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Has dips (F1)</td>
<td>bad, good (trap)</td>
</tr>
<tr>
<td></td>
<td>Length of ringing (I7)</td>
<td>bad, good (trap)</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>bad, medium, good (trap)</td>
</tr>
<tr>
<td>Model 3</td>
<td>Symmetry (I3)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Ampratio (I8)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Max dip size (F2)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Length of ringing (I7)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>mf1, mf2, mf3, mf4, mf5 (triangle)</td>
</tr>
<tr>
<td>Model 4</td>
<td>Symmetry (I3)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Ampratio (I8)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Max dip size (F2)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Length of ringing (I7)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Max amplitude of S-shape (I5)</td>
<td>low, good, high (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>mf1, mf2, mf3, mf4, mf5 (triangle)</td>
</tr>
<tr>
<td>Model 5</td>
<td>Symmetry (I3)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Ampratio (I8)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Max dip size (F2)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Length of ringing (I7)</td>
<td>bad, medium, good (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Mean amplitude of S-shape (I6)</td>
<td>low, good, high (trap, triangle)</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>mf1, mf2, mf3, mf4, mf5 (triangle)</td>
</tr>
</tbody>
</table>
### Table 3.6: Display names of parameters which were shown in the labelling application.

<table>
<thead>
<tr>
<th>Technical name</th>
<th>Display name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetry (I3)</td>
<td>Symmetry of the response</td>
</tr>
<tr>
<td>Ampratio (I8)</td>
<td>Amplitude of the response</td>
</tr>
<tr>
<td>Has dips (F2)</td>
<td>Flatness of frequency response</td>
</tr>
<tr>
<td>Max dip size (F3)</td>
<td>Flatness of frequency response</td>
</tr>
<tr>
<td>Ringinglength (I7)</td>
<td>Length of ringing</td>
</tr>
<tr>
<td>Max amplitude of S-shape (I6)</td>
<td>Amplitude of stimulus</td>
</tr>
<tr>
<td>Mean amplitude of S-shape (I5)</td>
<td>Amplitude of stimulus</td>
</tr>
</tbody>
</table>

### Table 3.7: Amount of rules per model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Amount of rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>5</td>
</tr>
<tr>
<td>Model 2</td>
<td>5</td>
</tr>
<tr>
<td>Model 3</td>
<td>12</td>
</tr>
<tr>
<td>Model 4</td>
<td>14</td>
</tr>
</tbody>
</table>

### Table 3.8: Evaluated model per expert

<table>
<thead>
<tr>
<th>Expert</th>
<th>Evaluated model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Model 1</td>
</tr>
<tr>
<td>2</td>
<td>Model 2</td>
</tr>
<tr>
<td>3</td>
<td>Model 2</td>
</tr>
<tr>
<td>4</td>
<td>Model 3</td>
</tr>
<tr>
<td>5</td>
<td>Model 3</td>
</tr>
<tr>
<td>6</td>
<td>Model 3</td>
</tr>
<tr>
<td>7</td>
<td>Model 4</td>
</tr>
</tbody>
</table>
3.4 Fuzzy modelling

(a) Membership functions after training.

(b) Membership functions after expansion to [0-1] interval.

Figure 3.9: Expanding input intervals by stretching the lower and upper membership functions.
### Model changes compared to previous version

<table>
<thead>
<tr>
<th>Model</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bell functions replaced by simpler triangle and trapezoid functions.</td>
</tr>
<tr>
<td>3</td>
<td>Has dips (F1) feature replaced by max dip (F2) feature, to have a continuous output. Number of membership functions changed to 3 for length of ringing feature. Number of output functions to changed 5 (finer grain of output score). More rules were added, due to the higher amount of output functions.</td>
</tr>
<tr>
<td>4</td>
<td>Membership functions were moved of ringinglength to be less strict on ringing. Changed epsilon for computation of ringing length to 0.015 Pa (instead of 0.001 Pa). Max amplitude of S-shape feature added. Weight of all rules set to 1.</td>
</tr>
<tr>
<td>5</td>
<td>Max amplitude replaced by mean amplitude of S-shape. Rollback of ringinglength membership functions to how it was in model 3.</td>
</tr>
</tbody>
</table>

Table 3.9: Model changes compared to previous version

### Checking during training

A new set of 50 unseen data samples was used as check data during the training phase. A check dataset prevents that the model over-fits to the training data, by keeping the structure as it is on the minimum checking error. This is because over-fitting starts when the checking error rises again and the training error still decreases [37].

### 3.5 Model tests and overview

All of the constructed models were tested on the datasets described in 3.2.1. Some data samples that are present in the second dataset were also used for training. However, the expert feedback on those samples was used for training rather than the OAE data itself, which was used for testing. Figure 3.10 shows an overview of the trained models and how they were tested. The training data was collected with the Java application.
Figure 3.10: Constructed models overview and test methodology.
3.6 Developing an Android application for automatic probefit evaluation

The final application was tested by sending several signals from a laptop to a smartphone running the application. These signals are described in 3.6.1. The general flow of the application is described in 3.6.2.

3.6.1 Test signals

In [3] a rectangular pulse of 80 $\mu$s was used as stimulus for the checkfit. The stimulus of the ILO-system was recorded with a Head And Torso Simulator (HATS). A similar signal was then obtained by generating a rectangular pulse of 80 $\mu$s at 50 Hz with a low-pass (cut-off 5000 Hz) and high pass filter (cut-off 500 Hz). This signal was sent through a probe and recorded with an in ear microphone (IEM) in a human ear. The click stimulus captured using the IEM was created using the DPOAE measurement system [7, 48]. The resulting impulse response was recorded with the same system. Both the IEM and the ILO signals were processed in Audacity®2.0.5. A low-pass (cutoff 5 kHz, 12 dB) and high-pass (cutoff 500 Hz, 12 dB) filter were applied, finally the signals were normalized. The normalization consisted of removing the DC offset and normalizing the maximum amplitude to -1 dB (default setting). This helps to reduce the noise. The filtering was done to keep the focus on the $[500,5000]$ Hz interval: in the ILO system only the first 6000 Hz of the spectrum are shown. A third synthetic signal was created by modifying the IEM recording with the pencil in Audacity. The process of the recording and the final testing is shown in Figure 3.11. It should be noted that the HATS impulse response passed through a cavity before reaching the microphone. This is not the case for the second signal: microphone and speaker are present in the same probe. The resulting signals are shown in Figure 3.12, together with the frequency spectrum.

3.6.2 Application flow

The application was developed in Eclipse IDE for Android Developers. The fuzzy module created in the Java application was reused. This enables to plug in new fuzzy models with other membership functions. If a new parameter were to be used, this parameter needs to be implemented (see Appendix A). The application uses the microphone input of the smartphone to receive analog signals. The general flow is shown in 3.13. First, a buffer of 100 ms (amount of
samples depends on sampling frequency) is filled. Then, it is checked if the user interface thread is not still drawing results from a previous buffer on the screen. It is also checked if the recording is initialised, i.e. if the position of the stimulus has already been calculated. If this condition is satisfied, the 100 ms signal is split up in five parts, and the average of the four last parts is calculated (the start of a pulse can be situated at the end of the first part, therefore only four complete parts are certainly present), resulting in an averaged signal of 20 ms. Subsequently the frequency response is calculated. Finally the result is calculated and is delivered to the UI thread for display on the screen. It should be noted that displaying the result on the screen is executed in a separate thread, namely the thread responsible for the user interface. This allows the screen to not freeze during extensive computations.

More information on the architecture of the application can be found in Appendix B.
3.6 Developing an Android application for automatic probe fit evaluation

Figure 3.12: Signals used for application evaluation.

(a) Waveform ILO signal, recorded with HATS.

(b) Frequency spectrum ILO signal.

(c) Waveform generated signal, recorded with IEM.

(d) Frequency spectrum generated signal.

(e) Waveform synthetic signal, created in Audacity ©2.0.5.

(f) Frequency spectrum synthetic signal.
3.6 Developing an Android application for automatic probefit evaluation

Figure 3.13: Application flow.
Chapter 4

Results

In this chapter the constructed machine learning models and fuzzy models are described and evaluated. In 4.1 and 4.2 the construction/training process of respectively the machine learning models and fuzzy models is given. In 4.3 all models are evaluated on the data sets described in previous chapter. Finally, in 4.4 the evaluation of the developed mobile application is handled.

4.1 Machine learning models

The set of models obtained through cross-validation is given by Table 4.1. The classification models were validated by measuring the AUC while the regression models were validated by measuring the MSE. The selected features for the models trained on the extended feature set are the same for each model (one set for classification models and one for regression models). These features are the ones obtained by using random forests as described in previous chapter. The shown AUC and MSE give an indication of the possible performance of the model. However, it can still happen that the model over-fits on the cross-validation [49], thus it can not be considered as the actual performance of the model. It is shown by the table that the best AUC is around 92% and the best MSE is around 200, which suggests an average error of 14 (although it is not completely the same as the error was squared, causing high errors to be penalized more). The linear regression model trained on the extended features has a remarkably high MSE of 675.08.

4.2 Fuzzy models

In this section the results of the Mamdani model evaluation are given, as well as the performance of the Mamdani models on the collected expert data. Finally, the construction of the neuro-fuzzy
### 4.2 Fuzzy models

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter values</th>
<th>AUC [%]</th>
<th>MSE</th>
<th># Feat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>leaf = 5, split = 6</td>
<td>87.18</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>GaussSVM</td>
<td>box = 13, scale = 4</td>
<td>92.46</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>LinSVM</td>
<td>box = 23</td>
<td>92.26</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>LogReg</td>
<td>lambda = 0.1</td>
<td>92.25</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>KNN</td>
<td>K = 19, weighting = inverse</td>
<td>91.81</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>DTALL</td>
<td>leaf = 10, split = 2</td>
<td>86.38</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>GaussSVMALL</td>
<td>box = 13, scale = 44</td>
<td>89.93</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>KNNALL</td>
<td>K = 43, weighting = inverse squared</td>
<td>89.05</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>LinSVMALL</td>
<td>box = 1</td>
<td>89.64</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>LogRegALL</td>
<td>lambda = 0.7</td>
<td>89.29</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>DTRReg</td>
<td>leaf = 5, split = 8</td>
<td>-</td>
<td>200.398</td>
<td>3</td>
</tr>
<tr>
<td>KNNReg</td>
<td>K = 12, weighting = inverse</td>
<td>-</td>
<td>194.2127</td>
<td>3</td>
</tr>
<tr>
<td>LinReg</td>
<td>mode = interaction</td>
<td>-</td>
<td>255.6015</td>
<td>2</td>
</tr>
<tr>
<td>DTRRegALL</td>
<td>leaf = 9, split = 6</td>
<td>-</td>
<td>228.71</td>
<td>44</td>
</tr>
<tr>
<td>KNNRegALL</td>
<td>K = 8, weighting = inverse squared</td>
<td>-</td>
<td>212.48</td>
<td>44</td>
</tr>
<tr>
<td>LinRegALL</td>
<td>mode = linear</td>
<td>-</td>
<td>675.08</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 4.1: Final machine learning models with cross-validation error. Models having an ‘ALL’-suffix are trained on the extended feature set. ‘DT’ stands for ‘decision tree’.
4.2 Fuzzy models

4.2.1 Labelling application data analysis

Table 4.2 shows the overall expert agreement with parameter classifications of the application. Different models use different membership functions for parameters and not all parameters are used by each model (see 3.4.1). The table shows that most experts agreed with the parameter feedback of ‘symmetry’ and ‘ampratio’: the average amount of agreement was 91.43%. The ‘ringinglength’ feature had a low epsilon (see calculation of I7 feature in previous chapter) for the first three models (0.005 Pa) and was therefore susceptible to noise, as became clear during the sessions. An example of a good fit that would get a medium score because of the ‘ringinglength’ feature with epsilon 0.005 Pa is shown in Figure 4.1. When evaluated using Mamdani model 3, this sample gets an overall score of 64.2% with an epsilon of 0.005 Pa and a score of 86.8% with an epsilon of 0.015 Pa. Furthermore, the value of the ‘ringinglength’ feature is classified as ‘medium’ with the first epsilon and is classified as ‘good’ with the second epsilon.

The fact that this parameter scored badly in the fourth model (26% agreement) is due to a bad choice of membership functions, which caused an evaluation that was too soft: the parameter was classified as ‘good’ for most of the checkfits. Therefore, in model 5 the membership functions were kept as in model 3. The epsilon was kept on 0.015 Pa for further evaluation of all models. During the labelling sessions it appeared that the ‘has dips’ and ‘max dip’ feedback did not
always correspond to the opinion of the experts, nevertheless the average agreement for ‘has dips’ was still 88 % and for ‘max dips’ 83.33 %. The ‘maximum amplitude’ parameter was evaluated by one expert only. The agreement was 78 %. This was because of the fact that the maximum amplitude only looks at one value, namely the maximum of the signal, and could therefore be less suited to model the ‘amplitude of the stimulus’. A possible solution is to look at the mean of the stimulus peaks, which was done in model 5.

Table 4.3 shows the mean error of the system score compared to the initial and final expert score (i.e. the score before and after system feedback). At least 10 % of the scores was changed per model, which could indicate a certain influence of system feedback on the expert’s decision. The amount of change is on average 13.8 %, which is not radical.
4.2 Fuzzy models

4.2.2 Mamdani models evaluated on new expert data

The five Mamdani models were tested on the collected expert data. It should be noted that for all tests the epsilon parameter for the computation of the parameter ‘ringinglength’ was set on 0.015 Pa (see 4.2.1), as it was clear from the labelling sessions that 0.005 Pa was too low. The result of the test is shown in Figure 4.2. It should be taken into account that the [0,20] and [80,100] intervals are the most important here. Indeed, experts will rather agree on the scores for extreme cases, i.e. very good and very bad fits, than on medium cases. The extremes are therefore more accurate. Models 3 and 5 show low errors in the [0,20] and [80,100] interval. This indicates that both model 3 and 5 are well suited for scoring. Model 4 shows large deviation in the [80,100] interval. This is probably due to the max amplitude parameter that caused very good samples to be scored less than just ‘good’ samples, in other words the parameter had too much unwanted influence on the result. In model 5 this seems to be solved: here the mean amplitude of the S-shape is used (see Table 3.5 and 4.2.1), and the membership functions were adapted to be less strict. Model 4 has the best overall error: a mean of 14.31 with a standard deviation of 11.87. Model 5 however has a mean overall error of 14.65 and a standard deviation of 12.14. Considering the extreme intervals model 5 seems the best option.

4.2.3 Neuro-fuzzy results

As both model 3 and 5 seemed promising, they were trained on the expert data, in order to fine-tune the membership function parameters. A checking set consisting of 50 data samples labelled by one expert was used to prevent over-fitting on the training data. After two epochs both training and checking error remained the same, as shown by Figure 4.3. The error distribution was again plotted for the training samples. This is shown in Figure 4.4. The purpose of these plots is to visualize the error distribution rather than visualize the overall error. As the samples were used to optimize the models, it is only logical that the overall training error will be lower than for the Mamdani models. The figures show however that the accuracy in the [0-20] interval has risen again compared to the Mamdani versions. This could indicate that more data is needed so that not only the overall error is low, but also the error levels in the [0,20] and [80,100] intervals.
Figure 4.2: Mamdani models tested on collected expert scores.
4.2 Fuzzy models

(a) Training of model 3. After two epochs the checking error does not decrease any more.

(b) Training of model 5. After two epochs the checking error does not decrease any more.

Figure 4.3: Training of the ANFIS models. The checking option was used to prevent over-fitting.

(a) Difference trained model 3 output and expert output (= training set).

(b) Difference trained model 5 output and expert output (= training set).

Figure 4.4: Comparing the predictions of the optimized model with the training data.
4.3 Model tests

After the model construction, tests were performed on given test sets to be able to compare all of the constructed models. Two datasets were used, as mentioned in previous chapter: an expert dataset and an OAE measurement dataset.

4.3.1 Expert data evaluation

The expert label set from [5] was converted to a binary label set by averaging the labels and splitting the set following a certain decision boundary. Three different sets were created using three different decision boundaries: -1, 0 and 1. In other words the first set contained a binary set with all samples having a label < −1 classified as ‘refit’ and the others classified as ‘no refit needed’, and so on. The averaged labels ranged from -3 to 2, as shown in Figure 3.2. The AUC was then computed using the classification scores (not the classification labels: these are binary) for the classification models, and simply the output for the regression models and fuzzy models. The classification scores are numbers between 0 and 1. More information on classification scores can be found in [50]. It should be noted that the classification scores are the opposite of regression/fuzzy scores: a high classification score means here that a refit is probably needed, while a high regression/fuzzy score means that the quality of the fit is good. Therefore the regression/fuzzy scores were inverted for this test. The result (in %) is shown in Table 4.4.

The first two datasets are well separated by most of the models (around 0.95 AUC), thereby indicating that they are capable of identifying bad fits. The third dataset is harder to separate, as this set includes label 0, which stands for ‘neutral’. Not many experts agree on this category. However, the linear regression model trained on only two features and the logistic regression model trained on eight features can still separate this dataset rather well (more than 0.9 AUC). The fact that linear regression scores good in general shows that the labelling strategy of the experts was rather straightforward. The difference between the Mamdani versions and ANFIS versions indicates that the optimized versions score worse in overall. The ROC curves for the ‘AUC3’ column are shown in Figure 4.5. The curves for the first and second column are not shown because of the high similarity between models (AUC close to 1). These curves could be used to compare models based on a minimum true positive rate that has to be achieved with a false positive rate that is as low as possible, i.e. if it is important to recognize X % of the bad checkfits with as few rejected good fits as possible. For example, if an important true positive rate would be 0.7, linear regression (expert set), logistic regression (extended set) and SVMs
score rather well: they have for a true positive rate of 0.7 a false positive rate of less than 0.2.

4.3.2 OAE measurements

Binary grouping

The OAE samples were split in two groups by using the output of the models. The output of the classification models was used as is, while the regression and fuzzy output was inverted and rounded (i.e. \( \text{floor}(\text{pred.} + 0.5) \)) to obtain a ‘refit needed’ label. The difference in interquartile range was computed (refit group interquartile range - no refit group interquartile range). The result is shown in Table 4.5. The interquartile range is higher for samples classified as ‘refit needed’, for both noise and signal level. A one-way ANOVA was used to test the difference between the two groups, both on signal and noise level. A one-sample t-test with reference value 0 was performed for each set, both for noise and signal deviation. This test was performed to see if the mean differed significantly from 0. The mean of the noise differs significantly from zero. The mean of the signal significantly differs from zero for ‘no refit needed samples’ (p < 0.05), for most of the models, except for KNN models for classification and regression (extended set). One-way ANOVA shows however no significant difference between two groups for signal differences. For noise differences however it is shown that two groups differ significantly (p < 0.05), for all models except for K-nearest neighbours (KNN) classification trained on the expert feature set. The box plots considering signal deviation for regression KNN and fuzzy model 5 ANFIS are shown in Figure 4.6: the first model has the most difference in interquartile range between groups, while the second model has the least difference. The box plots considering noise deviation for classification decision tree and Mamdani 1 are shown in Figure 4.7: these have respectively the most and least difference in interquartile range (for noise), as shown by column three in the table. In general, there is not much difference visible between models, therefore not all box plots are shown here. The figures show that the median for the signal difference is around 0, while the median for noise difference is higher for ‘refit’ samples. Both the interquartile range as the whisker distance is visibly larger for ‘refit’ samples.

Moving the acceptance threshold

To show the effect of moving a certain decision boundary, the interquartile range was plotted for all samples above a quality threshold (i.e. below the classification threshold) varying from 0 to 100. The threshold considered can be seen as an ‘acceptance threshold’: it can be used to make
### 4.3 Model tests

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC1</th>
<th>AUC2</th>
<th>AUC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>92.84</td>
<td>92.31</td>
<td>70.37</td>
</tr>
<tr>
<td>GaussSVM</td>
<td>97.84</td>
<td>97.06</td>
<td>88.7</td>
</tr>
<tr>
<td>LinSVM</td>
<td>97.03</td>
<td>99.32</td>
<td>78.15</td>
</tr>
<tr>
<td>LR</td>
<td>89.19</td>
<td>92.53</td>
<td>76.3</td>
</tr>
<tr>
<td>KNN</td>
<td>96.49</td>
<td>97.74</td>
<td>78.33</td>
</tr>
<tr>
<td>DTALL</td>
<td>92.84</td>
<td>92.31</td>
<td>70.37</td>
</tr>
<tr>
<td>GaussSVMALL</td>
<td>97.03</td>
<td>99.32</td>
<td>86.85</td>
</tr>
<tr>
<td>KNNALL</td>
<td>96.76</td>
<td>98.64</td>
<td>84.81</td>
</tr>
<tr>
<td>LinSVMALL</td>
<td>97.3</td>
<td>99.77</td>
<td>89.63</td>
</tr>
<tr>
<td>LRALL</td>
<td>97.3</td>
<td>99.32</td>
<td>90.74</td>
</tr>
<tr>
<td>DTReg</td>
<td>97.3</td>
<td>98.19</td>
<td>84.26</td>
</tr>
<tr>
<td>KNNReg</td>
<td>97.57</td>
<td>97.51</td>
<td>84.63</td>
</tr>
<tr>
<td>LinReg</td>
<td>97.3</td>
<td>100</td>
<td>92.96</td>
</tr>
<tr>
<td>DTRegALL</td>
<td>97.03</td>
<td>98.19</td>
<td>84.26</td>
</tr>
<tr>
<td>KNNRegALL</td>
<td>96.76</td>
<td>95.02</td>
<td>71.11</td>
</tr>
<tr>
<td>LinRegALL</td>
<td>99.73</td>
<td>96.83</td>
<td>74.19</td>
</tr>
<tr>
<td>1 Mamdani</td>
<td>98.38</td>
<td>96.95</td>
<td>85.19</td>
</tr>
<tr>
<td>2 Mamdani</td>
<td>98.38</td>
<td>95.81</td>
<td>81.76</td>
</tr>
<tr>
<td>3 Mamdani</td>
<td>97.84</td>
<td>97.06</td>
<td>81.94</td>
</tr>
<tr>
<td>3 ANFIS</td>
<td>93.24</td>
<td>95.93</td>
<td>82.78</td>
</tr>
<tr>
<td>4 Mamdani</td>
<td>97.30</td>
<td>95.70</td>
<td>83.89</td>
</tr>
<tr>
<td>5 Mamdani</td>
<td>97.30</td>
<td>97.96</td>
<td>84.91</td>
</tr>
<tr>
<td>5 ANFIS</td>
<td>98.65</td>
<td>92.76</td>
<td>76.30</td>
</tr>
</tbody>
</table>

Table 4.4: AUC [%] for each model, given the expert labels. The best option for each column is indicated in bold.
Figure 4.5: ROC curves for different models given the expert label set with decision boundary 1.
<table>
<thead>
<tr>
<th>Model</th>
<th>Interquartile range signal diff</th>
<th>Interquartile range noise diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>0.8</td>
<td>1.35</td>
</tr>
<tr>
<td>GaussSVM</td>
<td>0.85</td>
<td>0.875</td>
</tr>
<tr>
<td>LinSVM</td>
<td>0.9</td>
<td>1.025</td>
</tr>
<tr>
<td>LogRreg</td>
<td>0.9</td>
<td>0.925</td>
</tr>
<tr>
<td>KNN</td>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>DTALL</td>
<td>0.8</td>
<td>1.35</td>
</tr>
<tr>
<td>GaussSVMALL</td>
<td>0.825</td>
<td>0.95</td>
</tr>
<tr>
<td>LinSVMALL</td>
<td>0.775</td>
<td>0.8</td>
</tr>
<tr>
<td>LogRegALL</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>KNNALL</td>
<td>0.85</td>
<td>0.65</td>
</tr>
<tr>
<td>DTReg</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>KNNReg</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>LinReg</td>
<td>0.775</td>
<td>0.975</td>
</tr>
<tr>
<td>DTRegALL</td>
<td>0.8</td>
<td>1.05</td>
</tr>
<tr>
<td>KNNRegALL</td>
<td>0.925</td>
<td>0.725</td>
</tr>
<tr>
<td>LinRegALL</td>
<td>0.725</td>
<td>0.85</td>
</tr>
<tr>
<td>1 Mamdani</td>
<td>0.85</td>
<td>0.45</td>
</tr>
<tr>
<td>2 Mamdani</td>
<td>0.9</td>
<td>0.825</td>
</tr>
<tr>
<td>3 Mamdani</td>
<td>0.8</td>
<td>0.85</td>
</tr>
<tr>
<td>3 ANFIS</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>4 Mamdani</td>
<td>0.825</td>
<td>0.825</td>
</tr>
<tr>
<td>5 Mamdani</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>5 ANFIS</td>
<td>0.7</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 4.5: Difference in interquartile range of ‘refit’ and ‘no refit needed’ group for both signal and noise deviation [dB].
4.3 Model tests

(a) Signal deviations grouped by KNN regression model (extended feature set) output.

(b) Signal deviations grouped by fuzzy model 5 ANFIS model output.

Figure 4.6: Boxplots showing the signal deviations grouped by model output.

(a) Noise deviations grouped by decision tree classification model (expert features) output.

(b) Noise deviations grouped by fuzzy model 1 output.

Figure 4.7: Boxplots showing the noise deviations grouped by model output.
Figure 4.8: Effect on the interquartile range of the signal deviation by moving the acceptance threshold (classification models).

The model less strict or stricter. Ideally, an increasing line should be visible for classification models and a decreasing line for regression/fuzzy models: this would mean that the score is more or less inversely proportional to the expected deviation. The most important aspect is however that the curve is monotonous. The construction of a moving decision boundary was not possible to do for SVM models, as the scores are not distributed between a fixed interval. Therefore the SVM models were omitted here. The results are shown by Figures 4.8, 4.9 and 4.10. The plots stop at the point where the amount of accepted samples is lower than or equal to 50, to avoid noise. In general, the Mamdani models and the decision tree models show a monotonous decrease (increase for classification), but the transition is not uniform. The regression models, logistic regression, KNN and the ANFIS models have a more evenly transition. Model 5 ANFIS increases again at the end, which is not wanted (see Figure 4.10b). A uniform transition allows more flexibility in where to put the decision boundary. It should be noted that this does not mean that a non uniform transition is bad, there is however less choice between decision boundaries. For instance, for a regression decision tree it is not possible to give a decision boundary for interquartile range 1.5 dB, while this is possible for linear regression, namely 30 %.

Confusion matrix information

The OAE samples were split into two groups by using the following condition. A sample was considered to be a bad one if it had a total signal deviation of more than 2.5 dB or a signal deviation of more than 3 dB in at least one frequency band. The obtained dataset was compared
4.3 Model tests

Figure 4.9: Effect on the interquartile range of the signal deviation by moving the acceptance threshold (regression models).

(a) Regression models expert features.  (b) Regression models extended feature set.

Figure 4.10: Effect on the interquartile range of the signal deviation by moving the acceptance threshold (fuzzy models).

(a) Mamdani models 1, 2 and 4.  (b) Mamdani and ANFIS models 3 and 5.
4.3 Model tests

to the model predictions. The model predictions were rounded if the output was continuous, i.e. for regression/fuzzy models and logistic regression. The recall and the precision of ‘refit’ samples were computed. The recall corresponds to the amount of bad fits detected divided by the total amount of bad fits. The precision corresponds to the amount of bad fits detected divided by the total amount of samples predicted as bad fit. The result is shown in Table 4.6. The most important column is the ‘recall’-column: it is usually worse to not recognize a bad fit than it is to refit when it was not actually needed. Based on this table, it could already be assumed that fuzzy models provide a better recall-precision trade-off, given the accuracy and the high recall levels. This table shows however only the recall-precision for one single threshold, and the results could be different for other thresholds and if a lower recall than 80 % is needed. It could perhaps be that for a recall of 60 % a machine learning model provides a better precision. It is therefore useful to construct recall and precision curves for a moving threshold. Figure 4.11 shows plots for two classification models and Figure 4.12 shows plots for all of the regression models. It should be kept in mind that the classification score is the inverse of the regression score. While a high classification score denotes a high probability on refit, a high regression score denotes a high quality, and thus a low probability that a refit is needed. Logistic regression and KNN regression (expert feature set) achieve a precision of more than 42 % at a recall of 85 % (Figures 4.11a and 4.12a). This means that 69 out of 81 bad samples were recognized, while classifying about 163-164 samples as ‘bad’ and classifying 91-92 samples as ‘good’ (i.e. 79-80 which are actually good). The KNN regression model trained on the extended feature set achieves a precision of of 45.1 % at a recall of 85.19 %, which is even better (at the cost of more computation time, as the extended feature set is bigger than the expert feature set, see Figure 4.12b). The other machine learning models do not achieve notable high precision. Figures 4.13 and 4.14 show the recall and precision curves for the fuzzy models. Overall, the fuzzy models score reasonably better than the machine learning models. Model 1 for example achieves a precision of 45.22 % at a recall of 87.65 %, meaning that 71 samples out of 81 were recognized while classifying 157 samples as bad, therefore still recognizing 88 actual good samples. However, model 1 does not have a slowly increasing curve (it makes a large step around 50 %), allowing less choice in different thresholds. Model 5 achieves a precision of 47.95 % at a recall of 86.42 %, therefore recognizing 70 bad samples while classifying 146 samples as bad. 98 samples are correctly classified as ‘good’. In general, the trained fuzzy models seem to perform worse than the Mamdani models, as shown by Figure 4.13.
4.3 Model tests

<table>
<thead>
<tr>
<th>Model</th>
<th>FP</th>
<th>FN</th>
<th>TP</th>
<th>TN</th>
<th>Acc. [%]</th>
<th>Recall [%]</th>
<th>Precision [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>71</td>
<td>16</td>
<td>65</td>
<td>103</td>
<td>65.88</td>
<td>80.25</td>
<td>47.79</td>
</tr>
<tr>
<td>GaussSVM</td>
<td>80</td>
<td>14</td>
<td>67</td>
<td>94</td>
<td>63.14</td>
<td>82.72</td>
<td>45.58</td>
</tr>
<tr>
<td>LinSVM</td>
<td>79</td>
<td>15</td>
<td>66</td>
<td>95</td>
<td>63.14</td>
<td>81.48</td>
<td>45.52</td>
</tr>
<tr>
<td>LR</td>
<td>81</td>
<td>14</td>
<td>67</td>
<td>93</td>
<td>62.75</td>
<td>82.72</td>
<td>45.27</td>
</tr>
<tr>
<td>KNN</td>
<td>80</td>
<td>18</td>
<td>63</td>
<td>94</td>
<td>61.57</td>
<td>77.78</td>
<td>44.06</td>
</tr>
<tr>
<td>DTALL</td>
<td>71</td>
<td>16</td>
<td>65</td>
<td>103</td>
<td>65.88</td>
<td>80.25</td>
<td>47.79</td>
</tr>
<tr>
<td>GaussSVMALL</td>
<td>75</td>
<td>16</td>
<td>65</td>
<td>99</td>
<td>64.31</td>
<td>80.25</td>
<td>46.43</td>
</tr>
<tr>
<td>LinSVMALL</td>
<td>78</td>
<td>17</td>
<td>64</td>
<td>96</td>
<td>62.75</td>
<td>79.01</td>
<td>45.07</td>
</tr>
<tr>
<td>LRALL</td>
<td>76</td>
<td>17</td>
<td>64</td>
<td>98</td>
<td>63.53</td>
<td>79.01</td>
<td>45.71</td>
</tr>
<tr>
<td>KNNALL</td>
<td>81</td>
<td>19</td>
<td>62</td>
<td>93</td>
<td>60.78</td>
<td>76.54</td>
<td>43.36</td>
</tr>
<tr>
<td>DTRReg</td>
<td>69</td>
<td>21</td>
<td>60</td>
<td>105</td>
<td>64.71</td>
<td>74.07</td>
<td>46.51</td>
</tr>
<tr>
<td>KNNReg</td>
<td>70</td>
<td>17</td>
<td>64</td>
<td>104</td>
<td>65.88</td>
<td>79.01</td>
<td>47.76</td>
</tr>
<tr>
<td>LinReg</td>
<td>74</td>
<td>17</td>
<td>64</td>
<td>100</td>
<td>64.31</td>
<td>79.01</td>
<td>46.38</td>
</tr>
<tr>
<td>DTRRegALL</td>
<td>76</td>
<td>17</td>
<td>64</td>
<td>98</td>
<td>63.53</td>
<td>79.01</td>
<td>45.71</td>
</tr>
<tr>
<td>KNNRegALL</td>
<td>70</td>
<td>18</td>
<td>63</td>
<td>104</td>
<td>65.49</td>
<td>77.78</td>
<td>47.37</td>
</tr>
<tr>
<td>LinRegALL</td>
<td>77</td>
<td>18</td>
<td>63</td>
<td>97</td>
<td>62.75</td>
<td>77.78</td>
<td>45.00</td>
</tr>
<tr>
<td>1 Mamdani</td>
<td>17</td>
<td>56</td>
<td>64</td>
<td>118</td>
<td>71.37</td>
<td>79.01</td>
<td>53.33</td>
</tr>
<tr>
<td>2 Mamdani</td>
<td>14</td>
<td>70</td>
<td>67</td>
<td>104</td>
<td>67.06</td>
<td>82.72</td>
<td>48.91</td>
</tr>
<tr>
<td>3 Mamdani</td>
<td>11</td>
<td>81</td>
<td>70</td>
<td>93</td>
<td>63.92</td>
<td>86.42</td>
<td>46.36</td>
</tr>
<tr>
<td>3 ANFIS</td>
<td>16</td>
<td>71</td>
<td>65</td>
<td>103</td>
<td>65.88</td>
<td>80.25</td>
<td>47.79</td>
</tr>
<tr>
<td>4 Mamdani</td>
<td>14</td>
<td>70</td>
<td>67</td>
<td>104</td>
<td>67.06</td>
<td>82.72</td>
<td>48.91</td>
</tr>
<tr>
<td>5 Mamdani</td>
<td>15</td>
<td>69</td>
<td>66</td>
<td>105</td>
<td>67.06</td>
<td>81.48</td>
<td>48.89</td>
</tr>
<tr>
<td>5 ANFIS</td>
<td>18</td>
<td>73</td>
<td>63</td>
<td>101</td>
<td>64.31</td>
<td>77.78</td>
<td>46.32</td>
</tr>
</tbody>
</table>

Table 4.6: Confusion data given the model output (rounded output for regression models) and OAE data. FP = False Positives, FN = False Negatives, TP = True Positives, TN = True Negatives, with the positive class being the ‘refit needed’ class.
4.3 Model tests

(a) Effect on the recall and precision by moving the threshold (models trained on expert feature set).

(b) Effect on the recall and precision by moving the threshold (models trained on extended feature set).

Figure 4.11: Effect of moving the refit threshold on the model scores.
4.3 Model tests

(a) Effect on the recall and precision by moving the threshold (models trained on expert feature set).

(b) Effect on the recall and precision by moving the threshold (models trained on extended feature set).

Figure 4.12: Recall and precision curves for regression models (Linear Regression and Decision Tree).
(a) Model 3 and model 3 ANFIS comparison.

(b) Model 5 and model 5 ANFIS comparison.

Figure 4.13: Effect of moving the threshold (model 3 and model 5).
4.4 Android application

The three signals described in 3.6 were presented to the application. For signals 1 and 3, one fragment of 20 ms containing one pulse was looped. The results are shown in Figure 4.15. The resulting ILO signal (Figure 4.15a) looks more or less as it was displayed in the ILO system itself (Figure 4.16), but the ringing amplitude is higher. It should be noted that the signal in the ILO system was recorded in the ILO IEM, while the signal processed in Audacity and fed to the application was recorded with the HATS. This can make a difference as the cavity of the HATS ear coupler has a characteristic transfer function which affects the signal captured by the HATS microphone.

It is clear that the IEM signal needs additional preprocessing: there is still much ringing present, as shown in Figure 4.15b. The application evaluates the synthetic signal as expected. All of the parameters are classified as ‘good’, therefore resulting in a score of around 80 % (see Figure 4.15c). Furthermore, it should be noted that there was some fluctuation present on signal captured by the phone, causing scores to vary 20 to 30 %, while the same pulse was presented to the application. This could be caused by the sound card of the phone or the laptop sending the signals, or could be due to an automatic gain control in the phone [51].
4.4 Android application

(a) Signal 1 (ILO) application evaluation. (b) Signal 2 (IEM) application evaluation. (c) Signal 3 (synthetic) application evaluation.

Figure 4.15: Application checkfit evaluation results. ‘Amplitude of the response’ is calculated using the ampratio feature (I8), while ‘amplitude of stimulus’ is calculated using the mean amplitude of the stimulus peaks (I6).

Figure 4.16: The HATS recording of the impulse response shown in the ILO system.
Chapter 5

Discussion

Literature showed that the placement of the measurement probe is crucial to obtain an accurate OAE measurement \[9, 5\]. The goal of this thesis was to construct an expert system that is able to evaluate probe fits automatically. This would allow to use hearing screening in a non-clinical context without the need of a supervising expert. Two paths were explored to achieve such a system, namely machine learning and fuzzy logic.

5.1 Data collection

The Java labelling application produced two labels and scores per checkfit, namely the original score and label given by the expert and the final score and label after seeing feedback of the system. The reason that data was trained on the original scores rather than on the final scores is because of the possible influence the model could have on an expert’s decision. It could have been an option to give the system reference scores at the end, not after each sample, to avoid bias in the next labelled samples. The system reference score was however given after the labelling and scoring of each sample, so that the samples would be judged in the same manner. Two data sets constructed in previous work \[5\] were used to test constructed models on. The first set consisted of 48 checkfit samples and a label given by 18 experts per sample. These labels were averaged, so that each expert was given the same ‘weight’ in the result. It could have been another option to take the mode or the median, as the labels can also be seen as ordinal variables: this is something that can be considered in the future. The second dataset consisted of TEOAE signal deviations and noise deviations.

The first dataset was not used to train a model on as the amount of different samples was too
small. Therefore, models trained on this dataset would not generalize well. The reason that eventually data from fewer experts (i.e. data from the labelling application) was used with not much overlap, was based on the argument that it is more important to have many different samples than many different experts. For instance, averaging over 2 or 10 experts will not have a big effect on very bad or very good samples, while having 100 labelled samples instead of 20 will have a large effect on the generalization. However, the accuracy in the medium region could improve with labels averaged over many experts. This could be done in the future, to achieve accurate models that output meaningful scores (i.e. scores proportional to the expected deviation). Furthermore, the set was constructed with a restriction: each expert labelled the checkfits by means of a fixed distribution. This means that experts were restricted in their labelling. A model trained on this data would thus be biased somehow. However, an averaged dataset gives an idea of what a ‘bad fit’ could be: if the averaged score is low, this means that each expert classified this sample as bad. Therefore the dataset was still used as a test set.

The second dataset (OAE dataset) was also used for testing. An OAE test set was needed, as the final model should be able to avoid large deviations on the measurements. A part of the dataset was thus needed for testing, taking away samples fit for a potential training set. As the amount of samples left would be too low, this set would not be suitable for training.

In the end, 266 different checkfits labelled with the labelling application by experts 1 to 7 were used for training, whereof 128 checkfits from the OAE dataset. This means that the training and testing data is not completely independent. The checkfits were nevertheless included in the test set because the models were trained on expert labels and not on the OAE deviations. They were added to the labelling set in the first place to make sure that there would be enough variability: as the other samples were drawn randomly from a large database, most of them were probably ‘good’ fits. As the checkfits of the OAE dataset were done in suboptimal conditions, bad checkfits would certainly be present.

The 50 checkfits labelled by the last expert were only used for ANFIS model validation.

5.2 Machine learning

Five classification models were tested and three regression models. Two sets of features were considered. The first feature set only contained nine features derived from literature. As these are the features experts use often to evaluate the probe fit, they are probably correlated with the score they give. The other set contained additional features relating to the frequency response.
The description of a good frequency response is rather vague in literature and therefore features were considered relating to the flatness of the spectrum, such as spectral flatness, kurtosis and skewness. To eliminate useless features, feature selection was performed, using random forests. There are many feature selection methods available, and in further research these techniques could be explored to select better features. No transformation was used on the features (such as PCA) because this would mean that for each evaluation of a data sample all of the features need to be computed. This would slow down the evaluation of the probe fit.

Random forests were only used for feature selection, not for testing. This is because large random forests take relatively long to evaluate and consume a significant portion of memory. KNN and SVM models also need some memory, but this depends on the amount of training samples. Here this amount is rather low and this way the evaluation is fast enough and the amount of used memory is low. SVM models usually need less memory than KNN models because only the support vectors (which is usually a small portion of the training samples) need to be saved. Random forests selected 8 features for the classification models and 44 features for the regression models. From the cross-validation results it was clear that the high amount of features was highly disadvantageous for the linear regression model: the MSE was 675.08. This could have been improved if a regularization parameter had been used as was done for logistic regression. The high amount of features could also be disadvantageous for the KNN model and the decision tree, as they are prone to over-fitting.

5.3 Fuzzy logic

In total, five different Mamdani models were constructed, each one coping with problems encountered in the evaluation of the previous model. These problems were derived from expert feedback after each labelling session (see Table 3.9). However, the numerical data was analysed in the end, after the labelling sessions. The models were compared to the scores of the experts. Models 3 and 5 were further optimized using neuro-fuzzy optimization, as they were the most accurate in the [0,20] and [80,100] intervals.

5.3.1 Fuzzy model evaluation data

The evaluation data collected with the Java application were analysed in the end. The data confirmed the assumption that the parameters ‘symmetry of the stimulus’ and ‘amplitude of the impulse response’ were more or less correctly modelled: the average expert agreement was
91.43% for both parameters. The length of the ringing was evaluated too strictly: the noise threshold was a bit too low, i.e. a threshold of 0.005 Pa was used as noise threshold, resulting in a significant impact on the final model score. For later analysis 0.015 Pa was used as a threshold for the ringing. It appeared during the labelling sessions that experts did not always agree with the evaluation of the frequency spectrum, but an adequate agreement of 88.88% was however still achieved for the ‘has dips’-feature. Instead of checking for dips the shape seemed also of importance. It is however harder to score this objectively, a frequency spectrum scorer could for example be solved using machine learning. Finally, the interference of the system was analysed. At least 10% of the original scores was changed by the expert after seeing system feedback on the parameters and a system reference score. This suggests that the system has an influence on the expert’s decision. It is possible that this influence can be negative, but this will not be the case if the model has been evaluated thoroughly. Furthermore, the amount of change was small, therefore indicating that the amount of influence is limited.

5.3.2 Neuro-fuzzy optimization

Mamdani models 3 and 5 were chosen to be optimized, as mentioned above. The training of both models was performed with a checking set consisting of 50 data samples labelled by the eighth expert. This dataset served only one purpose: preventing over-fitting on the data. After two epochs the training stopped for both models. The average checking error was for both models around 20% (root MSE), which is acceptable: very bad fits will not be scored as a good fit and vice versa. The training error was around 11% (root MSE) for both models. This suffices to distinguish between very bad, bad, medium, good and very good fits. However, the data in the medium area is probably not very accurate, as there was not much overlap of samples labelled by different experts. Therefore it might be a good investment to have several experts label the same checkfits in the future, while keeping the total amount of samples around 250. When looking at the training error, it is notable that the error in the \([0,20]\) and \([80,100]\) intervals has a larger deviation than in the non optimized versions. This would not be a real problem if all labels would be accurate enough, because then the overall error would be a good indication of the model performance. However for the training set used here it is more certain that the labels in the extreme intervals, i.e. \([0,20]\) and \([80,100]\), are more accurate. Therefore the decrease in accuracy for those intervals is not wanted here and could have a negative influence on the test results.
5.4 Model evaluation

All of the constructed models were tested on both the expert dataset and the OAE dataset from [5].

5.4.1 Evaluation on expert data

Tests on expert data showed that all of the models could distinguish bad fits from medium and good fits. All models achieved an AUC of about 95 % to 100 %. It was harder to distinguish bad and medium fits from good and rather good fits. This is because of the vagueness of the medium label: experts do not easily agree on this level. Furthermore, it was shown in [5] that only one expert out of 18 would refit checkfits labelled with ‘-1’. This means that the need of refitting checkfits having an expert label ‘0’ is highly improbable. However, a simple linear regression model using only two features still achieved an AUC of 92.96 %. A logistic regression model using eight features achieved an AUC of 90.74 %. The first Mamdani model achieved an AUC of 85.19 %, which is the highest value among the fuzzy models, despite being one of the simplest fuzzy models using few parameters and rules. Furthermore, the regression models trained on the extended feature set (44 features) performed worse compared to the regression models in the expert feature set: the linear regression model trained on the extended set achieved an AUC of 74.19 % and the KNN regression model trained on the extended set achieved an AUC of 71.11 %, compared to 92.96 % and 84.63 % respectively. Given the fact that simple models perform the best on this test, this could indicate that experts did not look at many features.

5.4.2 Evaluation on OAE data

First, the checkfits from the second dataset were grouped by prediction output. It can be concluded that a ‘refit’ group always has more deviation on both the signal as on the noise level. There is significantly more noise present in measurements preceded by checkfits classified/scored as ‘refit needed’. More importantly, the interquartile range of the signal differences is larger for ‘bad fits’. A one-sample t-test showed that the mean of the signal differences for the ‘no refit needed group’ is significantly lower than 0 for most of the models. This is a result that was also found in [5]. However, one-way ANOVA showed no significant difference between the means of the signal differences of both groups. These findings show that all models are capable of identifying fits with larger deviations on the OAE measurements and with larger deviations on noise level.
Second, the effect on the interquartile range of the signal deviation (= signal differences) by moving the acceptance threshold was analysed. The noise was not analysed as it has been shown that noise depends on external factors [5]. Due to the small amount of data samples this could have a significant impact on curves constructed with a moving threshold. It can be concluded that for most of the models moving the threshold has the desired effect, i.e. the deviation increased if the threshold was decreased. If the deviation would not increase, this would indicate that the scoring of the model is not accurate: a sample of 70% should be better than a sample of 50%. For the ANFIS models, regression models, and the logistic regression model the increase in deviation was more uniform, therefore allowing more flexibility. Furthermore the score is then roughly proportional to the expected deviation, making it useful to display in an application. The more uniform the curve, the more choice in thresholds. If the curve makes a ‘jump’ from 1.8 dB to 1.4 dB, there is no threshold for 1.6 dB, the threshold at 1.4 dB needs to be chosen. Mamdani models and decision trees provide less flexibility as more samples are mapped on the same output, therefore not distributing the samples well. This does however not mean that a good threshold can not be found. For instance, it can still be said for Mamdani models 1 and 3 that the deviation is lower than 1.2 dB for all samples with a score greater than 50% (see Figure 4.10). It can be that a curve has a uniform transition but goes up again. A monotonously decreasing curve is more useful than a uniform decreasing curve that goes up again later. A distinct example of this can be seen in Figure 4.10b, where the deviation rises 1.1 to 1.3 dB for the ANFIS version of fuzzy model 5.

Finally, the OAE samples were split into two groups using a limit on the deviation a good fit can have. This resulted in a binary label set of 81 ‘refit needed’ labels and 174 ‘no refit needed’ labels. The model predictions were compared to this set. In overall, the recall was in the [75,83] % range. The precision, i.e. the amount of detected bad fits divided by the total of predicted bad fits, was in the [40,48] % range. This means that more than half of the refits would not actually be needed. It is nevertheless more important to avoid large deviations on the measurement than avoiding refits when not needed. This is an important trade-off that can better be analysed with moving threshold curves. Both logistic and linear regression seem to have a smooth transition, while a decision tree seems to work in discrete intervals, mapping samples on the same output and thus not having a good transition. It depends on the recall requirement which model is better suited. Again, a uniform transition allows more flexibility. Mamdani model 1 achieved a recall of 87.65 % together with a precision of 45.22 %. The
transition was however not very uniform, therefore allowing less flexibility: no threshold can be found in the [70,85] % interval. Once again this indicates that the suitability of a model depends on the context. If it is important to have both flexibility and intuitive scores that are proportional to the expected deviation or the expected recall, regression models or ANFIS models are more suited, these are trained on scores. If it is important to have several good thresholds to choose from to produce a classification label, Mamdani models are preferred.

5.4.3 Overall evaluation discussion

As OAE deviations are objective and therefore more important the tests suggest that Mamdani models are a good choice, even if they did not perform the best in the expert test. They only failed to separate the data well when medium checkfits were considered to be bad, but even then they could still achieve an AUC of [80,85] %. Moreover, as explained above, the findings in [5] suggest that the need of having to refit a checkfit labelled as ‘0’ is highly improbable. Another downside is the fact that Mamdani scores are not proportional to the expected deviation. This is something that is more visible for regression models and ANFIS models. As it is of most importance that a good threshold can be found which allows to reject bad checkfits, producing meaningful scores is of less interest. It could be interesting to produce such a score for the experts, but Mamdani models already show accurate parameter classifications, and coupling this to a ‘reject or not’ outcome is already useful to experts.

5.5 Android application

An application was created reusing the fuzzy module used for the data labelling application. Model 5 was used for the test. Three different signals were sent through a TRRS cable to a smartphone with an Android operating system. There was some fluctuation present in the signal received by the phone, which could be solved by averaging scores and updating the final score over a certain amount of time.

In the methodology chapter it was explained that additional filtering was needed on the recorded signal to obtain an impulse response similar to the ones displayed in the ILO-system, i.e. an impulse response without the frequencies outside of the [0,6000] Hz interval. Furthermore even after filtering the IEM signal had a poor evaluation score of 8 %, despite originating from an ideal fit. This could suggest that even more preprocessing is needed to cope with the ringing. Both filtering and additional preprocessing could be done on the digital signal processor (DSP)
of the DPOAE measurement system or on the phone. Using the first option would allow to keep
the application as it is now. However, as there is limited space available on the DSP, it could be
favourable to do the preprocessing on the phone. This could require additional optimizations to
still allow real time evaluation. Real time evaluation is important because it allows to refit the
probe and see the effect immediately.

The application was capable of evaluating the signals in real time. As Mamdani models scored
well in other tests, such a model is favourable over machine learning models. If speed should be
optimised, it could be that simple models such as linear regression and logistic regression are
favourable because of the faster evaluation. It should also be noted that the application reused
the fuzzy module, which has a modular design, therefore not optimised for speed. It could be
possible that most of the input parameters could be calculated all at once, i.e. by processing the
impulse response in one pass. The fuzzy module calculates all parameters separately, therefore
requiring multiple passes over the impulse response.

5.6 Future work

It can be concluded that a Mamdani fuzzy model can already give a good indication whether or
not a probe should be refitted. However, the actual numerical scores are not always intuitive as
many samples are mapped on the same score, making the model rather suited for classification
(e.g. bad, medium, good fit) by choosing one or more thresholds than for scoring of the fit. To
have meaningful and intuitive scores, i.e. a score that is proportional to the quality of the fit, it
is better to optimize using the neuro-fuzzy algorithm, or to use a machine learning model that
is trained on scores. It was shown that the ANFIS models had a training error of 11 % and a
checking error of 20 %. However, not all labels were that accurate, as there was not much over-
lap between experts. Therefore, it would be a big improvement if all training checkfits (about
250) were labelled by about 10-20 experts. This would take about three hours per expert. Then
minimizing the overall error would be favourable over minimizing the error in the [0,20] and
[80,100] intervals. Thus better regression and ANFIS models would be obtained, as they are
based on minimizing the overall error.

Another option could be to train model over time, by training it in an online manner. For
example, labelling sessions as were done with the labelling application could be created, and
experts could label a few samples randomly drawn from a database. The model would then be
optimized by providing it newly scored samples. The main challenge then is to have a diverse population of participating experts (enough experts with different backgrounds).

Machine learning could also be used to find relationships between checkfits and OAE data by training models on OAE data directly. This would be possible if a database would already be available, but as current existing databases usually only contain measurements taken in optimal conditions, this would require taking new measurements with different probe configurations as was done in [5] and seems therefore rather infeasible.

Moreover, it could be useful to train a simple machine learning model on frequency responses only, producing a score which can be presented as one input parameter to the fuzzy model. This way, the benefits of a fuzzy model would still apply while having a more robust parameter for the frequency response. Up until now, only the presence of dips was analysed in the fuzzy models.

Finally, both fuzzy modelling and machine learning could be applied to evaluate the final OAE measurement and/or to react appropriately. For example, if the checkfit was not perfect, an OAE measurement was performed and some of the OAEs have a small amplitude or the noise level is too high, a possible decision could be to suggest a new checkfit. In other words, decisions related to the combination of checkfit and OAE measurements could be taken automatically.
Chapter 6

Conclusion

The purpose of this thesis was to develop a technique that allows the automatic evaluation of probe fitting prior to OAE measurement. This evaluation had to reflect the possible deviation on the OAE measurements due to probe fitting. Two paths were explored, machine learning and fuzzy logic. In the machine learning part two different types of models were constructed: models trained on binary fit-refit labels and models trained on scores of fitting on a continuous scale between 0 and 100. In the fuzzy logic part models were constructed based on literature and expert feedback. In the end they were further optimized using scores. The binary labels and scores were collected by means of a labelling application which was used by seven experts.

Both the machine learning models and fuzzy models were able to recognize probe fits classified as ‘bad’ by experts, suggesting that not many parameters are needed to have a decent ‘refit-or-not’ suggestion. Additionally, all models were able to make a distinction between a fit that causes high deviation in OAE measurement and a fit that does not, using a relatively small amount of input variables (5-6). The strictness of a model was defined by means of a checkfit acceptance threshold. This threshold could be linked to a maximum allowed deviation on the OAE signal measurements or a minimum recall of checkfits that exceed a certain OAE signal deviation threshold. Regression models and neuro-fuzzy optimized models provide the highest flexibility in choosing such a threshold. Based on the recall-precision tests of checkfits with large deviations on the OAE measurements, Mamdani fuzzy models are preferred. An Android application was developed and the Mamdani models could be evaluated in real time.

Considering the test results and the fact that real time evaluation is possible, Mamdani models
are favourable. Moreover this type of models is based on expert knowledge and outputs are easier to interpret by experts. The main disadvantage is that Mamdani models have a simple structure and are more suited for rejecting and accepting fits than for providing a meaningful score: the scores Mamdani models give are not proportional to the expected deviation on the OAE measurement. A well-chosen threshold on the output can be used instead to determine the strictness of the model. This threshold then determines if a checkfit is rejected or not.

For applications where speed is an important aspect, more simple approaches such as logistic regression and linear regression can be used, as they already give good results. If it is important to obtain a meaningful checkfit quality score that is proportional to the expected deviation on the OAE signal measurement, the collection of scores given by several experts can be considered. When multiple experts score the same set of checkfits and the scores are then averaged, the final score will be more accurate. This would allow a better training of neuro-fuzzy models and regression models, resulting in better recall-precision trade-off. This would be a reason to prefer neuro-fuzzy optimized models over Mamdani models.
Appendix A

Fuzzy logic application using java

For this thesis an application was created to serve two purposes: collect labelled data samples and evaluate the quality of several fuzzy logic models. In this chapter the architecture of the application is explained to facilitate the reuse of the fuzzy module in future applications.

A.1 Global architecture

The application consists of four main modules:

- fuzzy module: contains fuzzy logic code
- datalogic module: contains data processing code
- gui module: contains user interface code
- io module: contains io code (import and export data)

It makes use of following external libraries:

- framework for fuzzy logic: jfuzzylite
- show plots of data samples: jfreechart, jcommon
- import data from web page: jsoup
- export data to Dropbox: dropbox-core, jackson-core
- other: javatuples
The most important one of these external libraries would be the jfuzzylite library, which is the main library used in the fuzzy module. Figure A.1 shows the dependencies between the packages. An arrow pointing to a package means that package is used by the package where the arrow originates.

A.2 Fuzzy module architecture

As the fuzzy module is the most important part of the application (should be reusable in e.g. an Android app), the architecture is explained here in more detail. Two main classes of the Jfuzzylite library were extended, namely the Engine class and the InputVariable class. Extending the InputVariable class made it possible to attach certain attributes to a fuzzy input parameter, such as ‘name’ and ‘description’. It also allows the parameter instance to compute its own value given a data sample (here a data sample consists of an impulse and frequency response). By extending the Engine class it is possible to define custom models, like the ones used in this thesis. The MyFuzzyModel class also provides methods which return objects of the MyFuzzyParameter class instead of objects of the InputVariable class (which does not allow

Figure A.1: The packages and their dependencies
the custom attributes as mentioned above). Finally, a MyFuzzyModelFactory class was created, which simply allows to get an instance of a model. It is possible to modify this class in a way that the factory knows by a passed parameter which model to return. It is possible to create new models (like Model3 and Model5) by extending the MyFuzzyModel class and adding the necessary initialisation code. Such code can be for instance obtained from a Matlab fis-file by converting it with the jfuzzylite jar-file and replacing the inputVariable instances by the corresponding MyFuzzyParameter classes. The architecture is shown in Figure A.2.
Appendix B

Android application

A simple Android application was developed, reusing the fuzzy module of the labelling application. In this chapter the architecture is explained. An open source application was used as a starting point, namely the ‘Android SPL Meter’-application [52]. The jfreechart library was again used for plotting the data. Figure B.1 shows the simplified package diagram and the class diagram for the levelmeter package. The LevelMeterActivity-class is responsible for processing audioframes and displaying the result on the screen. The MicrophoneInput-class runs in a separate thread, collects audio samples until a full buffer is reached. Subsequently ‘processAudioFrame’ is called from the LevelMeterActivity-class. The processing of the audioframe happens in the same thread, but the displaying of the result on the screen is performed in the UI-thread.

Figure B.1: Simplified package diagram and class diagram for the levelmeter package. External packages are omitted here.
Appendix C

Algorithms

In this chapter some of the feature extraction algorithms are described in detail.

C.1 Extraction of impulse response features

The algorithm used to find the start of the ringing is shown in Algorithm 2. The algorithm used to find the peaks of the S-shape is shown in Algorithm 3.

C.2 Extraction of frequency response features

The algorithm used to find the number of dips given a certain window and threshold is shown in Algorithm 4. The algorithm used to find the largest dip given a certain window and threshold is shown in Algorithm 5.
C.2 Extraction of frequency response features

Data: impulse response vector (y values)

Result: ringing start index

1 \([\text{ind1, ind2}] = \text{getFirstSecondPeak}(y)\);

2 if \(y(\text{ind1}) < y(\text{ind2})\) then

3 \(s = y(\text{ind2});\)

4 \(\textbf{while } \text{next } s \text{ available and } s \leq \text{next } s \text{ and } s < 0 \text{ do }\)

5 \(s = \text{next } s;\)

6 \(\text{end}\)

7 \(\text{ringingindex} = \text{index of } s;\)

8 else

9 \(s = y(\text{ind1});\)

10 \(\textbf{while } \text{next } s \text{ available and } s \geq \text{next } s \text{ and } s > 0 \text{ do }\)

11 \(s = \text{next } s;\)

12 \(\text{end}\)

13 \(\text{ringingindex} = \text{index of } s;\)

14 \(\text{end}\)

Algorithm 2: Finding the start of the ringing
Data: impulse response vector (y values), \( \epsilon \)

Result: S-peak indices of extrema

1 find first sample \( s \) where \( |s| > \epsilon \);
2 if \( s > 0 \) then
3     while next \( s \) available and next \( s \geq s \) do
4         \( s = \text{next } s \);
5     end
6     first_index = index \( s \);
7     while next \( s \) available and next \( s \leq s \) do
8         \( s = \text{next } s \);
9     end
10    second_index = index \( s \);
11 else
12     while next \( s \) available and next \( s \leq s \) do
13         \( s = \text{next } s \);
14     end
15     first_index = index \( s \);
16     while next \( s \) available and next \( s \geq s \) do
17         \( s = \text{next } s \);
18     end
19    second_index = index \( s \);
20 end

Algorithm 3: Finding the peaks of the S-shape
Data: frequency response vector (y values), threshold, window length

Result: startindices, dipindices, endindices

1 i = first sample;
2 while search window not at end do
3     found = false;
4     j = second sample in search window;
5     while not found and sample after j in search window available do
6         k = sample after j;
7         while not found and sample after k in search window available do
8             if first sample - j ≥ threshold and k - j ≥ threshold then
9                 found = true;
10                startind.append(i);
11                dipind.append(j);
12                endind.append(k);
13                move search window to sample k;
14             end
15         k = next sample k;
16     end
17     j = next sample j;
18 end
19 if not found then
20     i = next sample i;
21 end

Algorithm 4: Finding dips in the frequency spectrum
C.2 Extraction of frequency response features

Data: frequency response vector (y values), window length

Result: maxdip

1 \( i = \) first sample;
2 \( treshold = 0; \)
3 while search window not at end do
4 \hspace{1em} k = two samples after \( i; \)
5 \hspace{1em} while sample after \( k \) in search window available do
6 \hspace{2em} j = min sample between samples \( i \) en \( k \);  
7 \hspace{2em} if first sample - \( j \) \( \geq \) treshold and \( k - j \) \( \geq \) treshold then
8 \hspace{3em} treshold = min(first sample - \( j \), k - \( j \));
9 \hspace{2em} end
10 \hspace{1em} k = next sample \( k; \)
11 \hspace{1em} end
12 \hspace{1em} i = next sample \( i; \)
13 end

Algorithm 5: Finding the maximum dip in the frequency spectrum
Bibliography


