AALBORG UNIVERSITY
COPENHAGEN

Semester: 4th

Title: Environment dependent personalized hearing aid configuration

Project Period:
From 1/2/2019 to 6/6/2019

Semester Theme:
Master's Thesis

Supervisor(s):
Niels Koefoed

Knud Erik Skouby

Project group no.: 4SER 4.4

Members
(do not write CPR.nr.):

Christoffer Dyssegard

Pages: 150
Finished: 6/6/2019

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Semester Coordinator: Henning Olesen

Secretary: Maiken Keller

Abstract:

Fine-tuning of hearing aids can be required for a user to achieve a desired objective such as improving speech intelligibility or removing wind noise. People are different and have various techniques for achieving what they find to be the optimal listening experience. The objective can depend on factors such as the sound environment, mood, and fatigue. While modern hearing aid solutions do provide fine-tuning features, there is a need for automation. This project explores how a solution for learning user preferences in a specific sound environment can be designed.
The project researches multiple Convolutional Neural Network (CNN) architectures and reinforcement learning algorithms, and creates a solution that combines both techniques. The solution allows a user to carry out training sessions in order to teach the machine learning components about the user's preferences. By associating the preferences with an environment representation, personalized hearing aid configurations can be applied next time the user is in a similar environment.
A prototype is developed and tested. It is found that the CNN component is performing the best, however, the system's ability to adapt and recall user preferences overall needs more work. The design is still believed to be valid, and a number of improvements are listed to outline what should be done to achieve a solution with acceptable results.

This page is intentionally left blank

Environment dependent personalized hearing aid configuration

Christoffer Dyssegard

February - June 2019

Contents

Executive summary

There is a need for hearing aid solutions to evolve such that they can understand the audio and take user behavior into account. In order to provide the best experience for the hearing aid user, simply collecting, processing, and amplifying input audio does not suffice. Hearing aid users are not created equally, and there is a need to provide solutions that adapt to the preferences of the individual. Some solutions already do provide customizability, however, not in an intelligent and reusable manner. This report presents a solution for providing a personalized listening experience based on the current sound environments for hearing aid users. The project takes a starting point in the ReSound Smart 3D application which provides an interface for controlling hearing aids for users of a range of ReSound hearing aids. The Smart 3D application does provide users with various options for adjusting their hearing aids to cope with challenging situations. As an example, users can adjust noise reduction and speech focus levels. Furthermore, more advanced functionality is also provided allowing the user to adjust the amplification of bass, midrange, and treble with ± 6 dB. However, this presents two problems: Firstly, the adjustments are limited in reusability meaning the user has to take action to adjust the parameters each time the environment changes. Secondly, not all hearing aid users are comfortable with making such adjustments. By developing a solution which is capable of learning the user's preferences and correlating them with acoustic features of the environment, adjustments can be provided automatically. The user no longer needs to make the decision that an adjustment is needed to achieve a better listening experience. Moreover, the user no longer has to determine what the specific adjustments should be.

The project considers the speech focus, noise reduction, and wind noise reduction features from the ReSound Smart 3D application as well as the bass, midrange, and treble amplification adjustments. These parameters are treated as numerical values, and the report presents research on Convolutional Neural Networks (CNNs) and reinforcement learning algorithms in order to determine how to learn and predict user specific adjustments. By analyzing the research, it is found that the two techniques complement each other. A multi-task branched CNN architecture is chosen for environment analysis and for predicting general hearing aid adjustments. Such a network architecture allows for one network to perform both classification and regression. As a result, the environment can be labeled while also predicting adjustments. Furthermore, the architecture ensures that learnings are shared within the network such that information about how to predict the environment class can also benefit adjustment predictions. However, this approach is not appropriate by itself as training CNN models with such capabilities should not be done on a smartphone. Therefore, the reinforcement learning algorithm Sarsa(λ) is used to interact with the user and learn preferences. Sarsa(λ) is chosen due to its ability to utilize input data efficiently such that less user interactions are necessary. Furthermore, it is less prone to remembering user preferences that

might no longer be valid.

A design consisting of a smartphone application and a cloud entity is created. The application has an instance of a CNN model which is used to analyze the environment. However, in order to perform the analysis, an environment representation must exist. The project relies on a prestudy which found that a sound environment can be represented by Short-Time Fourier Transformation (STFT) spectrograms, and that such a representation can be well understood by CNNs. The application collects audio samples of the user's environment, creates spectrograms, and performs an analysis using the CNN model.

In order for the solution to learn a user's preferences, training sessions are carried out. During these sessions, Sarsa(λ) defines a state-space based on the user's input. Some states will be rated higher than others, and highly rated states indicate user preferences. Furthermore, the findings resulting from a training session are sent to the cloud entity together with a sample of the current environment such that a CNN model biased towards the user's preferences can be created. This adjusted model can then be downloaded to the smartphone application and used for later environment analysis. The output of the environment analysis performed by the CNN model acts as a pointer into the state-space that allows for locating highly rated states. When such a state has been found, appropriate hearing aid adjustments for the environment weighted by the user's personal preference has been determined.

A prototype following the above design is implemented and documented in the report. Due to complexity, only a subset of the hearing aid parameters is considered. The prototype focuses on being able to predict bass, midrange, and treble in three high-level environments.

Testing of the prototype reveals that the CNN model has a high classification accuracy (F1 Score of 99%). Furthermore, it predicts hearing aid configuration within ± 0.14 dB and ± 0.77 dB from the target depending on the environment and the parameter. Unfortunately, this accuracy is not reflected when bringing the prototype into real environments. It is found that an inconsistency in how spectrograms are generated on the cloud entity (during CNN model training) and on the smartphone (during environment analysis) is the likely cause of the discrepancy. It is expected that resolving the inconsistency will increase the accuracy. Further assessment of the prototype revels issues with recalling user preferences from the state-space and with the ability of the CNN model and Sarsa(λ) to learn. Results show that new learnings negatively affect the CNN model resulting in less accurate predictions without significantly adapting to the user's preferences. For Sarsa(λ), the training processes seems to have self-destructive characterizes resulting in long training sessions. The report suggests that these issues are due to incorrect configuration of the CNN architecture and the Sarsa(λ) algorithm. It is suggested that a deeper analysis is performed to determine the appropriate configurations for this solution.

The project aims to provide a proof-of-concept prototype to show that it is feasible to develop an environment dependent personalized hearing aid solution. Even though the prototype has some deficiencies it is still believed to be a valid design. However, more development is needed in order to prove that it is viable. Furthermore, the project is aware that user intent is an important aspect, however, due to the complexity of the problem, it is only treated on a conceptual design level.

To validate the idea, four hearing aid users are interviewed. It is determined that the majority is satisfied with the current solution. However, they all show interest in a solution like the one developed in this project. One interviewee even suggested such a solution himself unprovoked by the interviewer. Based on the four interviews, it can be concluded that these four people have a need for adjustments (to varying degree) and that they find a solution like this appealing.

The project does not aim to replace the ordinary fitting process that takes place at an audiologist. Only fine-tuning is of interest for the project. However, adding intelligence like this is a step towards self-fitting hearing aids. Such capabilities can be needed in the future when Over-The-Counter (OTC) hearing aids enter the market. This category describes hearing aids which can be obtained without having to see a hearing care professional. Hence, there is a need for guidance and intelligent procedures to ensure that the user gets the appropriate treatment.

Chapter 1

Introduction

The hearing care industry keeps innovating to create better and more advanced solutions to continuously provide an optimal experience for the hearing aid users. Companies such as GN Hearing [35], Oticon [75], and Widex [111] are incorporating artificial intelligence and bundling their hearing aids with smartphone applications. These applications provide the user with an interface that makes it possible to control the hearing aids. Such control is beneficial as the user can modify the configuration of the hearing aids to suit their preferences or a specific situation. Not all situations are alike and, as an example, noisy environments can present a challenge to a hearing aid user [86]. To limit the unwanted effect, some applications, e.g. the ReSound Smart 3D application, provide a number of listening programs and fine-tuning capabilities. A listening program is a collection of settings that changes the hearing aid configurations to combat the specific challenge [86]. Selecting the appropriate program can e.g. increase the hearing aid user's ability to understand speech or provide better comfort and sound quality. On top of the listening programs, the fine-tuning features allows the user to adjust specific parameters (e.g. amplification level of bass, midrange, and treble) to achieve a more fine-grained controller of how the problem is alleviated and the resulting listening experience. However, it requires knowledge and motivation from the user to evaluate and pick the best suited listening program or perform fine-tuning [86].

A prestudy to this project [23] found that only very few users actively utilize the listening programs, and that the programs are only available to a subset of the hearing aid users. Not all users can comprehend having to choose between multiple programs, however, 80% of the people can benefit from it [86]. It is acknowledged that there is a lack of automation in current hearing aid solutions which creates an inconvenience for the users [86]. As a response, artificial intelligence is becoming increasingly integrated into hearing aid solutions [102]. This results in products that automatically adapt to various environments and even provide a personalized experience. It is no longer just about processing the sound. Now, the hearing aids have to understand the environment and process the audio accordingly.

The prestudy [23] employed the ReSound Smart 3D application to prototype a solution that can understand and adapt to the environment. The purpose was to find a solution for automating the process of selecting the optimal listening program for a given environment. This facilitates a hearing aid solution that is less intrusive and hereby increases the level of satisfaction. Furthermore, it makes it possible to provide multiple listening programs to customers that are currently limited. However, it does not consider the important aspect of personalization. Not all users are equal [20]. Therefore, they might benefit from

different listening programs or prefer different fine-tuning configurations in specific situations. This project builds on top of the prestudy [23] with the objective to incorporate the personalization aspect. Instead of predicting a suitable listening program, the project focuses on how to provide user specific fine-tuning configurations. This requires an additional layer of intelligence. The project aims to construct this layer by utilizing machine learning techniques that are capable of learning a user's preferences in a given sound environment. The objective is to design a solution which can utilize this knowledge and provide the user with hearing aid adjustments that are appropriate for the current sound environment while taking the user's preferences into account.

Based on this, the following problem formulation can be extracted:

- How can machine learning be utilized to learn and predict a user's preferred hearing aid configurations for a given environment?

With the following sub-questions:

- How can a machine learning architecture be designed to accommodate prediction of multiple values from one input?
- How can personalized machine learning models be created and made accessible to a user?

The knowledge gained throughout the project is put into practice as a prototype. To make this possible, GN Hearing has provided a set of ReSound LiNX 3D hearing aids and a Software Development Kit (SDK) which facilitates the communication between a smartphone application and the hearing aids.

The project uses the ReSound brand as a foundation to conceptualize the idea. However, the research and conclusions in this report are not bound to the ReSound brand and can be applied to other similar cases.

This report describes the project from start to finish. The structure is as follows:

Chapter 2 elaborates on the problem faced in this project. The chapter presents the prestudy [23] and outlines how hearing aid configuration preferences can vary between users. The chapter is concluded with sections discussing the expected outcome and the scope of the project.

Chapter 3 describes the methodology followed through the project. This covers the process model, how information is collected, and how the solution is specified, designed, and tested.

Chapter 4 presents research on machine learning techniques that can be used to learn and predict a user's preferred hearing aid configurations.

Chapter 5 discusses the findings from the research and determines the appropriate technological choices for this project. Apart from the technical aspect, two types of hearing aid users are characterized, the risk and assumptions which this project is working under are presented, and solution requirements are defined.

Chapter 6 presents the design of the proposed solution. This includes the layout of the user interface as well as the solution architecture and how to represent the sound environment and hearing aid configurations.

Chapter 7 documents the prototype implementation. The chapter outlines the implementation specific details and deviations from the design.

Chapter 8 describes the results of the prototype testing. The chapter assesses all machine learning components and their ability to learn and predict user preferences.

Chapter 9 presents prototype and solution improvements. This chapter utilizes the findings from the testing chapter.

Chapter 10 discusses the readiness of the prototype and any issues that might be encountered. Hereafter, user interviews are presented which validates the need for a solution like the one developed in this project. Furthermore, the interviews are used to reevaluate a subset of the risks. Finally, the current and future development in the hearing industry is presented.

Chapter 11 concludes on the findings and observations made throughout the project and outlines whether the problem formulations has been answered.

Chapter 2

Problem field

Firstly, the Problem field presents the research done leading up to this project. Hereafter, the chapter outlines why there is a need for personalization in hearing aid solutions as well as how current products are tackling the problem. This is followed by a description of the process of developing machine learning solutions and the stages this project considers. As concluding remarks, the expected outcome and scope of the project are presented.

2.1 Introducing the prestudy

Prior to this project, a prestudy [23] was conducted with the purpose of being able to map a specific environment to a predefined listening program in the ReSound Smart 3D application. This section introduces the problem faced in the prestudy [23] and outlines the findings and conclusions which this master's thesis project builds upon. The prestudy report [23] has been handed in separately as part of an earlier project. For completeness, a copy of the prestudy report [23] has been distributed through different channels to the appropriate parties evaluating this master's thesis.

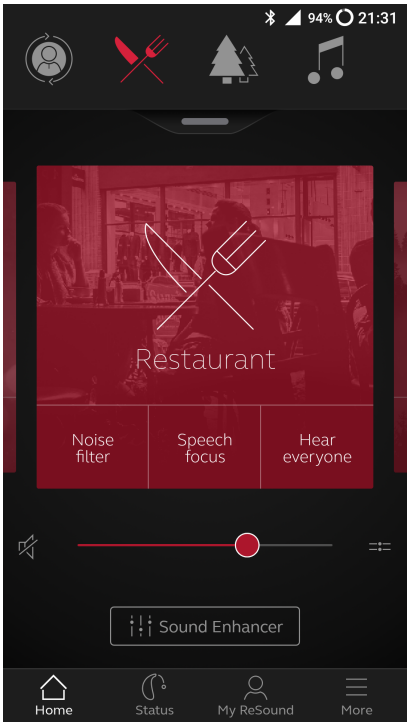


Figure 2.1: The ReSound Smart 3D application. A user interface for managing and adjusting ReSound hearing aids.

Figure 2.1 depicts the ReSound Smart 3D application with "Restaurant" as the selected listening program. The application contains multiple of these programs and the prestudy [23] worked with three of them: All-Around, Restaurant, and Outdoor . From a meeting with GN Hearing Audiology, it was discovered that the various programs do not necessarily map to one particular environment. Instead, the programs are designed to help the user when encountering challenging situations. The All-Around program is a multi-purpose program that equips the user with spatial awareness and is made to fit most situations. However, if a hearing aid user is in a very noisy environment, the Restaurant program is the best choice. This program tries to remove the noise and narrows the auditory field making it easier to have a conversation despite the noise. If the user is in a windy environment, the wind can get caught in the microphones of the hearing aid. This reduces comfort as well as the user's ability to hear other more important sources of sound. The Outdoor program has been designed to improve the experience in such situations. These three listening programs were chosen as it was found that multiple sources of speech, natural sounds (e.g. wind and walking on gravel), and cutlery are among the most often mentioned annoyances for hearing aid users [89]. The annoyance can either be due to the noise being unpleasant (acoustic annoyance such as nails on a blackboard) or because it is masking other desirable sources of sound (informative annoyance such as speech from the person you are having a conversation with being masked by other people speaking) [89].

The problem faced in the prestudy [23] was that the user has to make a conscious decision about the best suited program and needs to interact with either the hearing aids or the smartphone to make the change. The research utilized machine learning to find a solution capable of making this decision

without the user. It is a two-fold problem. Firstly, it requires a method for representing the audio from the environment in a meaningful format. Secondly, a way of analyzing this representation and making a mapping between the important features in the audio and the corresponding listening program has to be found. Succeeding in this would make the hearing aid solution less visible to the user and provide a better experience.

The prestudy [23] was successful and showed good results. Using Short-Time Fourier Transformation (STFT) spectrograms and a Convolutional Neural Network (CNN), a prototype capable of automatically determining the appropriate listening program for the current environment was developed. A CNN architecture is often used for image processing and has an ability to find and learn the important features of an input [18]. This makes it more dynamic and the developer does not have to make the decision of what specific acoustic features define the sound environment. As the computer can process more information than a human, the computer might be able to find patterns that a human cannot. To find a representation of audio that would also suit the sort of input a CNN is designed for, the STFT spectrogram was chosen. This makes it possible to represent the magnitude of each frequency component of an audio signal over time [19]. Multiple types of spectrograms exist, but the STFT variant was chosen for its simplicity and unbiased representation [19, 23].

The prototype performed particularly well when assessing environments that would fit the Restaurant and Outdoor listening program. On top of this, many situations were correctly classified in the All-Around program, however, some inconsistencies were observed. Based on the research and the prototype testing, a number of improvements were listed including the lack of personalization. Defining listening programs and suggesting them to a user based on generalized assumptions might not provide the best possible experience. All users are different and might try to accomplish various things that are outside the ordinary. This is not necessarily wrong as sound experience is a subjective concept, but it is something the prestudy prototype did not consider. Personalization becomes the objective of this project which builds on top of the prestudy [23] and aims to ensure that personal preferences, not a predefined listening program, are suggested for a given environment. The hearing aid personalization aspect is further explored in Section 2.3 (page 8).

2.2 Acquiring and fitting a hearing aid

This section outlines the process of getting a hearing aid including the procedure of adjusting it according to the specific hearing loss of the hearing impaired individual. The section does not give a detailed account of the entire process, but aims to provide a basic foundation for understanding how hearing loss is treated.

Before getting a hearing aid, the individual should be examined. This examination can be medical or audiological [105]. The medical examination ensure that the loss of hearing is not due to other problems such as infections or ear wax. The audiological examination is used to evaluate the degree and cause of the hearing loss [105]. This examination can be performed by a hearing aid dispenser or an audiologist who can then determine the appropriate hearing aid to treat the hearing disability [105]. After a hearing aid has been found, a fitting procedure must be performed.

The fitting process ensures that the hearing aids are adjusted to the specific individual's hearing loss and is based on rules and formulas. These rules and formulas provides a starting point which can then be used to tweak the fitting to the specific individual [53]. These rules and formulas can either be generic

and based on research (e.g. from universities), or they can be proprietary and defined by the hearing aid manufacturer [53]. Different fitting formulas might try to achieve different outcomes. As an example, the NAL-NL2 generic fitting formula tries to maximize speech intelligibility. Proprietary formulas might focus on providing comfortable audio such that people are less inclined to reject the hearing aids [53]. Furthermore, proprietary fitting formulas can take product specific features into account which generic fitting formulas cannot [53].

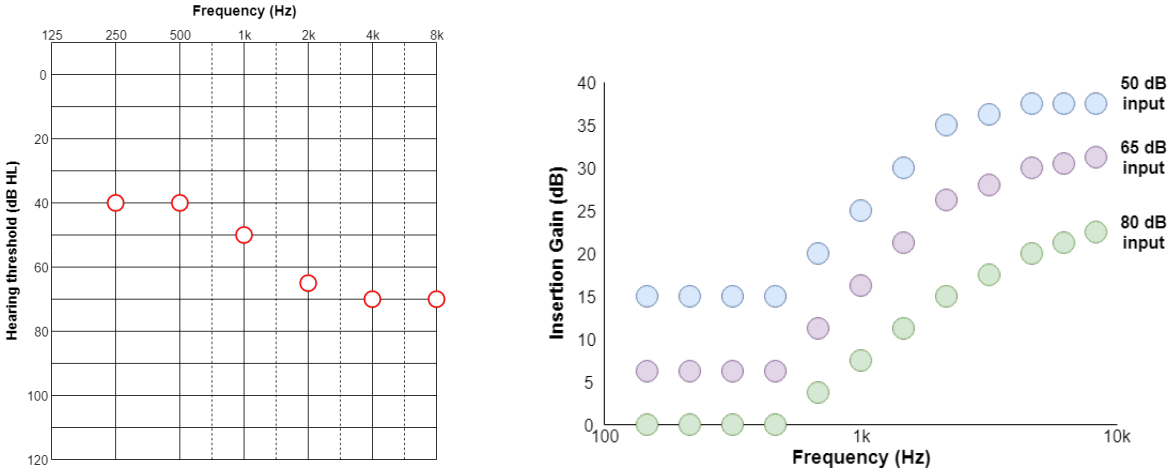


Figure 2.2: An audiogram of an individual with moderate hearing loss on the left and the appropriate frequency specific amplification based on input sound level as suggested by the NAL-NL2 fitting formula on the right. The suggested amplification is shown with dots. In reality, the amplification dynamics are described by a continuous curve. The images are recreated based on Keidser and Dillon [52]. Made with draw.io online tool.

Figure 2.2 depicts an audiogram on the left showing the hearing characteristics of a person with moderate hearing loss [52]. On the right is the suggested gain which the hearing aids should provide according to the NAL-NL2 formula [52]. The audiogram describes the threshold for when the individual can hear specific frequencies. As an example, based on Figure 2.2, it requires a sound pressure of 40 dB for the individual to detect a 500 Hz sound, but it requires 70 dB of sound pressure to detect a sound at 4000 Hz. An average conversation is between 10 dB and 50 dB across the frequency range [9].

On the right of Figure 2.2, the suggested amplification needed according to the NAL-NL2 formula is depicted. Three different suggestions are presented: The blue dots are for soft sounds (50 dB), the purple dots are for medium sounds (65 dB), and the green dots are for loud sounds (80 dB). This is needed as the perception of what is too load does not necessarily change. Only the lower-bound is shifted upwards meaning soft sound should be amplified, but loud sounds should not. Hence, the range from audible to uncomfortable is smaller for a person suffering from hearing loss compared to a person that is not hearing impaired [9]. The blue dots on the right of Figure 2.2 shows that soft sounds should be amplified more than loud sounds and the high frequencies needs more amplification than the low frequencies. This correlates with the audiogram on the left showing a bigger loss of hearing for high frequencies. Even though the amplification suggestions are shown with dots in this examples, they are defined as continuous lines in a real scenario.

This example simplifies the process of finding and fitting the correct hearing aid. The process can be affected by the characteristics and sort of hearing loss [53] as well as age, gender, experience with hearing aids etc. [52]. More work than outlined here is needed to providing a good fitting for a hearing aid user. As an example, tests must be carried to ensure the hearing aid is performing as expected. Keidser and Dillon [52] are hoping that trainable hearing aids can move some of the responsibility of providing the perfect fitting away from the audiologist into the hearing aid and the hands of the consumer. Over time, the hearing aid should learn the optimal fit based on how the user interacts with it (e.g. adjusting the volume) [52].

As mentioned, this presentation of the hearing aid process provides a simplified view. However, it should suffice for following this project. When the project mentions gain, amplification, and hearing aid adjustments, it should now be clear how it relates to the user's hearing and what is actually being adjusted.

2.3 Hearing aid users' listening preferences

This section provides deeper understanding of the differences in preferences among hearing aid users and why this is the case. Furthermore, the purpose of this section is to gain a better understanding of what parameters a hearing aid user will adjust in order to achieve a better listening experience in a given scenario. This information is used to align the expected outcome (Section 2.6, page 14), and to ensure that the project is dealing with the appropriate parameters.

Researchers have agreed that a generalized fitting does not provide an optimal experience for the individual patient [4, 20, 54]. Firstly, how pleasant various sounds are and if the sound quality is good is a subjective evaluation. Furthermore, this evaluation is not static as a user's mood, cognitive fatigue, time of the day, and the specific scenario can affect it [76]. Secondly, even people with the same level of hearing loss can have different ability to separate out e.g. speech from a noisy environment [54]. Hence, there is a need for personalization. This is recognized by hearing care solution companies who are bundling their hearing aids with smartphone applications allowing the users to interact with the hearing aids to make personal and environment specific adjustments. However, researchers argue that current hearing care solutions need more automation [47, 86]. The motivation behind this project is to provide a hearing aid solution that is easier to live with and which the user does not have to worry about.

Korzepa et al. [54] have been observing the behavior of hearing aid users. They found behavioral patterns in how and when the users change listening programs. However, they also found that sometimes the environment changes without the users responding to it. Korzepa et at. [54] speculate that this can be due to the users not being willing to put in the effort to change the program. Getting a suboptimal experience because the effort is to high might reduces the user's satisfaction. This shows that there is a need to automate the process such that the optimal experience can be achieved without putting effort into it.

By observing the behavior of the hearing aid users, Korzepa et al. [54] found that they have different approaches and different intents when personalizing their hearing aid configurations. Comparing two individuals, it was found that one would try to suppress background noise to increase the signal-to-noise ratio in a noisy environment with speech whereas the other test subject would increase the gain of the higher frequencies and increase the volume [54].

Similar results were found by Johansen et al. [47]. Five test subjects were followed over nine months to observe their hearing aid usage behavior and patterns. The study focused on brightness and noise reduction. Brightness is related to the gain in mostly high frequency components of the audio, and noise reduction is the detection and removal of non-voice sound. Increasing the brightness of a sound makes it possible to more easily separate out sound sources and can emphasize consonants. This increases the ability to understand speech but lowers the sound quality [47]. Increasing noise reduction provides a better signal-to-noise ratio making it easier to hear speech but unnaturally dampens the ambient sound [47].

Noise reduction can be achieved with a variety of methods. As an example, assume the purpose is to enhance speech. By detecting pauses in the speech, snippets of pure noise can be collected. When the noise signal is known, it can simply be subtracted from the signal that contains both speech and noise [58]. If an array of microphones is used, filtering can happen by adding the signals from each microphone. A slight time delay is required for this to work, and the delay has to be chosen such that the speech signal is added in phase and the noise signal out of phase [58]. A different approach is to place the microphones such that one is used to collect a noise signal and the other is used to collect a signal with noise and speech. As a result, the noise can be removed from the speech signal [58]. These techniques might make the noise removal process sound easy, however, the world is often not ideal and removing noise is hindered by events like reverberation and fluctuations [58].

Johansen et al. [47] concluded that one listening program is not sufficient to cope with various environments and that coping strategies varies between people [47]. Three of five test subjects chose programs with noise reduction and more directionality when encountering a challenging environment whereas the remaining two subjects increased the brightness of the sound [47]. It was found that patterns of behavior would depend on time, fatigue, context, and preferences. As an example, three of the test subjects often found themselves in work related meeting scenarios. Two of them preferred to increase the brightness, and the third one was in the group of people who tended to use noise reduction instead [47]. So, similar context does not necessarily yield the same hearing aid settings for different individuals emphasizing the need for personalization. Moreover, increasing the brightness does not provide a pleasant experience as sound quality is traded for speech intelligibility and the audio becomes harsh [47]. Furthermore, one of the test subjects changed preference during the study leading to the conclusion that personalization should take place continuously and adapt to the user [47].

Based on the two studies, it can be concluded that people try to adapt their hearing aid experience in various situations such as: Work related meetings, restaurants, bars, driving car and bus, and going for a walk in nature settings [47, 54]. In these scenarios, the users try to achieve a better ability to understand speech, to suppress unwanted background noise, or to be emerged in nature sounds such as leaves and ocean waves [47, 54]. The adjustments are made by changing the directionality of the hearing aid microphones, eliminating non-voice sound with noise reduction, or increasing the brightness of the sound [47, 54]. From this, the important environments, objectives, and parameters which this project should focus on can be extracted.

2.4 Trainable hearing aids

As outlined in Section 2.3, a generalized fitting is not satisfactory for the individual hearing aid user. This section explores the possibility of having trainable hearing aids. A trainable hearing aid learns a user's preferences over time. Such learnability can range from basic memory that will simply remember prior settings to environment classification which predicts the user specific environment preferences [20].

The need for trainable hearing aids arise both due to the need for personalization, but also because the audiologists in the clinics can benefit from it. The trainability enables the user to correct the settings in the hearing aid when they are in a challenging situation. The challenging scenario might have played out a long time prior to a session with an audiologist. In this time, the patient can loss detailed memory of the environment and what specifically made it annoying. Moreover, even if all the details are remembered, the patient might still find it difficult to describe the soundscape and the audiologist might have trouble understanding [20]. As a result, valuable information will be lost in translation. Furthermore, the audiologist has to translate the patient's description into what hearing aid parameters to adjust and by how much [20]. This is all avoided if the user can train the hearing aids in the specific scenario.

This can lead to questions regarding the user's capability of adjusting the appropriate settings. However, the user does not have to. The trainability can be hidden behind a graphical interface just asking the user if the audio is getting better and worse [20]. Behind the interface, specific parameters are being adjusted but this does not need to be visible to the user. However, such interface has to be intelligently designed. As an example, if the use is simply presented with a number of options (similar to the listening programs mentioned in Section 2.1, page 5) and asked to pick the preferred one, the user might not get the optimal setting. Though such approach has been found to be successful, it might also be too slow for any practical implementation [4, 59]. Therefore, the user interface could be made continuous to more easily allow the user to find the optimum. What exactly is meant by a "continuous interface" is exemplified later in this section.

The interface also links to the ease of use. As mentioned in the Introduction (Chapter 1), not all patients have multiple listening programs available in their hearing care solution as it can be too much for them to comprehend. A trainable hearing aid may not be for all patients either [20], but an intuitive and simple interface might help. However, the solution should still understand the user's current context and accommodate adjustment of settings such as frequency specific amplification [20].

Introducing smartphones provides hearing care solutions with less restrictive interfaces and more capabilities in terms of sensor, memory, and computing power [4]. This makes it possible to move into a new realm of hearing care that provides auditory cognition.

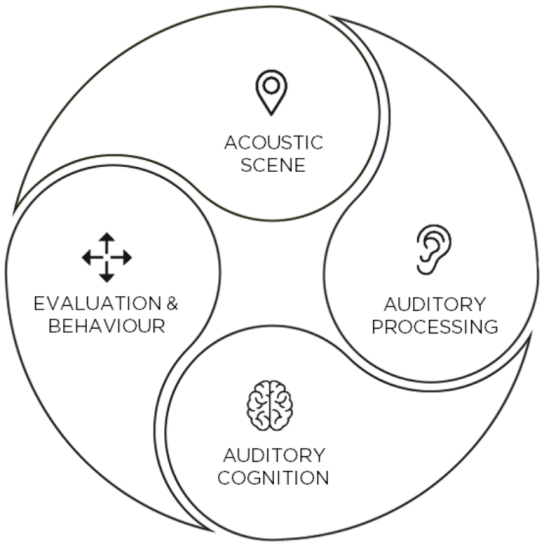


Figure 2.3: The four elements of the hearing process. A framework by Widex. Source: [46].

Consider Figure 2.3 which is a hearing process framework by Widex that describe the elements of the listening process [46]:

- **Acoustic Scene** is made up of all the individual sources of audio in the environment. All the sounds get aggregated into one signal that is received by the ear.
- **Auditory Processing** describes what the ears do to the received audio prior to it reaching the brain.
- **Auditory Cognition** is the process in the brain where the audio is decomposed and understood. As an example, speech understanding takes place at this stage.
- **Evaluation & Behavior** describes if the audio is perceived negatively or positively and how the listener acts based on it.

This framework outlines the direction in which Widex wants to evolve [46]. It also outline what is needed for trainable hearing aids to be successful. Previously, hearing aid solutions have been very focused on the auditory processing [46]. However, to provide personalization, the listening environment must be understood [4, 20]. This changes the focus to the auditory cognition stage where the hearing aid solution is trying to understand the audio input such that it can be intelligently processed.

Hearing care solutions from e.g. Widex and GN Hearing are moving into auditory cognition with the SoundSense Learn solution from Widex [111] and GN Hearing's newly announced Siri[1] integration [36]. The SoundSense Learn solution makes it possible for the user to compare two sound profiles (i.e. hearing aid configurations) and pick the preferred one. Using this information, the solution can learn the user's preferences [111]. As mentioned, such comparisons can require a lot of iterations to find the optimum. However, Widex has provided the user with a slider such that it does not become a binary decision.

[1]Siri is Apple's virtual personal assistant which tries to help the user accomplish objectives such as making phone calls, finding information, and controlling their smart home [8].

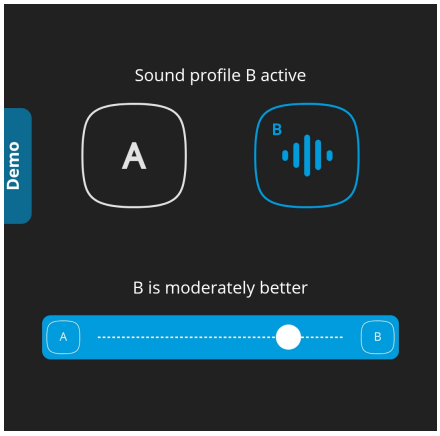


Figure 2.4: Snippet of the SoundSense Learn feature from the Widex Evoke application. The user can compare two sound profiles (A and B) and indicate preference using the slider.

The slider is depicted in Figure 2.4 and allows the user to indicate which configuration (A or B) they prefer and by how much. This is an example of the continuous interface mentioned earlier and might make it easier for the underlying algorithm to distinguish what parameters to adjust without having to expose the complexity to the user.

This section has presented why there is need for trainable hearing aids. Furthermore, a brief review of the current solutions and where the are heading to provide superior experiences through personalization has been presented. This outlines that there is a problem and what the industry is trying to do in order to solve it.

2.5 Machine learning development process

This section serves the purpose of presenting the process of developing machine learning solutions and where this project fits in. Furthermore, the category of machine learning which this project is dealing with is outlined. This is put in relation to the prestudy [23] and why the focus has shifted.

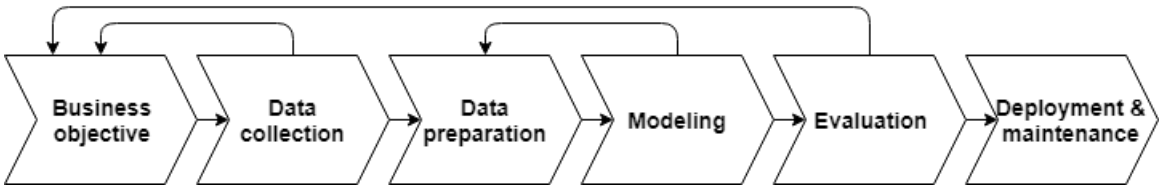


Figure 2.5: The machine learning solution development process. Inspired by: [14].

Figure 2.5 depicts the a slightly modified version of McKinsey's [63] data science approach to allow for explanation of how this project maps into it. Initially, the business objectives and use cases must be understood to determine how data science can be an enabler [63]. The following stage is about collecting the desired data. This might also have been done prior to determining the use cases, however, it is important to ensure the data quality and structure [63]. As mentioned by Hazel Clarke, a Senior

Data Scientist at the Microsoft partner Bluefragments, during her Tech Talk about recent deep learning solutions at Microsoft Development Center Copenhagen, a lack of data quality and structure can limit a company's possibilities. Therefore, this stage feeds into the business objectives as the company has to understand the limitations and possibly reevaluate.

When a realistic outcome has been determined and the data has been properly collected and stored, the data preparation can start. This stage applies human knowledge in order to extract the important features from the data [63]. The preparation stage can get affected by the modeling stage which covers the choice of technologies, the design of machine learning architectures, and the training. To exemplify, the prestudy [23] designed a solution where audio data must be prepared as an image to suit the chosen machine learning architecture.

After the modeling is finished, the resulting model is evaluated to determine if any changes are needed. This evaluation links to the business objectives. More development iterations might be needed before deployment can take place [113].

The prestudy [23] went through the process of understanding the user and the business objectives of GN Hearing. This was followed by data collection and preparation for the machine learning architecture. Hereafter, the solution was evaluated, and a number of improvements were found. The prestudy [23] ended in the evaluation stage and this project has to pickup and modify the project to once again align with GN Hearing's business objectives.

Through a meeting with Morten Lunde, a Product Manager at GN Hearing, the personalization objective arose. Furthermore, previous sections have outline why personalization is important in hearing aids and that some solutions already implement it. Moreover, from a business perspective, McKinsey states that personalization can increase the value of the customer to the business [63]. The business objective has been understood which puts this project in the data collection stage. Knowledge from the prestudy [23] facilitates progress through the data collection and preparation stages. However, for this project, the modeling has to change, and the data collection process will become a continuous task contrary to what is depicted in Figure 2.5.

Machine learning can be divided into various categories two of which are [49]:

- **Supervised learning** defines the process of training a model using labeled data. Such data contains both the input and what the desired output should be. Providing the model with labeled data enables the process of finding patterns between a given input and an output. After the training process, the model is capable of applying the learned patterns to new unseen inputs to predict an output.

- **Reinforcement learning** describes the process of teaching a system how to act on an input based on feedback from an environment. The feedback can be positive or negative depending on whether or not the system is achieving the desired objective. The system will correct accordingly and learn, over time, how to optimally achieve the objective from a given input.

The prestudy [23] collected training data (input) and manually classified each sample into a specific listening program ("All-Around", "Restaurant", or "Outdoor") from the ReSound Smart 3D application (output). Hence, the prestudy [23] utilized supervised learning to determine appropriate mappings between inputs and outputs. However, for this project, the learning process takes place based on input

from the user. The machine learning system will act on the soundscape by predicting a set of appropriate hearing aid configurations. If the prediction is suboptimal, the user will give feedback to the system. Therefore, the field of reinforcement learning is of interest to the project and it is discussed more in detail in the Research (Section 4.3).

2.6 Expected outcome

The section presents a preliminary architecture that serves to explain the expected outcome of the project. Furthermore, the specific settings from the ReSound Smart 3D application that are of interest to the project are presented.

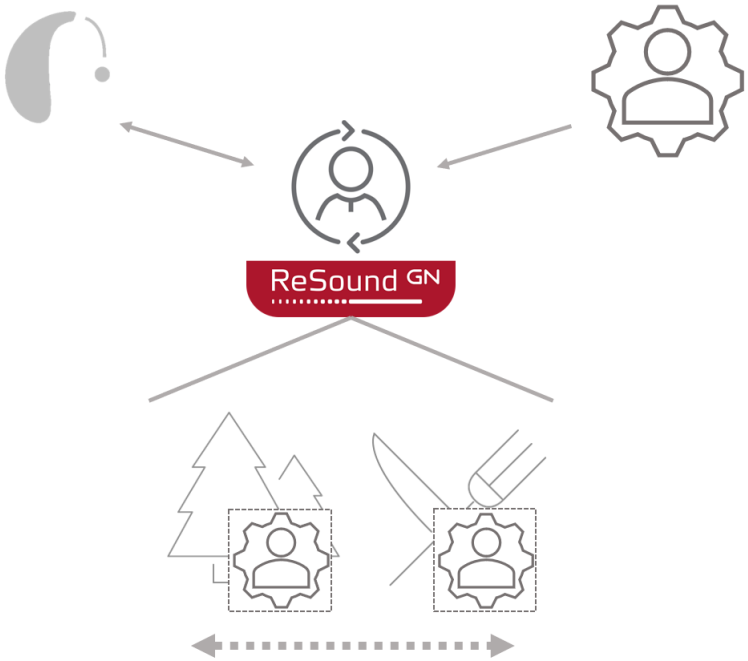


Figure 2.6: Preliminary prototype architecture.

The high-level preliminary prototype architecture is depicted in Figure 2.6 and contains a hearing aid, the ReSound Smart 3D application, and user preference input. The connection between Smart 3D and the hearing aid is bidirectional as the application acts as a remote control. Furthermore, the project assumes that sound can be collected and streamed from the hearing aid to the smartphone application. However, this assumption is known to be incorrect from the outset. Streaming from the hearing aid is not possible with the ReSound LiNX 3D hearing aids available for prototype development. To handle this limitation, Brian Dam Pedersen, the Chief Technology Officer (CTO) of GN Hearing, has suggested to use a small microphone. Alternatively, the audio could be collected by the internal microphone of the smartphone.

The user preference input is presented as the gear-icon in the top right of Figure 2.6. This knowledge is processed by the application in order to learn hearing aid configuration preferences in specific environments. These environments are depicted below the ReSound Smart 3D icon: The two trees symbolize

Outdoor, and the fork and the knife symbolize Restaurant. All-Around is not depicted, but also an environment of interest to the project.

Even though All-Around, Restaurant, and Outdoor is used to refer to listening programs in the ReSound Smart 3D application, this project also uses the naming to denote associated environments. As an example, an environment with challenging amounts of wind is denoted as an Outdoor environment. Throughout the report, the context of the mentioning should make it clear whether the specific text refers to the environment or the listening program.

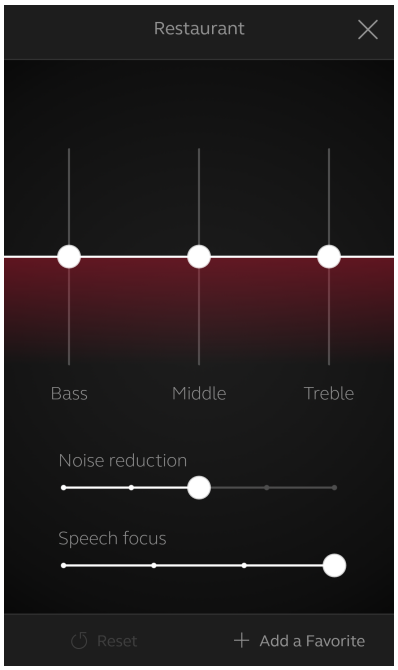


Figure 2.7: Sound Enhancer feature for the Restaurant listening program in ReSound Smart 3D.

The ReSound Smart 3D application has a feature called "Sound Enhancer" which makes it possible for the user to fine-tune a listening program for their specific situation and preference. As a minimum, the Sound Enhancer allows the user to adjust ±6 dB for bass, midrange, and treble. Figure 2.7 shows the Sound Enhancer for the Restaurant program. This program also allows adjustments to the level of noise reduction and the speech focus. For the Outdoor program, the speech focus adjustment is replaced with a wind noise reduction.

Adjustments can help the user achieve various goals. As an example, the Restaurant program has a predefined suggestion called "Hear everyone" which increases the gain 5 dB for bass, midrange, and treble as well as lowering the noise reduction and widening the speech focus. On the other hand, the predefined suggestion called "Noise filter" decreases the gain with 5 dB for bass, midrange, and treble as well as increases the noise reduction and ensures a narrow speech focus.

The expected outcome of the project is a prototype capable of predicting bass, midrange, treble, noise reduction, wind noise reduction, and speech focus for the individual environments based on the user's preferences and apply the changes to the hearing aids. This will ensure a dynamically changing experience that is always optimal and to the user's liking.

2.7 Scope

This section outlines what is within the scope of the project to create a clear understanding of what can be expected throughout the report and the functionality of the prototype. The following points list what is included in the project:

- Section 2.3 presents two studies focused on personalization. Both studies work with noise reducing parameters and frequency dependent gains. Therefore, it has been chosen that this project focuses on the bass, midrange, treble, noise reduction, wind noise reduction, and speech focus settings from the ReSound Smart 3D application. Furthermore, it has been verified with Brian Dam Pedersen, the CTO of GN Hearing, that these are appropriate parameters to work with. However, due to complexity (see Section 5.1.3, page 44), only bass, midrange, and treble are included in the developed prototype.
- Research on machine learning architectures for audio analysis which can be used for numerical predictions. Furthermore, the final solution must be capable of predicting multiple values from one input. Research is devoted to discovering how this is best done for the particular application.
- Research on how to include user preferences in the machine learning model. This covers both the aspect of how to represent user input as well as how to continuously include it in the model.
- A prototype is developed to verify the theoretical conclusion in the real world. The prototype serves as a proof-of-concept and is implemented as an Android application. Furthermore, having a mobile application facilitates testing which is carried out to ensure a satisfactory result and to discover potential improvements.

2.7.1 Limitations

The following list presents the elements that are considered out of scope for this project. Each element is excluded to either keep the project focused and within the time frame or due to technical limitations of the tools and devices utilized for the project.

- A set of ReSound LiNX 3D hearing aids is made available for the development of a prototype. It is currently not possible to stream sound from the hearing aids to a smartphone. For this reason, the prototype must be implemented to collect audio recordings from an alternative source.
- In an actual implementation, the product will have an end-user. Hence, the user aspect should be considered. However, no user testing is performed as this requires a high level of maturity. The prototype of this project served as a proof-of-concept and is not likely to reach that level of maturity. The User Experience department of GN Hearing is consulted to get insights about the user. Interviews with actual hearing aid users are carried out as well.
- To gain more knowledge about a user's preferences and intentions, having more inputs such as movement, time, and location can be beneficial to consider as well [47, 54]. However, to keep the focus of the project, only the soundscape and the explicit feedback from the user is within the scope of the project.

– The General Data Protection Regulation is relevant to consider when developing a solution like this which is capturing user data. A product that reaches the end-user should be compliant. However, GDPR is a major aspect to consider and it is beyond the scope of this project. It is the responsibility of anyone adopting this research to ensure that their solution is GDPR compliant.

Chapter 3

Methodology

This chapter describes the various methods and tools used throughout the project. The following sections outline the method of gathering information and how the project is managed. This is followed by a description of how users are modeled, and how the prototype is defined and documented. Lastly, the testing methodology is presented.

3.1 Information gathering

Multiple approaches are used throughout the project to collect the necessary information. The method of choice depends on the purpose and what sort of information that is to be collected. This section describes the methods and when each of them is used.

Literature review

To understand the technical aspect, literature review is primarily used in this project. Literature can give insights into the various methods and important technical details needed in order to solve the problem at hand. Research covering trainable hearing aids and personalization is of interest as it can give insights into limitations and already existing solutions. This can be combined with how the learning process can be achieved using machine learning in order to design an appropriate solution for this project. In the search for literature, Google Scholar and Primo[1] are the most used search engines in this project.

Public events

To assess the current state of solutions trying to archive an automated and personalized experience for the hearing aid user, the literature review is combined with event attendance. During the period of the project, a career fair arranged by DSE[2] will take place as well as a project exposition arranged by

[1] Primo is the search engine used for the online library of Aalborg University. Primo collects literature from a broad range of more than 200 databases.

[2] DSE is a student association trying to bridge the gap between students and companies. They aim to facilitate the contact between these two parties to ease the process of creating relations in the hope that internships, projects, and job opportunities arise [21].

Neural[3]. GN Hearing, Oticon, and Widex all have a booth at the career fair, and Eriksholm Research Centre (which is a part of Oticon [28]) is at the project exposition. Apart from gaining insights into the current and the future, attending events where the manufactures are present makes it possible to discuss specific technical aspects which can be brought into this project. The discussions with the companies are treated as unstructured interviews allowing for a dynamic interview structure such that the company representative can speak freely [24]. This is done to create a comfortable and natural discussion similar to what is found at such events. However, topics are prepared in advanced to ensure that the discussions are relevant for the project. Furthermore, it is speculated that Widex will display their SoundSense Learn solution (see Section 5.1.3, page 49) at the career fair. Therefore, a performance test of the solution is planned in advance hoping that the solution will at the event and possible to try out by visitors.

Specialist meetings

Apart from the unstructured interview at public events, meetings are arranged with GN Hearing employees from the Connected Apps team, Algorithm department, as well as the User Experience department. These meetings are all carried out as semi-structured interviews allowing for flexibility while being topic-centered [25]. Furthermore, they are formatted as meetings, and not interviews, to make the interviewee comfortable with the situations. To get the meetings started, the project is presented using a couple of slides to create a common ground and set the scope. Furthermore, the topic of the meeting is outlined in a few bullet points in the meeting invitation. This is both to ensure no surprises and to allow the interviewee to prepare if needed. All this is done in an effort to limit misunderstanding and improve the quality of the answers. A list of questions is prepared in advance, however, a conversational style of meeting is encouraged. The interviewee is more knowledgeable on the specific topic and might bring up new information and aspects that cannot be taken into account when preparing the list of questions. The information gathered from these meetings feed into the technical aspect of the project as well as the user modeling and prototype design.

User interviews

As the problem being solved in this project revolves around actual users, the user aspect should be taken into account too. To gather information related to what users think about the solution, four telephone user interviews are carried out (see Appendix H). The telephone approach is chosen due to convenience. Furthermore, some user are more comfortable talking about sensitive subject over the phone instead of face-to-face [26]. Hearing impairment is a disability and possibly a sensitive area for some.

Each user is specifically selected with the help from GN Hearing's User Experience department. They have a panel of users that have been equipped with ReSound LiNX Quattro hearing aids and the ReSound Smart 3D smartphone application. These users are ideal to interview for this project as they are familiar with the same smartphone application which this project is based upon. Hence, there is a common starting point which makes it possible to discuss specific features and improvements.

A list of questions is designed prior to the interviews and is kept consistent for all four users. The users all fit in the user types discussed in this project (see Section 5.2.2, page 58). Even though four

[3]Neural is a newly started community for students interested in artificial intelligence aiming to create a good environment for researchers and professionals. They arrange monthly events and annual symposiums and hackathons to facilitate knowledge sharing and networking [64].

interviews are not sufficient to make any statistically sound conclusions, the input can still bring feedback either confirming the project or indicate a need to pivot the idea.

3.2 Process model

This section presents the project management and software development methodology utilized in this project. They both aim to be flexible such that changes and new findings can be easily integrated into the project. Furthermore, the project management methodology tries to visualize work, and the software development methodology encourages a learn-by-doing approach in order to get early feedback.

3.2.1 Project management

The main project management methodology used in this project is Scrum [84]. However, as the project is carried out by a single student, only a subset of the Scrum components are used. The project has chosen to adopt the concept of a Product Backlog, Sprint, and Sprint Backlog as well as Sprint Planning, Review, and Retrospective. At the very beginning of the project, all the anticipated work was put into a Product Backlog. Furthermore, an initial prioritization was made and it was divided into Sprints. However, the Product Backlog is reviewed throughout the project and reprioritized as needed (see the final edition in Appendix I).

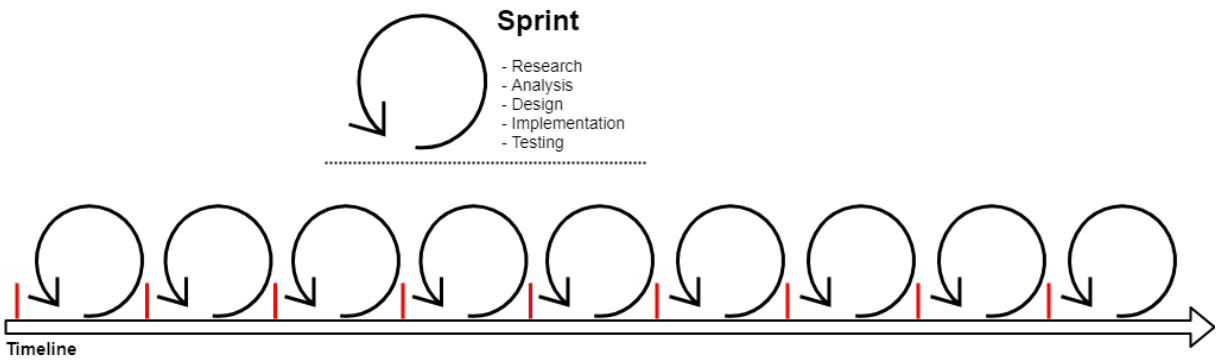


Figure 3.1: Illustration of the project Sprints (black circles) and associated Scrum events (red lines). This project has an identical structure to the prestudy [23] so the same illustration has been reused.

Figure 3.1 presents the timeline of the project which is divided into nine Sprints represented by the black circular arrows. Each Sprint is two weeks long except for the last one which is cut short by a few days to align with the thesis deadline. A Sprint can contain research, analysis, design, implementation, testing, or a combination of these. Between Sprints, the red lines indicate the Sprint related activities: Planning, Review and Retrospective takes place. These activities ensure that the Product Backlog is up-to-date and correctly prioritized, and that new tasks are pulled into the Sprint Backlog [84].

Apart from using this adjusted version of Scrum, the project also utilizes a Gantt chart (see Appendix I.1) and Kanban boards. The Gantt chart provides a visual representation of the entire timeline from start to finish. The aim of adopting the Kanban board is to manage and visualize the tasks defined by the

Sprint Backlog. The Kanban boards are created such that each Sprint has one. The Kanban methodology enforces Work in Progress Limits [6] which is not explicitly defined for this project. With that said, an effort is made to limit the amount of active work such that a flow is created. The Kanban board is made up of five columns: Sprint Backlog, Prioritized, Started, Done, Re-visit later. This structure highlights what has to be done, highly prioritized tasks, active work, and finished work.

With this combination of components and tools from different methodologies, the aim is to efficiently manage the project.

3.2.2 Software development – Rapid Prototyping

The project has many unknown variables as it tries to combine the two fields of hearing aids and machine learning. In order to allow for unknown, changing, or vague requirements, a prototyping approach is taken for the software development. Using Rapid Prototyping is favorable in unfamiliar situations as information can be gained from developing and evaluating a prototype [103]. These findings can then be utilized to formulate a better definition of the objective, requirements, or design. This makes it possible to discover issues earlier in the development process which would otherwise be expensive to fix [103, 48]. For this project, the cost is measured in time. The purpose is to avoid time-consuming changes late in the project due to unsuitable decision as a result of lack of knowledge earlier in the process. The simplicity and flexibility of the Rapid Prototyping methodology is desirable which is why it is the software development methodology of choice.

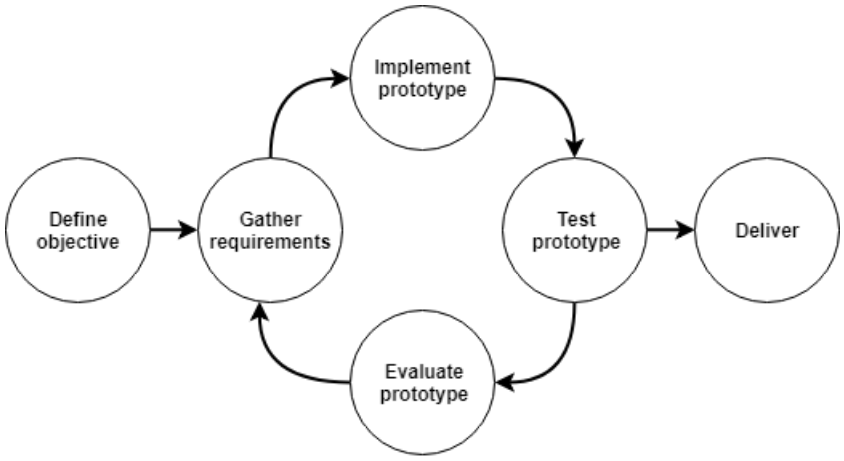


Figure 3.2: Illustration of the Rapid Prototype methodology followed for the software development in this project. Made with draw.io online tool.

Figure 3.2 illustrates how Rapid Prototyping is carried out in the project. Initially, the desired capabilities of the prototype are defined. Hereafter, requirements can be gathered and the implementation can be started. As implementation finishes, testing is performed. If the testing results are adequate, the prototype is delivered meaning whatever has been prototyped is ready for going into the process of becoming an actual product. If the testing shows that the prototype does not fulfill the objective, it is evaluated and the loop starts over.

The aim of the project is to do two loop iterations. After having done the majority of the research,

an exploratory prototype is developed in order to better understand the capabilities of the methods researched (see Appendix B for the documentation of the exploratory prototype). This makes it possible to gain a better understanding of the technical details which can then be used to specify the requirements for the next prototype. The second iteration develops and evaluates a prototype which has more functionality and aims to fulfill the objective. The prototype is presented and evaluated throughout the Design chapter (Chapter 6), Implementation chapter (Chapter 7), and Testing chapter (Chapter 8).

Due to the technical focus of the project, the aim is to make a vertical prototype. This defines a prototype with a narrow feature set, but a high level of functionality for the supported feature [65]. The purpose is to make a good technical solution which can then be delivered and integrated into already existing or future hearing care solutions.

3.3 User modeling

The user modeling performed in this project is based on two meetings with GN Hearing's User Experience department. The first meeting is used to understand what characterizes a user. This includes simple elements such as age and hearing loss, but it also consist of how they feel about hearing aids and their usage patterns. Hereafter, this information is compiled into personas, empathy maps, and scenarios which are then presented at the second meeting. At the second meeting, mistakes or misunderstands can be corrected to ensure the user model does reflect an average hearing aid user.

The first element of the modeling is the personas (see Section 5.2.1, page 57) which are based on an aggregate of data from the first meeting described above. Hence, one persona consists of an aggregate of information about multiple real users [41]. This information is used as input to the design process to ensure that a user actually exists in the receiving end of the solution being developed. Furthermore, the persona is used as a tool for communicating the characteristics of a hearing aid user and who the solution is being developed for.

The personas are followed by empathy maps (see Section 5.2.2, page 58) which are used to understand a user's behavior, pains, and possible gains [31]. These build upon the personas and, in this project, try to gain a deeper understanding of the user with the current solution (i.e. the ReSound Smart 3D application without any intelligent personalization). This makes it possible to highlight what the user is currently struggling with as well as what they currently like. This is used as input to the design such that the developed solution fits the user and possibly eases some of the pains that have been determined.

To finalize the user modeling, scenarios are used (see Section 5.2.3, page 62). The scenarios do not describe the user, but the user with the solution being developed [32]. Hence, the scenarios are utilized to make use examples more concrete. The aim is to make the solution less abstract and shows what it must be capable of. Hence, the scenarios are used to uncover requirements [32]. This information feeds into the requirement specification process and hereby affects the solution design.

3.4 Requirement specification

The project utilizes multiple inputs and iterations to define the requirements. An initial list of requirements is derived from the research, the technical analysis, and the user modeling. In order to verify the

validity and prioritization of this list, a meeting is arranged with a management individual from GN Hearing's Connected Apps team. During the meeting, additional requirements are added and reprioritization performed to reflect the needs seen from a company's perspective.

Both functional and non-functional requirements are defined (see Section 5.4, page 72 and Appendix D). The non-functional requirements are categorized using FURPS to underline what elements of the system they support [27]. Furthermore, both functional and non-functional requirements are prioritized according to the MoSCoW model to describe the level of importance for each requirement [2]. The model divides the requirements into "Must", "Should", "Could", and "Won't" have. "Must" defines requirements that must be fulfilled for the solution to be deliverable. "Should" states that the requirement is important but not vital or a workaround can be made. "Could" requirements are desired but can be left out. Finally, "Won't" defines requirements that are not going to be fulfilled in the current time frame [2]. This project applies this prioritization twice. First, it is done in relation to the prototype being developed in this project. Then, the same requirements are prioritized but as if it was a minimum viable product (MVP), not a prototype, being developed. Only the prototype prioritization is followed by this project, however, by also listing an MVP prioritization it shows how much extra work must be put into the solution to bring it to the next state. Furthermore, it provides a foundation for anyone adopting the idea of this project.

3.5 Prototype documentation

The prototype being developed in the project is mainly documented in the Design and Implementation chapters. The Design chapter (Chapter 6) specifies how the solution should be constructed. The Implementation chapter (Chapter 7) outlines the details of the actual implementation and discusses processes, constraints, and deviation from the design in details. The type of documentation utilized in the two chapters is a combination of standardized UML diagrams and other illustrations. From the UML standard, the following diagrams are utilized:

- Use case diagrams [70] are used to describe system functionality and interactions with an actor. The project utilizes this type of diagram as a requirement specification and visualization tool.
- Sequence diagrams [71] outline flows between system entities. The diagrams specify the order of messages as well as the content.
- Class diagrams [72] are used to describe the components which the system is made of as well as how they are structured and linked.
- Activity diagrams [73] are utilized to describe steps taken along a process flow in order to complete a task. The activity diagrams used in this project purposely deviates slightly from the UML specification. UML defines Fork and Join nodes which have the purpose of modeling splitting and merging flows [73]. These nodes are implicitly assumed in the diagrams used throughout this report when a flow splits or merges. Furthermore, UML defines Initial and Final nodes [73]. In this report, they are replaced with more informative and explicit oblong shapes containing the process name.

Apart from the UML diagrams, other diagrams are used to describe high-level system structures and communicate machine learning architectures. These diagrams are not based on a specific standard or naming but are simply created to fit the purpose and illustrate the context.

3.6 Prototype testing

Three different tests are carried out with the prototype to evaluate it. The testing is focus on the machine learning aspect as this is the main focus of the project. Firstly, the accuracy of the prototype is assessed (Section 8.1, page 117). As the prototype deals with both classification and regression, each aspect is assessed independently. In order to evaluate the regression accuracy, Mean Absolute Error (MAE). MAE is chosen as the resulting number is relatable and easy to interpret as no squaring etc. is done as seen in other error measures for regression problems [92].

In order to determine what measure to use for evaluating the classification accuracy, it must first be understood what sort of classification is being performed. The project works with multiple environments of which only one is assumed to describe the user's current environment. This is a multi-class classification problem [91] and the only important aspect is the model's ability to determine true positives (i.e. correctly predicting a certain environment to fit in a specific class).

Sokolova and Lapalme [91] present a number of measure adapted from binary classification such that they can used for multi-class classification. This project utilizes the Average Accuracy, Macro Precision, Macro Recall, and Macro F1 Score (the equations defining these measures are listed in Appendix G.1).

- Average Accuracy describes the ratio between correctly and incorrectly classified samples, but is affected by true negatives [91]. It is included to show the impact of true negatives.
- Precision describes how many of the samples evaluated to belong to a specific class actually do fit in that class [91].
- Recall describes how many of the samples from a specific class where recognized as belonging to that class [91].
- The F Score is a weighted measure combining Precision and Recall into one accuracy measure that does not include true negatives [91]. The weight can put more emphasis on either Precision or Recall. For this project, the weight is kept as 1 resulting in the F1 Score which is biased towards neither precision nor recall.

These measures also exist in a micro-variant, but those a less desirable for the project as they get biased if the evaluated classes do not contain the same number of samples [91].

The accuracy test is supplemented with a learnability test (Section 8.3, page 130). The same accuracy measure can be used for this test. Essentially, the learnability test just evaluates how much data must be provided to the machine learning in order to make it predict new values. These new values just become the new target values. Hence, the test methodology from the accuracy test can be utilized.

Finally, a third test is carried out to verify if the findings from the accuracy and learnability test are reflected in the real world. This test brings the prototype to a number of common environments in order to observe its behavior. If the prototype has shown a high accuracy and good learnability, this is expected to be confirmed by this test.

Chapter 4

Research

This chapter presents the technical background knowledge needed to design and develop a solution which is able to determine a user's preferences and predict the appropriate hearing aid adjustments based on the environment. The focus of the research is split between two possible approaches (neural networks and reinforcement reinforcement algorithms) and presents different designs and configuration techniques for both. The findings from the research goes into the analysis (Chapter 5) with the aim of determining the most suitable solution for the specific problem faced in this project

4.1 Activation and cost functions

An objective of the project is to perform regression such that an audio input in the form of a spectrogram can be turned into numerical predictions. For classification, Glauner [33] describes that a sigmoidal activation function combined with cross-entropy as cost function is often used. This is indeed what was utilized in the prestudy [23], but this now has to be replaced with an activation function and a cost function that is appropriate for multivariate regression.

This section delves into the common options that exist for activation [69] and cost functions [92]. This information is used in the Analysis chapter (Section 5.1.1, page 41) to make an appropriate choice for the prototype.

4.1.1 Activation functions

An activation function determines the output of a neuron in a neural network and what transformation is performed on the neuron's inputs [69]. Activation functions can either be linear or non-linear. Choosing a non-linear activation function enables the network to discover non-linear mappings from input data to output data [69].

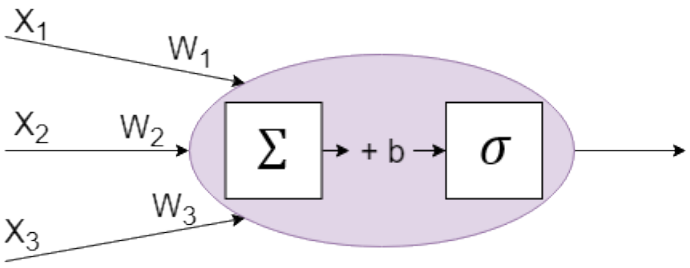


Figure 4.1: The building blocks of a neural network neuron. Source: [23].

Consider Figure 4.1 to understand how a single neuron in a neural network functions. The neuron takes in multiple weighted inputs which are summed up, and a bias is added as an activation threshold [23, 69]. The resulting value is processed by the activation function symbolized by σ in Figure 4.1.

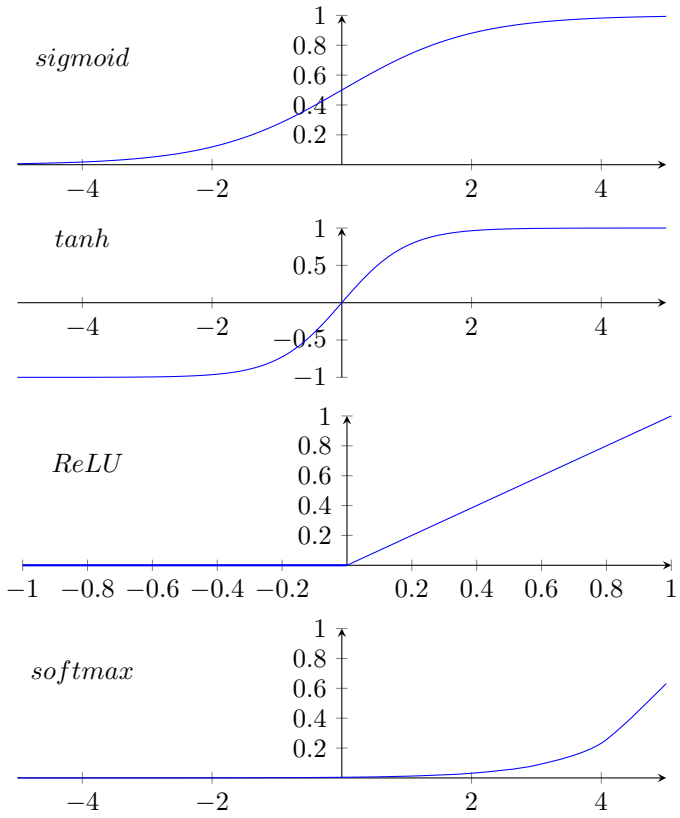


Figure 4.2: Common activation functions: Sigmoid, tanh, ReLU, and softmax. Source: [23].

Figure 4.2 depicts four activation functions: Sigmoid, tanh, ReLU, and softmax which are now briefly introduced:

Sigmoid ranges from zero to one. For this reason, it works well in the output layers of neural networks solving probability-based problems such as binary classification [69]. However, drawbacks include risk of saturation. Due to the function's shape, if the neuron's activation gets too close to

either tail, the learning procedure becomes slow and the neuron saturates [17].

Tanh (Hyperbolic Tangent) ranges from negative one to positive one and improves on the sigmoid function by being centered around zero which improves training of the network [69]. However, like sigmoid, tanh suffers from the saturation issue too due to the shape of the function [69].

ReLU (Rectified Linear Unit) is the most common activation function. It is commonly found in the hidden layer of a neural network combined with a different activation function in the output layer [69]. From Figure 4.2 it can be seen that ReLU is close to being linear. This provides good training characteristics due to easy optimization. Furthermore, it is easy to compute and does not suffer from the saturation problem [69]. However, regularization techniques might be needed as ReLU neurons have a tendency to overfit. Also, neurons can become inactive if the training process is too aggressive. This can cause an irreversible update in the neuron's weights pushing it to a state where activation will never occur [17].

Softmax was utilized in the prestudy [23] as it can be used for probability problems where multiple probabilities should sum up to one. This makes it appropriate in the output layer of neural networks for multi-class classification tasks [69].

Identity is not depicted in Figure 4.2 but is also a function to considered. The identity function is a linear function that simply returns the same value as it was inputted [88]. This makes it possible to get a continuous value as output which is not range limited. Shimobaba et al. [88] have used the ReLU activation function in the hidden layers of a Convolutional Neural Network combined with the identity function in the output layer in order to solve a regression problem [88].

The above list is not exhaustive. Multiple other activation functions exist as well as variations [69]. The variations have been made to offer less computationally heavy alternatives, introduce learnable parameters, or to fix a limitation of the original activation function. This should serve as a good basis for picking a suitable activation function for the regression problem faced in this project. The Analysis chapter (Section 5.1.1, page 41) argues and decides for the appropriate function to use.

4.1.2 Cost functions

Whereas the activation function modifies the output of a single neuron, the cost function describes the entire network's prediction error. The following text list the following common cost functions [92]: Mean Absolute Error (MSE), Mean Squared Error (MSE), and Root Mean Square Error (RMSE). However, first, consider Figure 4.3 to understand the functionality of a cost function.

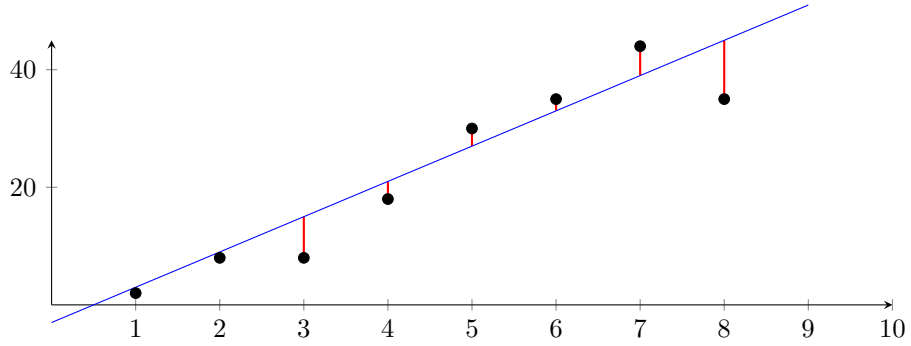


Figure 4.3: Illustration of the error a cost function is trying to minimize.

A regression problem tries to find a model that approximates a set of data points [92]. This model can then be utilized for predictions of new data points. The purpose is to find a good approximation with minimal deviation between predictions and actual data [92]. The purpose of the cost function is to describe this deviation such that a model with smallest deviation can be determined. Figure 4.3 depicts a model (the blue line) that tries to approximate a set of data (the black points). The red lines between the data points and the approximated function is the deviation, i.e. the error that should be minimized.

$$MAE = \frac{1}{k}\sum_{j=1}^{k}\left|y_j - h(x_j)\right| \tag{4.1}$$

Equation 4.1 describes how to determine the Mean Absolute Error (MAE) [92]. k denotes the number of samples being assessed, y_j is the jth data sample (actual value), and $h(x_j)$ is the predicted value. The measure sums up all the absolute differences between the actual values and the predicted values and divides with the number of samples to create an average. By design, the MAE measure is not penalizing outliers [92]. As a result, a model that is accurate most of the time but has high error for a few predictions can have the same Mean Absolute Error as a model that often deviates but only by a little [92]. To penalize rare but high error, the Mean Square Error (MSE) can be considered.

$$MSE = \frac{1}{k}\sum_{j=1}^{k}(y_j - h(x_j))^2 \tag{4.2}$$

As can be seen in Equation 4.2, the MSE measure looks similar to MEA. However, instead of summing absolute deviation, MSE finds the squared difference between actual values and predicted values [92]. Squaring the differences makes it possible to penalize outliers. Big differences are emphasized more than smaller distances. However, interpretation of the measure can be difficult due to the squaring [92]. This can be exemplified by using Figure 4.3 which has MAE = 4, MSE = 24.75, and RMSE = 4.97.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(y_j - h(x_j))^2} \tag{4.3}$$

Root Mean Squared Error (RMSE) is similar to MSE but takes the square root of the final result [92] as shown in Equation 4.3. As can be seen from the above exemplification, the RMSE values is closer to

the MAE value. RMSE keeps the emphasis on the outliers because it is based in MSE, but it is easy to interpret like MAE [92].

Determining what measure is appropriate depends on the application. If outliers are known to be present in the data samples, RMSE and MSE might not give a good indication of the model quality [92]. Furthermore, Willmott and Matsuura have published research describing why RMSE should be avoided [112]. They argue that MAE is the natural measure of error because it reflects the actual average error present in the model. RMSE models both average error and variance, hence, it is difficult to know what the underlying reason for a given RMSE value is [112]. As the RMSE measure is just a scaled version of MSE, their arguments must apply to MSE as well. However, with that said, a study tackling a problem using Convolutional Neural Networks for regression utilized the MSE measure with good results [88]. Furthermore, Ghosh et al. [30] did a study on the robustness of cost functions including MAE and MSE. The study was based on a classification problem but, nevertheless, showed that models created using the MSE measured performs marginally better than models created using the MAE measure in conditions with clean data samples [30]. So, Willmott and Matsuura might not like the RMSE (and hence the MSE) measure, however, the MSE measure appears to perform well.

The above list of cost functions is not exhaustive, but it does outline the three most common cost functions used for function approximation [92]. Hence, one of the presented functions should suit this project. The Analysis chapter (Section 5.1.1, page 41) presents which is used for the prototype development.

4.2 Multivariate regression with neural networks

The prestudy [23] was trying to solve a single task: Based on a spectrogram formatted audio input, predict what listening program is the most appropriate. This was modeled as a classification problem. However, as the objective is no longer to determine a listening program, but to determine the values of the underlying settings, multiple numerical predictions are need. This is a multivariate regression problem which can be solved in a variety of ways [22, 82]. The prestudy [23] found a solution using Convolutional Neural Networks (CNNs), and in order to benefit from the knowledge of the prestudy [23], the project further explores the realm of neural networks. This section presents ways in which the multivariate regression problem can be modeled using neural networks.

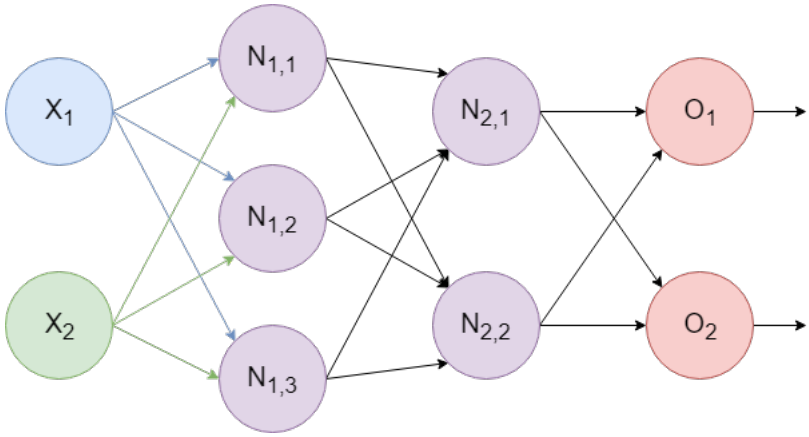


Figure 4.4: Simple neural network model for multivariate regression. Made with draw.io online tool.

Figure 4.4 depicts one way in which multivariate regression can be model using neural networks [22]. The green and the blue node are the input data given to the network. These inputs are processed by the purple neurons which make up the hidden layers. Figure 4.4 has two hidden layers: The first layer consists of three purple neurons and the second hidden layer consists of two neurons. The purpose of the hidden layers is to learn a mapping between the input and the output. If such mapping is learned, sensible predictions are outputted, and the multivariate regression is successful. However, this model assumes that all the neurons in the hidden layers are responsible for producing both outputs i.e. just one mapping from input to output is needed. This can be a challenging task for the network as the different outputs might actually require different input processing that is not closely related [22]. Hence, modeling the problem as depicted in Figure 4.4 can lead to a suboptimal regression model [22].

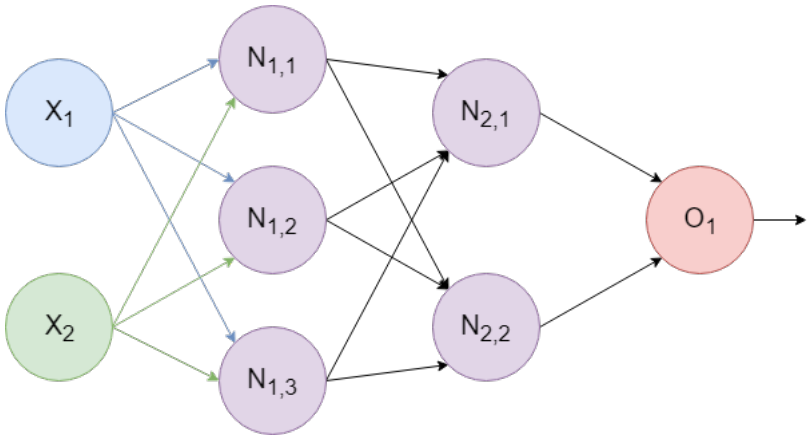


Figure 4.5: Single-task neural network for doing part of a multivariate regression. Made with draw.io online tool.

To avoid limiting a neural network to learn one mapping between multiple inputs and multiple outputs, an approach like depicted in Figure 4.5 can be employed [12]. Figure 4.5 is simply showing one neural network where multiple inputs produce a single output. Assuming that the multivariate regression would

require two outputs (like depicted in Figure 4.4), two separate networks would be trained. Having one network for each output variable makes it possible for each network to create its own mapping from input to output. The entire solution is no longer limited by one network having to create one mapping for multiple outputs.

The methods presented in Figure 4.4 and Figure 4.5 are known as multi-task learning and single-task learning respectively [12]. Even though, as mentioned, multi-task learning limits the neural network's ability to create specialized mappings for each of the outputs, this can serve as a benefit [12]. A neural network can become too good at representing the input data during the training process. This is known as overfitting and results in a less generalized model that preforms poorly on unseen data [38]. Using multi-task learning, the network has to find a mapping that satisfies multiple outputs which reduces the risk of overfitting [82].

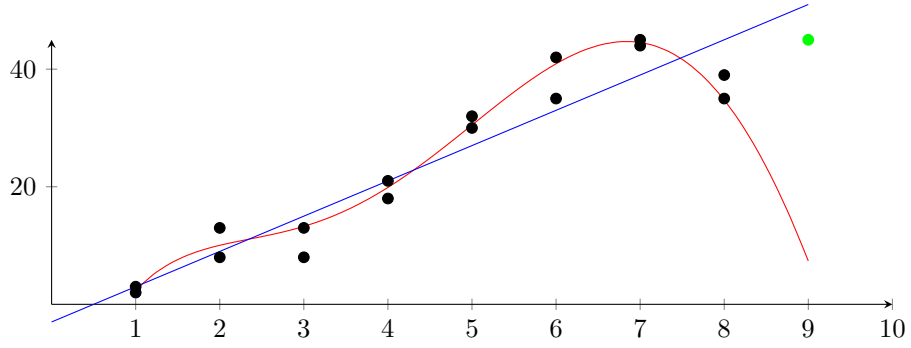


Figure 4.6: Illustration of the overfitting problem.

Figure 4.6 depicts the overfitting problem and visualizes why it is a problem. Relating Figure 4.6 to the neural network, the black points are input data and the red and the blue line is the function (or the mapping) the neurons are trying to learn from the input. The red function resemble the case of possible overfitting: The function fits most of the input points sublimely, however, fails to predict the new unseen green point. The blue function does not fit all the black points as well, however, it does a better job a generalizing the problem as it has a lower error when predicting the new green point.

Another advantage of the multi-task structure is that one task (the prediction of one output) can help the learning of another task [82]. Just like humans, the network can use the experience it has for solving one task to solve a related task. This also means that features or patterns in the data that benefit multiple of the outputs are emphasized which facilitates a more generalized model too [82]. However, this brings back the issue with the network having to find one mapping for multiple outputs which was stated as a negative thing in the beginning of the section.

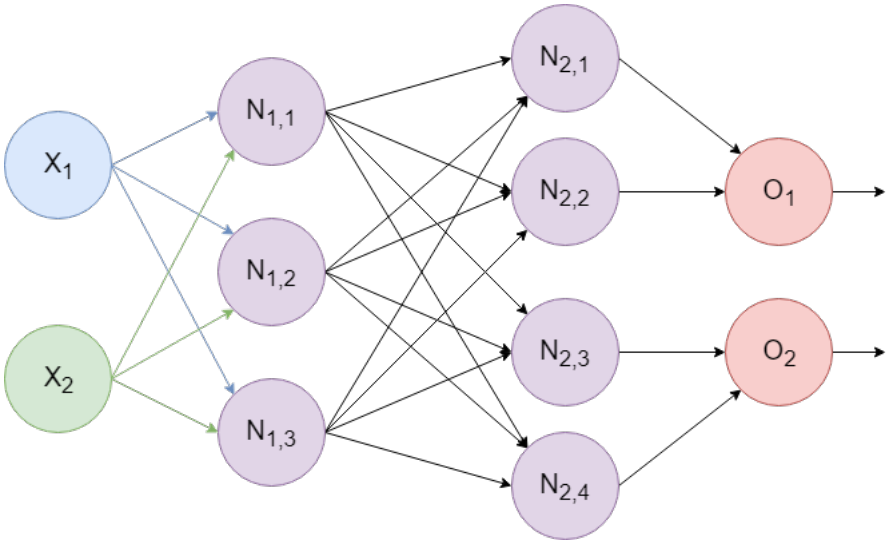


Figure 4.7: A branched multi-task neural network for multivariate regression. Made with draw.io online tool.

In order to find the balance between a very specialized model that has a high risk of overfitting and a model that is unable to represent the real world because a single mapping is not sufficient to represent the multiple outputs, the single-task and multi-task learning models can be combined [12, 82]. Such combination is exemplified in Figure 4.7 which depicts a branched neural network. The green and the blue nodes represent the inputs which are fed to the first hidden layer. This is a shared layer as it will contribute to solving both tasks (prediction of both outputs). This is followed by another hidden layer which is split in two private branches. The branches are private because the upper branch only contributes to the first output and the lower branch only contributes to the second output. Having a shared layer makes it possible for the network to learn some pattern in the data that benefits both outputs. Combining this with the private layers makes it possible for the network to also learn patterns that are specific to the two outputs [82]. In this way, not only one general mapping is learned. The mapping from input to output is slightly specialized which can be beneficial in scenarios where the tasks are not highly related [12].

When to make a branch in a network depends on the tasks at hand [82]. Sebastian Ruder [82] presents a computer vision research that utilizes a series of convolutional layers followed by a series of fully connected layers. The network splits into one branch of fully connected layers for each task after the convolutional layers [82]. This is possibly because all the tasks rely on the same feature extraction by the convolutional layers, but requires a different mapping from these features to the outputs. This individual mapping is created by the fully connected layers.

As presented, multi-task learning is interesting from a model quality perspective, but apart from this, it can also be interesting from an efficiency standpoint. One multi-task network is likely to be bigger and more complex than one single-task network, however, often not as computationally heavy as computing all the individual single-task networks [12].

This section has outlined how a multivariate regression problem can be modeled using neural networks. Figure 4.4 and Figure 4.7 have focused on what is known as hard parameter sharing [82]. In hard

parameter sharing, all tasks share parameters as they are part of the same network. Alternatively, the multi-task learning can be done with soft parameter sharing. In that case, each task has its own network like indicated in Figure 4.5, however, a knowledge sharing mechanism is put in place such that the networks can still share their findings.

Whether this project should treat the multivariate regression as a single-task or a multi-task problem, and what type of branching and parameter sharing strategy should be employed is discussed in the Analysis chapter (Section 5.1.1, page 39).

4.3 Reinforcement learning

The section explores reinforcement learning as a possible solution to learning a hearing aid user's preferences. This type of learning differentiates itself from other machine learning approaches by incorporating aspects of exploration and exploitation [93]. Learning approaches such as supervised learning (which Section 4.2, Figure 4.6, page 31 presents an example of) tries to achieve a generalization from a set of labeled data [93]. This generalization can then be used later to treat new unseen data appropriately. However, this project aims to find a solution than is personalized and can learn preferences. This requires interaction with the specific environment and it is unrealistic to collect training samples in advance for all users in all environments. Reinforcement learning fills the gap by providing mechanisms that can learn from experience [93].

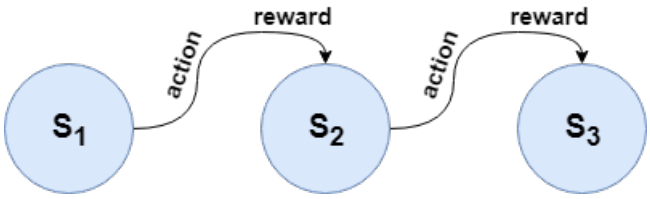


Figure 4.8: Basic reinforcement learning principle showing an agent moving between states and receiving rewards. Made with draw.io online tool.

Reinforcement learning terminology introduces agents taking actions based on policies to change states in order to maximize the agent's reward. The agent does not necessarily have to be a robot. It can be any component of a system that evaluates its current perceived environment and makes changes (takes actions) in order to influence the state of the environment [93]. As depicted in Figure 4.8, based on the current state and the chosen action, the agent reaches a new state and then receives a reward. Figure 4.8 only depicts one path, but the agent may have to choose between multiple actions bringing it to different states which all may return different rewards. Maximizing the total reward is the goal of the agent making reinforcement learning a goal-oriented problem [93, 80] and a close representation of how humans and animals learn too [93].

Learning from experience (instead of pre-collected data) and being goal-oriented leads to the inherent problem in reinforcement learning of balancing exploitation and exploration [93]. The learning agent wants to maximize the reward and to achieve this, it can take the actions that it knows will return the largest reward. This is exploitation [93]. However, the environment might change and, hence, the optimal choice of actions might change as well. The agent will never realize that the environment has changed

and that new actions are more optimal if it does not explore new actions and states. Furthermore, when the agent is first initialized, exploration is needed to gain the first understanding of the actions and what rewards they return [93].

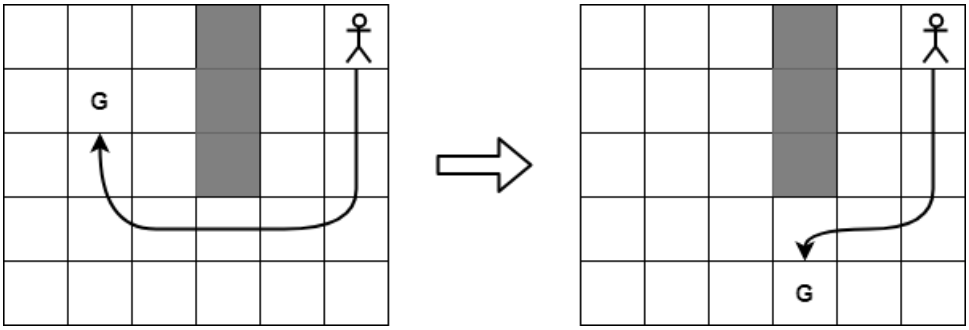


Figure 4.9: A reinforcement learning agent needs to adapt to changing conditions by exploration. Made with draw.io online tool.

Figure 4.9 depicts the need for exploration. Each square represents a state, the grey squares represent states that cannot be visited, the stick figure represents the agent, and "G" represents what the agent is trying to reach. This can be put into context of hearing aids: A state represents hearing aid configurations in a current environment and the goal is to take actions (i.e. make adjustments) to reach state G. State G then represents the user's preferred configuration for the environment. Figure 4.9 then shows a transition in user preferences as G has moved. The agent must encompass a level of exploration to learn where the new goal is. Otherwise, it would end up in the old goal location leading to suboptimal reward for the agent and a suboptimal experience for the hearing aid user.

Reinforcement learning techniques can be divided into an evaluation problem and a control problem [94]. The agent is said to follow a policy which determines its behavior [93] i.e. define what action to take based on a current state. The evaluation problem assesses the current policy and determines a value function [94]. The value function describes how much reward can be expected in the long run when being in a specific state and following the given policy [80]. The control problem aims to determine the optimal policy [94] i.e. determine how the agent should act. This describes how much reward can be expected when taking a certain action in a specific state and hereafter following a given policy [80]. The evaluation problem focuses on a value function for the states (state-value function) whereas the control problem focuses on a value function of the actions (action-value function) [94]. For this project, the interesting aspect is the control problem as the prototype solution being developed should learn how to act based on the current environment including the user preferences. The remaining of this section presents well-known and widely used algorithms which can be utilized to solve the learning problem faced in this project.

4.3.1 Q-learning

The following text introduces Q-learning which is one of the early reinforcement learning algorithms [94]. Even though it is one of the early methods, Q-learning and variations of it are widely used and found in systems ranging from communication networks to financial systems [107, 114]. Q-learning has also proven useful as a component for a user preference based indoor environment control systems. Cheng et

al. [13] developed a system for increasing comfort while decreasing energy consumption in an office space. The system was able to control the lights and the blinds to adjust the level of light in the room. The goal of the system was to use the sunlight instead of the electrical light while keeping the office worker comfortable. If the worker was blinded by the sun or if the room was too dark, this could be given as an input to the system. Using these inputs, the system learned the worker's preferences over time. It was found that the system behaved well, and that the users only needed to correct the system in the beginning of the usage period [13]. This is an example of Q-learning being used to learn user preferences and illustrates why it is an interesting approach to consider for this project.

As mentioned, reinforcement learning can be divided into evaluation and control problems. Q-learning is an interesting method to consider as it is a specific implementation of Temporal-Difference (TD) learning for solving the control problem [94]. TD is out of scope for the report, but it is one of the fundamentals for the theory of reinforcement learning [94]. It should suffice to say that TD is advantageous over the alternatives as it can evaluate the value of a state while exploring, and it does not require a model of the environment dynamics [94]. Returning to Figure 4.9 this means that TD methods can adjust the estimated state values based on the estimate of the coming state before reaching the final goal, G. As a result, learning is faster and problems that do not have a specific termination state can still be solved [94]. Furthermore, no model is required meaning the agent does not need to know the relationship between the states or the rewards [94]. Both Q-learning described in this section and Sarsa presented in Section 4.3.2 (page 36) are TD control problem algorithms [94].

A characteristic of Q-learning is the fact that it is an off-policy learning method [94]. Off-policy refers to how the method tries to approximate the optimal policy [95]. As mentioned, there must be a balance between exploitation and exploration. However, exploration is likely not going to yield the optimal behavior. As a solution, the algorithm can employ one policy to follow while learning about the second policy [95]. This is off-policy learning.

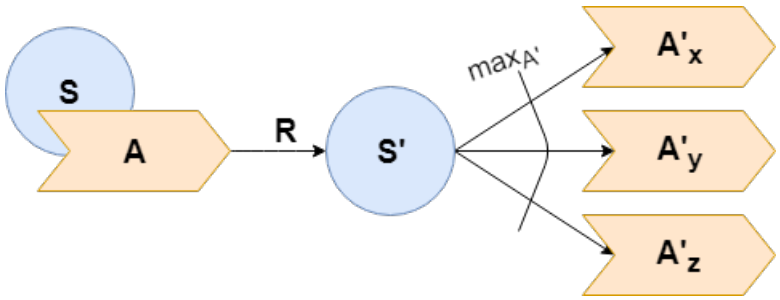


Figure 4.10: Backup diagram of Q-learning. Made with draw.io online tool. Inspired by: [94].

Figure 4.10 depicts how Q-learning works and can also serve as a visual representation of what makes it off-policy learning. The agent is initially in a state (S). In this state, a number of actions are available and the agent picks one (A) based on a behavior policy (which can allow for exploration through random actions). This brings the agent to a new state (S') and returns a reward (R). In the new state, the agent has to decide from a new set of actions ($\{A'x, A'y, A'z\}$) and picks one based on the target policy. For Q-learning, the target policy is to choose the action with the highest value ($max_{A'}$) [94]. From this explanation, it can be seen that the behavior policy can be different from the target policy allowing for

exploratory behavior while learning what actions are best in a certain state.

$$Q(S,A) \leftarrow Q(S,A) + \alpha * (R + \gamma * max_a(Q(S',a)) - Q(S,A)) \tag{4.4}$$

Figure 4.10 does not directly reflect how the algorithm learns. To understand this, consider Equation 4.4 [94]. $Q(S,A)$ estimates the action-value function as it describes the value of taking action A in state S. This should become as close as possible to the actual action-value function. To achieve this, the error between the current estimate and a better estimate must be found. The idea of an error is similar to what is presented in Section 4.1.2 (page 28). However, for Q-learning, the error is defined by the reward, the best action in state S', and the current estimate of the value of action A in state S [94]. The α value is a learning rate and it determines how quickly the current state-action estimate ($Q(S,A)$) should be adjusted by the error [80]. The γ value determines how much emphasis should be put on the value estimate of the current best action in state S' [80]. These two values are adjustable and together with the error determines how much the $Q(S,A)$ estimate is corrected. Next time the algorithm visits the same state S, it will have a new value for action A. As a result, A might no longer be the best choice, and the agent will change its behavior by taking a different action. On the contrary, A could also be further emphasized as the correct choice. In conclusion, updating $Q(S,A)$ allows the agent to learn how to behave optimally [94]. The full Q-learning algorithm presented by Sutton and Barto in chapter 6 of their reinforcement learning book [94] can be found in Appendix A.4. For a numerical example of how the action-value function estimates are updated, see Appendix A.1.

Equation 4.4 also demonstrate the need for exploration. The target policy focuses on actions returning the highest value. If only the highest valued actions are taken, $Q(S,A)$ is only updated for a subset of the states and actions [95]. As a result, the remaining states and actions are unknown territory and never get visited even though they might resolve in a better solution.

4.3.2 Sarsa

As an alternative to Q-learning, the following text presents the Sarsa algorithm which can also be used to solve the control problem. Sarsa can be found utilized in recommendation systems as well as in electric vehicles. In recommendation systems, Sarsa can be used to improve the exploratory aspect of the system [81] and find a balance between exploration and exploitation. In the electric vehicle project, Brusey et al. [10] successfully used a variation of Sarsa (Sarsa(λ) presented in Appendix A.3) to keep a comfortable temperature in a vehicle cabin while being energy efficient. The project did not directly include user input as it was argued that initial learning should be done with simulations. Hereafter, the system could be implemented into an actual vehicle and learn user preferences based on the inputs given to the climate control by the user [10]. Like Q-learning, Sarsa seems to be applicable in a variety of problems.

Sarsa gets its name from the events it takes into account during the action-value function estimation: State, Action, Reward, next State, next Action. Unlike Q-learning, Sarsa is an on-policy learning method [94]. Unlike off-policy learning, on-policy learning uses the same policy for the target policy as well as for the behavior policy [95]. This means that the policy which determines how the agent should act is also the policy that is being optimized. This creates a problem since just one policy then must allow the agent to explore the environment while being optimal. Exploration is not guaranteed to be optimal, hence, one might think that this requires two different policies like seen with Q-learning (Section 4.3.1, page

35). However, Sarsa gets around this issue by not promising an optimal policy, but a close to optimal policy which then allows for some level of exploration [95]. From this, the advantages and disadvantages of Q-learning and Sarsa can be extracted: On-policy approaches in general are simpler as they only need one policy, and they tend to converge faster [95]. However, for Sarsa to provide an optimal policy, the level of exploration must be reduced as learning takes place [94]. Off-policy approaches are said to have more uses and be better at generalizing [95].

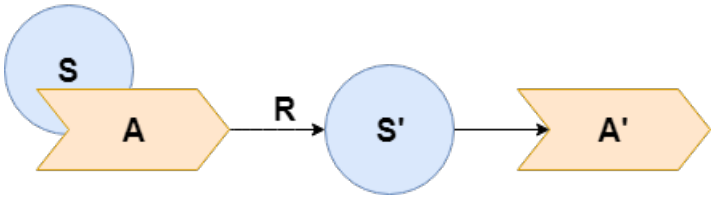


Figure 4.11: Backup diagram of Sarsa. Made with draw.io online tool. Inspired by: [94].

Figure 4.11 depicts how Sarsa incrementally works through states and actions. The agent is in an initial state (S) and picks an action (A) based on the policy. This brings the agent to the new state (S'), and the agent receives the appropriate reward (R) which is later used to adjust how well the choice of S and A was. Since Sarsa is on-policy learning, the agent again follows the same policy and determines a new action (A') to take [94]. Comparing with Q-learning depicted in Figure 4.10, the two approaches seems similar but differ in how they choose the next appropriate action (A'). This is also reflected in the action-value function estimation equation (Equation 4.5 [94]).

$$Q(S,A) \leftarrow Q(S,A) + \alpha * (R + \gamma * Q(S',A') - Q(S,A)) \tag{4.5}$$

Equation 4.5 describes how the action-value updates are performed in Sarsa [94]. $Q(S,A)$ denotes the current estimated value of taking action A in state S, and R is the received reward when taking A in S. How much emphasis should be given to the error term is determined by the adjustable learning rate (α), and the importance of the value of the following state-action pair ($Q(S',A')$) is dependent on the discounting factor (γ) [80]. Like Q-learning (see Equation 4.4, Section 4.3.1, page 36), the error is determined by the reward, the estimated value of the action in the next state, and the estimated value of the action in the current state [94]. However, Sarsa differs from Q-learning as follows: Q-learning determines the estimated value of the next state-action pair using $max_a(Q(S',a))$ as a policy. This always picks the action resulting in the largest state-action pair value for state S'. Sarsa simply uses the same policy to determine A' as it did to find A (which is not necessarily the action with the highest value) [94]. This underlines what is meant by on-policy and off-policy, and how Q-learning is using two separate policies whereas Sarsa is using just one. The full Sarsa algorithm presented by Sutton and Barto in chapter 6 of their reinforcement learning book [94] can be found in Appendix A.5. A numerical example of how the action-value function estimates are updated is presented in Appendix A.1. The example is based on Q-learning, however, serves as an example for Sarsa as well.

The differences between Q-learning and Sarsa might seem minute, but the advantages and disadvantages of on-policy and off-policy listed earlier in this section shows that it does make a significant difference.

4.3.3 Q-learning and Sarsa variations

Both Q-learning and Sarsa exist in different variations which are also of interest to the project. The following text briefly introduces the advantages and disadvantages to Dyna-Q (a Q-learning variation) and Sarsa(λ) (a Sarsa variation). The details can be found in Appendix A.2 and Appendix A.3 respectively.

Dyna-Q and Sarsa(λ) are more complex and computationally costly compared to the more basic approaches. However, on the other hand, they both have the advantage of being able to learn faster and extract more information from the available data [96]. Dyna-Q accomplishes this by creating a model of the environment which contains information about what actions must be taken to reach specific states and where rewards are located [97]. By creating such a model, Dyna-Q is able to create simulated data. As a result, an agent can be trained on few interactions with the real world. These few interactions are the basis for the model which then creates more simulated interaction for the agent to learn from [97]. Sarsa(λ) does not require a model but utilizes eligibility traces instead. This is a technique for keeping track of what states and actions have contributed to receiving a reward or reaching a goal [96]. Such information can be used to update multiple state-action pairs whereas the traditional Q-learning and Sarsa only update one state-action pair at a time [97, 96]. Furthermore, keeping track makes it possible to assign updates based on how much a specific state-action pair contributed to reaching that goal. Hence, an action taken in a state just before receiving a reward is valued higher than the actions taken multiple steps prior to this event.

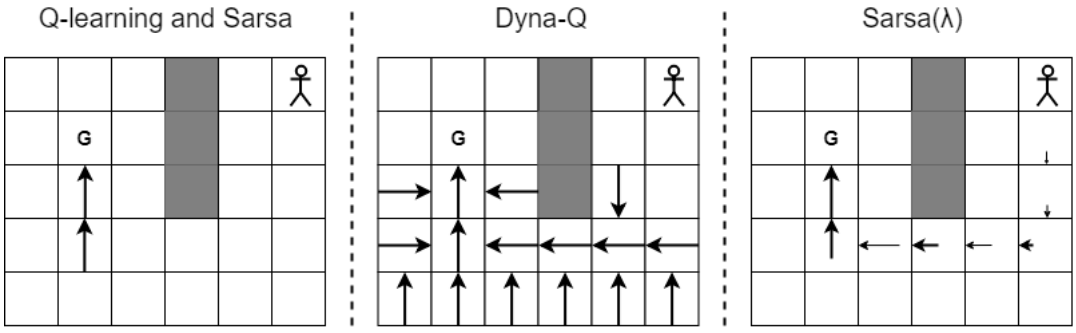


Figure 4.12: Comparison of Q-learning and Sarsa, Dyna-Q, and Sarsa(λ). Made with draw.io online tool.

Figure 4.12 visually presents how the methods are different. The image illustrates what an agent could have learned in two episodes. For Q-learning and Sarsa (the basic approaches), the agent has learned how to act in two states prior to the goal, "G". In the same time span, the Dyna-Q agent has, based on its model, realized what action to take in many of the states. This also makes the weakness of Dyna-Q apparent: If Dyna-Q learns an incorrect model or the environment changes, this might not be picked up by the algorithm and the agent can behave suboptimally as a result [97]. The Sarsa(λ) agent has not learned as much as Dyna-Q yet, but it has an idea of what path to take to get from the starting state to the goal. The varying sizes of the arrows in the Sarsa(λ) example illustrates that the state-action pairs closer to the goal are higher valued.

The fact that Dyna-Q and Sarsa(λ) can learn quicker and need less interactions with the real environment makes them both interesting approaches for this project. The Analysis chapter (Section 5.1.2, page 41) presents a discussion that aims to conclude which technological choices are best for this project.

Chapter 5

Analysis

This chapter utilizes the knowledge gained from the Research chapter by considering advantages and disadvantages in order to arrive at a technical solution that fits this project. This is combined with considerations regarding the scalability of the state-space, how to assess user context and intent, and how to determine an optimal hearing aid adjustment. This is followed by an assessment of who the typical user is for such a solution. All this information is used to evaluate the risks and assumptions of this project and to determine the requirements needed for the solution being developed.

5.1 Technical analysis

The technical analysis discusses the various machine learning techniques considered in this project to determine which is most appropriate. Neural network architectures and reinforcement learning algorithms are discussed separately, however, it is found that they can possibly benefit from each other. The section also presents solutions from Widex and Eriksholm Research Centre which have already been developed or are in the processing of being developed. The section dives into these two solutions to see what their approach is. This is followed up with a discussion regarding how to determine that a user's preference has been found. This is important as it can set the termination criterion for a training session. Hereafter, the section presents thoughts on how to collect environment information which is needed to reliably predict user preference. Lastly, all the findings are collected in a conclusion of the technical analysis.

5.1.1 Learning architecture - Neural networks

The Research chapter presents multiple approaches which can help solve the problem of predicting the preferred hearing aid settings of a hearing impaired user in a given environment. Among the discussed solutions are neural networks with a focus on multivariate regression (Section 4.2, page 29). Such networks have the ability to create a mapping between an input to a continuous numerical output [22]. The prestudy [23] used a certain type of neural networks (Convolutional Neural Networks) to create a mapping between an image (a spectrogram) and a class label. Combining the knowledge that a neural network can analyze spectrograms and that it is capable of producing a numerical output, makes it possible to predict numbers from spectrograms. This is interesting as these numbers can represent the hearing aid

configurations. This, of course, assumes that the neural network is capable of finding a relationship between patterns in spectrograms and the configurations. Hearing aid configurations consist of multiple parameters in this project: Gain of bass, midrange, and treble as well as speech focus, noise reduction, and wind noise reduction levels. The similarity between these parameters can affect how the neural network architecture should be designed [82]. As mentioned in Section 4.2, the problem can be solved with a rather simple architecture (see Figure 4.4, page 30) if only one mapping between the input (a spectrogram) and all output parameters is required. However, this assumes that the information used to predict one parameter can also be used to predict the remaining parameters [82]. If this is not the case, a more complex architecture is needed (as an example, see Figure 4.7, page 32). To make the problem more comprehendible, consider the problem of determining the area of a circle and a rectangle. Finding the area of a circle is a function that takes the radius and outputs the area. In other words, it is a mapping between the radius and the area. Using the same mapping to determine the area of a rectangle will not produce accurate results. In the same way, a neural network is a function approximation that creates a mapping [22], however, one mapping might not be suitable to predict the value of multiple parameters.

Whether a simple multi-task neural network architecture is preferable over multiple single-task networks or a branched multi-task network depends on the scenario. In terms of this project, there is a need to predict the value of six different parameters. Using multiple single-task architectures, the solution can focus on one parameter at a time. However, even though each individual single-task network can be smaller and simpler than one multi-task network, using multiple networks increases the complexity. More networks must be designed, and they might not be created equally. Furthermore, as single-task networks do not have the same regularization element built in by design [82], great care is needed for the design of each individual network. Moreover, they all have to be trained from scratch which might increase total training time compared to other architectures. The added complexity might not be justifiable when considering the type of parameters which will be predicted in this project. As an example, the ReSound Smart 3D application has a "Speech focus" mode for the Restaurant program. Enabling this feature makes changes to multiple parameters including an increase in both treble and noise reduction. A similar relationship between treble and noise reduction is found when choosing "Speech clarity" which is a mode available in the Outdoor program. Having the intent to focus on speech seems to be connected to an increase in both treble and noise reduction [98] meaning the algorithm might also benefit from knowing this relationship and take that into account in a combined mapping. This argues for this project to utilize a type of multi-tasking architecture.

Another argument for not choosing multiple single-tasked networks is the fact that the prestudy [23] demonstrated the suitability of Convolutional Neural Networks (CNNs) for audio analysis on spectrograms. The initial layers of a CNN are used to perform feature extraction on the input [18]. If multiple single-task CNNs are created, they each have to learn how to interpret a spectrogram and do the feature extraction. Using a multi-task CNN, the feature extraction knowledge can be shared. Alternatively, a branched multi-task CNN can allow for shared feature extraction while having individual mappings between features found in the spectrogram and parameter predictions.

A knowledge sharing mechanism could be put in place if multiple single-task networks were desired. However, this would further increase the complexity of having multiple single-task networks. This might be beneficial as the networks can be trained individually while still benefiting from shared knowledge. However, it is not believed that this project will benefit from training individual networks. With the

reasoning presented above, multi-task networks are advantageous compared to the single-task networks. Hence, this project should utilize the multi-task approach and limit branching to create a simpler network which shares as much knowledge as possible. The network has to adapt to the user over time. The data generated by the user might be scarce. Therefore, it is determined that the data available should be utilized thoroughly.

Activation and cost function

In order for the neural network to create the complex mappings and learn, an activation function and a cost function must be determined. The Research chapter listed a number of popular activation functions (Section 4.1.1, page 25) and cost functions (Section 4.1.2, page 27).

In the hidden layers of the neural network, a suitable choice seems to be the Rectified Linear Unit (ReLU) activation function. It is widely used and proven, but can cause overfitting [69]. However, by utilizing the multi-tasking neural network architecture, there is already made an effort to minimize this problem. In the output layer, the activation function must make it possible to create both positive and negative numbers as the bass, midrange, and treble parameters have a range from negative six to positive six. The Identity activation function is not limited to a specific range of numbers, and it has proven useful in other CNN regression problems [88]. It seems suitable for this project as well.

In terms of cost function, three were presented (Section 4.1.2, page 27): Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The MSE measure was found to perform well and has the benefit of punishing big mistakes more than small mistakes [92]. This is found beneficial to this project as being off by one decibel might not be noticeable to the user. However, being off by multiple decibels is noticeable and should be punished considerably more than something unnoticeable (more on when a change become noticeable in Section 5.1.4, page 55). Alternatively, RMSE could be used, however, it adds unwanted computational complexity. Hence, the cost function of choice is the MSE measure.

5.1.2 Learning architecture - Reinforcement learning

Discussing single-task and multi-task architectures does not necessarily lead to a suitable solution as the learning aspect has not been covered. This can be added by considering continuous learning techniques. Continuous learning covers a multitude of scenarios including the addition of new data that should be used as training data after an initial training has already finished [56]. However, this project is aiming to develop the solution as a smartphone application, and it might be infeasible to train neural networks on a smartphone. This issue was researched by Ran et al. [77] who recognize the limitation of a modern smartphones and determined that there is a need for a backend server to take the load when machine learning tasks become too demanding. Their study focused on labeling of content in a video stream using a CNN. It was found that the tested smartphones were not capable of processing and labeling the frames quickly enough. Depending on the image resolution and the machine learning model used, processing a frame took between 200 and 4500 milliseconds [77]. Similarly, they also recognized that services from companies like Google and Apple often do not run in real time or off-load the processing to more powerful backend servers [77]. This is backed up by reviewing some of the current machine learning frameworks such as TensorFlow Lite [99], Core ML [7], and Caffe2 [11] which enables a developer to deploy pretrained

machine learning models to smartphones. However, only being able to deploy pretrained models limits the smartphone to doing predictions only. With that said, Deeplearning4J is a deep learning library which does allow for training neural networks on Android devices [90]. However, the documentation of this functionality does state that such training is supposed to be done on powerful computers and will be slow on smartphones [90].

From this it can be argued that the neural network architectures presented in the Research chapter (Section 4.2, page 29) are not a suitable choice for this project as learning on the smartphone seems unviable. Not being able to learn on the smartphone can complicate the aspect of learning the user preferences. As an alternative, the Research chapter also presents reinforcement learning techniques (Section 4.3, page 33). These are rather simple approaches compared to a neural network and might be suited better for implementation in a smartphone. To verify this claim, an exploratory prototype has been developed and documented in Appendix B.

The reinforcement learning technique of choice must be able to quickly grasp what the user is trying to achieve (more on how quickly a training session should finish in Section 5.1.3, page 47). The research shows that Sarsa(λ) and Dyna-Q are capable of estimating the value of multiple state-action pairs at a time [96, 97]. This makes it possible to more quickly realize what the user is trying to achieve. Hence, this eliminates the more basic Q-learning and Sarsa methods as they require more data and more time to achieve a personalized result. Sarsa(λ) is advantageous as it does not develop a model of the environment. Such model can become outdated over time and not truly represent the actual environment. Because the model does not represent the actual environment, the agent behaves suboptimally and might never realize it [97]. Sarsa(λ) uses the eligibility traces instead, but they are not persistent in the same way as the model. Hence, this might allow Sarsa(λ) to learn a changing environment and generalize better. On the other hand, Dyna-Q is an off-policy approach allowing for learning about the optimal behavior policy while following a more exploratory policy. Exploration is needed in order to learn about the environment, but it does not guarantee that the optimal solution is found. Controlling the exploration factor might be needed for Sarsa(λ) as it only utilizes one policy.

Dyna-Q is capable of generating simulated data using its model and needs fewer interactions with the user as a result. As mentioned, if the user changes preferences, the model will become incorrect. This might happen, but probably not during one training period. The issue of changing preferences over longer time periods can possibly be solved by implementing the model such that it forgets over time or makes actions that have not been tried in a while more likely (like the Dyna-Q+ algorithm [97]). However, this might still not be sufficient in the short run. Even though the user preference does not change in the short run, users are not as consistent as computers. During a training period, the user might accidentally return an incorrect reward. If a model records such a mistake and tries to create simulated training data, this data will not be representative of the user's real opinion. All the training data is incorrect, but the system will not know that and will adjust according to it. This might become difficult to clean up and require many interactions with the user to correct. This is not to say that a similar mistake will not affect a Sarsa(λ) system, but Sarsa(λ) does not create simulated data and, hence, might be less affected by such a user mistake. The user interface should try to minimize such mistakes from happening, but a fault tolerant algorithm is desirable.

Both Sarsa(λ) and Dyna-Q have advantages and disadvantages. However, from the discussion above it can be concluded that Sarsa(λ) is favorable for this project. It is designed to utilize the training data

(i.e. user inputs) well without going to the degree of creating simulated data. However, there might be a need to control its exploration behavior at some point during a training process to ensure convergence towards an optimal state. How this is handled in the solution designed in this project is presented in the Design chapter (Section 6.3.2, page 85). Furthermore, the Design chapter covers how the states and actions required for Sarsa(λ) are defined (Section 6.3.2, page 84).

5.1.3 Comprehending the state-space

The state-space for the reinforcement learning has yet to be designed, however, it is safe to assume that it must represent the hearing aid configuration parameters: Bass, midrange, treble, speech focus, noise reduction, and wind noise reduction.

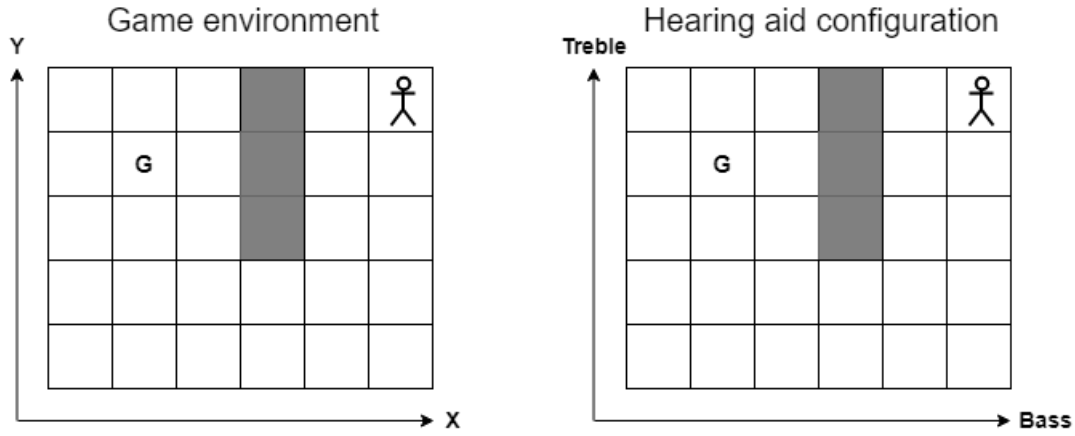


Figure 5.1: An illustration of how to translate the reinforcement learning game environment analogy to hearing aid configurations. Made with draw.io online tool.

Figure 5.1 depicts how the environments used for exemplification throughout the research on reinforcement learning (Section 4.3, page 33) translates to hearing aid configurations. The left of the figure illustrates that a user's (or a reinforcement learning agent's) location in a game environment can be described using a coordinate system. The optimal state can be described in a similar fashion. The agent can then change state in the search of the optimal state by changing location in the coordinate system. This can be used as an analogy for the hearing aid configurations too: The state-space can still be described as a coordinate system, however, with different dimensions. On the right of Figure 5.1, the stickman represents the current configuration state for the user's hearing aids. A state is no longer described by a pair of X and Y coordinates, but by bass and treble instead. Hence, the stickman represents a bass value of 6 and a treble value of 5 with the optimal state in this example being a bass value of 2 and a treble value of 4.

From this it can be seen that the game environment is a reasonable analogy for the hearing aid configurations. However, this project is working with six parameters, and not just bass and treble. As a result, the state-space becomes six-dimensional. Furthermore, in order to understand how the user preferences link to a specific sound environment, the state-space must also contain environment information which increases the dimensionality even more. Six dimensions might not be a problem in

itself, however, it quickly scales the number of possible states making it more difficult for the stickman in Figure 5.1 to find the optimal state.

Feature	**Possible states**
Bass	A gain from -6 dB to +6 dB = 13 states
Midrange	A gain from -6 dB to +6 dB = 13 states
Treble	A gain from -6 dB to +6 dB = 13 states
Speech focus	Narrow, medium, wide = 3 states
Noise reduction	Off, mild, moderate, considerable, strong = 5 states
Wind noise reduction	Off, mild, medium, strong = 4 states

Table 5.1: The possible states for each of the dimensions (parameters) considered in this project.

Table 5.1 presents the six parameters (dimensions) this project considers. Individually, the number of states seem low and manageable. However, assuming that all six parameters are available in any sound environment, each environment has $13 * 13 * 13 * 3 * 5 * 4 = 131820$ possible states.

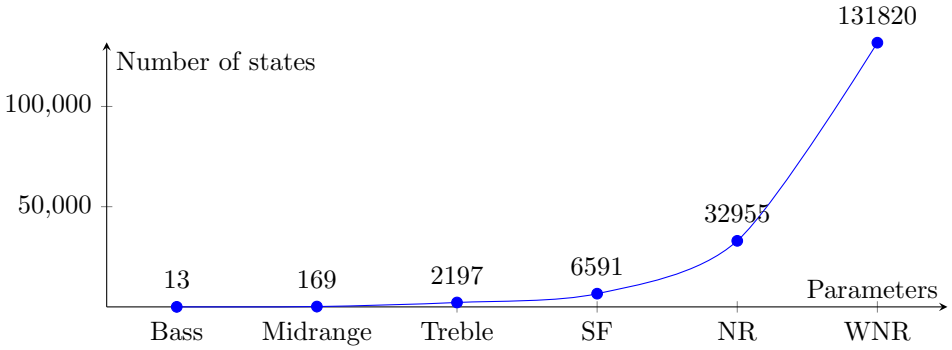


Figure 5.2: Illustration of the state-space growth as more dimensions (parameters) are considered.

This rapid growth in state-space as a result of increasing the number of parameters to learn, is depicted in Figure 5.2. It can be seen that even though bass, midrange, and treble are the parameters with most possible states, considering only those three parameters limits the space to 2197 states. Of course, a similar calculation can be made if only considering Speech focus (SF), Noise reduction (NR), and Wind noise reduction (WNR) which leaves only 60 states to be explored. This shows that adding dimensions scales very poorly and it should be considered if it is feasible to find the optimal state among 131820 possibilities based on user input.

User context and intent

The size of the state-space could be limited by considering the current context and what the user's intent is. Specific hearing aid adjustments might not be applicable in certain environments, and a user's intent might indicate what sort of adjustment the user is likely to prefer.

Parameter	Speech clarity functionality	Noise filter functionality
Bass	-2 dB	-5 dB
Midrange	+4 dB	-5 dB
Treble	+5 dB	-5 dB
Noise reduction	Strong	Strong
Wind noise reduction	Strong	Strong

Table 5.2: Comparing Speech clarity and Noise filter functionality from the Outdoor listening program in the ReSound Smart 3D application. These two features are predefined and apply the listed adjustments to make it easy for the user to achieve a certain goal. These predefined features are known as "Smart Buttons" which is elaborated in Section 5.2.1 (page 57).

As an example, if the user of the ReSound Smart 3D application has selected the Outdoor listening program and enables a predefined feature named "Speech clarity", the application applies the hearing aid adjustments presented in the second column of Table 5.2. If the user enables "Noise filter", the adjustments listed in the third column of Table 5.2 are applied. It can be seen that there are some tendencies: Both scenarios require significant noise reduction with a negative gain on low frequencies, but for Speech clarity the gain for midrange and treble is increased while it is decreased for Noise filter. Furthermore, it can be found that the Speech focus parameter is not available in the Outdoor environment. Hence, the state-space can be limited by disregarding the Speech focus parameter when the user is in an Outdoor environment.

An example provided by Brian Dam Pedersen, the CTO of GN Hearing, outlines the importance of capturing the user intent when fine-tuning hearing aid parameters: Consider going to a jazz venue with your wife to listen to the music. Then consider going to a jazz venue with your wife but with the intention of enjoy the evening and have a chat. The context is the same, but the intent is very different. In the first scenario, the hearing aids should be adjusted for listening to music where as in the second case, they should be adjusted for conversation and have music as a secondary.

Similarly, Korzepa et al. [55] argue that it is necessary to capture the intention of the user in order to ensure optimal hearing aid configuration. Correctly classifying the environment is not sufficient [115]. However, a hearing aid is no longer just a hearing aid, but part of a bigger solution. By utilizing the smartphone, more information about the users and their context is available [55]. Yet, it is still difficult to extract the intention as it requires explicit feedback from the user. Instead, Korzepa et al. [55] propose a guideline of contextual clues that can help infer the user's intention. These clues include how the user is moving, what time it is, and what acoustic scene the user is in (e.g. party, supermarket, or forest) [55]. Some of the information can be extracted from the hearing aids themselves and other information can be provided by sensors in the smartphones as well as location and motion services [55]. However, this still does not provide a solution to the jazz venue example presented earlier. To solve such a case, Korzepa et al. [55] suggest making a solution that is "voice-aware" and able to recognize familiar voices. The research followed a number of hearing aid users and was able to conclude that being able to predict the intent depends on the user and if they have preferences which can be explain by the context [55]. In some cases, such as lunch in a noisy company canteen, it would be possible to infer an intent based on the acoustic scene, location, and time. However, in other cases the user preferences might be unexplainable

or contain only weak patterns [55]. More thoughts on user intent is provided in Chapter 9 (page 140).

Among the authors of that research [55] are two representatives from Eriksholm Research Centre which is a part of Oticon [28]. Michael Kai Petersen, Senior Scientist Augmented Hearing at Eriksholm Research Centre, is one of these two representatives, and he also took part in a student project exposition event by Neural on the 2nd of April 2019 to present current work and make people interested in doing projects with Eriksholm Research Centre.

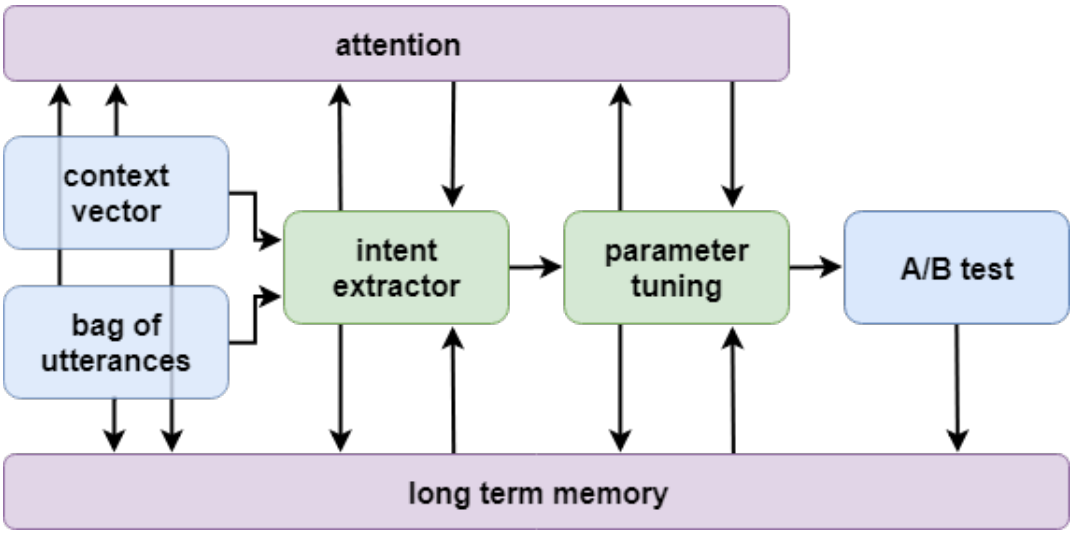


Figure 5.3: Determining user preferences based on environment and speech input. A project by Eriksholm Research Centre. The figure is recreated based on a slideshow from the stand of Eriksholm Research Centre at the Neural event.

Figure 5.3 depicts a project Eriksholm Research Centre is currently working on in order to determine what hearing aid adjustments a user would like in a specific environment. The details on the project were a bit vague and no publications of the research were to be found. However, from Figure 5.3 it can be seen that the system tries to estimate the intent of a user based on context information and speech input. The inferred intent is then used to suggest new hearing aid adjustments which can then be presented to the user as an A/B test. According to Michael Kai Petersen, the context is defined by a classification performed by the hearing aid (using information such as the noise floor, signal-to-noise ratio, and sound pressure level) and motion information from a smartphone. He also emphasized the need for intelligent solutions as these can allow a change in focus for the future audiologist. Furthermore, a solution that tries to adjust to the user, can convince people, who need a hearing aid, that they can benefit from it, according to Michael Kai Petersen.

Being able to infer as user's intent seems like a complicated and error prone task based on the research by Korzepa et al. [55]. However, inspiration could be taken from the SoundSense Learn solution available in the Widex Evoke application. SoundSense Learn allows a user to perform A/B testing of multiple different sound profiles such that the system can learn the user's preferences in the current environment. The output of a training session makes it possible for the user to apply the settings temporarily to the current listening program or to save the settings as a separate listening program such that the user can

apply the adjustments again on-demand at a later time.

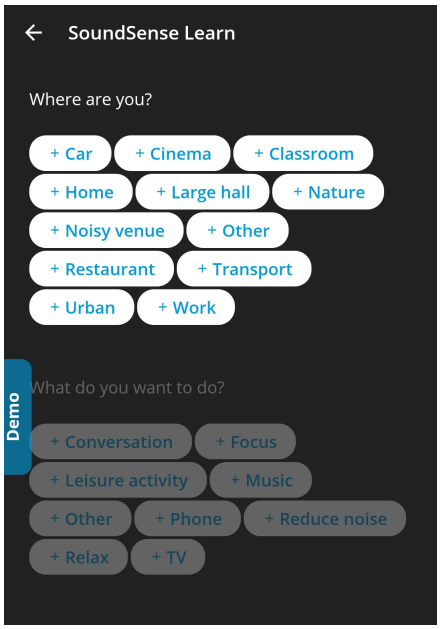


Figure 5.4: Widex SoundSense Learn screen prior to training. The user is asked to input what environment they are in and what intent they have.

Figure 5.4 shows the initial screen a user of the SoundSense Learn feature is met with. The screen asks the user to pick what sort of environment they are in (the white buttons at the top) followed by what intent they have (the darker greyed out buttons at the bottom). This solution was presented by Anders Trier Poulsen, Machine Learning Specialist at Widex, at the DSE career fair on the 10^{th} of April 2019. According to him, the learning algorithm can benefit from knowing this information.

It can be concluded that researchers and hearing care companies emphasize the need for knowing both context and intent. They are aware that it impacts how the hearing aid configurations should be adjusted. Hence, user context and intent are also important in this project and should, therefore, be taken into account as well. However, inferring the intent might be too unreliable, complicated, and require a study on its own. Therefore, the solution developed in this project should find a different way of dealing with user intent and possibly draw inspiration from the SoundSense Learn solution.

Determining when optimum has been reached

During a training session, the solution should converge towards a set of hearing aid adjustments which are optimal. When this optimal point has been found, the training session can terminate. However, this requires the solution to determine when it has reached this point. Using the word "converge" carries some of the explanation of what to look for: A definitive limit or a meeting point. Such a point would be where all the hearing aid parameters meet at maximum user satisfaction.

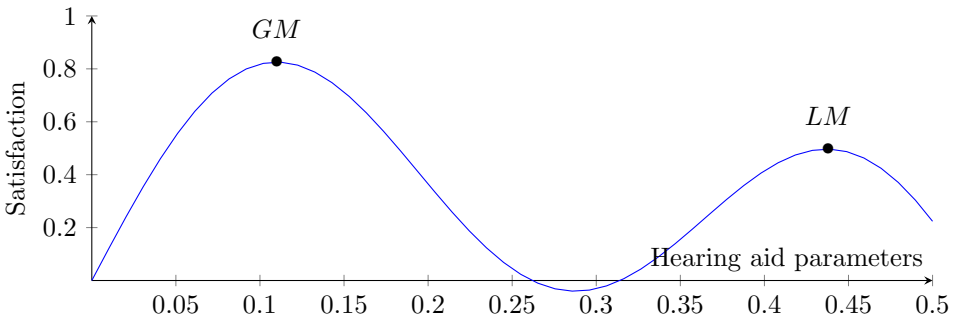


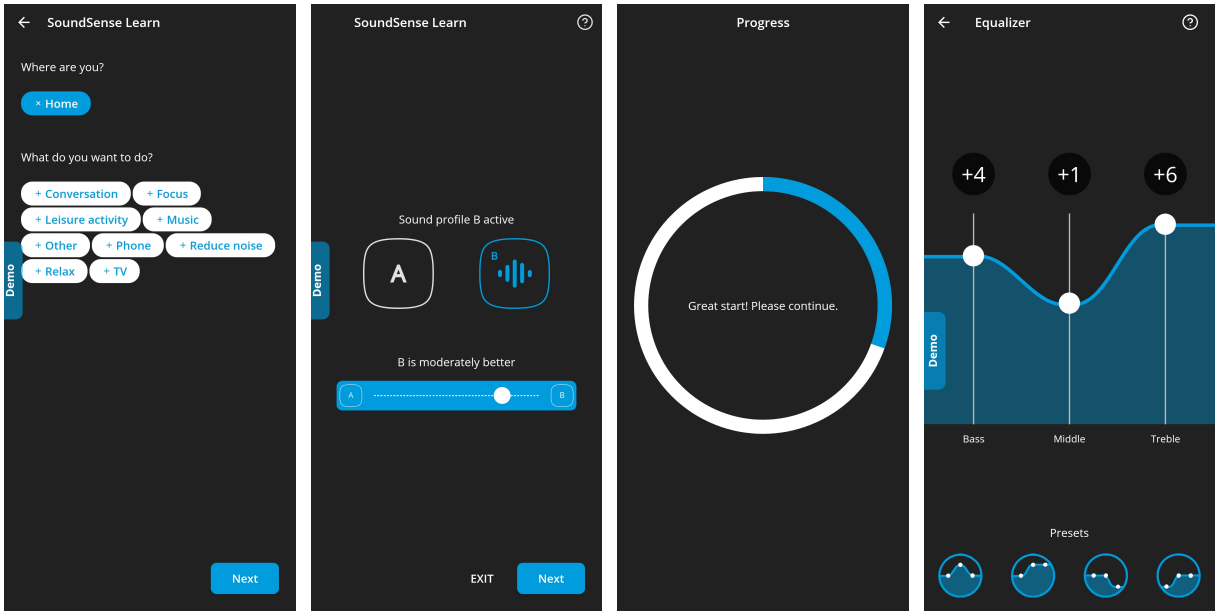
Figure 5.5: Illustration of the global and local maximization problem.

A meeting point like this is illustrated by point *GM* in Figure 5.5. The figure shows hearing aid parameters aggregated into one dimension on the horizontal axis and satisfaction on the vertical axis. This makes it possible to illustrate the problem but also simplifies it greatly as more dimensions are needed for the actual solution. Visually finding point *GM* is not difficult. It can be described as the point with highest satisfaction value. If the hearing aid parameters are adjusted right or left, the satisfaction will decrease making it easy to detect that an optimal point has been found. However, consider point *LM* which can be described in a similar fashion: If hearing aid parameters are adjusted either left or right, the satisfaction value will decrease. However, that does not make *LM* to an optimal solution. Hence, this way of defining optimal only promises as local optimization and not a global optimization. In the case of Figure 5.5, checking the full range might be feasible as it is rather limited. However, as the system scales to more dimensions and larger ranges, more points have to be tested in order to ensure a global optimization which might not be feasible.

This problem is not unique to this project, Anders Trier Poulsen gave a similar explanation to what Widex is trying to achieve with their solution: Adjust the hearing aid parameters such that one of these optimal points are found. He did not state it as being a necessity to find the global maximum satisfaction. The SoundSense Learn solution is a commercial implementation of a Ph.D. thesis written by Jens Brehm Nielsen [66]. The thesis emphasizes that global optimization is more important than good generalization. Having a model that is good at predicting global optimal solutions has a higher value than a general solution as the general model would try to describe other regions too. However, these other regions are not of interest as they provide suboptimal hearing aid adjustments [66]. With that said, Nielsen [66] also points out the computational complexity linked with finding the global maximum. Therefore, he employs a method that assesses the gradients like exemplified with the local and global maximum in Figure 5.5. As mentioned, this does not promise that the global maximum is found. However, performing the same gradient assessment with different starting points ensures that multiple local maximums can be considered when trying to find the global one [66].

The work presented by Nielsen [66] uses Gaussian Process Regression (which has not been considered in this project) to find the optimal hearing aid configuration. The work claims that between 10 and 20 iterations are needed in order to determine the optimal configuration. This project shares, apart from the global maximization problem, a dimensionality problem with Nielsen's research [66]. Figure 5.2 (page 44) showed the rapid growth in state-space as dimensions are added. Nielsen [66] reports that scalability issues *"occur somewhat above four dimension"* [66]. Once again, the methods applied are different and

the dimensions might not be comparable, but a scaling issue has been noticed. To get input on how the SoundSense Learn solution functions, and if it is able to find an optimal hearing aid configuration in 10 to 20 iterations, an experiment was conducted.



(a) (b) (c) (d)

Figure 5.6: Test run of SoundSense Learn in the Widex Evoke application. Context is set to be a "Noisy venue" and the intent is set to "Conversation" (a). Multiple sound profiles are compared and rated (b) to determine user preference. During training, the user gets feedback in the progress (c). Upon training session termination, the preference prediction is applied to the Equalizer (d).

A set of Widex Evoke Fusion 2 hearing aids were made available to the author of this project for a short period of time at the DSE career fair. The environment was an open area with lots of people having conversations making it very noisy. The "Noisy venue" environment was selected with the intent of having a conversation as shown in Figure 5.6a. This initiates the training session as depicted in Figure 5.6b where the user is asked multiple times to compare two sound profiles and rate the better one. The user can pick from "Neither is better", "X is slightly better", "X is moderately better", and "X is much better" where X can be either profile A or B. As shown in Figure 5.6c, during the training session, the user is made aware of the progress indicating how much more SoundSense Learn has to learn in order to suggest a new hearing aid configuration. The user can at any time terminate the session, and the best preference guess at that time is applied to the hearing aids. Furthermore, the adjustments are reflected in the Widex Evoke Equalizer where Figure 5.6d shows that the preferences of the author of this project in a noisy venue with the intent of having a conversation is a gain of +4 dB for bass, +1 dB for midrange, and +6 dB for treble. However, it should be mentioned that the author of this project is not hearing impaired. Furthermore, according to Anders Trier Poulsen, the hearing aids were not fitted to any particular person. Instead they had a flat gain curve adding slight amplification across the entire frequency range.

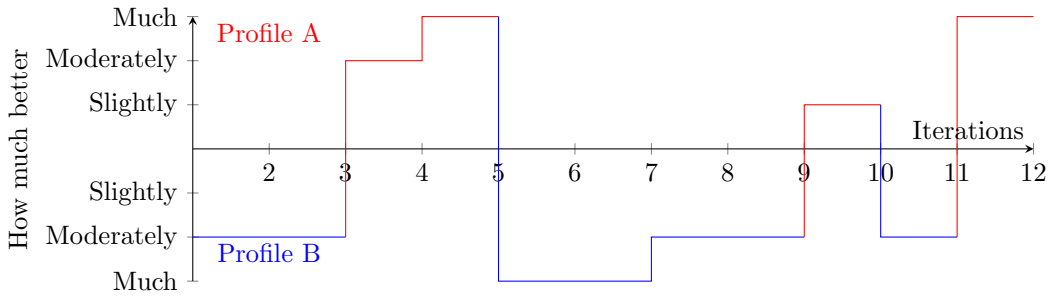


Figure 5.7: Sound profile rating at each iteration during the SoundSense Learn test in the "Noisy venue" context with a "Conversation" intent.

Figure 5.7 illustrates the changes in sound profile preferences over the course of a training session initiated as depicted in Figure 5.6a. It can be concluded that the training session finished in 12 iterations (which is within the range of 10 to 20 iterations stated by Nielsen [66]) and that some major preferences changes were made during the training. As an example, at iteration four, sound profile A was determined to be much better than the alternative, but at iteration five, sound profile B was rated as much better. Appendix C presents more sound profile rating plots like Figure 5.7 (one for context: "Noisy venue" with intent: "Conversation" and "Reduce noise", and one for context: "Nature" with intent: "Relax") and similar behavior is found. The learning algorithm finishes in 12 iterations and the preferences varies significantly. Anders Trier Poulsen explained that the implementation requires a minimum of 12 iterations before SoundSense Learn can claim to have determined the user's preference, however, it can continue with more iterations if needed. All the training sessions were done in the same environment and were based on the "Universal" listening program from the Widex Evoke application as a starting point. As mentioned, the author of the project is not hearing impaired, however, tried to achieve a bright and "open" sound during all the training session to ensure consistency.

Context	**Intent**	**Adjustments**
Noisy venue	Conversation	Bass: +4 dB Midrange: +1 dB Treble: +6 dB
Noisy venue	Conversation, reduce noise	Bass: +4 dB Midrange: -2 dB Treble: +6 dB
Nature	Relax	Bass: -5 dB Midrange: +2 dB Treble: +2 dB

Table 5.3: Hearing aid configuration adjustments made upon finishing SoundSense Learn training session with various contexts and intents.

Table 5.3 lists the adjustments applied to the hearing aids after each of the SoundSense Learn training sessions. All three tests where carried out in the same sound environment, but the context and intent

provided to SoundSense Learn varied. It can be found that the goal of reaching a bright and "open" sound produced similar adjustments when the context and intents were kept similar. However, if the context and intent was changed, it also affected the resulting adjustments made to the hearing aids. This can indicate that Widex might be limiting or biasing the algorithm to search among specific adjustments. However, no conclusion can be made in relation to this as test data is sparse.

This, however, does not outline how the SoundSense Learn approach decides that an optimum has been reached. Moreover, the gradient assessment method described earlier does not necessarily provide an accurate representation of the optimal hearing aid settings. The method will find a current maximum (possibly a global one), however, new training input from a user can possibly shift these maximums. In fact, Nielsen [66] does not state that the gradient assessment method is used to determine the optimal hearing aid adjustment. Instead, the method is utilized to determine what data point needs to be examined in order to provide the most useful input to the algorithm [66]. This is due to the approach taken by Widex (Gaussian Process Regression) and does not apply to the reinforcement learning techniques discussed in this report as a state selection mechanism is already in place.

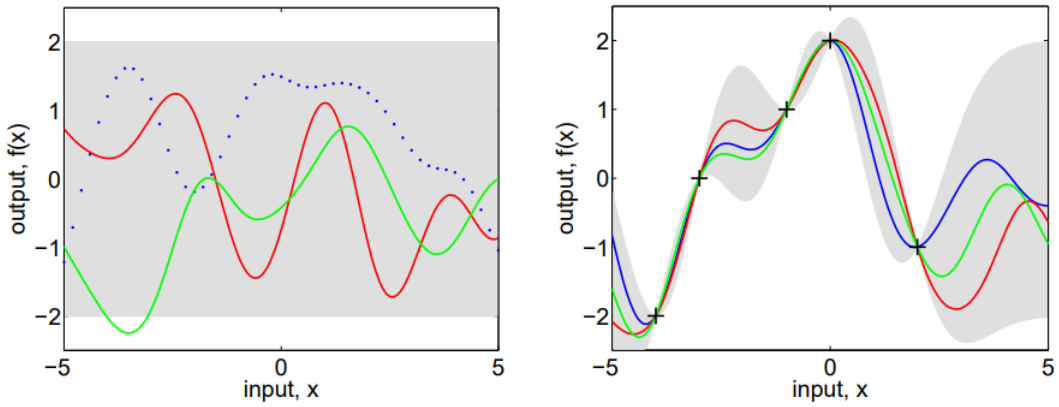


Figure 5.8: Example of Gaussian Process Regression. New data limits the possible functions. Source: [78].

The Gaussian Process Regression approach is illustrated in Figure 5.8. On the left, no data has been collected yet, and all sorts of functions can be used for prediction. Furthermore, the grey area shows the uncertainty which is huge before data is collected. On the right of Figure 5.8, the crosses illustrate collected data points. As not all functions can be used to describe these data points, the function-space is reduced, and the uncertainty is less [78]. The right of Figure 5.8 still shows high uncertainty for x = 5. This might be the data point that will provide the most valuable input resulting in the greatest reduction in uncertainty. Finding these points is how Nielsen's [66] work is utilizing the gradient assessment method. It can be speculated that SoundSense Learn decides that an optimal solution has been found and training can be terminated whenever the uncertain is below a certain threshold. However, as the reinforcement learning approaches does not model uncertainty, it is difficult to apply this thinking to this project.

Even though that exact uncertainty thinking does not apply to this project, statistics could be utilized. Assume that the reinforcement learning is able to determine what states a user does not like by visiting the particular states only one time. Furthermore, assume that to determine the optimal state, the solution

has to circle around the optimal state presenting the user with close-to-optimal states. As a result, the optimal state and the close-to-optimal states around it are visited more. Many states will be visited once, and a few states will be visited multiple time. Counting state visits makes it possible to calculate the average visits for any state as well as the variance. This information can be used to calculate a z-score which is a measure that indicates how far an observation is from the mean [3]. However, the z-scores can be misleading for small sample sizes [67] which is the aim of the solution (it should learn the user's preferences based on as few interactions as possible). Moreover, an outlier test using the z-score assumes that the data is normally distributed [3] (i.e. has a bell shape) meaning few states are visited only few time, most states are visited a number of times, and few states are visited a lot of times. This assumption might not hold true for a training session with the solution being developed in this project. Alternatively, the interquartile range can be used as a measure of when an observation is an outlier [3]. If an observation is more than one and a half times above the upper quartile or below the lower quartile, it can be classified as an outlier [3]. Imagine having a set of observations (a list of states that have been visited) ordered by value (how many times each state has been visited), split the observations into two data sets at the median, and then find the median of the two data sets. The median of the data set containing the low values is the lower quartile, and the median of the data set with the high values is the upper quartile. The interquartile range defines the distance between the upper and lower quartiles [3].

The idea of keeping track of state count and estimating outliers, however, might not be a robust approach. Firstly, the user should not have to visit very many states before reaching an optimal state. Few interactions with the user (which is desirable) leads to few observations to base the statistics on (which is undesirable). Secondly, a high visit count for a few states might be caused if a user, by accident, indicates a preference for those states. The reinforcement learning solution will think that the user is about to reach an optimal state while, in truth, the user is trying to correct the accidental preference indication.

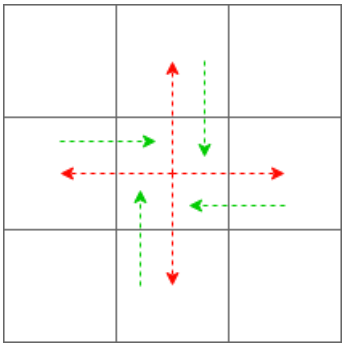


Figure 5.9: Illustration of state-action values around a possibly optimal state. Made with draw.io online tool.

Instead of relying on statistics, each state can be evaluated by the state-action values of itself and its neighbors. Consider Figure 5.9 depicting nine states and the value of the state-actions concerning the center state. The red arrows indicate that a negative reward is given to an agent making such a transition, and green arrows indicate a positive reward associated with the transition. It can be seen in Figure 5.9 that an agent leaving the center state would receive a negative reward no matter what action

is taken but receives a reward for any of the actions leading into the center state. This idea can be used to assess the entire state-space giving an indication of areas that are potential candidates for optimal states. Based on this, it could be evaluated if the user is currently in such a state. If that is the case, the training session can be terminated. Outlining the entire state-space in this manner might not be computationally viable so focusing on a subset of the state-space where the user is currently located might be favorable. However, focusing on a subset will not guarantee global maximization.

Instead of focusing on state-action values, state value could be a measure to determine if an optimal state has been reached. The value of a state can be estimated as defined in Equation 5.1 [94].

$$V(S) \leftarrow V(S) + \alpha * (R + \gamma * V(S') - V(S)) \tag{5.1}$$

$V(S)$ defines the state-value estimate of state S which can be updated using the current estimate ($V(S)$), the value estimate of the following state ($V(S')$), the reward received by transitioning to the next state (R) as well as a discounting factor (γ) and a step size (α). This can be performed as an iterative approach giving all states a value of zero initially [94]. The reward is determined based on user input and the remaining parameters are implementation details. This would make it possible to evaluate the value of a state given a specific policy (derived from user input) and the user reward input. This approach is known as Temporal-Difference Prediction [94] and is similar to the approach presented in Figure 5.9, but instead of evaluating individual state-action values it focuses on the rewards received upon transitioning between states by following a policy. However, since this approach follows a policy, it might not get global coverage. Furthermore, there is a trade-off between exploration and exploitation. Not allowing for exploration would resolve in greedy optimization and find the nearest optimum just like the problem exemplified with Figure 5.5 (page 48). Allowing for exploration makes it possible to determine that more than one highly valued state exists. All the highly valued states can then be taken into account before evaluating which one is the best and whether the user is in that state currently. If yes, the training session can be terminated. However, in order to achieve this, a high exploration factor might be needed making state value evaluation slow due to the high number of states. Furthermore, it just moves the issue: The initial problem was to decide when to terminate the training session. Now, it is how to determine that a sufficient amount of states have been explored. Alternatively, less exploration combined with multiple evaluation runs could solve this problem. An evaluation has to start in some state. Starting multiple evaluation runs each from different states makes it possible to discover multiple local optimal states while keeping exploration. An evaluation run would then simply terminate when a state with high value has been found and the states around it all have lower values. A highly valued states is a local optimal state, and this can be compared with the outcome of the other evaluation runs.

Possibly a combination of the above suggestions can be used, or the users can simply be asked to terminate the session themselves whenever they are happy with the hearing aid configuration as seen in a study concerned with self-fitting by Mackersie et al. [60]. However, asking the user to determine this increases the effort a user needs to put into a training session. Nielsen [66] puts emphasis on developing an intuitive system and to not impose a high load on the user. Making the user terminate a training session by clicking a button might not seem like it can impose an additional load on the user. However, the care seeker who is not comfortable with new technology (see Section 5.2.2, page 59), might not be comfortable making that decision either.

This section has been focused on finding a global optimization, however, it can be argued whether

this is even necessary. As indicated in Figure 5.2 (page 44), the entire state-space is large even when only adjusting a few parameters. It is not feasible to develop a solution where the user has to visit all states to find the optimal state. For this reason, it does not make sense to search globally for an optimal state either. The information is not available and, hence, "global" can be narrowed down to the states the user has visited and rewarded. This makes it more feasible to consider all the local optimums that might be created in the state-space. Combining this with the information about the user's current state location, and whether a specific local optimum keeps being emphasized, can be used to evaluate if the user's preferences has been determined. If this is the case, the training session can be successfully terminated. The exact implementation of how this should be achieved is covered in the Design chapter (Section 6.3.2, page 88). The above discussion has provided a range of ideas which can be used to achieve this. Using Temporal-Difference Prediction with low exploration initiated from different states seems like an interesting approach.

Describing the environment

As discussed throughout the section "User context and intent", the intent and the context of the user is important. However, this project has not yet covered exactly how the context should be presented to the system. It has been seen that Eriksholm Research Centre is trying to collect this information from the smartphone (Figure 5.3, Section 5.1.3, page 46) and Widex is asking the user to provide the information (Figure 5.4, Section 5.1.3, page 47). Widex's solution seems very simple as the user has to decide from a predefined list where Eriksholm Research Centre's approach is more complicated combining various inputs to describe the context. Based on discussions with Ditlev Munk Raboel, DSP Software Developer at GN Hearing, Anders Trier Poulsen, Machine Learning Specialist at Widex, and Michael Kai Petersen, Senior Scientist Augmented Hearing at Eriksholm Research Centre, it is clear that hearing aids from GN Hearing as well as Widex and Oticon all have internal environmental classifiers. From discussion with Ditlev Munk Raboel and Anders Trier Poulsen, it can be concluded that these environmental classifiers are rather simple and limited. Much more advanced environment classification can be made, however, a hearing aid does not have the power to do so. Michael Kai Petersen did not emphasize the simplicity of Oticon's environmental classifier. However, he did mention similar basic features used to determine the environment such as signal-to-noise ratio, noise floor, and sound pressure level. Even though the environment classification is simple, if the classifier is reliable, it could be used to determine the context of the user. However, accessing the environmental classifier in the hearing aid puts requirements on the hearing aid: It must be able to communicate this information.

To not rely on the hearing aid, neural networks can be used to classify environments based on audio [23]. Section 5.1.1 discusses the learning architectures for learning user preferences using neural networks. It might be beneficial to combine these with the reinforcement learning architectures discussed in Section 5.1.2. Based on what is presented in this project, the reinforcement learning methods are simpler than the neural networks. The idea is that due to the simplicity, the reinforcement learning approaches can be used to interact with the user and quickly gain and understanding of the user's preferences. However, a disadvantage might be their lack of ability to contain complex preference information. Furthermore, they do not know what environment the user is currently in. By using a neural network (more specifically a Convolutional Neural Network (CNN)), the environment can be defined by the audio around the user [23]. The audio is given as an input to the CNN and the output is an environment classification. This

project has discussed neural networks for regression, not classification, but learnings from the prestudy [23] make it possible to design a classification network. Furthermore, a network that combines the classification and regression problem might be beneficial. A branched network could be an interesting approach where one branch is responsible for predicting the user's hearing aid settings for the specific environment (regression), and the other branch is responsible for determining what environment it is (classification). This information can feed into the reinforcement learning which now has information about what environment the user is in as well as a starting point in the state-space (the hearing aid adjustments predicted by the CNN). In this case, the CNN is used to understand what the user generally prefers in a specific environment. The reinforcement learning method is then used to perfect the suggestion by user interaction. Ideally, this approach provides shorter training sessions with the user because an initial suggestion has already been made and the environment is taken into account.

A combined solution like suggested above does create a more complex overall architecture. Furthermore, introducing an environment dependent state-space is similar to adding an additional dimension. Despite the disadvantages, the solution seems advantageous and just has to be designed such that the complexity is not apparent to the user.

5.1.4 Concluding remarks on the technical analysis

This section warps up the technical analysis by extracting the conclusions throughout the analysis as many aspects have been discussed. The analysis found that the solution being developed in this project possibly can benefit from combining neural networks and reinforcement learning. Throughout the research they were presented as separate approaches which could replace each other. However, they both have disadvantages and advantages. By combining them, the two approaches can complement each other by diminishing the disadvantages while keeping the advantages. As an example, the advantage of the neural network is the generalization power, but it is complex and likely not something a user can interact with in real-time. However, the reinforcement learning is limited in terms of complex generalization but is sufficiently simple to run on a smartphone and interact with a user in real-time. The analysis found that multi-task neural networks would fit best for the regression task in this project, however, with limited branching. Furthermore, the prestudy [23] has shown Convolutional Neural Network's ability to classify environments. This can be combined such that a multi-task Convolutional Neural Network can be used for environment classification and initial general hearing aid adjustment prediction. This requires at least two branches for the multi-task network, but both the regression task and the classification task are likely to benefit from the same feature extraction. This information is then fed into Sarsa(λ) which has been chosen as the reinforcement learning approach due to its ability to utilize sparse training data while still being able to adjust to a changing environment.

The analysis also revealed how adding dimensions (hearing aid parameters to adjust) scales the state-space. Increasing the state-space is undesired as more states might have to be assessed before finding an optimal one. To limit the number of states to consider, the user intents can be taken into account. This can give an indication of what direction the learning algorithm should search. Intent will not be assessed as its own dimension but can still indicate a trajectory in the state-space. Furthermore, it is possible that the algorithm can intelligently skip similar states. As was pointed out by Ditlev Munk Raboel, DSP Software Developer at GN Hearing, as a rule of thumb, a change of 3 dB is noticeable to

a human which is supported by other sources too [40]. The learning algorithm should present the user with noticeable differences. This improves two aspects: Firstly, it limits the number of states that needs to be assessed leading to shorter training session. Secondly, making more perceivable changes might also provide a better experience and increase the user's patience with the system.

By examining the Widex SoundSense Learn solution combined with a discussion with Anders Trier Poulsen, Machine Learning Specialist at Widex, made it apparent that the Widex solution only adjusts bass, midrange, and treble. Developing a prototype for this project with limited scope initially might be a good idea. This will yield some learnings, and these can be used to expand the prototype and include speech focus, noise reduction, and wind noise reduction. Furthermore, focusing on bass, midrange, and treble might also be more useful for the user. Statistics provided by GN Hearing show these three parameters are more frequently adjusted compared to speech focus, noise reduction, and wind noise reduction. This information might not necessarily indicate preferences towards those three parameters though. Not all users have listening programs that allow for adjustment of the three latter mentioned parameters which might bias the data. However, as not all uses can adjust the latter three parameters this emphasizes why it is important to focus on bass, midrange, and treble first.

The interface for the user to interact with during a training session can also be inspired by the SoundSense Learn solution. Nielsen [66] points out that it is easier for people to perform pairwise comparisons. The user should not be asked to evaluate if adjustments are better at timestep T compared to at $T-1$. Instead, the user should be asked to assess multiple collections of adjustments (i.e. sound profiles) for each timestep. This should make the user find it more intuitive and perform more consistently [66]. A similar approach is indicated in Figure 5.3 (page 46) from a project by Eriksholm Research Centre. It can be concluded that the interface for the user should be some sort of A/B test similar to what is depicted in Figure 5.6b (page 49).

This section has briefly covered most of the conclusions throughout the analysis. This information feeds into the requirement specification (Section 5.4, page 72), and the Design chapter (Chapter 6) outlines how the system must be designed to incorporate all this. The remaining of the Analysis chapter presents the user and the risks associated with this project before going into the requirement specification.

5.2 Modeling the user

The following subsections present the type of users which the solution of this project is designed for. The users are modeled using personas and empathy maps, and scenarios are created to show the situations wherein the solution can be helpful.

The information used to model the user has been acquired through a meeting with Adam Heleniak who is a Senior UX Designer in GN Hearing's User Experience department. The meeting with Adam made it clear that more than one type of hearing aid user exists, and they persist different personality trades that makes it natural to separate them into different groups. Therefore, the following subsections models two types of users: The cautious and careful person, and the curious and experimenting person. Respectively, these are grouped as "care seeker" and "proactive". To ensure the modeling does represent an average hearing aid user, the following text has later been verified by Adam Heleniak.

5.2.1 Personas

This section presents two personas: A care seeker type named Alice and a proactive user named Bob. These two personas are carried over into the empathy maps (Section 5.2.2, page 58) and scenarios (Section 5.2.3, page 62) as well to create a complete understanding of the two types of user.

Alice - The care seeker

Alice is 70 years old, retired, and has two children, but now lives alone as her children have moved out and her husband has passed away. She is an experienced hearing aid user and have been using hearing aids for more than two years. She is the quite type who does not like radical changes, and it was a major change for her to start wearing hearing aids. As she lives alone, she did not really notice a deterioration of her hearing ability, but age has caused her to have a moderate hearing loss. Her children managed to convince her to get a pair of hearing aids even though it was something she was not familiar with and felt unsure about. Alice is now a happy hearing aid user, and her children are helpful and supportive too. THey bring her to the audiologist whenever needed. This provides comfort for Alice as the children will hear what the audiologist says. If she becomes unsure or forgets something, the children can help her with the hearing aid solution.

As Alice has been categories as a "care seeker", the audiologist at the local hearing center has put some extra effort into ensuring a good experience. The audiologist has provided Alice with just one listening program (the All-Around program) such that she does not have to worry about what hearing aid listening program should be used in a certain environment. Furthermore, as Alice has a compatible smartphone, the audiologist has instructed her in how to use the basic features of the ReSound Smart 3D application such that Alice knows how to adjust the volume and enhance speech clarity and filter noise using the Smart Buttons[1]. Alice is now comfortable using these features as they are easily available in the home view of the Smart 3D application, and she trusts the expertise of the audiologist. However, she has not explored other more advance functionality in the application as she is afraid, she might do something wrong.

Alice rarely uses the application as she spends a lot of time at home on her own. However, it does happen that she uses the Smart 3D application when she is with her friends. Alice is a member of a club for elderly people which arranges various events once a week. If an event is popular and a lot of people show up, she does use the smartphone application to improve the auditory experience. She is very pleased with the effect it has. Alice prefers a hearing aid solution that can help her in all environments while being invisible and not require any attention from her. The fact that she only has one listening program and is not required to do complicated adjustments does provide a certain level of simplicity. However, she feels awkward whenever she has to use the Smart 3D application when she is in an environment that requires her to enable a Smart Button to have a good experience. Alice would like the hearing aid to be like a pair of glasses. You put them on and they work everywhere without her having to think about them.

[1]Smart Buttons refer to one-click adjustments a hearing aid user can apply from the ReSound Smart 3D application. These adjustments are presented as buttons named according to what they enhance. These buttons are a quick and easy way to adjust bass, midrange, treble, speech focus, noise reduction, and wind noise reduction levels without the need to know how to adjust these individual features. For the Restaurant listening program, three Smart Buttons are available: "Noise filter", "Speech focus", and "Hear everyone" (See Figure 2.1, Section 2.1, page 5).

Bob - The proactive user

Bob is 60 years old and lives with his wife. Together they have two children who no longer live at home. Bob is a proactive type who likes gadgets and tries to keep up with technological advances and new products. Due to having worked in an industrial manufacturing plant close to noisy machinery for many years, he now has a mild hearing loss. Bob likes his job and has now been moved to a position where he is not working with the noisy machinery. Bob recently noticed that he was having trouble with his hearing and contacted a hearing center which provided him with a set of hearing aids.

He has been using the hearing aids for a few months and is getting used to them. It has been a major change for him to start wearing hearing aids. The audiologist categorized Bob as a curious and proactive person who can manage and benefit from advanced functionality. Hence, the audiologist explained to Bob that he could download the ReSound Smart 3D application to his smartphone and play with various settings. This made hearing aids more appealing to him as they become more like a gadget. The audiologist had enabled different listening program for him to play with including All-Around, Restaurant, and Outdoor.

Bob is a regular user of the Smart 3D application and finds it enjoyable to play with the settings until the sound is just right for the environment. He is still getting used to how the hearing aids and configurations work but does not limit himself to just listening programs and Smart Buttons. He tries to create custom listening programs and adjust individual settings including bass, midrange, and treble using the Sound Enhancer[2]. This customizability is very appealing and helpful to him as he finds himself in many different environments during the week. Work requires him to take part in meetings and discussions, at home he wants to reduce noise and relax, at dinner parties he must be able to comprehend many speakers at once etc. Therefore, he is an active user of the Smart 3D application and uses it daily.

If any problems or annoyances arise, he does his best to solve them by using the application. If he arranges an appointment with the audiologist, it is because he has a problem he cannot fix himself, even after much effort. In these cases, Bob is quite aware and capable of explaining what the issue is.

Even though Bob does like gadgets and mostly find the adjustability of the hearings aid appealing, some situations, such as important meetings, does not allow him to fiddle with his smartphone to improve the auditory experience. In these cases, Bob would like a solution capable of adjusting to the environment automatically based on settings he has made in a similar environment.

5.2.2 Empathy maps

Based on the personas of Alice and Bob, this subsection presents an empathy map for each persona. The empathy maps try to model what the user says, thinks, and feels as well as how they behave. The empathy maps extend on the users as described in the personas. The solution developed in this project has not been introduced yet. In this way, it becomes apparent what the user is missing and what the pains and goals are with the current solution. The current solution is assumed to be the ReSound Smart 3D application with compatible hearing aids.

[2]Sound Enhancer refers to a more advanced sound optimization features available to users of the ReSound Smart 3D application (see Figure 2.7, Section 2.6, page 15). This feature allows a user to adjust bass, midrange, and treble as well as speech focus, noise reduction, and wind noise reduction dependent on what listening program is selected. It enables the user to make finer adjustments to the auditory experience.

Alice - The care seeker

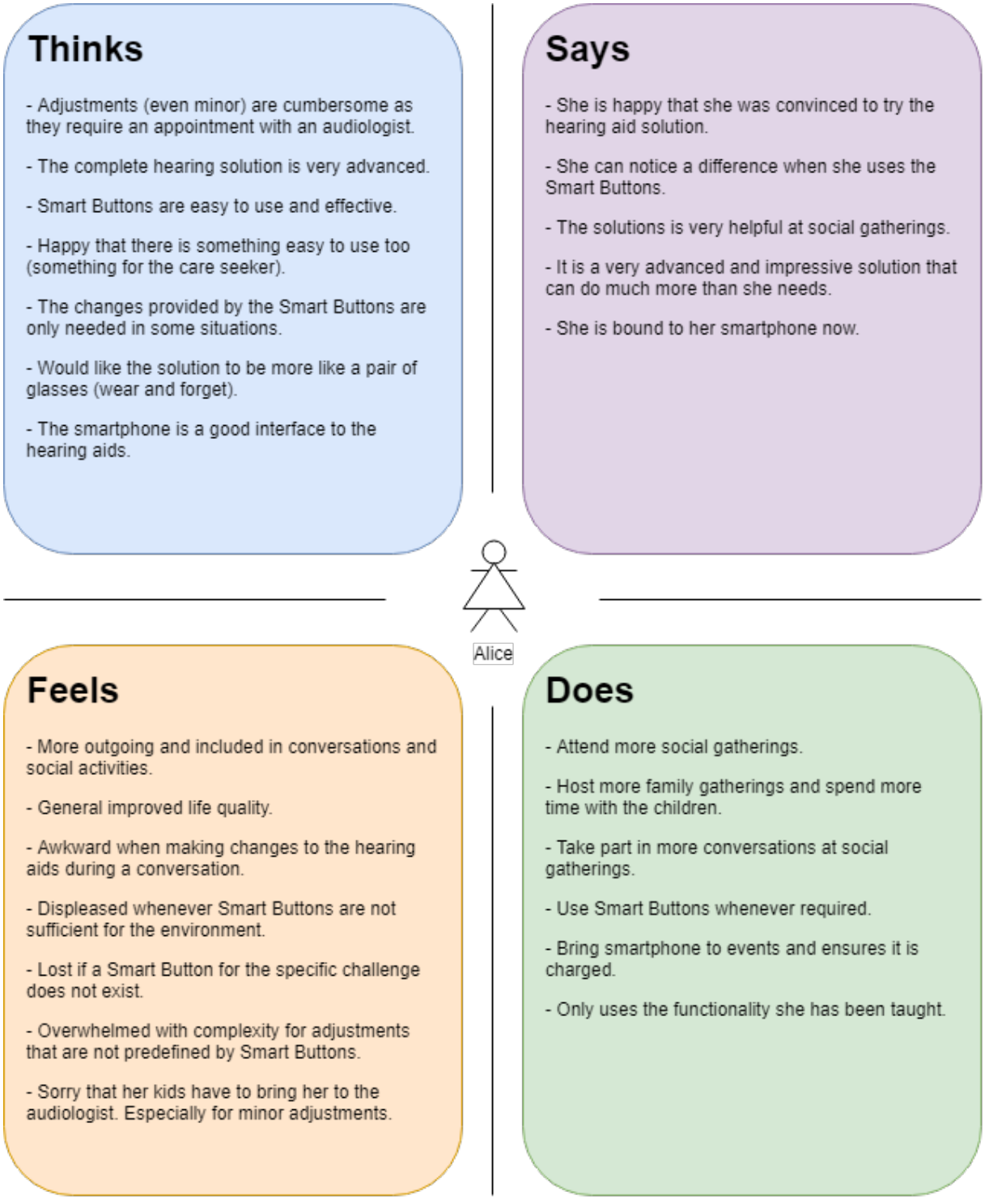


Figure 5.10: Empathy map of Alice the care seeker persona. Made with draw.io online tool.

From the empathy map in Figure 5.10 it can be concluded that Alice is happy with the current solution as it provides her with the configuration possibilities that she needs. On the other hand, the application has an overwhelming number of features that she does not use. Furthermore, she does not tend to explore these due to her care seeking nature even though they might provide her with a better experience in some situations. The empathy map lists that she feels lost if the Smart Buttons does not provide the good experience she needs. It is possible that the Sound Enhancer would allow her to find this good experience. However, she is not comfortable with such advance adjustments, and shy away from it. This is where the solution of this project could be beneficial to her. This solution would make it possible to abstract these advanced configuration possibilities into an easy to comprehend interface. Through this interface, Alice would follow a simple flow that would learn her preferences and then ensure an optimized experience. If the interface is easier to comprehend, it is more likely that Alice will try it out and succeed in finding optimal settings without using the advanced features of the Smart 3D application which she cannot comprehend. Moreover, it would automatically adjust whenever she is in a similar situation again. This could possibly also save trips to the audiologist if it is only related to minor adjustments.

Apart from outlining Alice, the empathy map has provided an idea of what needs to be done before she can benefit from a solution like the one developed in this project. It must be presented such that it is easy to comprehend and provide her with a feeling of assurance that she cannot do anything wrong.

This knowledge about the care seeker is utilized in the scenarios found in Section 5.2.3 and also feeds into the requirement specification in Section 5.4.

Bob - The proactive user

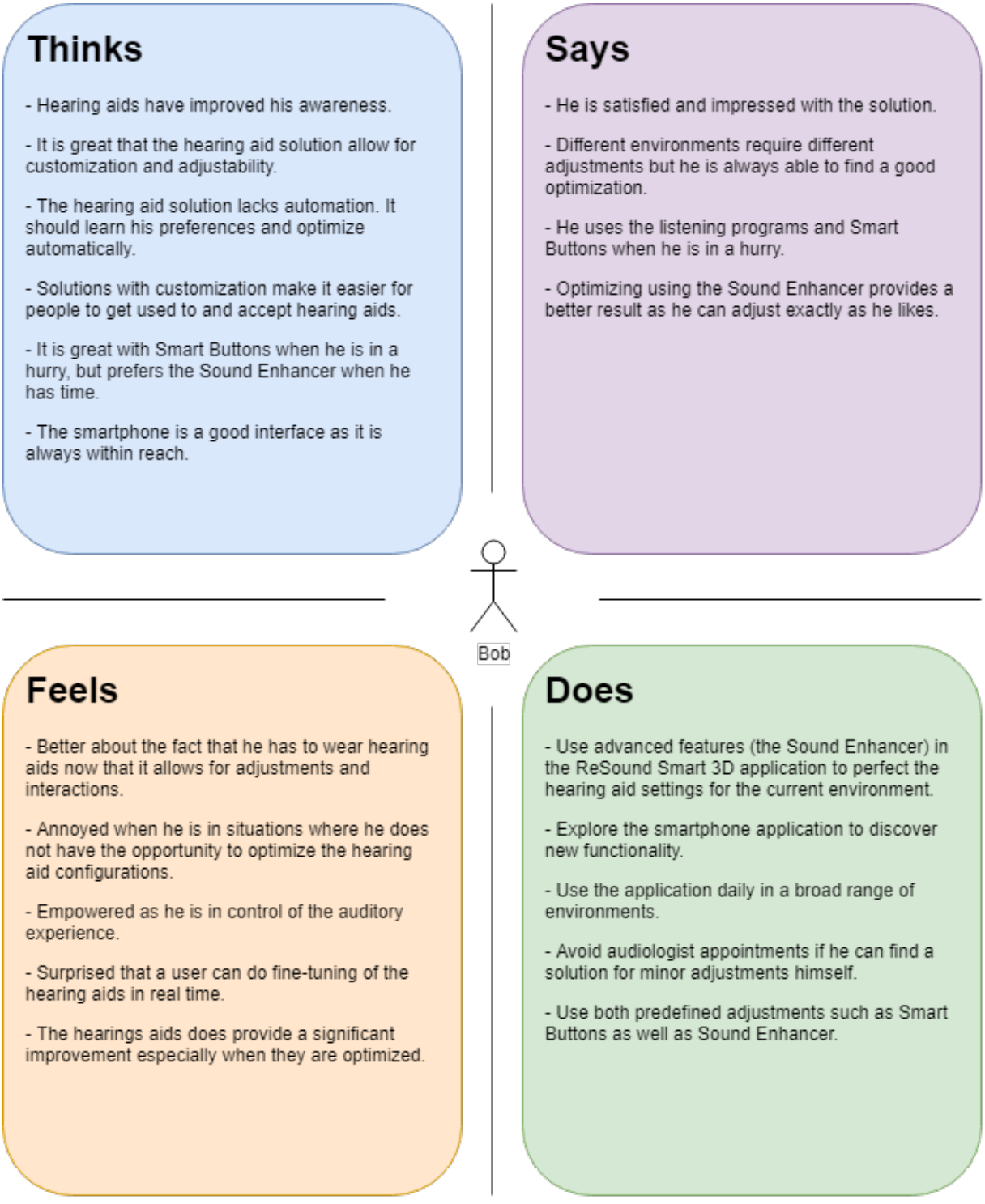


Figure 5.11: Empathy map of Bob the proactive persona. Made with draw.io online tool.

Figure 5.11 describes Bob who is the proactive user and explores the ReSound Smart 3D application more compared to Alice. He even utilizes the advanced features such as the Sound Enhancer when he has time for it, but also benefit from the Smart Buttons when he just needs a quick adjustment. However, he sometimes finds himself in situations where he does not have the opportunity to take out his smartphone and make adjustments. In these situations, he would like a sort of automation which could provide him with a good experience based on earlier input. However, he still likes to be empowered and wants the optimal auditory experience.

The solution developed in this project should provide him with both the learning aspect and the empowerment. As he is the type of person who is able to adjust settings in the Sound Enhancer, this could be used as a correction input. The system should implement a level of scrutability such that it can learn from user corrections. In this way, Bob can keep his empowerment and the system can provide better suggestions next time.

Bob can comprehend more advanced features compared to Alice, however, the solution must be created such that both users can use it. Hence, the advanced features should be hidden and not intimidating for Alice, but visible enough for Bob to realize they exist. Arguably, advanced features are a nice to have and not a need to have. This solution should not put effort into providing advanced features to satisfy Bob until the idea has been deemed feasible and viable. Yet, this knowledge about Bob is valuable. The knowledge about Bob and the proactive persona revealed from these empathy maps is utilized in the Scenarios found in Section 5.2.3 as well as in the requirement specification in Section 5.4.

5.2.3 Scenarios

The following subsections present a number of scenarios which the users might find themselves in. They are based on the personas, and the solution of this project is now introduced to the user. The scenarios communicate how it is supposed to work in the hands of the user and what complications can occur. Furthermore, these scenarios make requirements for the solution apparent. This is utilized to create the requirement specification presented in Section 5.4. The scenarios also make it possible to extract risks and assumption which this project is working under. These risk and assumptions are presented in Section 5.3.

First-time usage and onboarding

Alice's children have noticed that it is becoming increasingly challenging to have conversations with her, especially during family gatherings. They have suggested Alice to get her hearing tested and see if she might need hearing aids. Alice is not fond of this idea and does not think her hearing is that bad. However, the children have managed to convince her.

The results from the hearing test made the audiologist recommend Alice to get a set of hearing aids. The audiologist explained to Alice that the current solution on the market has clever algorithms that will make the transition to using hearing aids much easier as long as she has a smartphone. However, this is worrying Alice. She does have a smartphone but only uses it for calling and text messages. One of her children has shown her how to use the smartphone to read news articles so she does that too, but she is generally not very comfortable with this sort of technology. The audiologist prepares a set of hearing aids for Alice and shows her how to download an application that can communicate with the hearing aids.

Together they ensure that the hearing aids are connect, and the audiologist tells Alice that shortly a message will pop up on the screen asking her to go through an onboarding procedure. The message pops up while Alice is still with the audiologist who can then assist Alice if needed. The pop-up presents the "USense SoundLearner" functionality which the audiologist points out to be this clever algorithm that will make it easier for Alice to use the hearing aids. They will automatically adapt to her preferences. The pop-up describes USense SoundLearner as being a functionality that aims to learn the user's preferences in different environments. By learning the preferences, the USense SoundLearner promises optimal auditory experience in the environments it has be trained for. The audiologist suggests that Alice turns on this feature. Alice is then presented with a new message asking Alice if it should monitor the surroundings for her and suggest training session when it is not capable of confidently determine personalized settings. Alice decides that she would like this. Alternatively, she will have to decide when to start a training session, but she feels she might forget about the feature and is worried she might not know when she should train it. Hereafter, the algorithm needs to get to know Alice and she is asked what environments she normally finds herself in. She selects "Home" and "Social gatherings" which is an initial input to the algorithm to indicate what optimization might suit her and what intentions she often has. The setup is now finished, and Alice found it easier to interact with than expected.

While Alice is finishing up with the audiologist, a notification pops up on her smartphone. It says that USense SoundLearner has detected a new environment which it does not know Alice's preferences for. It asks her to start a training session. Alice opens the notification and is presented with the options of starting a training session or ignoring the request. Even though Alice is about to leave, she asks the audiologist if they can do this training session together. Alice wants to make sure she knows how to use it and is not doing something wrong. The audiologist knows that a training session is often very short and agrees to help Alice out. Alice starts the session and is initially presented with a screen listing several environments asking her to pick which one she is currently in. She determines that "Home" and "Office" are both not quite correct and decides to pick the more general "Indoor" environment. Hereafter, she is presented with a screen that makes it possible for her to vote for different sound profiles (i.e. hearing aid fine-tuning configurations). After a few iterations, the algorithm has learned what Alice's preferences are in this environment leaving Alice with a hearing aid configuration to her liking. Alice is now confident that she can go home and comfortably start using this hearing care solution.

Reinforcing and relying on prior learnings

Bob started using hearing aids just a few months ago and is still getting used to them. His hearing care solution includes the USense SoundLearner which he has been using from the very beginning. He most often finds himself at work or at home and have trained the USense SoundLearner to recognize and apply his preferences in meeting, canteen, and home environments. During meetings, Bob likes to hear everyone around the table, in the canteen he aims to filter all the background noises yet still be able to have a conversation with his colleagues, and at home he prefers a configuration that softens the sounds making it more comfortable after a long day at work.

Today is a regular day in the office for Bob, he has a few meetings to attend and some desk work to do. He is a busy man and is a bit late for an important meeting. He realizes that the meeting is in a different building and decides to take a shortcut and go through the manufacturing facilities. These facilities are noisy and not a place he usually comes anymore. He has not been in this environment since

he got his hearing aids, and it is now apparent to him how noisy it actually is. USense SoundLeaner detects that this is a new environment. It suggests Bob to start a training session by making a notification on his smartphone. Bob knows that these sessions are quick, but he does not have the time and clicks the "Ignore" button. This brings up a new dialog asking if Bob would like to be notified less often. Bob actually likes the frequency of the notification, but just does not have the time to act on this one. He swipes away this message and carries on to his meeting. The meeting environment is known to USense SoundLearner, and it automatically changes to Bob's preferences upon detecting this environment without having to bother Bob with notifications or training sessions.

In the evening, Bob decides to go out for dinner with his wife. This is not something they do often, and the hearing aids will be subject to another new environment. At the restaurant, a new notification pops up on Bob's smartphone. It is USense SoundLearner again. However, this time it does not suggest a training session, it simply asks if Bob likes the current hearing aid settings which he confirms. What is not apparent to him is that the USense SoundLearner has found a similarity between the environment at the canteen at his workplace and the restaurant. The knowledge gained during previous training sessions has been utilized in this new, yet similar, environment, and Bob has confirmed to the USense SoundLearner that this was a good choice. Bob and his wife carry on with their dinner without any discomfort or new notifications.

Misinterpreted user intent leading to incorrect configuration

Bob has been invited for the annual strategy meeting at work where the managers from his department get together to discuss future plans and what needs changing. Bob is not very excited for the meeting as it does not really interest him, and it is late in the day, so his motivation is low. Yet, he feels like he needs to be there such that he can at least have say if needed. He decides to bring his computer to the meeting and sit in the back, so he can answer some emails in the meantime.

Shortly after the meeting has started, USense SoundLearner detects that the environment has changed and switches to Bob's usual preference which is to open up for sound input and make it possible to hear everyone at the meeting. However, this time, his intent is not to take part in the meeting as he usually does, and this configuration is bothering him. Bob is not able to focus on answering his emails and he brings out his smartphone. USense SoundLearner thinks it has performed as it should and does not bother Bob with notifications asking for feedback every time it makes a change. This makes it a bit more cumbersome for Bob to input the fact that he is displeased with the settings. However, Bob manages to find a button in the application allowing him to start a training session. Bob inputs that he is in a "Meeting" environment and that his intent is to "Concentrate". This is input to the training session and allows USense SoundLearner to get an initial idea of what Bob is trying to achieve. After a few iterations, the new preference is discovered and applied to Bob's hearing aids. Before ending the session, the application notifies Bob that the meeting environment usually is linked to a difference preference setting asking him if this new training session should adjust the current meeting environment preference. Bob knows that this is not his usual preference and tells the application that it is a one-time adjustment that should not affect his usual preference. Bob is now able to focus on answering his emails while still participating in the meeting. Furthermore, next time he will be in a meeting situation, the hearing aids will adjust to his usual preference.

5.3 Risks and assumptions

Based on the personas (Section 5.2.1), empathy maps (Section 5.2.2), and scenarios (Section 5.2.3) it can be found that this project is developed under some assumptions and risks that can affect whether the solution will succeed or not. This section lists these risks and assumptions and follows by plotting them based on how significantly the solution will be affected if they materialize and how much is known about them at this stage. The following text lists the risks and assumptions. When they have been prioritized, the high-risk rated elements are described in more detail.

Risks

- **R1** The solution is incapable of capturing user preferences during training sessions.
- **R2** A training session is too time-consuming discouraging the user from using the learning functionality rendering the solution useless.
- **R3** The solution is not able to distinguish between different environments to the level of detail required. As a result, user preferences cannot be linked to a distinct environment making long-term learning difficult.
- **R4** Long-term learning is incapable of finding similarities between outcomes of multiple training sessions in the same environment. As a result, it become unable to generalize and represent the user which leads to an inability to provide a suitable hearing aid adjustment next time the user is in that specific environment.
- **R5** A user's preference or intent changes too often in a specific environment making it difficult for the solution to establish a general preference. As a result, the user has to perform training sessions more often.
- **R6** Fine-tuning bass, midrange, treble, speech focus, noise reduction, and wind noise reduction are not sufficient to reach the optimal auditory experience for the user.
- **R7** The solution is not capable of determining a user's intent leading to suboptimal preference predictions.
- **R8** Including the learning functionality which provides the personalization makes people uncomfortable with the entire hearing care solution.
- **R9** The solution requires the user's attention too often (e.g. notifications) eventually making the user annoyed with the solution. As a result, the user has a bad experience and might use it less.
- **R10** Improper onboarding flow leading to wrong or limited use of learning functionality.

Assumptions

- **A1** The solution can be designed and implemented such that it can serve both the care seeking persona and the proactive persona without making it too advanced for the care seeker.
- **A2** The users of the solution constantly carry a charged smartphone with them.

– **A3** Sound can be collected by the hearing aids and streamed to the smartphone for real-time analysis.

– **A4** The users frequently connect their smartphone to the internet allowing the application to send data off to long-term learning and receive new long-term learning models.

– **A5** Convolutional Neural Networks can be utilized to determine the current environment based on the soundscape.

– **A6** The state-space can be limited intelligently such that there is no perceived reduction in the available optimization-space where the user will find the preferred settings.

A3 is defined as the system requires sound input to determine the environment (see Section 5.1.3, page 54). Furthermore, see Section 5.1.4 (page 55) for a description of why **A4**, **A5**, and **A6** are necessary. The remaining risk and assumptions are based on the user modeling.

5.3.1 Risk evaluation

A risk plot is used as a tool for prioritization [44]. It visually makes critical risk and assumptions apparent and, hereby, makes it easier to determine where to invest resources to mitigate a risk. The risk plot performs the classification using two dimensions: Low risk to high risk and known to unknown.

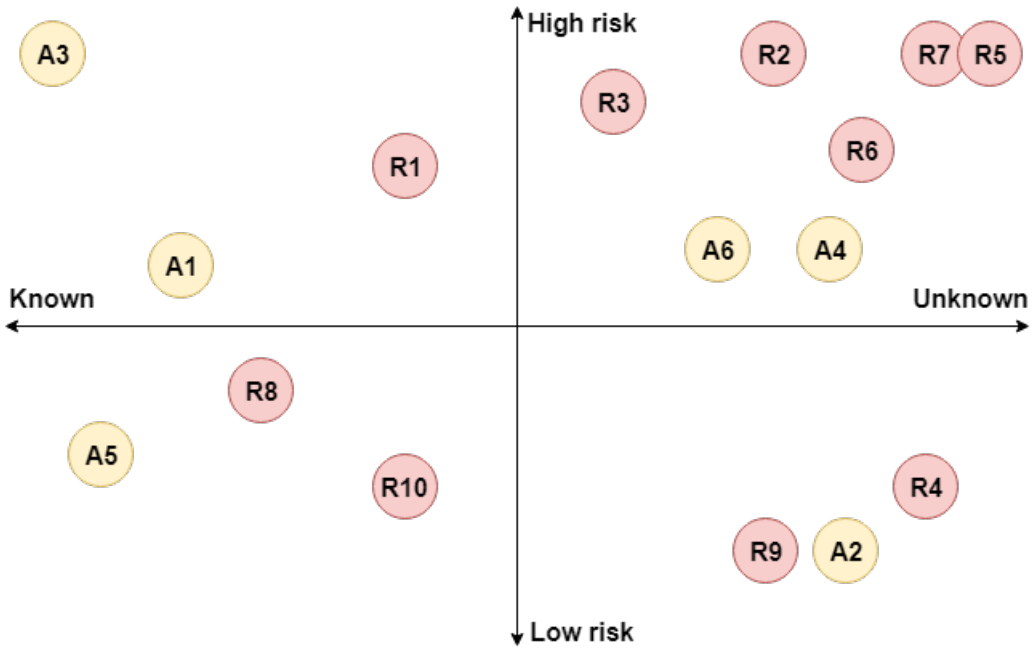


Figure 5.12: Prioritization of identified risks and assumptions based on risk and knowledge level. Made with draw.io online tool.

Figure 5.12 depicts the prioritization of the extracted risks and assumptions. Based on the plot, **A3**, **R3**, **R2**, **R7**, and **R5** have been identified as high-risk elements and are particularly important to gain

some information about or mitigate to ensure a feasible solution. The rational for each of them is as follows:

- **A3:** The user's preferences need to be linked to the environments, and an environment is described by the soundscape. For this reason, it is a must to record the sounds surrounding the user. As the smartphone might be in the user's pocket, this does not necessarily provide an accurate representation of the sounds present. The hearing aids have built-in microphones, and they are connected to the smartphone. Hence, theoretically, they can be used to collect the audio and stream it to the smartphone for analysis. This is a high-risk assumption as the feasibility of the solution is greatly reduced if it does not hold true. Based on the assumption's location in Figure 5.12, it can also be found that it is a "known" assumption. As mentioned in Section 2.6 (page 14), it is not possible to stream audio from the hearing aids that are available for the prototype development in this project. With that said, this project can still show the potential for personalized hearing aids. The prototype just have to utilize a different audio source than would be seen in a consumer-ready product.

- **R3:** This risk is concerned with the ability of the prototype to differentiate between environments with a sufficient granularity. This is important as the user's preferences depends on the environment. Assume a user has trained the prototype to apply one set of preferences in a restaurant environment and a different set of preferences in a forest environment. In the first case, the user might like to focus on a conversation in front while reducing background noise. In the second scenario, the user might want to get as many sound inputs as possible such that all the bird tweets and sound of wind in the trees is not muffled. If the prototype is not able to differentiate these two environments, it will not be able to apply the appropriate preferences. As a result, the user will not be satisfied with the solution and might stop using it. Until this aspect has been tested, this is an unknown and high-risk element.

- **R2:** The state-space is rather large as the prototype considers bass, midrange, treble, speech focus, noise reduction, and wind noise reduction. As a result, it might be difficult for the solution to pinpoint exactly what the optimal configuration is for the particular user in a given environment. If it is difficult to find the optimal configuration it can require a lot of interaction with the user before a solution is found. This is a rather unknown risk which requires testing to be performed before it is revealed whether it is a problem or not. Long training sessions might discourage the user from starting them or make the user lose motivation during the session.

- **R7:** The environment might not be sufficient to determine the user's preferences. As scenario "Misinterpreted user intent leading to incorrect configuration" describes, the environment might not change, but what the user wants to achieve in that environment can change. It is highly unknown if the solution proposed in this project is capable of encoding the user intent as well as the environment. If not, the suggested user preferences might not actually fit the user.

- **R5:** Assuming the environments and intents can be captured by the solution, a risk regarding changing preferences arises. It is unknown how much input the solution will require in order to establish a general preference for a given user in a specific environment. This is intended to work

such that over time, the user runs multiple training sessions in the same environment. This makes it possible to aggregate the outcome of all the sessions and determine what the user generally likes. However, if the user's preferences change (e.g. due to fatigue or changes in required attention level) at a rate which is higher than the collection of data, the solution might not be able to keep up. As a result, it cannot generalize and will provide preference suggestions that no longer fit the user.

The risk prioritization plot in Figure 5.12 reflects the prioritization at this stage in the project, and it might look different for anyone else adopting the research. As an example, assumption **A3** regarding audio streaming from the hearing aids, might not be an assumption at all if one is working with hearing aids that implements this functionality. The risk plot is not supposed to serve as a definitive list, but as a snapshot of this project at this stage.

Based on testing of these risks and assumptions, conclusions can be made in regard to the overall feasibility. Hereafter, it can be evaluated if the solution needs to pivot. This project does not aim to perform an extensive feasibility study of a consumer-ready solution. However, the Discussion chapter reevaluates some of the risks and assumptions based on user interviews (see Section 10.2, page 145).

To support the risk plot and define what must be done to mitigate the risks, the following tables (Table 5.5-5.8) are created. The tables evaluate each of the risk on a scale of probability of occurrence and impact in case of occurrence. As a result, the severity of each risk can be classified. Table 5.4 illustrates the probability and impact dimensions, and how they affect the risk rating. To reduce risk impact or likelihood of occurring, a mitigation plan is provided for each risk.

	Impact			
Probability		**Minor**	**Moderate**	**Major**
	Very likely	Medium	High	High
	Likely	Low	Medium	High
	Unlikely	Low	Low	Medium

Table 5.4: Risk rating evaluation dimensions.

Risk	Probability	Impact	Risk rating	Mitigation plan
R1 Incapable of capturing user preferences	Unlikely	Major	Medium	Reduce that parameter scope. The state-space is likely to be the cause if this risk materializes. Reducing the parameters (bass, midrange, treble etc.) that are taken into account during a training session reduces the dimensionality and, hence, the complexity. As a result, it is easier to learn the user's preferences and the risk is mitigated.
R2 Too time-consuming solution	Likely	Major	High	Similar to **R1**, the state-space size must be reduced. Alternatively, Sarsa(λ) can be biased by other inputs (e.g. user intent, geolocation, or time of the day). This will focus the state search making it easier to find an optimal state. Furthermore, the search can be done more coarsely such that similar states are not considered until convergences seems to occur.
R3 Incapable of determining the environment	Unlikely	Moderate	Low	The architecture must be sufficiently complex to understand the input, but no more complex than that. Furthermore, the data must be plentiful and balanced such that the architecture has sufficient data to learn from and is not biased by unbalanced data (unequal number of samples for all environments). Alternatively, the environment scope can be reconsidered. Ensure that the solution is only trying to cover necessary environments.

Table 5.5: Risk impact evaluation and mitigation plan for **R1**-**R10**.

Risk	Probability	Impact	Risk rating	Mitigation plan
R4 Lack of long-term learnability	Unlikely	Moderate	Low	Collect more audio data representing the sound environment. This gives the CNN more material to learn from. Alternatively, consider reducing the environment scope (e.g. combine "Home" and "Office" to "Indoor"). Having fewer and more general environments creates more training samples for each environment. However, this solution might increase the probability of **R5** materializing.
R5 Frequent changes in user preferences or intent	Likely	Major	High	Increase the complexity of the state-space such that intent can be taken in to account as well. However, this increases the risk of materialization for **R1** and **R2**. Instead, accept that preferences change often, and intent is difficult to model. Focus on ensuring efficient training sessions and reducing the state-space. As a result, training sessions are sped up making it less of a headache for the user to go through more often.
R6 Insufficient adjustment possibilities	Unlikely	Moderate	Low	The hearing aid has additional gain handles which can be adjusted. These can be taken into account. However, this might increase the state-space creating other issues (e.g. **R1** and **R2**). Possible rely on information from the current state-space (i.e. parameters) to adjust the additional gain handles instead of including them in the state-space directly.

Table 5.6: Risk impact evaluation and mitigation plan for **R1**-**R10** continued.

Risk	Probability	Impact	Risk rating	Mitigation plan
R7 Incapable of determining intent	Very likely	Moderate	High	Use a similar approach as seen with the Widex SoundSense Learn solution: Have the user provide their intent. This solution should be temporary until a better intent prediction method is developed. Advanced intent inference could be based on specific and known voices, level of activity, orientation in relation to main sound source etc. Further elaboration on such method can be found in Section 9 (page 140).
R8 User uncomfortable with solution	Unlikely	Moderate	Low	Provide proper communication from the marketing team, the audiologist, and within the application. The audiologist should spend time with the user introducing them to the solution. Furthermore, the onboarding session should clearly state the functionality and how users can cancel and revert changes. If this is not sufficient, adopt similar approach as is seen today: Limit such that only users that are comfortable with it have the functionality.
R9 Demands attention too frequently	Likely	Moderate	Medium	Accept higher level of uncertainty in predictions and automatically apply suggestions without getting confirmation from the user. If it provides a bad experience, the user will act on it. If no action is taken, the suggestion is sufficiently good.

Table 5.7: Risk impact evaluation and mitigation plan for **R1**-**R10** continued.

Risk	Probability	Impact	Risk rating	Mitigation plan
R10 Improper onboarding	Likely	Moderate	Medium	Similar to **R8**, the onboarding session must provide all the necessary information in an easy-to-consume manner. If this is not sufficient, nudging can be utilized. If a user has not yet used the functionality, or have not used it for a long period of time, the user can be made aware of it. Communicate that it can provide an optimized and automated experience if used.

Table 5.8: Risk impact evaluation and mitigation plan for **R1**-**R10** continued.

R2, **R5**, and **R7** have been rated "High" which means they are most problematic for the solution being developed in this project. **R7** is related to predicting the user's intent and it is accepted that intent is difficult to predict. Defining a method for inferring intent is out of the scope for this project. This is reflected in the requirements (**NFR7** is rated as a "Won't have" requirement for this prototype. See Table 5.12, page 77). Complexity can be added whenever the basics of the prototype have been proven. Similar argumentation goes for **R5**. As for **R2**, Section 6.3.2 (page 87) describes how the training session is designed to efficiently determine that an optimal state has been found.

The "Low" risks (**R3**, **R4**, **R6**, and **R8**) are all unlikely to occur and no additional effort are put into these risks for the prototype. From the "Medium" rated risks (**R1**, **R9**, and **R10**), both **R9** and **R10** are concerned with user annoyance and proper instructions which is not needed to prove the technical viability of the solution. The requirements (e.g. **FR18** and **FR23**) are prioritized accordingly, and it is not evaluated to be a problem for the prototype. **R1** does need to be handled and the mitigation plan stated for **R1** in Table 5.5 is followed. The prototype should only include bass, midrange, and treble.

5.4 Requirement specification

This section presents the requirements for the prototype being developed in this project. The requirements are derived from the technical analysis as well as the user modeling to ensure a solution that is both technically sound and usable by the average hearing aid user.

The prioritization of the listed requirements are done in accordance to the MoSCoW model. Furthermore, correctness of the prioritization has been verified with Anders Udahl who is Senior Manager for GN Hearing's Connected Apps team. During the verification session, requirement **FR19**, **FR26**, **NFR19**, and **NFR20** where created and prioritized. Furthermore, during a user modeling verification session with Adam Heleniak, Senior UX Designer in GN Hearing's User Experience department, requirement **FR20**, **FR22**, **FR27**, and **FR28** were created and prioritized.

The requirements are prioritized twice: Prototype and MVP. The prototype prioritization lists the requirements in accordance to what is expected as the outcome of this project. The MVP prioritization

lists the requirements as if the solution were to reach the customers.

Further elaboration of each requirement can be found in Appendix D.1 and Appendix D.2 for the functional and non-functional requirements respectively. All functional requirements are denoted as "**FR#**" with # being the requirement number. Similarly, non-functional requirements are denoted as "**NFR#**" where # is the requirement number.

Some requirements dictate usage of Short-Time Fourier Transformation (STFT) spectrograms. This is a way of representing audio data and has previously been used as input to the CNN for environment classification in the prestudy [23]. STFT spectrograms are not covered in this project, however, this project adopts some of the knowledge gained in the prestudy [23]. The STFT spectrograms are utilized for this prototype, and the requirement specification must reflect that.

The requirements use the following names to describe specific elements of the prototype: "the solution", "the cloud", "the smartphone application". When the prototype as a whole (all the components) have a shared responsibility, "the solution" is used to denoted this. When the requirement is specific to the part of the prototype running as a smartphone application, this is denoted as "the smartphone application" or "the smartphone". Lastly, "the cloud" is used when a requirement is concerned with server-side functionality accessible over the internet. The requirements do not specifically demand a cloud solution when "the cloud" is mentioned. It is up for interpretation and merely indicates that an instance accessible over the internet with sufficient resources is needed.

To create a better overview, each of the follow tables have a group of requirements that apply to a specific part of the system. Where applicable, the requirements within one group has been split into a functional and non-functional table as well.

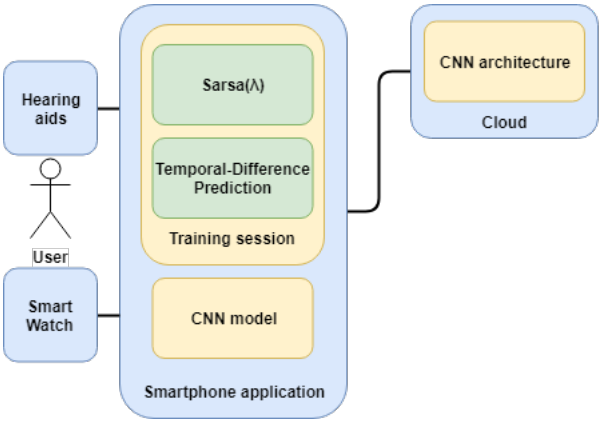


Figure 5.13: High-level diagram showing location of components mentioned by the requirements. Made with draw.io online tool.

Figure 5.13 provides a very high-level view of the components in the system mentioned by the requirements. The diagram shows where each component is located and how it is connected to the rest of the solution. This is provided to accompany the requirements and ease understanding. A more detailed visualization is made using two use case diagrams. These are located in Appendix E and describes the actors of the system and what interaction possibilities the actors have. Furthermore, each component is label with the requirement which it is responsible for.

Solution general requirement (functional)	Prototype	MVP
FR1 The solution must be able to predict the user's preferred adjustments of bass, midrange, and treble in environments similar to previous training sessions	**M**	**M**
FR2 The smartphone application must collect audio of the user's soundscape	**S**	**W**
FR3 The solution must perform training of CNN models for environment classification and hearing aid adjustment regression in the cloud	**M**	**M**
FR6 The smartphone application must be able to download and implement newly trained CNN models	**S**	**M**
FR7 The cloud must provide trained models which can be used for inference on the smartphone	**S**	**M**
FR8 The smartphone application must be able to process the collected audio into STFT spectrograms	**S**	**S**
FR9 The smartphone application must provide STFT spectrograms with user preferences and an environment label to the cloud as input for CNN model training	**S**	**M**
FR10 The smartphone application must store STFT spectrograms with preference and environment information until it can establish a connection to the cloud solution	**C**	**S**
FR11 The cloud solution must start training a new CNN model when new STFT spectrograms with preference and environment information is received	**S**	**W**
FR14 The training session must terminate when the optimal state has been found	**S**	**S**
FR16 When a training session is terminated, the best user preference prediction must be applied to the hearing aids	**M**	**M**
FR17 The solution must suggest new personalized hearing aid adjustments if the environment changes	**S**	**S**
FR27 Adjustment suggestions can be applied automatically without confirmation from the user if the user has agreed to this during the onboarding session	**S**	**S**

Table 5.9: A list of functional requirements that applies to the solution as a whole. The list is prioritized using the MoSCoW model.

During the requirement verification session with Anders Udahl, it was pointed out that not even a full-scale solution would need all parameters. Focusing on what is listed in **FR1** is sufficient possibly even for a full scale solution. Furthermore, it was pointed out that **FR2** is not applicable to anything but a prototype. When the solution is to transform into an MVP, there is a need to find a better way to collect audio (possibly not base the decision on audio or get it streamed from the hearing aids). Similarly, **FR11** does not scale to anything but a prototype and, hence, is not applicable to an MVP.

Solution general requirement (non-functional)	**Prototype**	**MVP**
NFR1 The solution must consist of a smartphone application and a machine learning cloud instance	**M**	**M**
NFR2 The smartphone application must implement and use Sarsa(λ) to learn the user's preferences	**S**	**S**
NFR8 All hearing aid adjustment suggestions must be predicted by a Temporal-Difference Prediction routine, not by the CNN model alone	**S**	**S**
NFR14 The STFT spectrograms must cover 1 second of audio	**M**	**M**
NFR15 The solution must sample the environment every 5 seconds	**C**	**S**
NFR16 The solution must determine an environment change based on 6 samples	**C**	**S**
NFR20 The solution must run the environment observation as a background service	**W**	**W**

Table 5.10: A list of non-functional requirements that applies to the solution as a whole. The list is prioritized using the MoSCoW model.

NFR15 and **NFR16** mentions some specific values related to environment classification. Testing is needed to verify if the values specified are appropriate. Hence, requirements are subject to change as more information is collected.

User interaction specific requirement	Prototype	MVP
FR12 The smartphone application must ask for the user's intent when starting a new training session	**W**	**S**
FR13 The training session must be a pairwise comparison of sound profiles	**S**	**S**
FR15 The smartphone application must allow the user to terminate a training session at any time	**C**	**S**
FR18 The user must receive a notification on their smartphone when new adjustments are being suggested	**C**	**S**
FR19 The user must receive a notification on their smartwatch when new adjustments are being suggested	**W**	**C**
FR20 All notifications can be rejected by the user	**W**	**W**
FR21 The user can try the new hearing aid adjustment suggestion before accepting or rejecting it	**S**	**S**
FR22 Upon three consecutive notification rejections, the user must be asked if they would like notifications less often	**W**	**W**
FR23 First-time users must go through an onboarding session to determine who must initiate training sessions and how often notifications must come	**W**	**S**
FR24 The smartphone application must allow the user to start a training session	**M**	**M**
FR25 The smartphone application must suggest starting a training session if the environment is new or if it is uncertain about the quality of the suggestion it can provide	**C**	**S**
FR26 A consent message must be presented and accepted by the user before the machine learning components of the solution are enabled	**W**	**M**
FR28 The system must allow a user to overwrite the old adjustments if a training session resolves in an unexpected outcome dissimilar to previous knowledge	**W**	**C**

Table 5.11: A list of requirements that applies to the smartphone application element of the solution. The list is prioritized using the MoSCoW model.

The majority of the requirements specified in Table 5.11 are prioritized as "Won't" requirements for the prototype. This is due to all the requirements being related to the behavior between the prototype and the user. A lot of it is not necessary to prove that the solution works. As a result, it is not needed for the prototype to implement these.

Sarsa(λ) specific requirement	**Prototype**	**MVP**
NFR3 The state-space must consist of the following dimensions: Bass, midrange, treble, and user environment (All-Around, Restaurant, and Outdoor)	**M**	**M**
NFR4 The solution must determine the user's preferences in 12 (or less) iterations	**C**	**C**
NFR5 Sarsa(λ) is allowed to adjust hearing aid configurations $\pm$3 dB for each training iteration	**S**	**S**
NFR6 Sarsa(λ) must use the environment and user preference prediction from the CNN model as a starting point for the training session	**M**	**M**
NFR7 The user's intent information must bias the trajectory of Sarsa(λ)	**W**	**C**
NFR9 An optimal state has been found when the following are aligned: A Temporal-Difference Prediction run, negative state rewards, and continuous user emphasis.	**S**	**S**
NFR17 An accepted hearing aid adjustment suggestion provides +10 reward to that Sarsa(λ) state	**S**	**S**
NFR18 A rejected hearing aid adjustment suggestion provides -10 reward to the Sarsa(λ) state representing those adjustments and $+\frac{10}{2}$ reward for the state representing the current adjustments	**S**	**S**

Table 5.12: A list of requirements that applies to the Sarsa(λ) algorithm. The list is prioritized using the MoSCoW model.

Requirement **NFR3** states that only a subset of the hearing aid parameters are adjusted. The non-functional requirement is linked to **FR1**. As a result, it can be concluded that **NFR3** is also sufficient even for a full-scale solution. Both **FR1** and **NFR3** are "Must" requirements meaning they are important. This is just to point out that it is important to cover all three parameters (bass, midrange, and treble), but not necessary to cover anymore than these.

CNN specific requirement (functional)	**Prototype**	**MVP**
FR4 The CNN models must be able to classify any environments into one of the following classes: All-Around, Restaurant, and Outdoor	**S**	**S**
FR5 The CNN models must be able to predict the user's environment and general hearing aid adjustments for that environment	**M**	**M**

Table 5.13: A list of functional requirements that applies to the Convolutional Neural Network. The list is prioritized using the MoSCoW model.

Similar to **FR1**, limited functionality is allowed also for an MVP in relation to what environments are supported (**FR4**), according to Anders Udahl. Focusing on All-Around, Restaurant, and Outdoor is sufficient for a minimum viable product. Furthermore, it has been categorized as a "Should" meaning it is not vital even for an MVP. It might be possible to develop workarounds to stay on schedule.

CNN specific requirement (non-functional)	Prototype	MVP
NFR10 The CNN architecture must be a multi-task network with one classification branch and one regression branch	**S**	**S**
NFR11 The CNN regression branch must use Mean Squared Error as cost function	**S**	**S**
NFR12 The CNN regression branch must use ReLU in the hidden layers and the identity function in the output layer as activation functions	**S**	**S**
NFR13 The STFT spectrograms must be used as input to the CNN model	**M**	**M**
NFR19 Each CNN model must be specific to the user	**S**	**W**

Table 5.14: A list of non-functional requirements that applies to the Convolutional Neural Network. The list is prioritized using the MoSCoW model.

Requirement **NFR19** is prioritized lower for an MVP than for the prototype. The prioritization is a result of the requirement verification session with Anders Udahl. The prototype wants to provide a very personalized solution to the user. However, providing individual CNN models for each user is, according to Anders Udahl, not immediately a viable solution for an MVP or full-scale solution.

Chapter 6

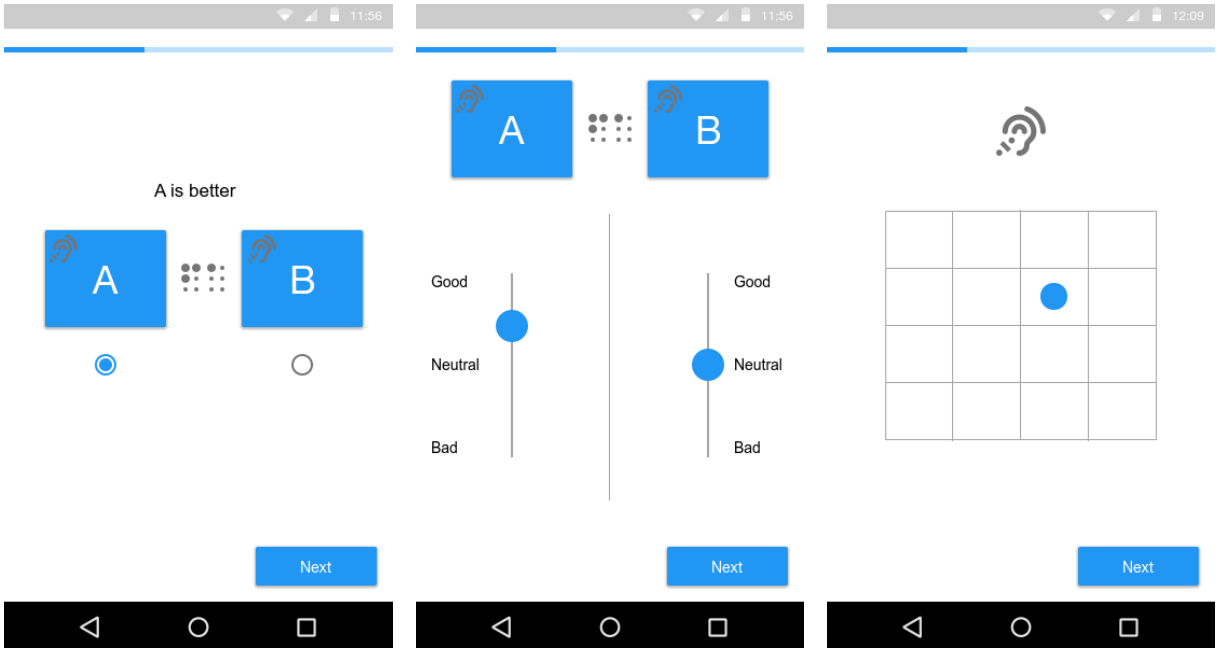
Design

The following section describe the design of the prototype solution. This includes everything from the visual layout and high-level diagrams outlining the components of the system, to detailed descriptions of state-space design and how termination is ensured. The design is based on the Analysis chapter and the specified requirements.

The following text mentions "USense SoundLearner" which mainly refers to the smartphone application. However, supporting components such as an Online training entity is needed. Therefore, USense SoundLearner can be used to refer to the entire solution. The context of the usage of the name makes it clear exactly what it refers to.

6.1 Training session design

This section proposes multiple layouts that can all be used for the USense SoundLearner training sessions. The purpose of the layout is to collect information about user preferences. Requirement **FR13** states that this must be done as a pairwise comparison. Furthermore, Section 5.2 shows that the solution must apply to two very different types of users: One of which is very careful around unfamiliar technology while the other is more exploratory. Both the requirement (**FR13**) and the different user types create some constrains in terms of how the layout can be designed. Widex (Figure 5.6, Section 5.1.3, page 49) has created one elegant way of collecting the user input. The following layouts present variations inspired by Widex, but also presents a new way of thinking of a pairwise comparison.



(a) (b) (c)

Figure 6.1: Three proposed USense SoundLearner training session layouts for collecting user preference information. Made with proto.io online tool.

Figure 6.1 presents three layouts which can be used to collect a user's preferences such that an optimal hearing aid adjustment can be achieved. Figure 6.1a presents the most basic layout. Two sound profiles (two states from the state-space i.e. two hearing aid adjustments) are presented to the user, and the user can click and listen to both. Before clicking "Next", the user has to pick one as the best. This is a simple pairwise comparison, and possibly too simple. The user does not have the opportunity to say that neither is better. This problem is solved by the layout presented in Figure 6.1b. This layout does allow a user to give a neutral rating, however, it also increases the complexity of the comparison. The layout is designed such that a user has to listen to both sound profiles and also rate them both. This is more work for the user, but allows the user to express that a sound profile is dissatisfactory, neither good or bad, or good. Both profiles can be rated in this manner independently of each other.

Figure 6.1c takes a different approach to sound profile comparison. Instead of giving the user profile "A" and profile "B" to choose from, the user can navigate a grid by moving the blue dot to find the best sounding profile. When the dot enters a new cell in the grid, the sound profile (i.e. hearing aids adjustment) represented by that cell is applied to the user's hearing aids. This is analogues to the color picker one might have interacted with in a graphics editor. However, instead of determining a color by sight, a preferred adjustment is determined by ear. Essentially, this layout presents a subset of the state-space to the user but in a two-dimensional format. The grid does not have to be made up entirely of similar states (i.e. states near to each other in the state-space). It might be advantageous to present states that are audibly different. The same goes for the other layouts as well of course. However, more states can be presented at once with the grid layout. As defined by requirement **NFR5**, sound profiles can change by three decibels for each training iteration. This can be used to define the state similarity

in a grid layout as well. Given the user can figure out how to use the layout, the final outcome should be the same as with the layouts in Figure 6.1a and Figure 6.1b. However, the algorithm might more quickly get an idea of the trajectory the user likes if the layout from Figure 6.1c is utilized.

Common for all the layouts are a progress bar at the very top and a "Next" button in the lower right corner which allows the user to continue to the next sound profile comparison. The progress bar is included to indicate to the user how far the training session has come. The idea is that the visual feedback might help the user stay focused and engaged. A similar progress indication is implemented in the Widex SoundSense Learn. However, progress might be difficult to measure as a user can oscillate between multiple optimal states. It is likely that Widex is faced with the same problem. When using their solution (presented in Section 5.1.3, page 49), it is noticeable that the progress indicator does not move a specific amount after each sound profile comparison. Hence, a progress bar for the solution of this project might not need steady and predictable increments either to be appreciated by the user. This is speculative and user testing is needed to actually confirm this hypothesis.

To ensure simplicity such that all types of user can use the solution, this prototype should include the layout from Figure 6.1a. If this is found to not provide enough information to Sarsa(λ) to finish a training session in 12 iterations (according to requirement **NFR4**), the other layouts can be considered for later prototype implementations. However, it should be ensured that the desired user group is comfortable with interacting with the chosen layout if the complexity increases.

6.2 High-level system architecture

The system consists of multiple components. This section describes these components and outlines how they are connected. The high-level architectural overview also describes internal modules and functionality of each component. This section presents the entire system, and the individual technical details of the relevant components and modules are then outlined throughout the remaining of the design chapter.

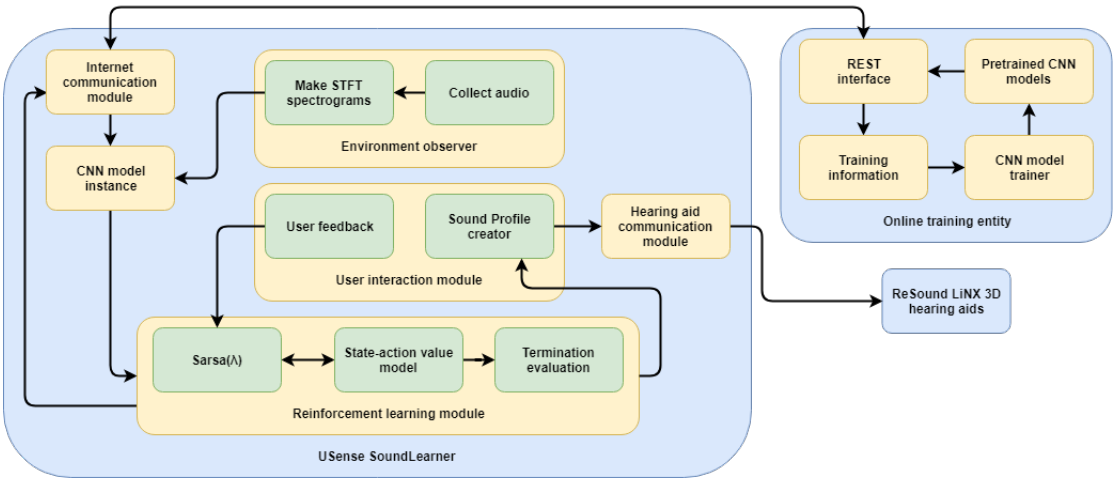


Figure 6.2: High-level architectural overview of USense SoundLearner and related components. Made with draw.io online tool.

The high-level architectural overview is depicted in Figure 6.2 and contains three main components: the USense SoundLearner smartphone application, an Online training entity, and the hearing aids. The components are color-coded to clarify encapsulation and responsibility boundaries. Blue shapes indicate overall components. Components can contain yellow functionality elements or modules which are responsible for a more narrowly defined units of the solution. Furthermore, yellow modules may group multiple green functionality elements with a common purpose.

Online training entity (OTE)

As it is not viable to train Convolutional Neural Network (CNN) models on a smartphone, the requirements state that a cloud instance must be setup to handle this task. In this project, the term "cloud" is up for interpretation and just denotes an online service that can be reached over the internet which has sufficient resources to train CNN models. Hence, the entity has been vaguely named in this section too.

As depicted in Figure 6.2, the responsibility of this component is to train new CNN models and make them available through a REST interface. The models are created based on data packages uploaded to this entity. Such a package consists of "Training information" which is Short-Time Fourier Transformation (STFT) spectrograms representing the user's environment, an environment label, and the user's hearing aid configuration preferences. From this, the CNN has an input (STFT spectrograms) and the corresponding output (environment label and user preferences). Using this information, new CNN models can be produced an made available for download.

No user preference information is available until the first training session has been completed. However, a training session needs to know the current environment such that the learned user preferences can be associated with the correct context. Therefore, a general CNN model must be made available. This general model can be used by the USense SoundLearner until sufficient user information has been collected to create a personalized CNN model. A general model can be created as audio samples where classified according to their environment with GN Hearing Audiology during the prestudy [23].

USense SoundLearner

The USense SoundLearner smartphone application is the main component of the solution. The responsibility of the application is to perform environment classification, predicting user preferences, running training session to learn a user's preferences, and interact with the user. Furthermore, modules for communication with the OTE and for hearing aid communication must be implemented.

This component downloads CNN models from the OTE and runs an instance of the models locally. The application records audio from the user's surroundings and represents this as spectrograms. Combining these two features makes it capable of classifying environments and predicting hearing aid adjustments. This input can then be used during reinforcement learning training session to guide the algorithm. Based on multiple pairwise comparisons of sound profiles, the user's current hearing aid configuration preferences can be determined. This can be applied to the hearing aids and also used by the OTE to create new CNN models.

Figure 6.2 depicts an Environment observer module and only a one-way arrow for the hearing aids. The hearing aids used for the prototype development cannot stream audio to the smartphone. For this reason, audio must be collected by the application. Furthermore, as the machine learning is running

locally, and the user should not be required to constantly be connected to the internet, processing of the audio must be done locally as well. The OTE is using STFT spectrograms for the training of new CNN models. As a result, the smartphone application must support processing audio into STFT spectrograms.

6.3 Machine learning architecture

The complete solution consists of two types of machine learning. This section outlines the network design of the Convolutional Neural Network (CNN) as well as the state-space design which is utilized by the Sarsa(λ) algorithm. The last section describes how the two machine learning approaches are linked and how the CNN feeds into Sarsa(λ).

6.3.1 Convolutional Neural Network design

Convolutional Neural Networks are constructed using different layers with certain abilities [18]. A convolutional layer is able to extract important information from the input [18]. Such a layer is combined with a pooling layer which is responsible for reducing the amount of data while keeping the important information [18]. After extracting and reducing information, a fully connected layer can be used to create the mapping between input and output [18]. The following text describes how this prototype is arranging these layers and how branching is performed.

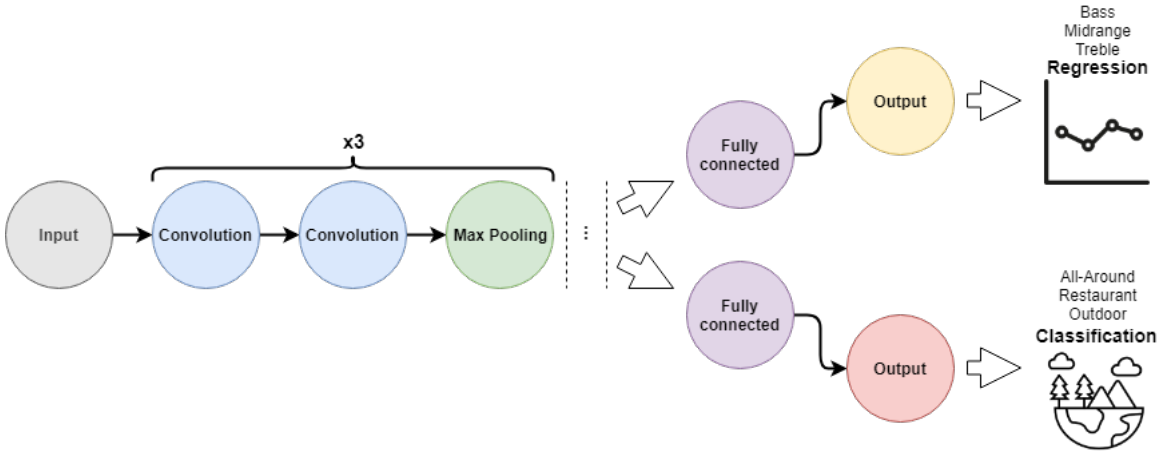


Figure 6.3: Branched Convolutional Neural Network architecture to allow for both regression and classification. Made with draw.io online tool.

In order to design a network that is capable of doing both regression and classification, a branching strategy must be utilized. For this prototype, it is evaluated that the tasks of predicting bass, midrange, and treble are very similar. For this reason, the regression task is taken care of by a single branch allowing for maximum parameter sharing in the network. The classification task differs from the regression task, but is likely to benefit from the same findings in the input. Therefore, the regression task and classification tasks get their own fully connected layer, but share all the input features extraction prior to the fully connected layers. This makes it possible for the network to creates one mapping from input to bass, midrange, and treble, and a different mapping from input to environment class. As a result, the network

looks as depicted in Figure 6.3. Hence, it can be seen that this network has two branches with one output layer each. One output layer (the yellow output node on the top right of Figure 6.3) produces a prediction of the bass, midrange, and treble adjustments based on the input. The other output (the red output node on the lower right of Figure 6.3) creates environment labels by classifying the input into either All-Around, Restaurant, or Outdoor.

As mentioned, a CNN includes convolutional and pooling layers. This project will build on the findings by the prestudy [23] and utilize the same architecture for feature extraction. The prestudy [23] found that serializing three chunks consisting of two convolutional layers followed by a max pooling layer is capable of comprehending a Short-Time Fourier Transformation (STFT) spectrograms. The grey input node on the left of Figure 6.3 represents the input layer which is receiving the raw STFT spectrograms for processing by the neural network. This architecture is similar both for training and inference, and it is important that the inputs are consistent. According to the requirements, the audio must be processed into a STFT spectrogram by USense SoundLearner (**FR8**), and these spectrograms must cover one second of audio (**NFR14**). The CNN models are trained to recognize a specific type of input (STFT spectrograms). If a different audio representation is used for some reason, the model is not able to understand the input and will not produce reliable outputs.

6.3.2 Sarsa(λ) state-space design

The following text explains how the state-space is designed to accommodate multiple environments. Apart from representing hearing aid parameters, the state-space must also reflect the context wherein these adjustments are applicable.

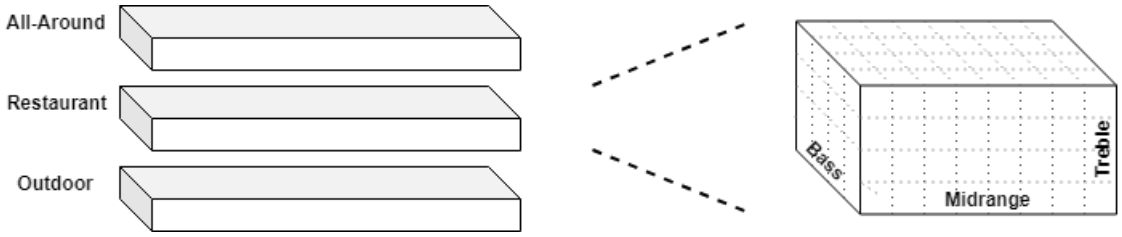


Figure 6.4: Illustrating the state-space designed for this project. Made with draw.io online tool.

Figure 6.4 depicts the state-space design for this project. The left of the figure illustrates the entire space, and the right of the figure shows a zoomed view illustrating the details. What can be seen on the left is three layers each denoted by one of the environment classes considered in this project. Each environment classes are linked to an identical subset of the state-space. This subset is depicted on the right of Figure 6.4 and consists of a three-dimensional space made up of bass, midrange, and treble parameters. Each state in the three-dimensional state-space illustrates one possible combination of bass, midrange, and treble adjustments. The entire space represents all possible adjustments. Combining this with the three environment labels creates a state-space with a total of four dimensions.

As was depicted in Figure 5.2 (Section 5.1.3, page 44), adding dimensions to the state-space scales poorly. Despite the scaling problem, the user should still achieve personalized hearing aid adjustments after only few iterations of a training session (12 iterations or less according to **NFR4**). The state-space

depicted on the left of Figure 6.4 has $3 * 13 * 13 * 13 = 6591$ unique states. However, two thirds are eliminated by the environment classification which brings the number of states a user needs to assess during a training session down to 2197. Furthermore, the actions taken by Sarsa(λ) to move in the space can be defined such that some states are skipped. Each state differs from the surrounding states by one decibel. The requirements (**NFR5**) specify that adjustments up to three decibels are allowed. Hence, some state can be left unconsidered effectively resulting in a reduction of states-space. Reducing the state-space in this manner is somewhat artificial as the states do not disappear. Therefore, the implementation should still allow for movements less than three decibels, and skipped states should still be affected by rewards.

Movements less the three decibels are necessary when the current state is not a multiple of three. This may happen due to the CNN which will give Sarsa(λ) a starting point. The limits of the space are negative six and positive six for each parameter. If a starting point is defined to be a positive gain of four, Sarsa(λ) is not able to move to states representing higher gain for this parameter unless it can move less than three decibels as it will otherwise be out of bounds. Furthermore, whenever the user's preferred state is in the vicinity, moving three decibels might be too coarse of a resolution. Hence, whenever a user has moved back and forth between two states that are three decibels apart, the solution should try the states in between to see if this is what the user is trying to reach.

Balancing exploration and exploitation

The Analysis chapter (Section 5.1.3) discussed a number of methods that can be utilized to determine when a user has reached a preferred state. Based on this, the requirements (**NFR9**) states the termination decision must be based on Temporal-Difference Prediction, what state the user is currently in, and if the rewards for leaving that state are all negative. However, this brings up the issue of exploration and exploitation. Sarsa(λ) should explore the state-space to locate where the user's preference can be found. However, as soon as the algorithm is in the vicinity of the preference state, exploration should be limited to allow the user to locate and emphasize the optimal state. In this case, a high level of exploration might bring the user away from the optimal state. To solve this, exploration can be a function of knowledge about a state. Tokic [101] is one of the researches who have realized the need for such a function. Exploration should be high in the beginning of the process and then gradually decrease as the environment becomes less uncertain [101]. Tokic [101] describes a method that becomes less likely to suggest random actions as the changes in state values becomes less. According to Tokic [101], little or no change in value of a state indicates that the environment is more certain. A simpler approach is taken by Rekleitis [79] who suggested using a strategy that chooses a random action by chance but the chance is weighted by the number of states that have been visited. As a result, if more states are known (i.e. have been visited), there is a lower chance of exploration to happen [79]. Vázquez-Canteli and Nagy [107] concludes that most studies (of those they have evaluated) use ϵ-greedy due to its simplicity. ϵ-greedy covers the method that simply by chance selects a random action without taking number of visited states or their value into account. This strategy is mentioned by Sutton and Barto [94] as well.

The approach suggested by Rekleitis [79] is using a ratio between number of states visited compared to number of total states. This might not be appropriate for this project as some states are eliminated by the environment classification or skipped by the coarse action selection. Furthermore, implementing a strategy that is dependent on the value of a state might not be fitting for this project either. State-value

information should not be cleared between training sessions, but a new training session should start out being exploratory to determine if the user's preferences has changed significantly since last time. A strategy based on state-value would be biased by the information learned from earlier sessions and limit the exploration as a result. Instead, the approach should be to use an ϵ-greedy strategy and simply lower the probability by the absolute number of states visited, or, as suggested by Sutton and Barto [94], simply defining the likelihood as $\frac{1}{t}$ where t is the number of timesteps. By counting timesteps or how many states has been visited in a training session, information from earlier training session can still be utilized without affecting the exploration parameter. Counting timesteps or counting states non-uniquely yields the same outcome. Counting states uniquely does not decrease the exploration factor when a user is close to their preferred state and alternating between surrounding states. In this case, the exploration factor should be lowered such that it is clear this is the state a user wants to be in and not because it has been chosen by chance. As a conclusion, the solution implements the ϵ-greedy strategy where the likelihood of choosing a random action is defined by $\frac{1}{t}$.

Biasing the state-space toward user intent

Requirement **NFR7** states that this prototype will not take user intent into account during the training session. However, this section gives a brief explanation of a way in which the trajectory of Sarsa(λ) can be biased towards a specific intent. This section is made as it is understood that user intent is an important aspect of determining the user's preference. Hence, the method outlined in this section should be consider for later prototype iterations.

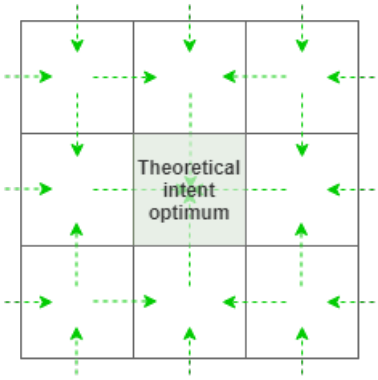


Figure 6.5: Illustration of how user intent can bias the state-action values to favor theoretical best state for a given user intent. Made with draw.io online tool.

Figure 6.5 illustrates how state-action values can be biased such to favor a theoretical optimum state given a specific user intent. The green arrows indicate that actions leading towards the theoretical optimum are positively biased such that the algorithm is given an incentive to seek that state. Actions close the theoretical optimum state are biased more than actions in states further away (indicated by the shorter green arrows on Figure 6.5). However, two problem might occur which can affect the state-action value bias: Firstly, the state-space is likely to have stored information from previous training sessions that might or might not agree with biasing. Secondly, the intent state is only a theoretical optimum and

might not reflect what the user actually wants to achieve.

The second issue can be solved by adding intent as a dimension just like the environments. However, the state-space scales poorly with added dimensions. Therefore, a simpler approach might be beneficial. Simply recording which state a training session terminates in for a given environment and intent might make it possible to make the "Theoretical intent optimum" less theoretical. When the same environment and intent is selected again, the "Theoretical intent optimum" becomes whatever state resulted from last time (or an average over a period of time). In cases where no information has been recorded for a given environment and intent pair, the theoretical optimum becomes whatever audiologists, or similar experts, have define as optimum.

The first issue regarding the state-space containing user information from earlier training session that might not favor the biasing can be solve in multiple ways. One way is to gently force the user toward the "Theoretical intent optimum" independently from what earlier training sessions have indicated. This would be achieved by adding a temporary bias to the state-space that would, independently from earlier training sessions, give positive state-action values to all actions leading to the theoretical optimum. The bias would be so slight that a negative user reward would counter the bias. Upon a few negative rewards, the temporary bias could be removed as a result of accepting that this is not what the user is trying to achieve. This method could be expanded such that an intent would bias the state-space less if the intent has been chosen often. The idea is that if a user often picks e.g. the "Conversation" intent, the state-space created by the user over time will also reflect this. In this case there is only a need to bias the state-space when the user picks an intent which is rarely chosen.

An alternative solution to the biasing issue would be to utilize the pairwise sound adjustment comparison (as specified by requirement **FR13**) wisely. Each sound profile would be based on different state-spaces. One adjustment could favor the "Theoretical intent optimum" whereas the other would simply reflect the state-space which has been trained by the user over time. Depending on which sound profile the user favors during a training session, this can be used to conclude whether the bias or unbiased state-space is favorable in the given situation.

This alternative solution does give a bit more freedom an does not force the user in a direction they do not like which is a waste of training session iterations. Hence, this is the favorable solution for this application. However, it is an additional feature that is not necessary to verify the idea of the prototype. **NFR7** has been classified as a "won't" requirement and the above-mentioned solutions will not be implemented in this prototype iteration.

Training session termination

The Analysis chapter presented multiple approaches to determining whether an optimal state has been found. It was concluded that global optimization is not necessarily needed and possibly not even feasible. This section presents a combination of approaches already discussed in the Analysis chapter in an effort to design a solution that can reliably ensure users the adjustments they like.

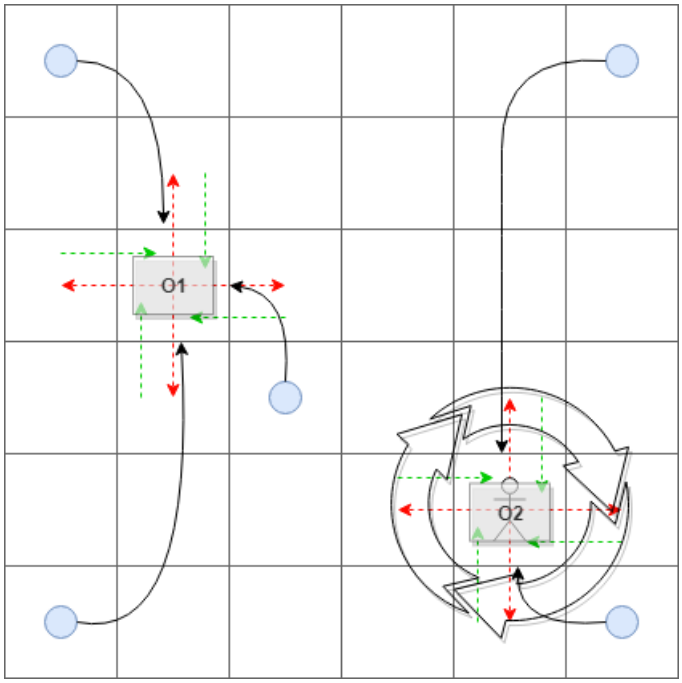


Figure 6.6: Illustration of how to determine that a user has reached a preferred state. Made with draw.io online tool.

In order to determine if a user has reached a preferred state, and that the training session should be terminated, multiple criteria must be fulfilled. Firstly, the states must be evaluated by multiple Temporal-Difference (TD) Prediction runs each starting in a randomly selected location. This is depicted by the blue dots in Figure 6.6. Each run returns a locally optimal state. In the figure, three evaluation runs find state **O1** and the two remaining runs find **O2**. It is now known that **O1** and **O2** are candidates for preferred preferences. However, the system cannot decide which one is most appropriate using this criterion alone. Both states might have been emphasized as an optimal state as a result of earlier training sessions. Therefore, the user's current location must be taken into account as well. Figure 6.6 shows that the user is currently in **O2**. Just because the user is currently in **O2** does not necessarily mean it is the user's preferred state. The user might just by coincidence be in this state which has been emphasized as an optimal state based on earlier training sessions. However, if the user's behavior is monitored, a conclusion can be made in relation to whether or not the training session should terminate. The circular arrows around the user and **O2** in Figure 6.6 illustrate that the user is circling around **O2** and continuously voting action leading into this state as preferable. Combining all this information, it can be concluded that a local optimum has been found, the user is currently in that optimum, and the user keeps emphasizing that optimum. As a result, it can be concluded that the user has found a state (which represents specific hearing aid adjustments) which the user likes. The training session can be terminated, and the adjustments applied to the hearing aid.

6.3.3 Combining CNN and Sarsa(λ)

The Convolutional Neural Network (CNN) architecture and Sarsa(λ) have been described individually. This section describes how they are interconnected and how they benefit from each other.

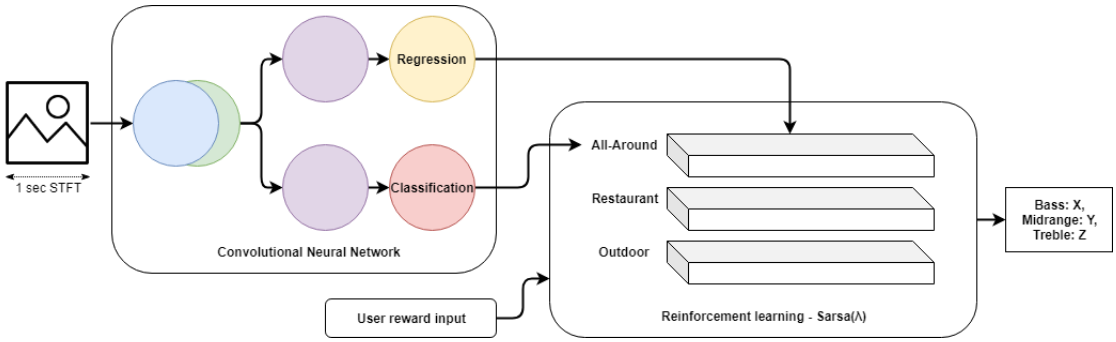


Figure 6.7: Illustration of how the CNN and Sarsa(λ) interconnects in order to provide personalized hearing aid adjustments. Made with draw.io online tool.

Figure 6.7 depicts how the CNN and Sarsa(λ) are linked. The CNN model provides two separate inputs to Sarsa(λ) with different entry points. Firstly, the CNN is responsible for labeling the user's environment as All-Around, Restaurant, or Outdoor. It is important that states encode something unique about the user's current situation such that e.g. being in a restaurant can be differentiated from walking in the forest. As an example, Figure 6.7 shows that the environment has been classified as "All-Around". This input points to a specific subset of the state-space used by Sarsa(λ). Hence, the CNN helps Sarsa(λ) limit the state-space making it easier to find the specific user preferred state.

Apart from labeling the environment, Figure 6.7 also shows that the regression branch of the CNN model provides an output that points to a specific location in the All-Around state-space. Using this information, Sarsa(λ) has a starting point for the training session. The CNN model is not tasked with predicting the preferred user state as such. The purpose of the CNN regression is to get a general sense of what the user tends to like. This tendency is indicated by the regression prediction and can then be fine-tuned through training sessions where the user interacts with Sarsa(λ).

From this explanation, it can be concluded that the CNN is used to ease to job of Sarsa(λ) and ensure that training sessions terminate quicker. The purpose is that the CNN should learn more complex relationships between environments and user preferences but on a general level. However, general predictions are not sufficient for a personalized hearing care solution. Therefore, Sarsa(λ) is responsible for keeping track of the latest preferences as well as user interactions to determine the latest preferences. The weakness of one approach is minimized by the other approach. This is done with the aim of achieving short training sessions and good personalized hearing aid adjustments for the users.

6.4 Solution sequence diagrams

This section presents two sequence diagrams in order to outline the order of actions and messages between the USense SoundLearner application and the Online training entity (OTE). The sequence diagrams also present the data that is carried in the communication.

6.4.1 Adjusting to the environment

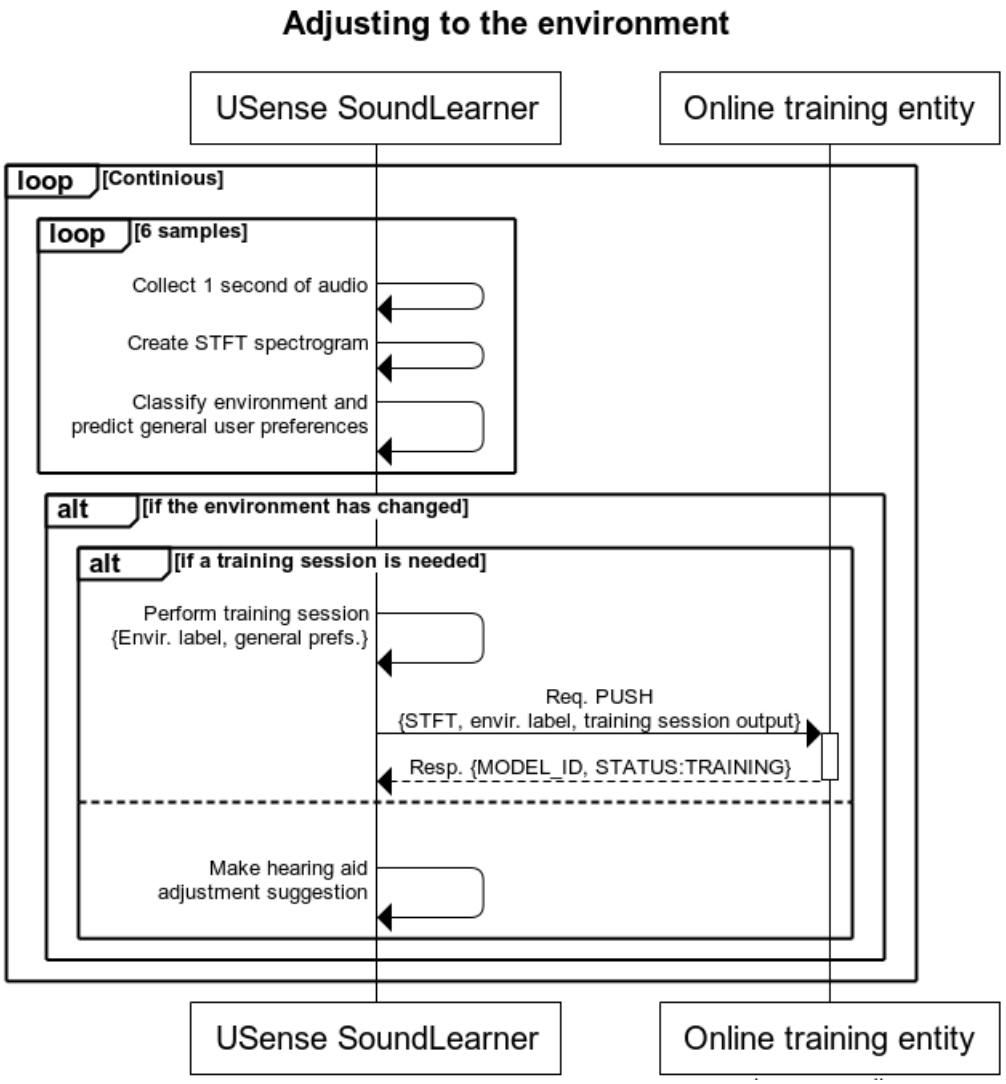


Figure 6.8: Sequence diagram presenting the process of observing the environment and adapting to it. Made with WebSequenceDiagrams online tool.

A sequence diagram describing the environment observation and the process of adapting to the environment is presented in Figure 6.8. The diagram depicts that most of the process is taken care of by the USense SoundLearner application which constantly monitors the environment and collects six audio samples of one second each. Each sample is represented as a Short-Time Fourier Transformation (STFT) spectrograms. This is used as input to the CNN model which predicts an environment label and general hearing aid adjustments. If there is no change in environment detected, the process continues monitoring. If it is determined that the environment has changed, action must be taken. USense SoundLearner either suggests new personalized hearing aid adjustment based on previous knowledge, or initiates a training session if new information is required. As a result of a successful training session, environment informa-

tion (STFT spectrograms and associated environment labels) is sent to the OTE as a package with the outcome of the training session (bass, midrange, and treble user preferences). The OTE starts training a new model and returns the status of the process with a model identifier. The identifier can be used by USense SoundLearner to later download the model when it is done training.

6.4.2 Receiving a new CNN model

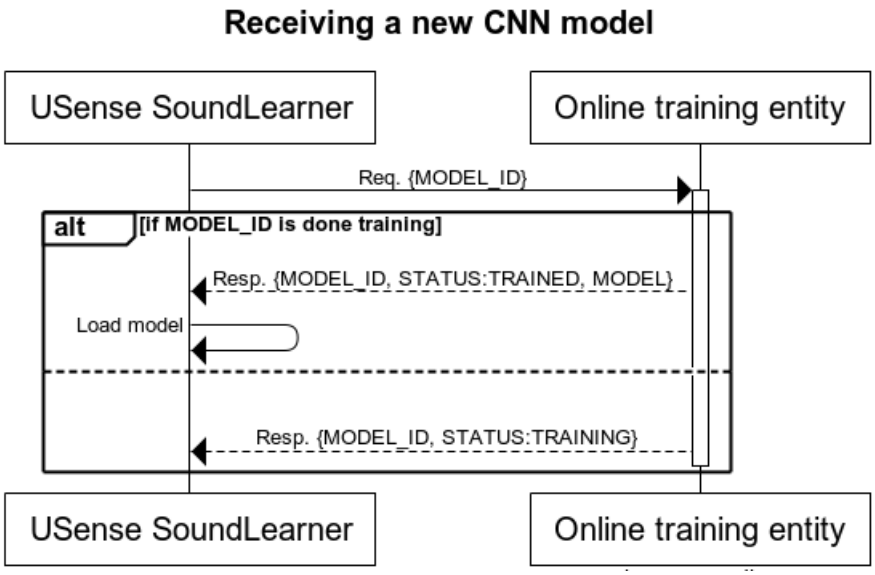


Figure 6.9: Sequence diagram describing the communication required to download a newly trained CNN model. Made with WebSequenceDiagrams online tool.

The process of downloading a CNN model from the Online training entity (OTE) is depicted in Figure 6.9. This process takes place when a model training has been initiated as a result of a successful training session. While the USense SoundLearner has internet connectivity it regularly checks with the OTE what the status of the training is. To get this status, the OTE requires a model identifier to be sent in the request. If the model is done training, the response contains the model along with the identifier and the status. USense SoundLearner can then store and load the new CNN model. If the model has not finished trained yet, the OTE simply responds with the model identifier and the status. In this case, USense SoundLearner has to check back at a later time.

6.5 Solution class diagrams

The section presents two class diagrams: One describing the design of the smartphone application implementation, and a second one describing the cloud entity. Like mentioned earlier, cloud simply refers to computational resources that can be reached over the internet and are sufficient to train the required models.

6.5.1 Smartphone application

The USense SoundLearner smartphone application is the most comprehensive part of the complete solution. This section describes each of the classes that needs to be implemented. This design is specific to the prototype and might need adjustment or additional components added to represent an MVP or full-scale solution.

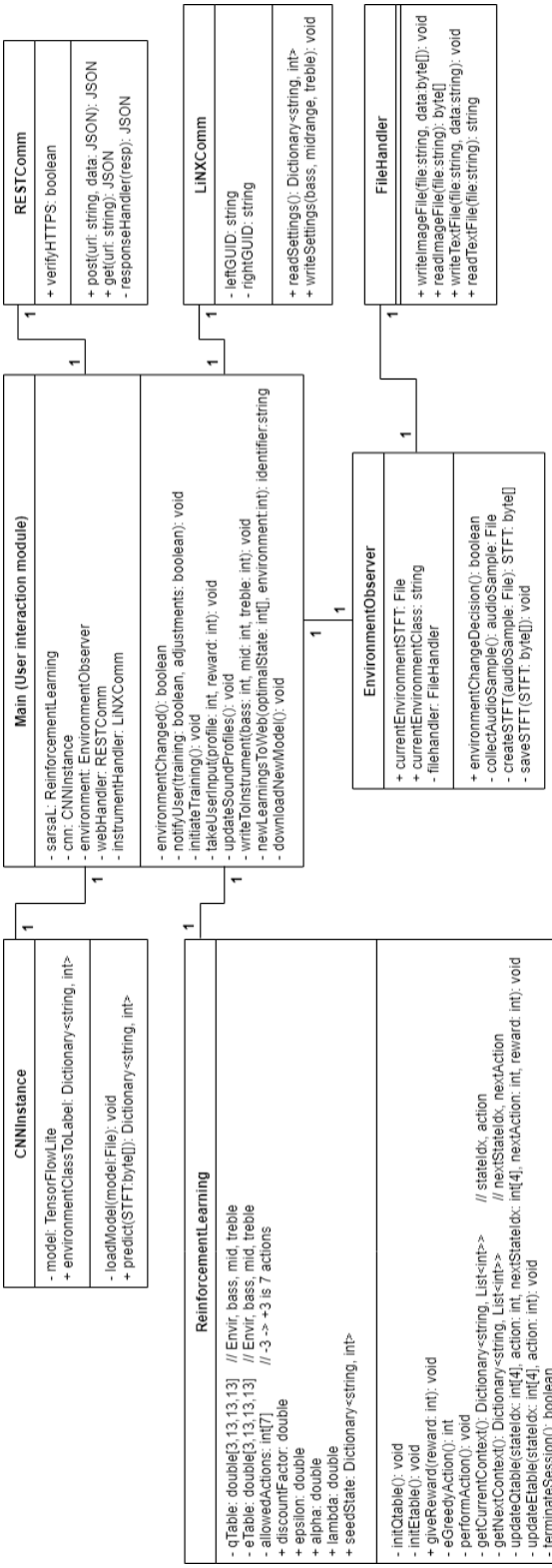


Figure 6.10: USense SoundLearner smartphone application class diagram. Made with draw.io online tool.

Figure 6.10 depicts the classes and functionality that must be implemented for the smartphone application. The following list outlines the responsibility of each class:

- EnvironmentObserver is used to monitor the environment and ensuring that changes does not go unnoticed. Furthermore, it is responsible for collecting audio samples and converting them to Short-Time Fourier Transformation (STFT) spectrograms. If utilizes the FileHandler for saving spectrograms.
- CNNInstance is running the pretrained model from the cloud. This makes it possible to make environment label predictions as well as user preference predictions.
- ReinforcementLearning describes what is needed to implement Sarsa(λ). The Main class provides the code needed for the user to interact with Sarsa(λ), however, all the learning takes place in the ReinforcementLearning class. This class is also responsible for determining when an optimal state has been found such that this adjustment can be applied to the hearing aids.
- For saving information locally on the user's smartphone, the FileHandler class is used.
- Communication with the hearing aids is done through the LiNXComm class. For this prototype, the communication is exclusively focused on applying new fine-tuning parameters.
- For supplying the cloud entity with new training data, RESTComm needs to be implemented.. Furthermore, this class is also used to download pretrained CNN models.
- To bind all these different classes and provide the user with an interface, the Main (User interaction module) class is used. This class ensures that the user can run training sessions, that the resulting data is sent to the cloud for training, that new models are downloaded, and that the user is notified about suggested adjustments.

6.5.2 Cloud instance

The prototype of this project utilizes a cloud instance to train new CNN models. This instance must implement functionality that allows training data and trained models to be exchanged. This section presents the design of this functionality.

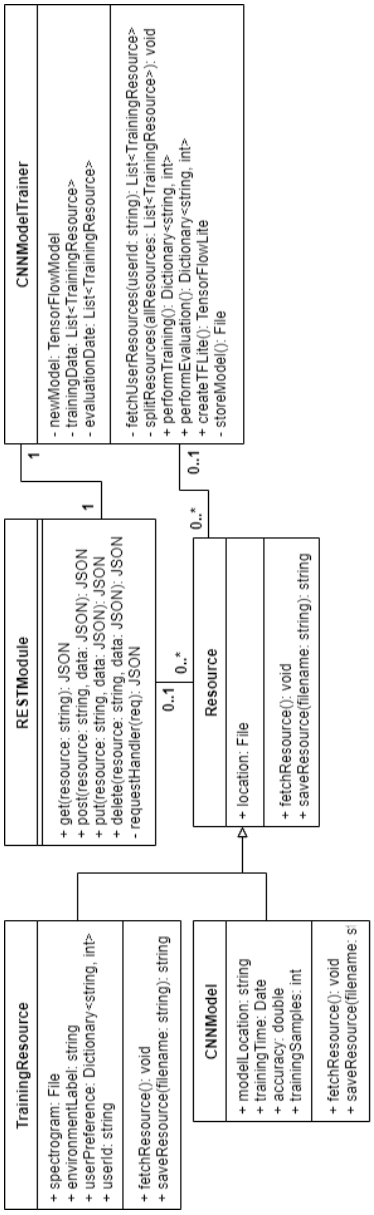


Figure 6.11: USense SoundLearner cloud entity class diagram. Made with draw.io online tool.

Figure 6.11 depicts a class diagram describing the functionality which the cloud instance must implement. It is all centered around the RESTModule which ensure that the cloud instance can respond to incoming requests. This module both needs to save incoming data which is used for training and provide pretrained models. Storing new training samples is taken care of by the TrainingResource class. Both TrainingResource and CNNModel are subclasses of the more generic Resource class.

The process of training new CNN models is taken care of by the CNNModelTrainer which creates TensorFlow models. However, such models are not compatible with the USense SoundLearner smartphone application. Hence, functionality to convert such model into TensorFlow Lite models must be implemented. When a new CNN model has been trained, it becomes available for download.

Chapter 7

Implementation

This chapter describes the implementation of the prototype. The first section describes the high-level view and present the different entities that are needed for the solution to function. This is followed by a description of selected key processes which are discussed in more details. The chapter is concluded with a review of what requirements are fulfilled by the implementation.

7.1 High-level system description

The architecture of the implemented prototype differs from the designed architecture presented in Figure 6.2 (Section 6.2, page 81). The change in architecture is due to a change in connections between some components in order to enable code reuse. Furthermore, the prototype implementation is subject to resource limitations. As a result, the prototype needs additional entities in order to provide an end-to-end solution. This section presents the prototype architecture and describes each entity in order to outline functionality and responsibility.

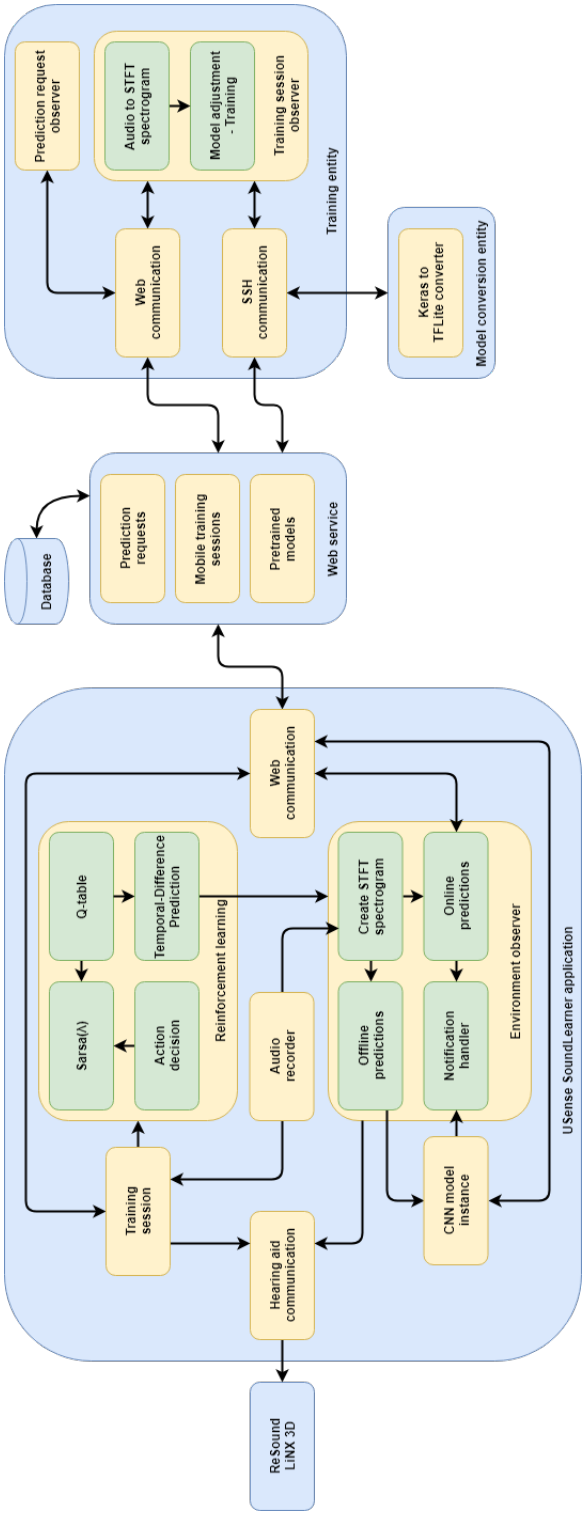


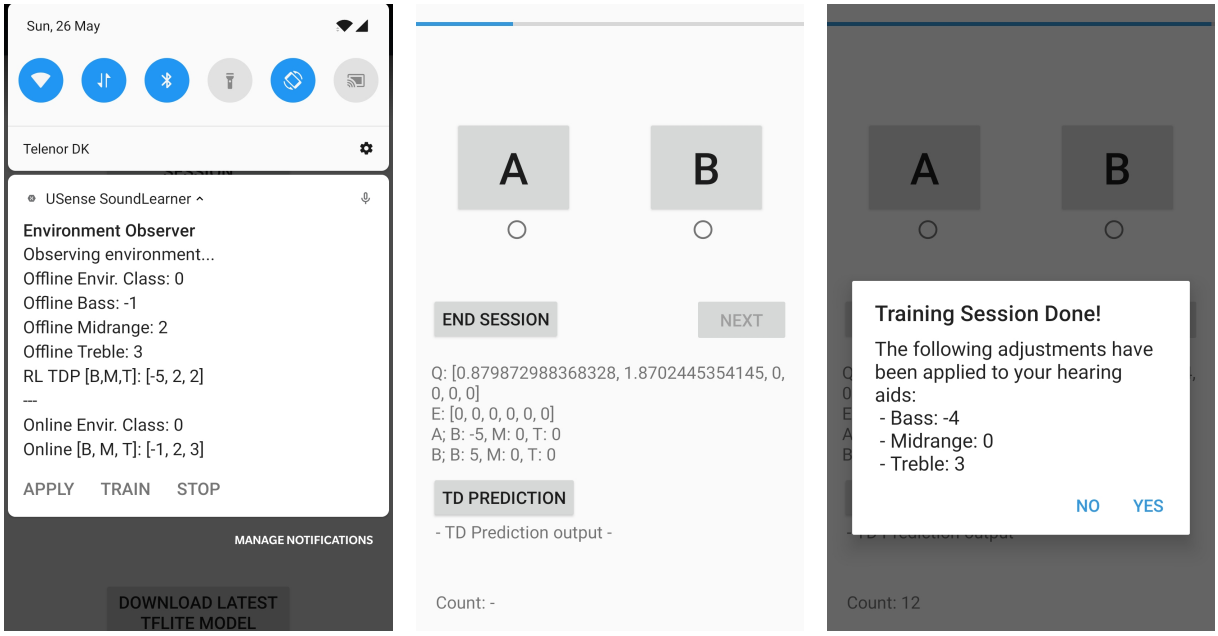
Figure 7.1: High-level architecture of the implemented prototype. Made with draw.io online tool.

Figure 7.1 depicts the high-level architecture of the implemented prototype. The coloring indicates encapsulation: Blue is an outer container and describes an entity. A blue container can hold yellow modules which provide a specific functionality which can either be exposed outside the entity or simply used within the entity to accomplish a greater task. Yellow modules can hold green components which define a specific routine or element of a module. USense SoundLearner application is implemented in C#, the Web service is PHP, and the Training entity and Model conversion entity are implemented in Python. Each entity is described in the following paragraphs.

USense SoundLearner application

The USense SoundLearner application is what the user is directly interacting with. This is a smartphone application providing the user with hearing aid adjustments that are appropriate for the current environment. This functionality is provided by the Environment observer module which interacts with the Audio recorder module. As audio is recorded, it is turned into Short-Time Fourier Transformation (STFT) spectrograms which can then either be used for offline or online predictions. If offline predictions are used, the spectrogram is provided to the Convolutional Neural Network (CNN) model entity which performs and analysis and returns an environment classification with suggestions of how the hearing aids should be adjusted. If the online predictions are used, the spectrogram is sent to the Web service where it is picked up and analyzed by the Training entity. Hereafter, the prediction can be received and used by the smartphone application. How online predictions are accomplished are described more in details using a sequence diagram in Figure 7.9 (Section 7.2.1, page 109).

The outcome of the CNN model's environment and hearing aid adjustment prediction is used as input to the Temporal-Difference Prediction routine. This is then supposed to recall the user's exact preferences for the situation. The information is compiled into a notification which is displayed to the user.



(a) (b) (c)

Figure 7.2: Screenshot of the implemented prototype showing the notification (a), the training session activity (b), and the training session termination pop-up (c).

A screenshot of this notification is seen in Figure 7.2a. It contains information about what the prototype is currently doing as well as the result of the latest environment analysis. Currently, the notification shows that the environment is being observed (an audio recording is being made) this is then followed by the creation of a spectrogram used for environment analysis. It can be seen that the latest predicted environment class is "0" which equates to All-Around. Furthermore, it can be seen that the CNN model has suggested to apply the following adjustments to the hearing aids: {Bass: -1 dB, Midrange: +2 dB, Treble: +3 dB}. The Temporal-Difference Prediction routine has taken this as input and suggested {Bass: -5 dB, Midrange: +2 dB, Treble: +2 dB} as the user's preferred adjustments. So far, the analysis is all done locally on the smartphone and does not require internet connection. However, the implementation has been made such that predictions can be performed on the Training entity as well. For this prototype, it is not strictly needed as the Training entity and USense SoundLearner application use the same CNN model. As a result, the outcome of an offline prediction is the same as an online prediction. However, this feature does provide some flexibility. If a smartphone is not capable of running the offline analysis, the solution can still be used, however, then requires internet connectivity.

The notification is both used for information purposes but also because Google has put restrictions on background services as of Android 8 [37]. The prototype of this project is targeting Android 9. Therefore, the Environment observer has been implemented as a foreground service which displays a notification, but does not have the same limitations.

Apart from showing information, the notification also makes it possible to apply the adjustment suggestions to the hearing aids (the ReSound LiNX 3D entity in Figure 7.1) by clicking the "APPLY" button. If the user no longer wants the environment observer to be running, it can be stopped using

the "STOP" button. The "TRAIN" button launches the Training session activity which, as depicted in Figure 7.1, utilizes the reinforcement learning implementation including the Sarsa(λ) and the Temporal-Difference Prediction components. These two components both use the Q-table (which contains all state-action values for the entire state-space) which is adjusted by the user to reflect current preferences. The Q-table is persisted such that preferences are saved.

The Training session activity looks as depicted in Figure 7.2b. Like Widex SoundSense Learn, it provides the user with an "A" and a "B" button. Each button represents a sound profile i.e. a specific bass, midrange, and treble adjustment. Clicking the buttons applies the adjustments to the hearing aids. When the user has chosen the preferred one, the "NEXT" button is enabled bring the user to the next comparison. At the top is a blue progress bar which indicates how far the user is in the process. However, it is not intelligently coupled to the learning process in this prototype implementation. The user can at any time end the training by clicking "END SESSION".

Ending a session can either be triggered by the user, or programmatically when the application thinks it has determined the user's preference. When a session is terminated, the pop-up shown in Figure 7.2c is displayed. It simply tells the user what it thinks the user's preference is and applies it to the hearing aids. The user can either decline and continue the training, or accept and end the training.

Below the "NEXT" and "END SESSION" buttons some information used during development is shown. The information shows the Q-table and eligibility trace values for the current state. Below this are the sound profile details for "A" and "B". This is followed with information about the Temporal-Difference Prediction routine and the number of comparisons made.

If web communication has been enabled, audio is recorded during the training and uploaded to the Web service along with the determined preferences. The Training entity can then collect this data and start training a custom CNN model. The adjusted CNN model is then made available for download afterwards. This is described in more details by a sequence diagram in Figure 7.13 (Section 7.2.2, page 114).

Figure 7.3: The main activity of the prototype providing functionality to start the various features of the application.

Figure 7.3 is the main activity of the prototype and simply acts as glue between the different features. The layout makes it possible to start a training session from a desired starting point (environment, bass, midrange, and treble values), forget all learnings by clearing the Q-table, starting the Environment observer, and downloading the latest adjusted CNN model from the Web service. Furthermore, a switch is added to manually toggle whether functionality interacting with the Web service should be active or not.

Web service

The Web service provides a communication interface between the USense SoundLearner application and the Training entity. Comparing with the Design chapter (Figure 6.2, Section 6.2, page 81), it can be found that the Web service is an additional entity that has been added. The design assumes that the Training entity and the Web service is one entity. However, due to resource limitation on the Web service, it is incapable of training a CNN model. Furthermore, the Training entity does not have the ability to expose services like the Web service. Hence, they have been separated into two entities.

The Web service provides functionality that makes it possible to request predictions, upload and download audio resources and user preferences as a result of a training session, and upload and download trained CNN models. The messages and information exchanged for each of these processes are described Section 7.2.

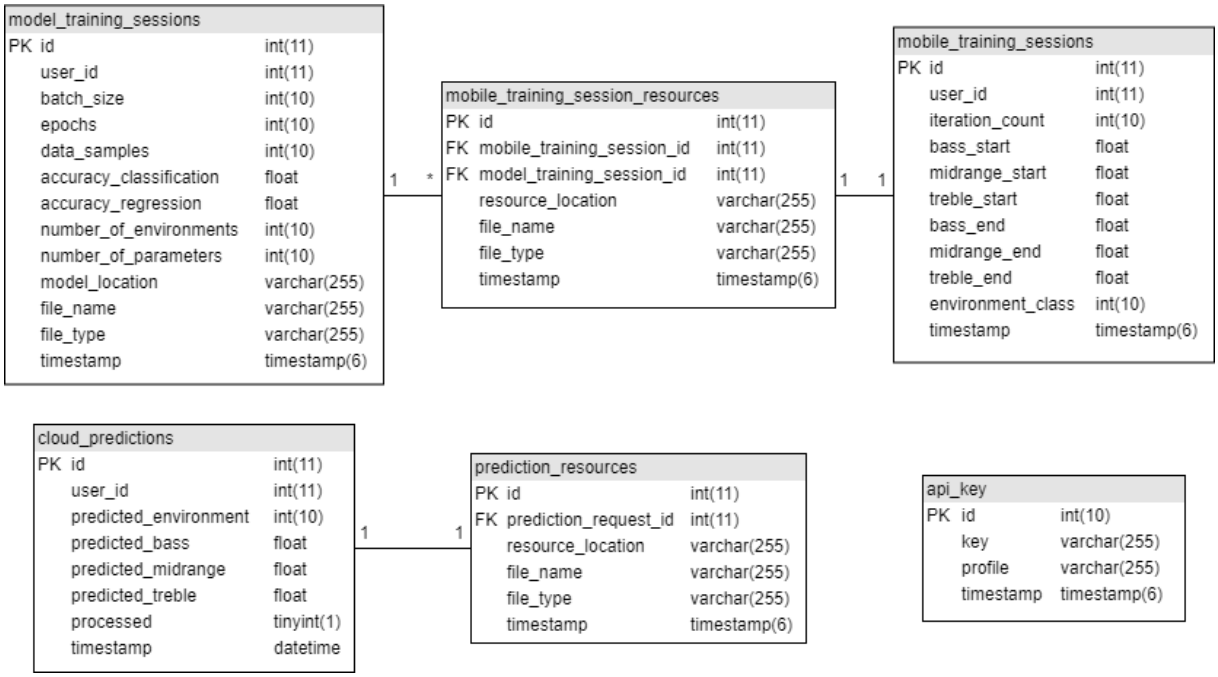


Figure 7.4: Layout of the database structure. Made with draw.io online tool.

The Web service entity severs as a storage place and is connected to a relational MySQL database to ensure structure. The database structure is depicted in Figure 7.4 and consists of six tables:

- model_training_session stores information related to CNN models. When the Training entity creates a new version of a CNN model, the model gets uploaded along with information such as how many data samples were used to create it, how many environment classes it can predict, where the model is stored etc. which is kept in this table.
- A model training session is based on a mobile training session. New mobile_training_session entries are created by the USense SoundLearner application. When the user finishes a training session, information related to the user's preference, how many iterations it took to find, what the environment is etc. is uploaded. This information is used to adjust the CNN model accordingly.
- Mobile training sessions produce an audio file which is required to perform a CNN model training session. mobile_training_session_resources is created to keep track of these audio files and associate them with model training session and mobile training sessions. The implementation only allows for one resource being upload for each mobile training session. However, multiple resources can be used to perform a model training session.
- cloud_predictions is used to request the Training entity to provide a prediction based on a representation of the current environment. USense SoundLearner causes a new entry to be created in this table. However, most of the information is related to the prediction outcome which is later populated by the Training entity.

– Predictions requests are based on an environment representation (Short-Time Fourier spectrograms) which is uploaded along with the request. prediction_request is used to associate these representations to a prediction request. The implementation only allows for one resource per request.

– api_key contains keys of which one must be provided in order to retrieve or publish resources. For a prototype, this approach is sufficient and ensures that not everyone can write information to the Web service or database.

Training entity

The Training entity is monitoring new uploads to the Web service. If either a prediction request is made, or the user finishes a training session in the USense SoundLearner application, the Training entity starts processing. The Training entity has an instance of the latest CNN model running. Upon detecting a prediction request, the environment representation (STFT spectrogram) uploaded by the smartphone application to the Web service is downloaded and processed by the Prediction request observer. If the user has performed a training session on the smartphone, an audio file and the user preferences are uploaded to the Web service. This information is downloaded by the Training session observer and the audio file is converted to a series of spectrograms which are then used to adjust the CNN model.

After creating the adjusted CNN model, a conversion must be performed in order to make it possible to run the model on a smartphone. The processed used for the conversion in this project is not supported by Windows. However, the Training entity for this prototype is a Lenovo P51 running Windows 10. Therefore, the adjusted CNN model is copied to the Model conversion entity. This entity is running a UNIX-based system. The SSH communication module is used to transfer the model to the Model conversion entity where the conversion takes place. The converted model is returned and now needs to be uploaded to the Web service. However, converted models are 100 megabytes. This exceeds the limit of 96 megabytes set by the hosting company [74]. To get around this issue, the model is moved to the Web service using SSH too.

When the model has been uploaded, it can be downloaded by the USense SoundLearner application to provide environment analysis customized to the user's preferences.

An initial CNN model must be created before all the above-mentioned customization can happen. The initial CNN model used for this prototype is trained with data collected during the prestudy [23]. A detailed description of the data collection equipment and process can be found in Appendix E of the prestudy [23]. For completeness, a summary of the environments and main sound characteristics is presented here:

– **Transportation – Car:** Medium traffic driving on paved roads at speeds between 40 and 70 km/h. No passengers were present, and the radio was turned off resulting in noise from just the car. This represents the All-Around environment.

– **Forest – Walking and sitting:** While walking, sounds of walking on gravel, grass, and forest-type soil was collected. Furthermore, the sound of a bike passing on the gravel was recorded. While sitting, sounds from bikes, people, and dogs passing on a gravel road was collected. The recording also included bird song, an airplane, and a noisy car in the distance. Forest is defined as an All-Around environment.

- **Transportation – Train and metro:** A mix of talk and no talk was recorded. However, train proved difficult for the CNN model to understand in the prestudy [23]. Therefore, it is also being left out in this project. However, if included, it would represent an All-Around environment.
- **Intersection:** A busy intersection with a mix of people, bikes, cars, and buses was observed at distance. This represents the All-Around environment.
- **Shopping street – Walking:** A walk down Strøget was done on a Saturday afternoon. These recordings include the sound of people walking, talking, and biking as well as street music, construction work, and traffic. It maps into the All-Around environment.
- **Train station:** These recording contains the sound of trains leaving and arriving and train doors opening and closing as well as the sound of other people and announcements over the speakers. The project defines it as an All-Around environment.
- **IBM canteen:** The prestudy [23] was done as an internship with IBM. Sound recordings for the Restaurant environment were done in their canteen and contained sounds of people talking right next to the recording setup as well as in the background. Furthermore, the soundscape consisted of cutlery, plates and glasses. This represent the Restaurant environment.
- **Aalborg University Copenhagen quayside:** This location provided recordings with long bursts of wind. The quayside is not to far from a large road which might have resulted in some traffic noise also being recorded. This recording represents the Outdoor environment.

7.2 Process descriptions

Both the environment observation as well as the training session are key processes for this prototype. This section describes the processes in more detail. The descriptions do not go into the technical details, but lists the steps that must be taken in order to provide the functionality.

7.2.1 Environment observation process

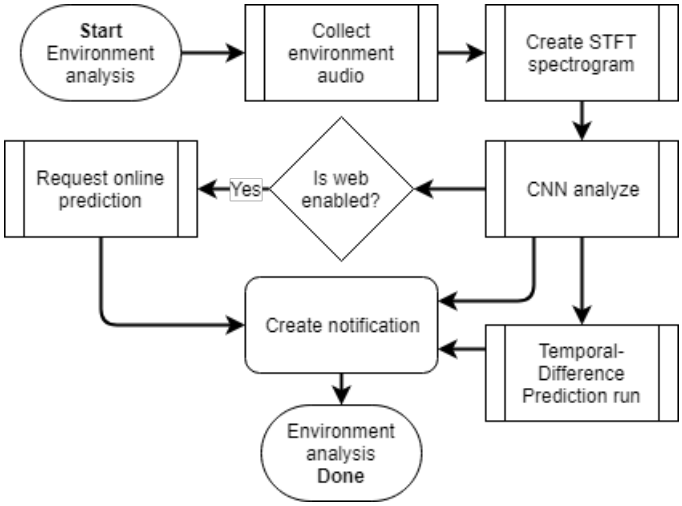


Figure 7.5: High-level flow of processes needed in order to provide the user with a hearing aid adjustment suggestion. Made with draw.io online tool.

Figure 7.5 describes the process of observing and analyzing the environment in order to suggest a new hearing aid adjustment to the user. This figure is rather high-level and is defined by a number of other sub-processes. First, audio of the environment must be collected. This can be turned into Short-Time Fourier Transformation (STFT) spectrograms which can be analyzed by the Convolutional Neural Network (CNN) model instance. This provides an offline prediction. If an online prediction request is desired, this is initiated by ensuring that web communication is enabled (ensure the switch shown in Figure 7.3 is enabled). The output of the offline prediction is used as input to the Temporal-Difference (TD) Prediction routine. Finally, the user notification is created allowing the user to apply the adjustments, start a new training session, or stop the analysis.

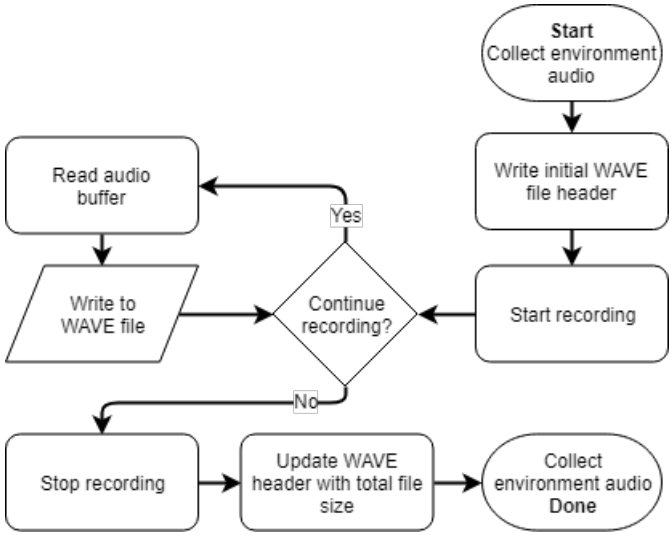


Figure 7.6: Depiction of the process of collection audio samples of the user's environment. Made with draw.io online tool.

The audio collection process is described by Figure 7.6. This process is used both by the Environment observer and by the training session process. However, the process has two different termination criteria. In case of the Environment observer, the process will stop recording after two seconds as the spectrograms are only one second long, and only one spectrogram is used to provide the prediction. In comparison, during a training session, the audio recording will continue until the session is terminated, or until the size of the audio file reaches 94.5 megabytes as the Web service hosting company has a file size limit of 96 megabytes [74]. The audio recording has a sample rate of 48 kHz and each sample is stored as two bytes. Hence, a total of just over 16 minutes of audio is allowed to be recorded for each training session. Apart from the fact that the hosting company does not allow for upload of larger files, it might also be undesirable to upload large files as it will consume the user's mobile subscription data plan if not connected to WiFi.

The process of recording audio start by writing an initial WAVE header. The audio used in this project is raw uncompressed samples to ensure that any compression does not degrade the prototypes ability to correctly analyze the environment. Collecting the samples is supported by Android, however, saving as a WAVE file is not. Therefore, a process for writing a WAVE header is implemented.

WAVE header		
Description	**Value**	**Comment**
Specifies Chunk Start	'RIFF'	
WAVE + Data Chunk Size	-	Bytes
Specifies WAVE Chunk Start	'WAVE'	
Specifies Format Chunk Start	'fmt'	
Size of Format Information	16	Bits
Format Tag	1	PCM data = 1
Channels	1	Mono = 1
Sample Rate	48000	Hz
Byte Rate	96000	Bits/second
Sample Size With Channels	2	Bytes
Sample Size	16	Bits
Specifies Data Chunk Start	'data'	
Data Chunk Size	-	Bytes

Figure 7.7: Description and values of the fields that make up the WAVE header of the audio files created by the prototype [50, 62]. Made with draw.io online tool.

Figure 7.7 depicts the structure of the WAVE header. It can be seen that it requires the length of the file to be set (see "WAVE + Data Chunk Size" and "Data Chunk Size" fields). As the audio is not recorded yet, this value is unknown for now. When the initial header is written, the recording starts. The samples are collected and appended to the file as they come in. This process continues until one of the termination conditions is met. The recording is stopped, and it is now clear how large the data is. The WAVE header file size fields are updated, and the audio collection process is finished.

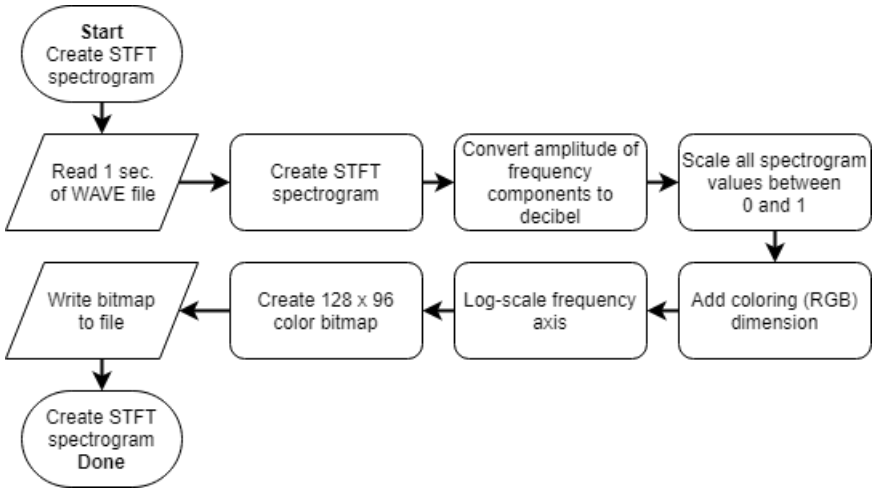


Figure 7.8: Illustration of the steps required in order to create the STFT spectrograms used for environment analysis. Made with draw.io online tool.

After the audio has been collected, it must be turned into a spectrogram which is the type of envi-

ronment representation the CNN model expects. The process described in Figure 7.6 collect two seconds of audio. The spectrogram creation process described in Figure 7.8 extracts one second of that audio with half a second of margin on either side. This margin might not be strictly needed, however, early version of the audio collection process had issues with clipping sound when the microphone was turned on. Having a bit of margin solves that issue. After reading the raw audio data, this is input to the STFT spectrogram creation process.

The prestudy [23] used LibROSA [61], a Python library, to generate spectrograms with good results. As the USense SoundLearner application is implemented in C# for an Android smartphone, the Python library is not available. The project decided to put a lot of effort into reimplementing the behavior of LibROSA in C#. LibROSA depends on other Python libraries: NumPy [68], SciPy [85], and Matplotlib [43]. Selected functionality from all the mentioned Python libraries is reimplemented in C# in an effort to recreate the same STFT spectrograms. The time and effort needed to do so is high, but is needed in order to ensure similar spectrograms for CNN model training and prediction.

When the audio samples are read from the file, a windowing function is prepared. The windowing function limits spectral leakage [83] creating less noise in the final spectrogram. The specific function used in this project is the Hanning window [83] which is chosen as it is the default used in the LibROSA library. The audio samples are then sliced into the desired window size, and the prepared window function is applied. Hereafter, the project is utilizing the Accord.NET framework [1] to perform a Fast Fourier Transformation on each of the slices.

This results in a basic spectrogram, and the process outlined in Figure 7.8 continues. The energy of the frequency components in the spectrogram are converted to the decibel scale and then normalized to fit between 0 and 1. This normalization makes it possible to use the Matplotlib color map named "magma". This create a spectrogram that ranges from deep purple to very bright orange. This color map is chosen as it gives a good contrast in the spectrograms. Furthermore, it is the default color map used by LibROSA.

After coloring the, the spectrogram gets logarithmically scaled on the vertical axis. The vertical axis is representing the range of frequencies. By scaling the axis logarithmically, more emphasis is put on the low frequencies. This concludes the process of creating the spectrogram. It is then scaled to fit the image width and height expected by the CNN model and saved as an image file.

This image is fed to the CNN model for analysis which results in an environment classification and a hearing aid adjustment suggestion. Both the Convolutional Neural Network and the state-space design for the Sarsa(λ) algorithm has been implemented exactly as specified in the Design chapter (Figure 6.3, page 83 and Figure 6.4, page 84 respectively). The technical configurations of the Convolutional Neural Network can be found in Appendix F.3.

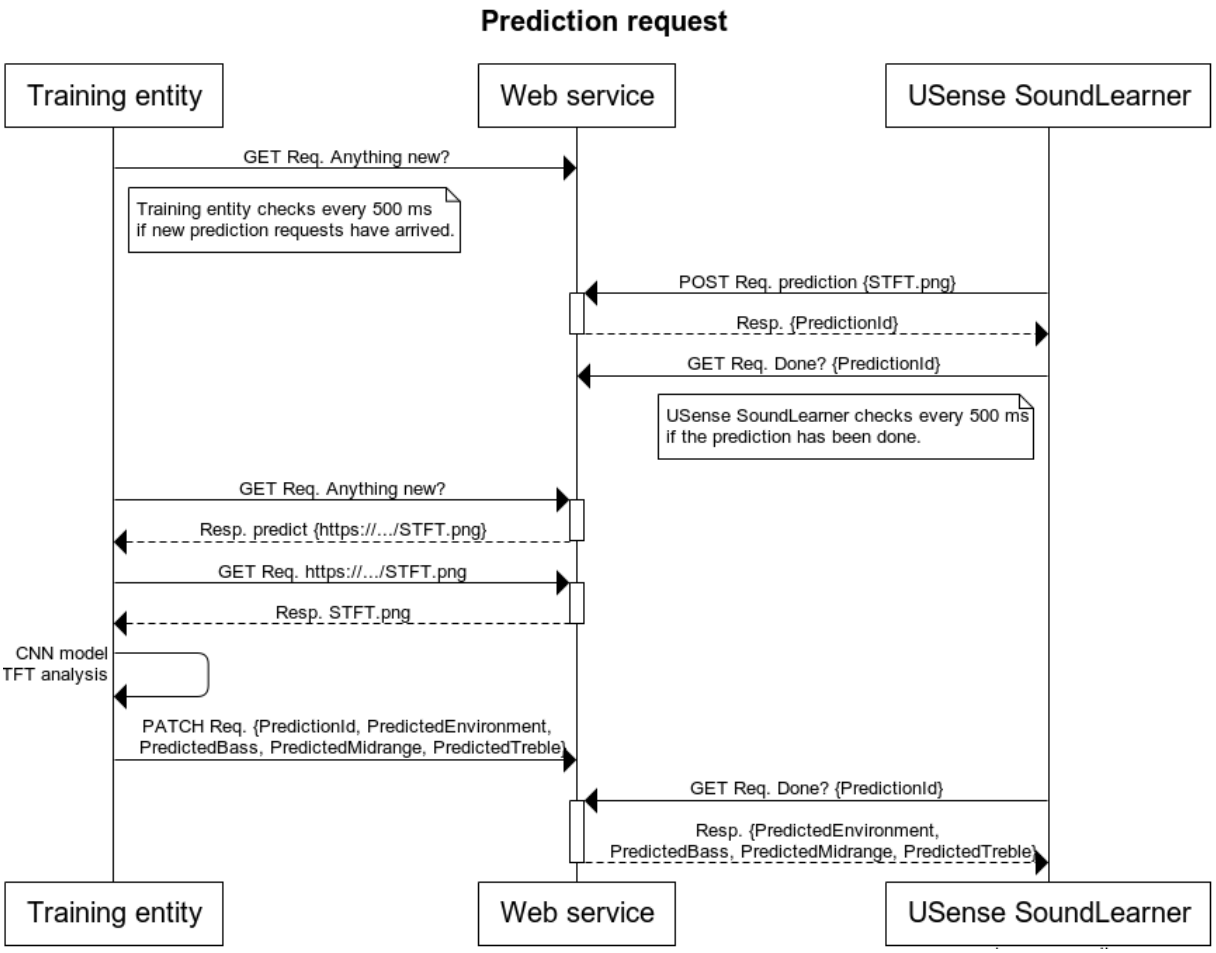


Figure 7.9: The sequence of messages needed to create an online prediction request and receive the answer. Made with WebSequenceDiagrams online tool.

If web actions are enabled (by enabling the blue switch depicted in Figure 7.3), the spectrogram used for the offline prediction is uploaded to the Web service and a new prediction request database entry is created. The Training entity is responsible for detecting this and processing the request. The messages needed to complete the process are shown in Figure 7.9. Figures describing this as process flows instead of a sequence diagram can be found in Appendix F.1.

Initially, the Training entity repeatedly asks the Web service whether a new prediction request has arrived with 500 milliseconds of delay between requests. At some point, the USense SoundLearner application creates the prediction requests. The Web service responds with an identifier which can later be used to retrieve the prediction. The smartphone application now starts to check if the prediction request has been processed by continuously sending requests to the Web service with 500 milliseconds of delay in between (waiting more than 10 seconds total will cause the prediction request to time out). The Training entity picks up that a new prediction request has been made and downloads the associated spectrogram. The spectrogram is fed into the CNN model instance which is running on the Training entity. This analysis results in an environment class and bass, midrange, and treble values being predicted just like in an offline prediction performed by the USense SoundLearner application. The newly predicted

values are used to update the prediction request database entry on the Web service. The prediction request has now been processed. Next time the smartphone application requests the result, it receives the predictions that have been uploaded to the database.

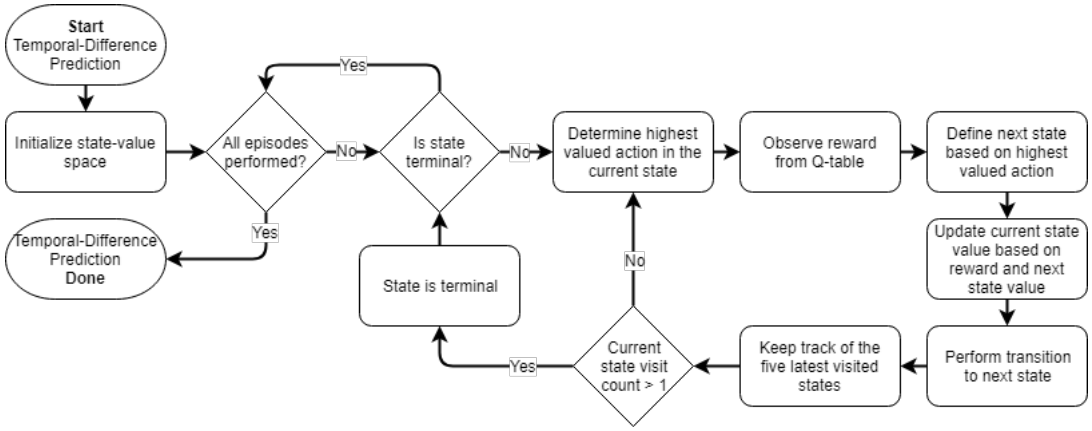


Figure 7.10: The Temporal-Difference Prediction process responsible for recalling the user's latest preference given an offline prediction result as input. Made with draw.io online tool.

Temporal-Difference (TD) Prediction is a specific method used to evaluate policies within reinforcement learning [94]. This project uses it as a method for recalling the user's preferences which have been learned during USense SoundLearner training sessions. Figure 7.10 describes the process of recalling the preferences. The TD Prediction process is used both as part of the environment observation process and during the training session. However, the method of using it differs. The environment observation process uses this routine only once and uses the offline spectrogram analysis as a starting point (not depicted in Figure 7.10). For the training session, five of these routines are compared and the starting points are randomized in the state-space. All the details on the training session process follows in Section 7.2.2.

Initially, an empty state-value space is created, and the routine enters two loops. The inner loop tests if the evaluation has reached a terminal state, and the outer loop ensure that this is done multiple times to support that this is the optimal terminal state. In the inner loop, the initial state is taken as a starting point, and it is evaluated which action should be taken (actions are plus or minus one decibel of adjustment to either bass, midrange, or treble) in order to reach the highest reward. When the action has been chosen, the reward is observed from the Q-table (which the user has provided input to during training sessions), and the next state is defined. The value of the current state in the state-value space is then updated based on this information followed by a transition to the next state. If this next state has been visited recently (the prototype remembers the last five states), it is assumed that the agent circles around this state and an approximate optimum has been found. If the state has not been visited recently, the process of determining the best action, updating state values, and making state transitions until a terminal state is found continues. Finding the terminal state exits the inner loop. The inner loop is started over again until the terminal stats has been emphasized a sufficient number of times (50 is the limit set in the prototype implementation) to call it an optimal state. When both the inner and outer loop finish, the optimal state is returned as the user's preference given the original input.

The information from the offline prediction and the TD Prediction routine as well as the online pre-

diction (if performed) goes into a notification which appears on the user's smartphone. This notification then allows the user to apply the suggested hearing aid settings, start a new training session in order to correct the predicted preference, or stop the environment observation. Due to some issue with the prototype's ability to recall user preferences (as described in the Testing chapter (Section 8.2, page 128)), the adjustments applied to the hearing aids are the outcome of the offline prediction, not the TD Prediction routine.

This concludes the environment observation process. All the subprocesses depicted in Figure 7.5 have now been presented. This process repeats itself only with a short delay in between runs such that optimal suggestions are provided continuously.

7.2.2 Training session process

Two types of training session can take place in the prototype. The user is training the reinforcement learning in the USense SoundLeaner application. This results in the Convolution Neural Network (CNN) model being adjusted by the Training entity. The main objective of this section is to outline the reinforcement learning training session. The details on the CNN model adjustments are described in Appendix F.2.

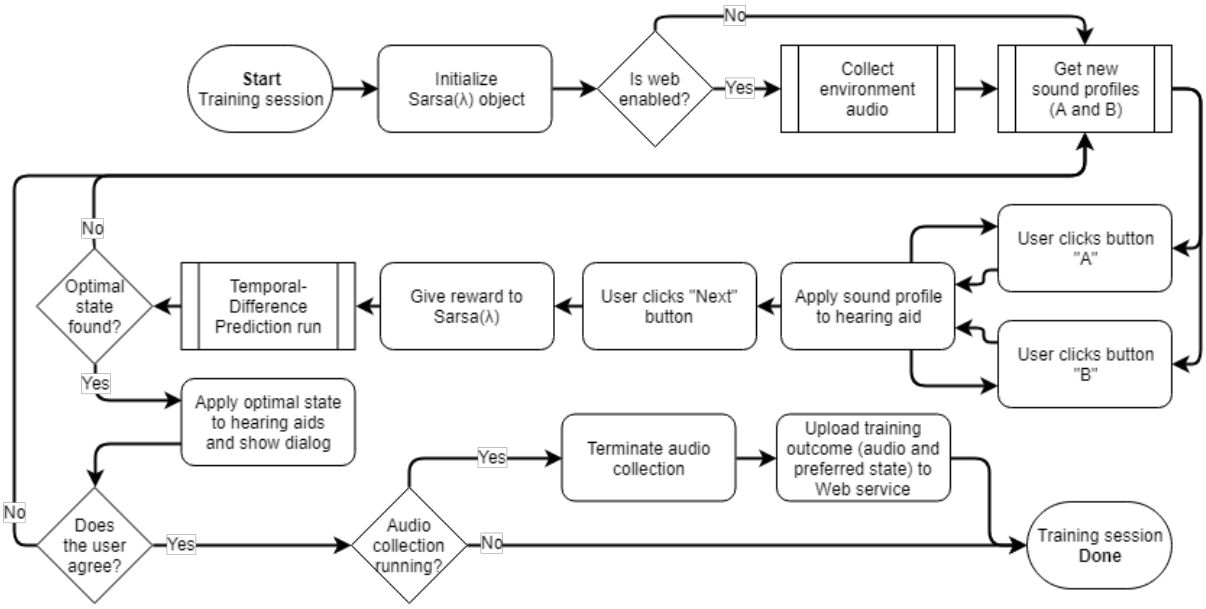


Figure 7.11: The process of running a training session in the USense SoundLearner application. A training session teaches the system about the user's preferences in the current environment. Made with draw.io online tool.

In order for the prototype to learn about the user's preferences, training sessions must be carried out by the user. The training session process is illustrated in Figure 7.2b.

Initially, a Sarsa(λ) instance is created as this is the reinforcement learning algorithm which the user interacts with. If web communication is enabled, the training process starts recording audio. Otherwise, it goes directly to present the user with an "A" and "B" sound profile. A sound profile is a hearing aid

adjustment (e.g. {Bass: -2 dB, Midrange: +2 dB, Treble: +2 dB}), but is denoted as a sound profile during training session to differentiate from the suggestions made by the environment observation process (Section 7.2.1).

Each sound profile is associated to a button for the pairwise comparison. If the user clicks one of the buttons, the sound profile associated with that button is applied to the hearing aids. The user can go back and forth between the two sound profiles in order to determine which one the user prefers. When the user has decided which is best, the user clicks the "Next" button to continue to the next sound profile comparison (see Figure 7.2b, Section 7.1, page 99 for the user interface). Based on which sound profile the user chose as the preferred one, a reward is given to Sarsa(λ). A positive reward is given to the preferred sound profile, and a negative reward to the other. The Temporal-Difference (TD) Prediction routine is then carried out to determine if the user has reached an optimal state. Five independent routines are performed each starting from a random place in the state-space. The TD Prediction routine is described in detail by Figure 7.10 (Section 7.2.1, page 110).

If an optimal state has not been found, two new sound profiles are generated and presented to the user. Figure 7.11 differs slightly from the actual implementation as the act of giving the reward is what generates the new sound profiles. However, to ease the illustration, it is shown as happening after the termination decision has been made.

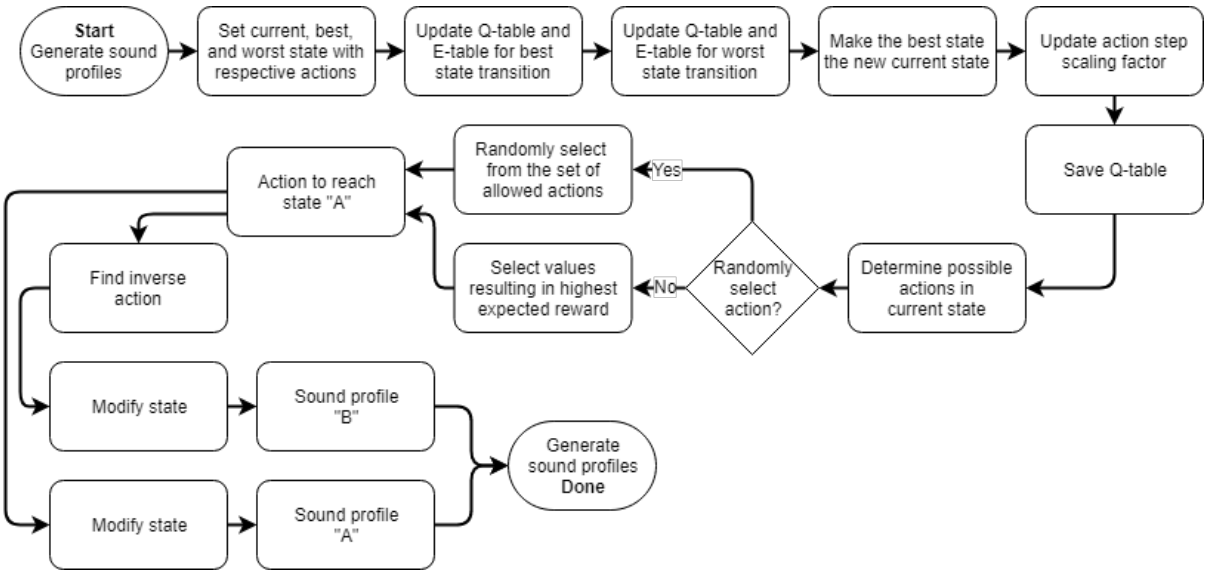


Figure 7.12: Illustration of what is required in order to generate two sound profiles for the pairwise comparison. Made with draw.io online tool.

The process of generating the sound profiles is depicted in Figure 7.12. Before new sound profiles can be generated, the best possible actions to take in the current state as well as in the states represented by the two sound profiles must be found. These actions are chosen such that the action in that specific state is expected to give the highest possible reward. The Q-table and the eligibility traces are then updated using this state and action information as well as the reward inputted based on the user. The Sarsa(λ) procedure then changes the current state such that it is represented by the user's preferred sound profile. This allows the algorithm to continue the state-space exploration in an intelligent manner.

An action scaling factor is then updated, and the current Q-table is saved such that learnings are not lost in case of training session termination. The action scaling factor is specific to this project and allows for bigger deviation between the two sound profiles in the beginning of the training session. This has been implemented in an effort to more quickly gain a view of what the user is trying to achieve to limit the number of states that should be tested.

All the preparation for generating sound profiles is now done. Till now it has been assumed that a new set of sound profiles are being generated based on an already ongoing training session. However, if the training session has just been initiated, and the first set of sound profiles needs to be generated, this preparation describe above is skipped. The following text describes the process of generating a new, or initial, set of sound profiles.

Based on the current state, a list of possible actions is generated. Usually, this list contains plus and minus for bass, midrange, and treble. However, if the current state is at the edge of the state-space, not all actions are allowed. As an example, the maximum value for any of the three parameters is +6 dB. Hence, if the current state is defined by: {Bass: +6 dB, Midrange: +4 dB, Treble: +2 dB}, the action of adding to base is not allowed limiting the choice of action to five instead of six possible actions.

When the list of possible actions has been generated, it must be decided which one to take. Upon starting a training session, a value ϵ is defined. This value defines how likely Sarsa(λ) is to pick randomly between the actions. This value can change during the training session. The state-space design section talks about an ϵ-greedy strategy (Section 6.3.2, page 85), and this is what it refers to. It is mentioned that the prototype implementation should reduce ϵ over time. However, it has been excluded due to poor performance during prototype implementation. Instead a static ϵ-value is used.

Returning to Figure 7.12, if the action has not been chosen randomly, it is chosen based on which one provides the highest expected reward. This is the action used to define sound profile "A". Another action has to be chosen for sound profile "B". The "inverse" action is chosen. As an example, if +1 dB midrange was chosen for "A", -1 dB midrange is chosen for "B". If the "inverse" action is not in the list of allowed actions, an action is picked randomly. Now that two actions have been found, they are each applied to the current state. The state does not change, but the algorithm observes which states would occur if these two actions where taken. The observed states are then applied as the new "A" and "B" sound profiles.

The process of comparing sound profiles, picking the best one, and generating a new pair goes on until the user is found to be in an optimal state. The decision of whether the user is in an optimal state or not, is defined by whether the user is in a state that has been visited recently (within the last five comparisons) and if more than one of the TD Prediction routines ends up in a state within ± 2 dB of the user's current state. If this is the case, the adjustment represented by the optimal state is applied to the hearing aids, and the user is asked if the prediction is acceptable or if the training session should continue. If the user decides to continue, the process of generating sound profiles etc. is started. If the user decides that the adjustments are acceptable, the session terminates. If audio has been collected during the training session, the recording is stopped, and the file uploaded to the Web service along with the outcome of the training session.

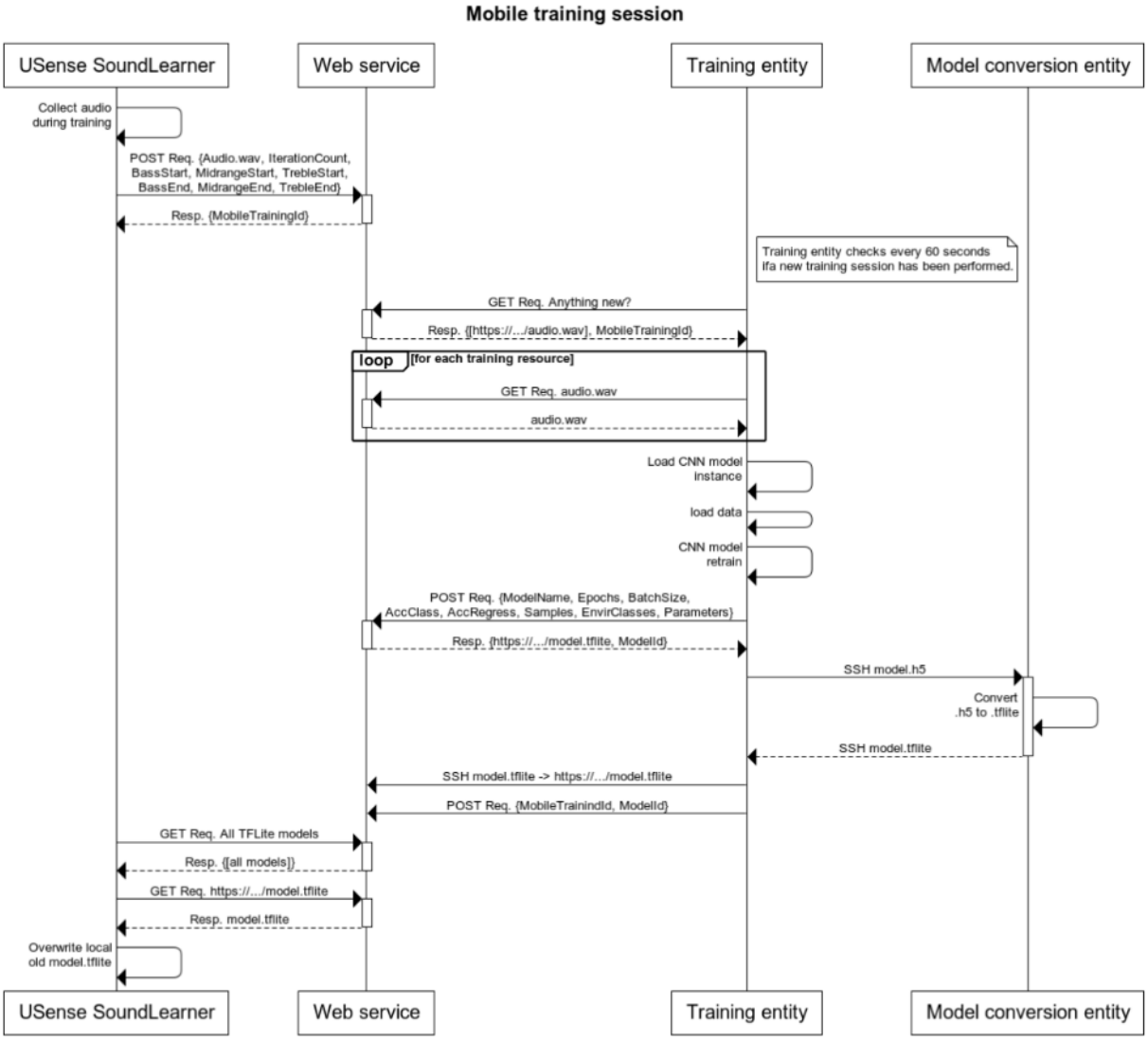


Figure 7.13: The sequence of messages exchanged between the USense SoundLearner application, Web service, Training entity, and Model conversion entity in order to update a CNN model with the latest user preferences. Made in WebSequenceDiagrams online tool.

Uploading an audio file as a result of a training session in the USense SoundLearner application, results in a CNN model adjustment. This is a training session taking place on the Training entity in order to adapt the CNN model to the user's latest preferences. The process is described with the sequence diagram in Figure 7.13. A process flow similar to the previous illustrations describing the Training entity's perspective more in detail can be found in Appendix F.2.

Figure 7.13 describes that audio is being collected by the USense SoundLearner as part of a training session. This is then uploaded to the Web service along with information about initial state for the training session, the resulting optimal state, and how many iterations it took to get there. The Training entity regularly (every 60 seconds) queries the Web service to see if new training data has arrived. When new mobile training session resources are detected, they are downloaded. More than one resources can be

used to adjust the CNN model. With the resources downloaded to the Training entity, the audio files are converted into STFT spectrograms which are divided into a training set (80% of the spectrograms) and an evaluation set (20% of the spectrograms). The spectrograms are loaded, prepared, and then used to adjust the current CNN model. When the CNN model is done training, information about the accuracy, number of supported parameters and environments, the number of training samples etc. is uploaded to the Web service. However, the model is not ready yet. It is currently a Keras model (Keras [15] is the name of the Python library used for creating the CNN models in this project) and needs to become a TensorFlow Lite [99] model (the sort of model supported by the smartphone). The conversion process is not supported by the Training entity. Therefore, the Keras model is transferred to the Model conversion entity over SSH. SSH is used as that is the communication channel available. After the conversion has taken place, the TensorFlow Lite model is transferred back to the Training entity which then uploads the TensorFlow Lite model to the Web service. Once again SSH is used because of a file size limitation by the hosting company of the Web service [74]. The model is moved to a location which has been agreed between the Training entity and the Web service. Lastly, the training resources on the Web service is associated with the model to show that they have been used.

A new and adjusted CNN model is now available for download. The USense SoundLearner application can download the new CNN model, overwrite the old one, and rely on new predictions that take the user's most recent preferences into account.

This concludes the process which takes place every time the user decides to start a training session in the USense SoundLearner application.

7.3 Requirement evaluation

A number of requirements have been defined and prioritized in Table 5.9-5.14 (Section 5.4, pages 74-78). The prioritization has been done both for the prototype implementation and for an eventual MVP implementation. This section evaluates if the prototype prioritization is fulfilled by the implemented prototype.

The following requirements are prioritized as "Must" have for the prototype: **FR1**, **FR3**, **FR5**, **FR16**, **FR24**, **NFR1**, **NFR3**, **NFR6**, **NFR13**, **NFR14**. The implementation has been prepared to fulfill all these requirements. However, testing will have to show if they are actually fulfilled. This mainly refers to **FR1** which specifies that the solution must be capable of recalling user preferences in similar sound environments to where training sessions have taken place previously. Section 8.2 (page 128) describes a test where the prototype is brought into real environments which gives an indication whether the requirement is fulfilled or not. The same is valid for **FR5** which specifies that the Convolutional Neural Network (CNN) model must be capable of environment classification and suggesting hearing aid adjustments.

NFR1 dictates that the solution must consist of a smartphone application and a cloud instance. From the implementation description in this chapter, it is concluded that this requirement is fulfilled, but not as intended. The responsibility of the cloud instance is divided between the Web service, Training entity, and Model conversion entity. This is not as initially designed but provides CNN training capabilities in the cloud (as required by **FR3**) and works as a proof-of-concept. The remaining "Must" requirements listed are implemented as specified.

The "Should" have category consists of the majority of the requirements of which most are implemented. The following are not implemented: **FR17**, **FR21**, **FR27**, **NFR8**, **NFR17**, and **NFR18**. **FR17** specifies that the user must be notified and suggested new hearing aid adjustments when the environment changes. The prototype does not keep track of the environment state, it simply performs an analysis constantly showing a notification suggesting the optimal adjustments. **FR21** specifies that the user must be able to try the suggested adjustment before applying it. This is currently not supported and is also linked to why **NFR17** and **NFR18** are not implemented. These two requirements specify a rewards structure for accepted and rejected suggestions. However, the implemented notification does not allow for rejection, hence, this reward strategy does not make sense. **FR27** is concerned with automatically applying the hearing aid adjustment suggestion if allowed by the user. This would be allowed during an onboarding session which is not implemented. Hence, the permission cannot be given and the need for supporting requirement **FR27** disappears.

NFR8 specifies that the Temporal-Difference Prediction routine must be used to recall the user's preferred hearing aid adjustments in a specific environment. However, due to some poor testing results presented in Section 8.2 (page 128), it was decided to let the CNN model make the final suggestion for this prototype.

Requirement **FR9** and **FR11** are also implemented differently than specified. Instead of providing spectrograms for the Training entity to use for CNN model adjustments, audio files are provided. The process of creating a spectrogram is a lot of work for the smartphone. This is offloaded to the Training entity be giving it an audio file instead.

The entire solution is not made for supporting user specific models. The database does associate resources to a user, however, the Training entity does not keep track of multiple models in the current implementation. Therefore, **NFR19** is not fully implemented. However, GN Hearing also specified that user specific models might be too resource demanding and prioritized it as a "Won't" requirement for an MVP.

Lastly, **NFR5** is implemented with some modification. The implementation is allowed to adjust ± 5 dB for the first few iterations of a training session in the USense SoundLearner application. It is then scaled down to the ± 3 dB specified by **NFR5**.

The majority of the "Could" have requirements are not implemented. They mostly refer to refinement of the current prototype, ensures better usability, and more thorough environment analysis. With that said, **FR18** specifies that the user must be notified when new hearing aid adjustments are suggested. This is implemented due to very little effort required as a result of how the environment observation process is done.

Of the "Won't" have requirements, **NFR20** is implemented due to convenience. It specifies that the environment observation process must run as a background service. A foreground service is used in the prototype implementation, but the service is still running in the background from the user's perspective which qualifies it as a background service in this project. The remaining "Won't" requirements are not implemented as they mainly specify additional features which can be added when the basic functionality is working. Whether the basic functionality is working is tested and documented in Chapter 8.

Chapter 8

Testing

This chapter presents the results of testing the prototype documented in Chapter 7. The focus of the testing is on the prototype's ability to learn and predict a user's preferences. The following section presents accuracy testing in order to evaluate how good the solution is at recognizing environments and predicting hearing aid adjustments. This is followed by a test aiming to verify if the same behavior is observed in real environments. Finally, an assessment of the learnability of each component is presented. The test results are used as input to create the list of improvements presented in Chapter 9.

8.1 Prototype accuracy assessment

In order to provide the user with appropriate hearing aid adjustment suggestions, the prototype must analyze the current sound environment. The aim of this section is to evaluate whether the prototype is capable of correctly classifying the environments and providing the appropriate hearing aid adjustments.

The section only covers an accuracy assessment of the Convolutional Neural Network (CNN). Currently, meaningful accuracy testing cannot be performed on the reinforcement learning component as it is unreliable in its current configuration. This is further described in Section 8.2.

8.1.1 Accuracy - Convolutional Neural Network

The Convolutional Neural Network (CNN) is responsible for predicting the user's sound environment as well as a suggestion a suitable hearing aid adjustment. This input is fed into the reinforcement learning which then is supposed to make the final hearing aid adjustment prediction while taking the user's most recent preference into account. This section tests the CNN's ability to correctly classify an environment and predict the suitable adjustments.

The model used for this testing is based on seven environment classes consisting of a total of 12090 examples. Each example is a Short-Time Fourier Transformation (STFT) spectrogram and represents one second of audio of an environment. All the spectrograms are presented to the CNN a total of nine times in order for it to learn the mapping between input STFT spectrogram and output environment classification and adjustment suggestion. Nine is found to be the limit for this prototype before it starts showing signs of overfitting. With the hardware available for this project, the training process takes three

hours.

The environments used to train the model are: Walking down a busy shopping street, driving a car, walking and sitting in the forest, train station, road intersection, wind, and canteen. The environments are outline in more details in Section 7.1 (page 103).

Environment	**Training examples**	**Evaluation examples**	**Prediction examples**	**Total**
City walk	2052	386	126	2564
Driving (car)	1274	238	78	1590
Forest (sitting and walking)	2388	450	142	2980
Train station	708	132	42	882
Intersection	992	186	60	1238
Wind	1084	202	66	1352
Canteen	3592	674	222	4488
Total	*12090*	*2268*	*736*	*15094*

Table 8.1: Distribution of training, evaluation, and prediction examples of the seven environment classes.

Table 8.1 describes how the samples are distributed among the seven classes. Each environment sample set is split into training (80% of the total number of samples), evaluation (15% of the total number of samples), and prediction (5% of the total number of samples) sets. The training and evaluation sets are used during training of the CNN model whereas the prediction set is used to perform the following testing. Splitting the environment samples like this ensure that the machine learning model has never seen the testing data before. Testing on the training and evaluation data would only make it possible to conclude whether the model is capable of recognizing data it has seen before. Stronger conclusion in terms of the model's accuracy can be made when using unseen data.

The prototype has not been developed to handle seven environments. The focus has been on the more general environments: "All-Around", "Restaurant", and "Outdoor". However, it was found during the prestudy [23] that splitting the "All-Around" environment into sub-categories made for a more accurate machine learning model. The environments listed in Table 8.1 are denoted as the "low-level environments" in this project. City walk, Driving, Forest, Train station, and Intersection all map into the "All-Around" environment. Canteen maps into the "Restaurant" environment, and Wind maps into the "Outdoor" environment. This classification has been performed with Peder Jonathan Thyme from GN Hearing Audiology during the prestudy [23].

Environment	**Training examples**	**Evaluation examples**	**Prediction examples**	**Total**
All-Around	7414	1392	448	9254
Restaurant	3592	674	222	4488
Outdoor	1084	202	66	1352
Total	*12090*	*2268*	*736*	*15094*

Table 8.2: Distribution of training, evaluation, and prediction samples of the three general environment classes.

Table 8.2 outlines the distribution of samples in the more general environments (denoted as "high-level environments" in this project). The following testing is performed on both the low-level and the high-level environments. However, the low-level environment classification is not important as such. It merely gives and indication of what the model is capable of. The success criteria are correct high-level environment classification and hearing aid adjustment predictions.

Environment	**Bass**	**Midrange**	**Treble**
All-Around	0 dB	0 dB	0 dB
Restaurant	-2 dB	+3 dB	+5 dB
Outdoor	-5 dB	-5 dB	-5 dB

Table 8.3: Hearing adjustments the CNN model is trained to suggest in the three high-level environments.

The CNN model is trained to suggest hearing aid adjustments depending on the environment as listed in Table 8.3. The table assumes that the hearing aid user does not need any adjustments in general (the All-Around environment), has an intent to focus on speech when in a Restaurant environment, and wants to remove noise when in the Outdoor environment. Whether a user would prefer this to be default is unknown. However, this test is just to confirm that the CNN model is capable of learning how to suggest hearing aid adjustments.

According to the mapping between Table 8.1 and Table 8.2, Outdoor is synonymous to wind. It was clarified during the prestudy [23] by Peder Jonathan Thyme that All-Around, Restaurant, and Outdoor in the ReSound Smart 3D application does not correspond to a specific environment, but to a specific challenge. As an example, speech intelligibility in noisy environments is supposed to be handled by Restaurant. Furthermore, being outdoor does not require a user to select "Outdoor". Outdoor is for reducing the annoyance of wind in the hearing aid microphones. For this reason, the adjustment suggestions listed in Table 8.3 are assumed to be a good baseline.

The evaluation is two-fold and consists both of a multi-class classification problem and a regression problem. The classification accuracy is assessed using Average Accuracy, Macro Precision, Macro Recall, and Macro F1 Score as described in the methodology (Section 3.6, page 24). The regression problem is assessed using the Mean Absolute Error (MAE).

Classification accuracy

Confusion matrices are needed in order to calculate the Average Accuracy, Macro Precision, Macro Recall, and Macro F1 Score. These matrices have been left out from the main report for brevity but can be found in Appendix G.2. Average Accuracy describes the ratio of correctly classified examples (including both true positives and true negatives) to all classified examples (true positives, true negatives, false positives, and false negatives) [91]. Precision describes the ratio between correctly classified examples for a given class (true positives) and all examples that were classified to belong to the specific class (true positives and false positives) [91]. Recall is a measure for the ratio of correctly classified examples (true positives) and the number of examples that should have been recognized to belong to the specific class (true positives and false negatives) [91]. The F1 Score is a weighted measure that combines Precision and Recall into a single measure in an effort to describe accuracy without including true negatives. The weighting denotes the order of the F Score. This project is using a weight of 1 (hence, F1 Score), but the weight can be adjusted if one wants to bias towards Precision or Recall [91]. The measure exists in both a micro-variation and a macro-variation. The micro-variations have a tendency to favor larger classes whereas the macro-variations do not have such bias [91]. Hence, the macro-variations of the mentioned measure are used in this project to avoid any biasing. Macro deviates from the ordinary measures by averaging the results over all possible classes (see Equation G.1 - G.4 in Appendix G.1).

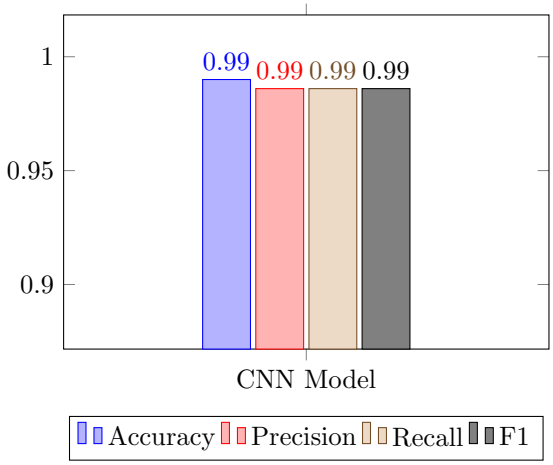


Figure 8.1: The Average Accuracy, Macro Precision, Macro Recall, and Macro F1 Score for the high-level environment predictions. The bars show similar accuracy (99%) but differ in height. This is due to rounding. Only two decimals are shown in the figure, but they differ on the third decimal.

Figure 8.1 depicts the accuracy of the CNN model when predicting the high-level environment classes (All-Around, Restaurant, and Outdoor). The blue bar is the Average Accuracy, the red bar is the Macro Precision, the brown bar is the Macro Recall, and the black bar is the Macro F1 Score. The figure shows that the model has a very high accuracy for predicting the high-level environment classes. The unit of measure on the y-axis is percent. It can be concluded that the CNN model correctly classifies the prediction examples in 99% of the cases for the high-level environments.

Environment	Correctly predicted	Actual
All-Around	445	448
Restaurant	214	222
Outdoor	66	66

Table 8.4: Number of correctly classified examples compared to the number of total samples in the high-level environment classes.

Table 8.4 shows that almost all examples of the high-level environment classes are correctly classified supporting the high accuracy value shown in Figure 8.1. All Outdoor examples are correctly classified meaning this class likely stands out as being unique. A few spectrograms are incorrectly classified for All-Around and Restaurant showing that these classes might be more difficult to predict. However, the number of misclassifications is minute. If it turns out to be a problem that some spectrograms are incorrectly classified, an average of multiple predictions can resolve the problem.

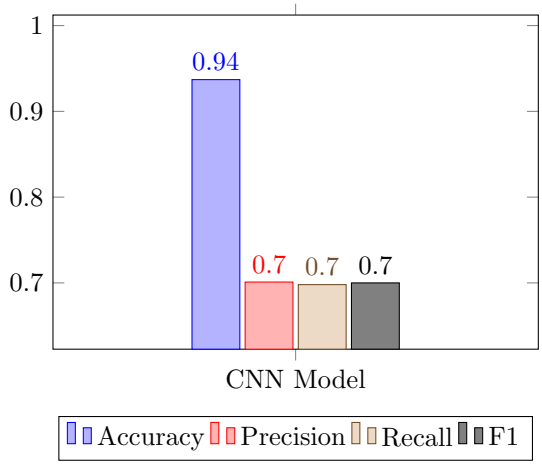


Figure 8.2: The Average Accuracy, Macro Precision, Macro Recall, and Macro F1 Score for the low-level environment predictions. Precision, Recall, F1 Score shows same accuracy (70%) but are not level. This is due to rounding.

For completeness, Figure 8.2 shows the accuracy results of the low-level environment classification. Like Figure 8.1, the blue bar is the Average Accuracy, the red one is Macro Precision, the brown one is Macro Recall, and the black bar is the Macro F1 Score. The y-axis is in percent showing a lower accuracy compared to the classification into the high-level environment classes. This is not surprising as it is more challenging for the CNN model to make more detailed classification. Figure 8.2 also show why the more complex F1 Score is favorable over the Average Accuracy measure. According to the Average Accuracy, 94% of the examples are correctly classified. However, according to the F1 Score, 70% are classified correctly. This is due to the fact that the Average Accuracy includes true negatives. It is easier for the CNN model to correctly predict that a given example does not fit in a certain class (true negative) compared to correctly predicting what class that example actually fits into (true positives). As the model can choose between seven class there is a high chance of true negatives even if making random guessing.

This underlines why the F1 Score is the most reliable measure. Hence, it can be concluded that the CNN model correctly classified the low-level environment examples (listed in Table 8.1) in 70% of the cases.

Environment	**Correctly predicted**	**Actual**
City walk	47	126
Driving (car)	57	78
Forest (sitting and walking)	142	142
Train station	5	42
Intersection	42	60
Wind	66	66
Canteen	214	222

Table 8.5: Number of correctly classified examples compared to number of total samples in the low-level environment classes.

Table 8.5 outlines how many examples where correctly classified for each of the low-level environment classes. Not surprisingly, Canteen and Wind are correctly predicted in almost all cases. Forest is also predicted correctly in all cases. Something must make this stand out and be unique. However, City Walk, Driving, Train station, and Intersection, have a lot of misclassifications. As a result, the overall accuracy is reduced and reflected in the F1 Score in Figure 8.2. However, the misclassification must be all within the same high-level class, otherwise the F1 Score in Figure 8.1 would suffer too.

As already mentioned, it is not necessary for the prototype to be able to classify in these details. However, it can be found that Forest stands out. Hence, it might be possible to provide the hearing aid user with specific adjustments for the Forest. Such adjustments could be trying to embrace bind song and wind whistling in the tree tops. Because the Forest class stands out, it can be reliably predicted and would not affect the user in other environments.

Regression accuracy

Apart from classifying the environment, the CNN model must also be able to suggest appropriate bass, midrange, and treble adjustments for the hearing aids. This is a regression problem and a different error measure must be used. The CNN model is optimized for Mean Squared Error (MSE) to punish larger errors more than small errors. However, for ease of interpretation, the error measure used to communicate the regression testing results is the Mean Absolute Error (MAE).

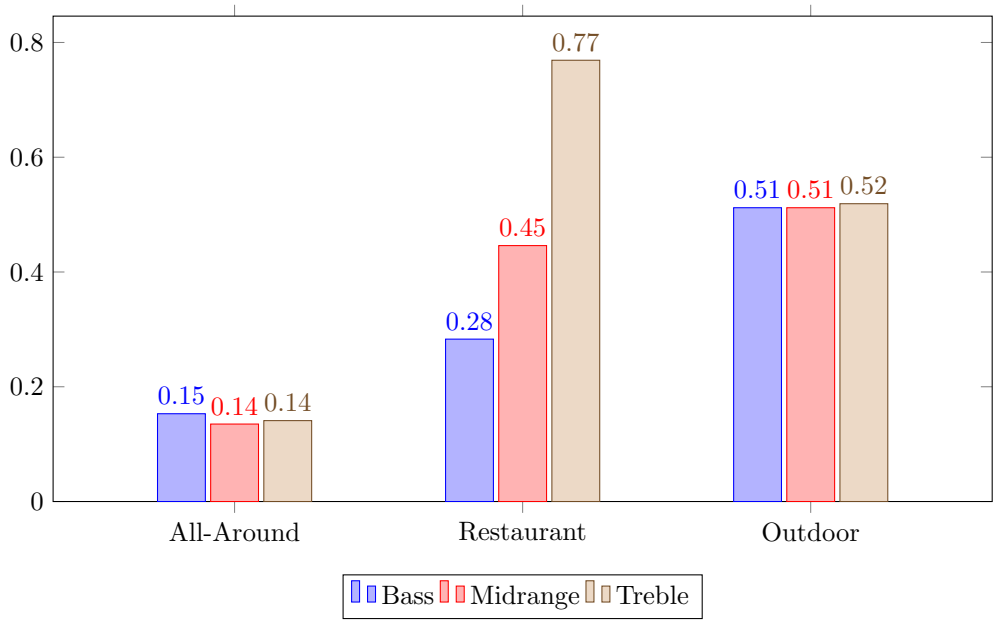


Figure 8.3: The average regression accuracy across the high-level environment classes. Measures are absolute error in dB. Some bars do not align even though they show the same error. This is due to rounding.

Figure 8.3 depicts the average error for bass (the blue bars), midrange (the red bars), and treble (the brown bars) for each of the high-level environment classes. All the measures are in decibel, and it can be found that the maximum average error is treble in the Restaurant environment. Table 8.3 shows that this environment should result in an +5 dB adjustment to treble. However, according to the results, the model's predictions deviates from the +5 dB with 0.77 dB on average. However, this behavior might actually be correct. The Restaurant environment is not static and differs just like any other environment. Not all time periods have the same amount of noise (e.g. cutlery or distant conversations) and therefore might require different hearing aid adjustments. The CNN model has been taught that STFT spectrograms representing the Restaurant environment should yield a specific hearing aid adjustment (as specified in Table 8.3), and it is utilizing regression achieve this. It tries to create a mapping between spectrogram features and the hearing aid adjustment. However, the spectrogram at time T might have less of the feature which corresponds to an increase in treble compared to a spectrogram at time $T + 1$. As a result, the predicted treble adjustment for the first spectrogram is less.

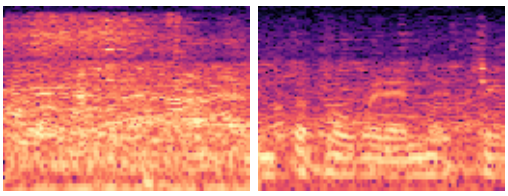


Figure 8.4: Exemplification of why hearing aid adjustment predictions might differ. Both spectrograms are created based on audio from a canteen (Restaurant environment).

This reasoning is exemplified by Figure 8.4. Both STFT spectrograms are created based on audio recorded in the same canteen within two minutes of each other. It is clear the left spectrogram represents more energy and have some high frequency components which are not to be found in the right spectrogram. It seems unlikely that one mapping can create the same hearing aid adjustment prediction for both spectrograms. The testing has not been extensive enough to conclude if this is the cause of the error seen in Figure 8.3. However, it does exemplify the situations, and it does make sense that not all representations of a given environment yield the same hearing aid adjustment prediction. Furthermore, assume that a spectrogram might have components which also represents a second environment class with a different hearing aid adjustment rule. The regression outcome might take this into account and find the happy medium resulting in a prediction that is not quite one or the other.

Treble for the Restaurant environment has been taken as the example, however, the discussion applies to the other errors as well. It is also worth mentioning that the prototype applies the adjustment to the hearing aids in increments of one. The errors seem acceptable and testing in a real environment (Section 8.2) will have to show if the errors presented in Figure 8.3 are actually problematic.

8.2 Real environment testing

Section 8.1 describes the prediction accuracy of the prototype based on a data set specifically made for testing. However, this data set only comprises of a limited number of environments. Furthermore, the environments being test against are the same as what the machine learning has been trained for. The purpose of this section is to describe how the prototype behaves in similar, yet different, environments. This is done by bringing the prototype to real environments. If the prototype is able to generalize its learnings, good performance is expected in this test as well. By generalization is meant that the prototype is able to recognize e.g. the sound environment at the Aalborg University canteen as a Restaurant environment even though training data for the Restaurant environment has been collected in the canteen at IBM. It is expected that these two different locations have a similar sound composition made up of background conversation, chairs being dragged, and cutlery making contact with plates. Such generalization is important as it is not viable to teach the prototype for every single existing environment.

To perform this testing, the prototype is brought to the following environments: A forest, two shopping centers (Rødovre Centrum and Field's), three intersections (Valby, Rødovre, and Sydhavnen), the group area for ITCOM and ICTE students at Aalborg University Copenhagen, the main canteen at Aalborg University Copenhagen, transportation (car, bus, and train), the quayside at Aalborg University Copenhagen, a train station (Valby station), and a conversation over the dinner table. A detailed description of all the environments visited can be found in Appendix G.3.

The testing is done with the prototype implementation described in Chapter 7 running on a OnePlus 6T smartphone with Android 9. For all training session with the reinforcement learning, a set of ReSound LiNX 3D are worn and connected to the USense SoundLearner application. The hearing aids are modified such that earplugs have been fitted around the hearing aid receivers (i.e. the speakers that are in the ear canal). This does not necessarily reflect the hearing characteristics of a hearing impaired person, but is done to block as much outside sound as possible. Blocking outside sound is found to make it easier to hear bass adjustments. It is worth mentioning that the hearing aids used for testing are simply fitted with a slight amplification across the frequency range (250 Hz to 6 kHz) as the author of this project is

not hearing impaired. As a result, the outcomes of the training sessions might not reflect a real hearing impaired user's preferences.

The following paragraphs describe the observations made while testing the prototype. The observations in relation to the Convolutional Neural Network (CNN) model is presented first followed by the testing results of the reinforcement learning.

Forest

The forest environment showed the expected results. The CNN model predicted the environment to fit in the All-Around class and suggested not to make any hearing aid adjustments. Samples of the environment were taken in different location including while walking on tracks of gravel and forest-type soil as well as while sitting still. Furthermore, a location was found where a slight breeze could be felt but it did not seem to register with the prototype. The environment was not classified as Outdoor and no hearing aid adjustments where suggested. It can be argued whether the breeze should have triggered a change or not. It seemed very light, though noticeable in the hearing aids. The training data for the Outdoor environment had much stronger wind. It makes sense that this is not registered as wind.

The first training session was carried out in this environment. The thoughts on the training session performance are presented when discussing the reinforcement learning component later in this section.

Shopping center

The shopping center is a good example of a context where a user's intent might not be consistent. If the user is alone and just need to get some shopping done, the user might prefer to suppress as much noise as possible. However, if the user is in the shopping center with someone else, it is possible that the hearing aids should adjust to allow for easy conversation. For this reason, it can be a difficult for the prototype to make appropriate hearing aid adjustment suggestions. As already mentioned in Section 5.1.3, user intent is a difficult problem to solve and not covered by this project.

First off, Rødovre Centrum was visited. The CNN model consistently classified the environment as Restaurant, and suggested hearing aid adjustments ranging from slight adjustments: {Bass: 0 dB, Midrange: +1 dB, Treble: +1 dB} to significant adjustments: {Bass: -2 dB, Midrange: +3 dB, and Treble: +5 dB}. This behavior was observed when sitting on a bench in an open area with people walking by. Larger adjustments seemed to be linked to higher levels of speech present in the background noise. This same pattern also showed when the location was changed to a different open area with tables belonging to a fast-food restaurant and a coffee shop. Some of the tables were occupied by people who were having conversations, and significant hearing aid adjustments were suggested more often.

During the testing in the first open area, a single All-Around environment classification accompanied by the following adjustment prediction: {Bass: -1 dB, Midrange: +2 dB, Treble: +4 dB} was observed. This shows that the environment classification and the adjustment suggestion is not strictly dependent on each other. This does makes sense as the CNN architecture has been designed such that separate mappings can be created for the regression and classification tasks. This is seen as an advantageous feature as the user might be in a situation that does not represent e.g. Restaurant, however, still requires emphasis on speech.

Three training sessions were carried out: One session in the first open area aiming for a pleasant sound, and two sessions in the open area with restaurants aiming for a trade-off between pleasantness

and speech focus. The training session outcomes were: {Bass: +3 dB, Midrange: 0 dB, Treble: +1 dB}, {Bass: -3 dB, Midrange: +3 dB, Treble: +3 dB}, and {Bass: -3 dB, Midrange: +4 dB, Treble +6 dB} respectively. During the training sessions, environment audio was recorded and used to adjust the CNN model.

Later, the updated CNN model was tested out in the second shopping center: Field's. The open areas of the two shopping centers seemed comparable. Neither Rødovre Centrum nor Field's were crowded with people when the testing took place. The open area in Field's did include more background noise due to escalators and more noticeable music from shops. Both All-Around and Restaurant classifications were observed, and suggested hearing aid adjustments ranged from no adjustment to {Bass: -2 dB, Midrange: +3 dB, Treble: +4 dB}. The adjustments suggested in Field's were generally lower than what was observed in Rødovre Centrum. The observations did not make it clear whether the environment just seemed different to the prototype or if this was caused by the training sessions performed in Rødovre Centrum. Section 8.3.1 (page 130) presents the CNN model's ability to learn in more details.

Entering a grocery store did create scenarios where more significant hearing aid adjustments were suggested. These suggestions occurred when the prototype was in the vicinity of noisy refrigeration units or moving shopping carts. However, simple speech or store music would only trigger slight adjustments: {Bass: -1 dB, Midrange: +1 dB, Treble: +1 dB}.

Office space, canteen, and dinner table

These three environments might seem very different, but combining the findings during the tests makes it possible to draw conclusion in relation to how speech is affecting the output of the CNN model.

In the office space (the ITCOM and ICTE group area at Aalborg University Copenhagen), one other group of people where having a discussion a few meters away behind a whiteboard. No other noise was present and the prototype, as expected, classified the environment as All-Around and suggested no hearing aid adjustments (a single test sample suggested a +1 dB increase in midrange and treble).

At the dinner table, which had five people, the CNN model primarily classified the environment as Restaurant and suggested no or slight adjustments. A single prediction did state All-Around and significant adjustments. The adjustments suggested throughout the test seems to be related to the power and clarity of the voice. Similar prototype behavior was observed in the canteen at Aalborg University Copenhagen. Scenarios with distant speech or high levels of just noise would result in the environment being classified as Restaurant and adjustments around: {Bass: -2 dB, Midrange: +3 dB, Treble +5 dB}. However, for situations with noise but clear speech, the adjustments suggested would be lower and around {Bass: -1 dB, Midrange: +2 dB, Treble: +3 dB}. This indicates that the CNN model tries to predict adjustments such that speech is improved. If it only sees noise, significant adjustments for improving speech intelligibility are suggested. However, if speech is detected and dominant compared to the background noise, less aggressive adjustments are suggested. Furthermore, if no noise is present, no adjustments are suggested.

Three training sessions were carried out in the canteen environment in an effort to bias the CNN model towards more comfortable audio. The three training sessions resulted in the following preferences: {Bass: 0 dB, Midrange: -3 dB, Treble: +6 dB}, {Bass: +1 dB, Midrange: -2 dB, Treble: +6 dB}, and {Bass: 0 dB, Midrange: -3 dB, Treble: +6 dB}. The adjusted model did seem to suggest a +2 dB gain in midrange whenever +3 dB would otherwise have been predicted. However, it also caused the office space

to be classified as Restaurant and suggestions of slight adjustments became more likely. This behavior is unwanted. A more thorough analysis of how the CNN model learns is covered in Section 8.3.1 (page 130).

Transportation and traffic

Three different sorts of transportation were tested: Car, bus, and train. It was found that low-speed driving (70 km/h and below) would most often be classified as All-Around and no hearing aid adjustment would be suggested. For highway speeds (110 km/h and above), the chance of classification as Restaurant increased and slight adjustments: {Bass: -1 dB, Midrange: +1 dB, Treble: +2 dB} were suggested more often. A bus produced similar results to a car at high speed in terms of suggested hearing aid adjustments. Environment classification varied between All-Around and Restaurant. For trains, the environment was classified as Restaurant if the train was not stationary. No strong conclusion can be made in terms of how speech affected the hearing aid adjustment prediction. However, it can be concluded that the noise from the train itself would cause the CNN model to suggest adjustments around: {Bass: -1 dB, Midrange: +2 dB, Treble: +3 dB}.

Similar behavior was observed at Valby station. No adjustments were suggested, and the environment classified as All-Around for general background noise. If the level of noise increased (e.g. due to arriving and leaving trains at the tracks next to the test setup), the environment would be classified as Restaurant and adjustments around {Bass: -1 dB, Midrange: +2 dB, Treble +3 dB} were suggested.

Furthermore, it was found that being in the sidewalk at busy intersections would often categorize the environment as Restaurant and suggest no or slight adjustments. When high-noise vehicles such a lorries or motorbikes would go by, the adjustment suggestion were increased to {Bass: -2 dB, Midrange: +3 dB, Treble +5 dB}. Visiting less busy intersections or keeping a bit of distance to the busy intersection (about 10 meters of distance was used for these tests) would make an All-Around environment classification and no hearing aid adjustments suggested more likely.

Comparing the observations from the transportation and traffic environment to the canteen environment shows that different types of noise create different CNN predictions. It seems like the prototype is much more sensitive to noise from speech compared to the mechanical noise of a vehicle moving. High levels of mechanical noise will trigger a major hearing aid adjustment, however, the testing seem to suggest that the prototype is much less sensitive to it. With that said, it can be argued whether the adjustment suggestion is correct. In a traffic environment, a user might rather want the hearing aids to simply reduce the volume, not focus on speech. However, the prototype has only been trained for a few environments which do not including train, and the intersection samples were recorded at distance, not right next to the road. Furthermore, the CNN has been taught that lowering bass, midrange, and treble is only associated with the Outdoor environment which contains only wind. As a result, it can be concluded that the predictions made by the prototype might not be ideal, however, they do reflect what it has been taught given the environment.

Quayside

Unfortunately, it was not possible to find a sufficiently windy location for testing prototype behavior in wind. It should result in an Outdoor classification and a suggestion for lowering the bass, midrange, and treble. This test was attempted in the forest and in a city environment next to a less busy intersection.

The Outdoor environment classification was observed a few times, but nothing conclusive. Testing at the quayside at Aalborg University Copenhagen did not create any conclusive results either.

The wind data used for creating the CNN model was recorded on a very windy day. It might be the case that the amount of wind present at the testing location simply was not enough. Lightly blowing into microphone of the smartphone, does trigger the Outdoor environment classification and an adjustment suggestion of {Bass: -3 dB, Midrange: 0 dB, Treble: +1 dB}. However, it is very challenging to create a scenario that mimics the windy environment by blowing into the microphone. No conclusive results were found in relation to the prototype's ability to classify and adjust for windy environments.

Wind was noticed in the hearing aids during testing. The prototype should probably be more sensitive to wind. As a solution, the CNN model should be trained with more examples of winy environments including a broader range of wind intensities.

Reinforcement learning component

The previous paragraphs have discussed the observed behavior of the CNN model. However, the CNN model does not have the final say in what hearing aid adjustments are suggested to the user. It is merely a suggestion that is fed into the reinforcement learning which is then supposed to make the final prediction based on learned user preferences. This paragraph covers the observed behavior of the reinforcement learning.

It should be underlined that the reinforcement learning requires the user to train it over time. However, for it to be usable, it should become accustomed to the user's preferences rather quickly. Therefore, it is expected to perform poorly at least until the first training session. Indeed, it did perform very poorly in the initial testing environments. The outputs were inconsistent and deviating. This was expected to become more consistent and actually reflect preferences after a training session. The first training session took place in the forest with an aim to emphasize bird song. Starting from {Bass: 0 dB, Midrange: 0 dB, Treble: 0 dB}, the training session took 57 iterations and resulted in {Bass: 0 dB, Midrange: +3 dB, Treble: +3 dB}. This shows that Sarsa(λ) is capable of finding an appropriate adjustment, although a bit slow. After this train session, it would be expected that given the same starting point, the final outcome would be in the vicinity of the training session result. However, this turned out to not be the case. No pattern was found in how the reinforcement learning chose hearing aid adjustments after the first training session. During the testing days, more training sessions were carried out in Rødovre Centrum and the main canteen at Aalborg University Copenhagen. However, without improved results.

As mentioned, the first training session in the forest took 57 iterations which is significantly more than the 12 iterations specified by requirement **NFR4**. However, training sessions in Rødovre Centrum took 35, 35, and 34 iterations respectively. In the canteen of Aalborg University Copenhagen, it took 25, 11, and 18 iterations respectively to finish the training sessions.

As the reinforcement learning state-space is divided into All-Around, Restaurant, and Outdoor, the training session in the forest has not improved the training time in the shopping center or the canteen. However, the shopping center and canteen are both classified as Restaurant. Hence, the learnings in the shopping center has likely been the cause of the reduced training time in the canteen.

It can be concluded that the reinforcement learning does learn as more training sessions are performed. It more quickly suggests that it has found the user's preference during training sessions. However, the ability to recall these user preferences afterwards presents a greater challenge, apparently.

Environment test - Final remarks

Based on the observation made in the various environments, it can be concluded that the CNN model behaves as expected. The reinforcement learning does seem to have some challenges which needs to be solved before it becomes usable. The time needed to perform the first few training sessions is much longer than planned. This might be acceptable as the number of iterations required is quickly reduced. However, the lack of ability to recall the preferences is not satisfactory.

The prototype seemed to be capable of differentiating between different sorts of noise. Moreover, it was able to focus on speech while avoiding overcompensating in situations where noise was present, but the voice was clear. However, the results of the testing in the real environments do not reflect the same accuracy as presented in Figure 8.1 (page 120) and Figure 8.2 (page 121).

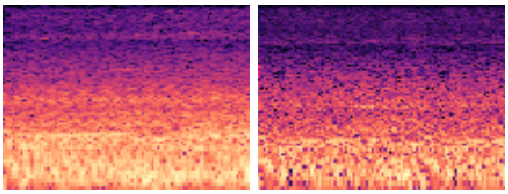


Figure 8.5: Comparison of two STFT spectrograms representing the same audio collected next to a busy intersection. Left spectrogram has been created using the Python library LibROSA, and the right spectrogram is created by the USense SoundLearner prototype application.

To understand why this might be, consider the two Short-Time Fourier Transformation (STFT) spectrograms in Figure 8.5. The left spectrogram has been created using the Python library LibROSA whereas the one on the right has been created by the USense SoundLearner application. The implementation in the USense SoundLearner application has been made such that it should mimic LibROSA. However, there is still a slight difference, and LibROSA seems to apply some sort of smoothing or brightening which is missing on the right spectrogram. All the CNN models are trained on a computer using spectrograms generated by LibROSA. This model is later moved to the smartphone application and used by USense SoundLearner to classify the environments. The environment classification is done by analyzing a spectrogram of the current soundscape. However, the spectrogram of the current soundscape is created by the USense SoundLearner application. The problem is that the CNN model is trained using representation that looks like the left spectrogram, but asked to make prediction based on examples like the right spectrogram.

	Environment class probability	**Suggested adjustments**
LibROSA STFT spectrogram	Forest: 86% certainty Canteen: 5% certainty	Bass: -0.03 dB Midrange: +0.1 dB Treble: +0.15 dB
USense SoundLearner STFT spectrogram	Forest: 48% certainty Canteen: 51% certainty	Bass: -0.38 dB Midrange: +0.7 dB Treble: +1.13 dB

Table 8.6: Comparison of the prediciton outcome when analyzing the spectrograms from Figure 8.5

Table 8.6 shows the prediction results when analyzing each of the spectrograms. The analysis is done using the same CNN model which was used for the environment testing described in the previous paragraphs. It can be seen that the discrepancy between the spectrograms makes a difference. Firstly, the USense SoundLearner spectrogram is classified as belonging to the Canteen environment (equal to Resturant for the high-level environments). Secondly, the suggested adjustments also reflect an intent to focus on speech. With that said, the LibROSA spectrogram is also incorrectly classified as a Forest environment. It should have been classified as Intersection. However, the Forest environment is included in All-Around and so is the Intersection environment making it acceptable for this prototype.

The probabilities shown in Table 8.6 do not add up to 100% as the remaining five low-level classes have been left out for brevity. It can be seen that the environment classification certainty of the USense SoundLearner spectrogram is almost equal for Forest and Canteen. Hence, making multiple samples of the environment before making a prediction might be advantageous. However, this does not solve the issue in relation to the difference between the two spectrograms. It can be concluded that the implementations used for generating the spectrograms must be completely alike.

8.3 Model learnability

The prototype consists of two learning components: The Convolutional Neural Network (CNN) and the reinforcement learning algorithm Sarsa(λ). This section evaluates the learning behavior of each component as implemented in the current prototype. The aim is to assess whether the prototype is able to adapt to the user, and reveal what might need to be changed in order to improve this aspect.

8.3.1 Learnability - Convolutional Neural Network

During the environment testing, three training sessions were carried out in the main canteen of Aalborg University Copenhagen. However, the CNN model did not seem to change. At the same time, it caused the CNN model to classify the office space as a Restaurant environment.

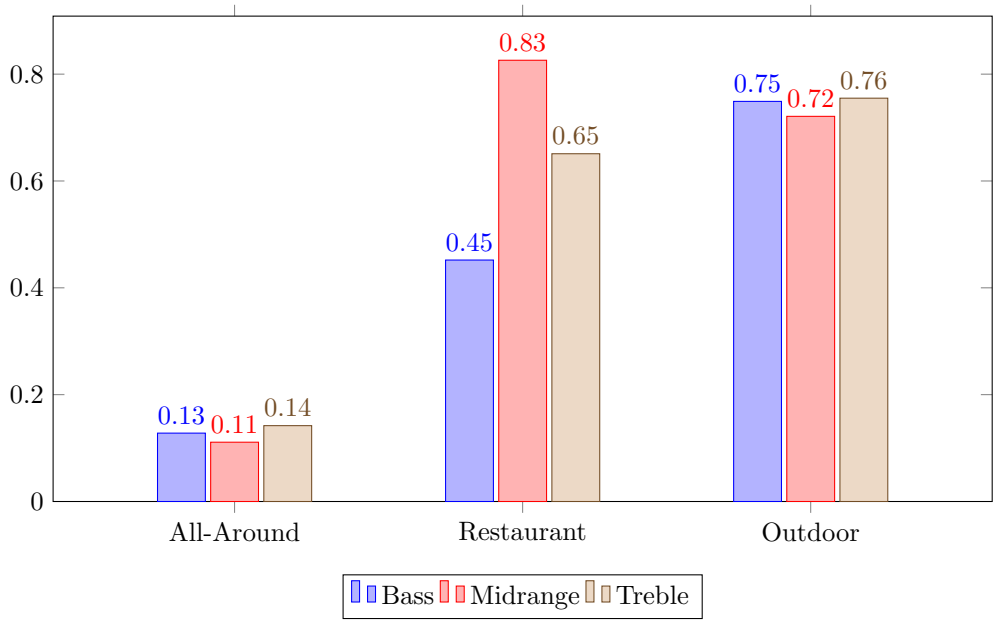


Figure 8.6: The average regression accuracy across the high-level environment classes for the adjusted CNN model. The model is based on three training session carried out in the main canteen at Aalborg University Copenhagen. Measures are absolute error in dB.

Figure 8.6 depicts how the average error of the CNN model has changed after having been biased by the training sessions. This figure should be compared to Figure 8.3 (page 123). Focusing on the Restaurant class in Figure 8.6, it can be seen that the error has increased. This makes sense as the model is still assessed according to the what is specified in Table 8.3. The changes in error seems reasonable as the model has been told that the user's preference is {Bass: 0 dB, Midrange: -3 dB, Treble: +6 dB} for two of the training sessions and {Bass: +1 dB, Midrange: -2 dB, Treble: +6 dB} for the remaining one. Originally, the CNN model has be trained for {Bass: -2 dB, Midrange: +3 dB, Treble: +5 dB}. The difference between this and the user preferences is mainly to be found for bass and midrange. As a result, they change more, which results in a higher error in Figure 8.6. This is expected and not necessarily a bad thing as the model is trying to adjust to the user's input. However, it can also be seen that the error for the Outdoor environment has increased which is undesirable. The training sessions took place in the Restaurant environment and should only affect that class.

The three training session collected 427 samples of the Canteen environment which was used to adjust the CNN model. Hereby, the CNN model learns the user's preferences in hearing aid adjustments as well as how the user's specific environment looks like. At the time of testing, each of the 427 samples were presented to the CNN model 30 times in order to adjust it. Consider the initial CNN model which predicts {Bass: -1.76 dB, Midrange: +2.62 dB, Treble: +4.33 dB} on average for the Restaurant environment when evaluated with the prediction examples presented in Table 8.2. The adjusted model with the error shown in Figure 8.6 predicts {Bass: -1.55 dB, Midrange: +2.18 dB, Treble: +4.55 dB} on average which is only a minor change (just as was observed when the testing was carried out in the canteen).

Multiple parameters can be adjusted to affect the learning process when creating CNN models. If the learning rate is adjusted such that the CNN model learns 10 times faster, the predictions for the

Restaurant environment is {Bass: +0.18 dB, Midrange: -2.23 dB, Treble: +4.84 dB} on average after presenting the 427 samples to the model just 14 times. The model that has been subject to more aggressive learning reflects the user's preference much better. However, it also results in the CNN model being incapable of predicting the Outdoor environment as well as a decrease in the All-Around prediction accuracy.

This example shows that the parameters chosen for updating the CNN model in the current prototype makes it rather slow at learning user preferences. However, it also shows that simply changing the parameters makes it capable of learning, but that it must be done in a clever manner to not degrade the overall performance.

8.3.2 Learnability - Sarsa(λ)

During testing in the real environments (Section 8.2), a number of training sessions were carried out. Initially the reinforcement learning seemed rather slow. It took 35 iterations to finish the first training session in the shopping center. However, later training session carried out in the canteen took only 11 iterations for the quickest one. This seems to suggest that the prototype is learning and becoming more capable of determining when the user has found a preferred hearing aid adjustment. This section aims to explain the learning behavior of the reinforcement learning component and discuss whether the earlier findings are just a coincidence.

The testing carried out in this section is done in two stages. First, the results based on the prototype as configured during real environment testing are presented. Hereafter, the configurations are changed to show how simple modifications can change the algorithm's learning behavior. In order to do a valid comparison, all prior learnings are clear before starting each test. All training sessions were started from state: {Bass: 0 dB, Midrange: 0 dB, Treble: 0 dB}.

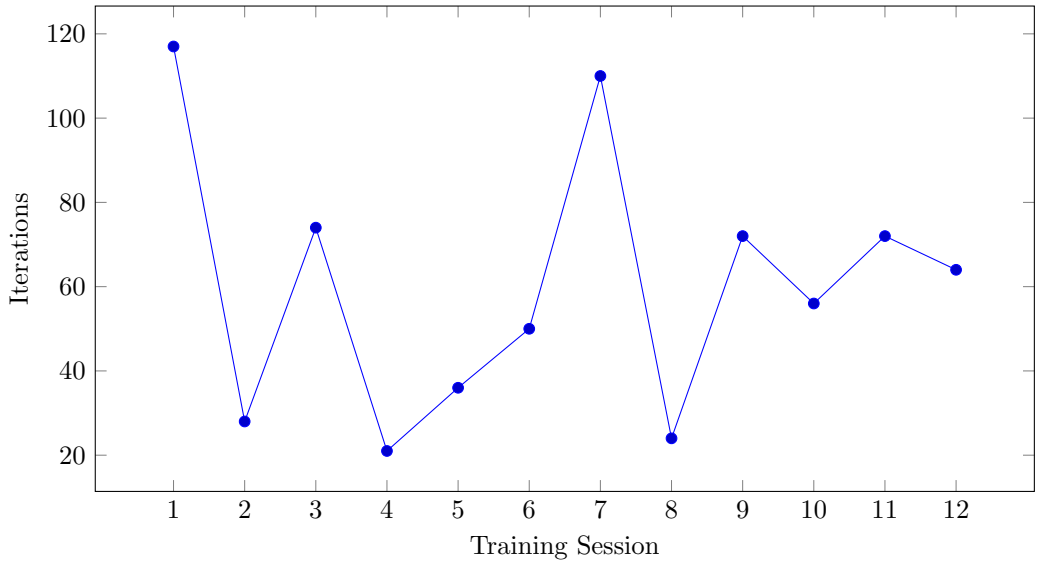


Figure 8.7: Number of learning iterations needed to reach state {Bass: -3 dB, Midrange: +3 dB, Treble: +3 dB} for 12 consecutive training sessions.

Consider Figure 8.7 which shows how many iterations each of 12 training session took to reach state: {Bass: -3 dB, Midrange: +3 dB, Treble: +3 dB}. It can be seen that the initial training session is the one which requires the most iterations to reach the desired state. Already at second training session, the number of iterations needed has decreased immensely. Looking at those two points individually would indicate that the model has learned the user's preferred state from the first training session. However, considering the following training sessions, it can be seen that the number of iterations fluctuates. Seventh training session requires almost just as many iterations as the first one. This is not a very good indication of learning. Moreover, this many iterations seem like a lot in order to move from {Bass: 0 dB, Midrange: 0 dB, Treble: 0 dB} to {Bass: -3 dB, Midrange: +3 dB, Treble: +3 dB}. Requirement **NFR5** specifies that adjustments of three decibels can be made each training iterations. Ideally, it should only take three iterations to reach the desired state in this test.

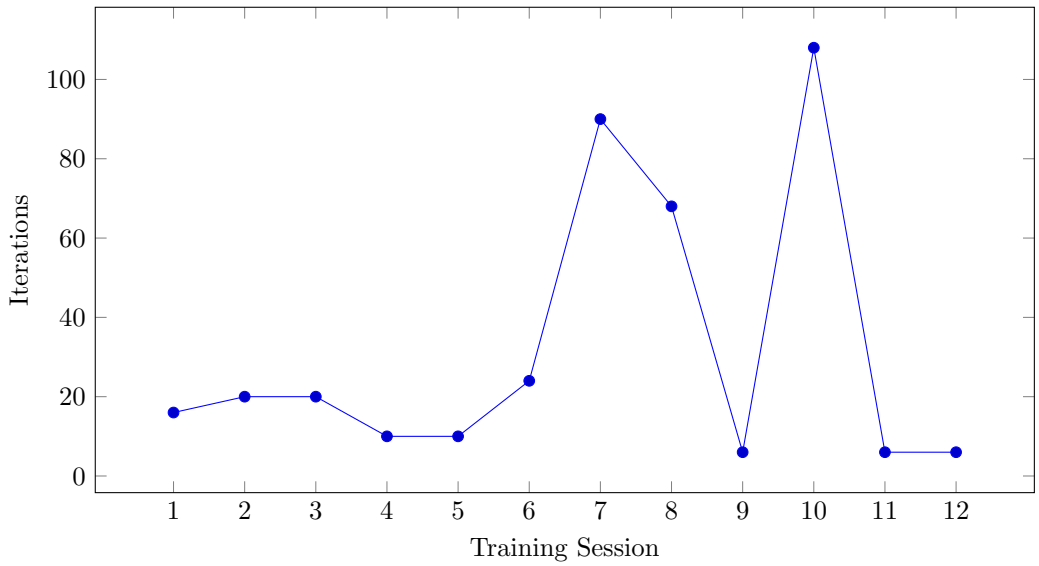


Figure 8.8: Number of learning iterations needed to reach state {Bass: -3 dB, Midrange: +3 dB, Treble: +3 dB} for 12 consecutive training sessions. The discouting factor has been reduced from 0.9 to 0.4 giving less emphasis to future rewards.

Adjusting the configuration of the learning parameters changes the behavior as depicted in Figure 8.8. For this second test, the discounting factor is adjusted such that much less emphasis is given to rewards expected in the future. As a result, the learning algorithm is more focused on immediate rewards. Figure 8.8 shows that, initially, this results in shorter training session compared to Figure 8.7. However, the fluctuating behavior returns at the seventh training session.

The prototype must also adapt in the event that a user changes preference. Another four training iterations were done with the desired state being {Bass: +3 dB, Midrange: +3 dB, Treble: +6 dB}. These sessions where completed in 11, 5, 32, and 7 iterations respectively. Two things can be extracted from this: Firstly, only a few iterations are required even though the desired state has changed. Secondly, another spike in the iteration count is seen at the third training session.

The data shown in this section does not immediately seem to correlate with the results found from the real environment testing. This deviation can be explained by the fact that this section has been trying to

reach one specific state. During the environment testing, a state would be accepted if it seemed fitting. More than one state might seem fitting which reduces the time required to find it. With that said, the three training sessions had very similar outcomes. Whether this is due to coincidence or actual learning cannot be decided based on the testing carried out in this section.

It can be concluded that changes to the parameters of the learning algorithm can significantly change its behavior. The current implementation could benefit from a deeper analysis of the specific configurations required for this particular problem. Furthermore, the reward structure might need a revision. Currently, positive rewards are much greater in comparison to the negative rewards. As a result, states that are, by accident, given a positive reward needs to be visited and given a negative reward many times before the mistake is corrected. This might be the cause of the spikes seen in Figure 8.7 and Figure 8.8. At each iteration of a training session, the user has to choose between two sound profiles (i.e. suggested hearing aid adjustments). A positively rated sound profile might be better than the alternative at time T. However, at a later time, $T + 1$, the previously positively rated profile might no longer be the best one. Over time, incorrect positive rewards might be accumulated resulting in the spiking behavior which has been observed in this section.

8.4 Test conclusion

The chapter has covered multiple aspects of both the reinforcement learning and the Convolutional Neural Network (CNN) components of the developed prototype. The findings show that the initial CNN model has a high classification accuracy of 99% within the three high-level classes: All-Around, Restaurant, and Outdoor. Furthermore, more detailed classification of the low-level environment classes also shows good results. This indicates that it might be possible to expand and include more environment classes. Similar good results are seen when assessing the initial CNN model's ability to predict hearing aid adjustments.

The good results from the accuracy testing were not entirely reflected in the real environment testing. The results were still good as the prototype seemed capable of distinguishing between different sorts of noise and only put high emphasis on speech whenever needed. Some misclassification and inaccurate hearing aid adjustment suggestions were observed. However, this can partly be blamed on the environment representation. As CNN model training is done on a computer, but environment analysis is done on a smartphone, two separate implementations for generating Short-Time Fourier Transformation (STFT) spectrograms are needed. The testing showed a slight difference between the output of the two implementations which did have an effect on the environment analysis.

As the user interacts with the prototype, the initial CNN model is supposed to become a custom model specifically capable of recognizing the user's common environments and predicting the user's preferred hearing aid adjustments. In order to learn the user specific preferences, the reinforcement learning algorithm Sarsa(λ) is used. The testing shows that the current configuration of Sarsa(λ) results in time-consuming training sessions, which is undesirable. Furthermore, testing suggests that it might have self-destructive learning characteristics making it sensitive to user error and difficult to reach a desired state. This problem was more noticeable during strict testing and less so during real environment testing. During testing in real environments, training sessions were, generally, finished within reasonable time. However, there is still a need for improving the efficiency and performance. The ability to recall the preferences again at a later time also proved challenging. Even though the user carries out training

sessions, the reinforcement learning state-space is likely still only sparsely populated or populated with possibly outdated information. Modifying the prototype to handle this might improve the ability to recall user preferences.

Performance improvements are also needed for the CNN model to ensure user specific preferences are given a higher importance. Real environment testing showed that user input as a result of reinforcement learning training sessions had a low impact on the CNN predictions. Moreover, the model seemed to degrade more than adjust to the user's input. With that said, it was shown that using slightly different configurations made the model able to fully adapt to the user input. However, the model's classification accuracy suffered from these configuration changes.

It can be concluded that both the reinforcement learning and Convolutional Neural Network component can benefit from a thorough analysis targeted towards finding the optimal parameters.

Chapter 9

Improvements

The prototype does not fulfill all requirements, and the testing revealed some problematic areas. As a result, this chapter highlights some of the improvements which can greatly increases the usability and value provided by the designed solution. The following text presents thoughts on how to solve the issues related to learning and recalling user preferences. Furthermore, the chapter also outlines the ideas of including more fine-grained environment classification as well as user intent.

Ensuring consistent environment representation

The testing showed that the current prototype has a discrepancy between the spectrograms used for training Convolutional Neural Network (CNN) models and when using the models for predictions. This difference did not render the prototype useless, however, the prediction outcomes were somewhat affected. Environments could be classified incorrectly, and hearing aid adjustment suggestions did not necessarily accurately reflect what the CNN model had been trained for.

Based on this, an obvious improvement for the prototype is to ensure that only one implementation is used to generate spectrograms. This ensures that the environment representations are similar both when training and predicting. Alternatively, multiple implementations can coexist (e.g. one in the smartphone application and one in the Training entity) as long as the procedures are exactly the same such that the produced spectrograms are identical. This prototype is using a Python library on the Training entity and a self-implemented procedure trying to mimic that library for the smartphone application. Even though great effort was put into replicating the behavior, the resulting spectrograms are different.

Another aspect is the microphone used for recording the audio. The prototype utilizes audio recordings from the built-in microphone in the smartphone. These recordings are used for environment analysis as well as sent to the Training entity after a user has carried out a training session. However, all the audio recordings used for training the initial CNN model has been made with the Sennheiser AMBEO Smart Headset which is made for high quality audio recording [87]. There might be differences in the frequency response between the AMBEO headset and the smartphone microphone. This can result in deviation between expected accuracy and observed accuracy in real environments. For a prototype, this problem can be resolved by recording training data using the smartphone microphone. However, when user testing is needed, the users might have different smartphones with different microphone resulting in different behavior. Furthermore, using the microphone of the smartphone might not even be viable in

the long run. If people have the smartphone in their pocket, the audio recordings are unlikely to actually represent the environment as experienced by the user.

The difference in frequency response could also become a problem for shared CNN models. During the requirement verification session with Anders Udahl, Senior Manager for GN Hearing's Connected Apps team, it was mentioned that one CNN model for each user might not be a scalable solution. If multiple users have to share one CNN model, but the input given to the model is generated by multiple smartphones, which are all representing similar environments slightly differently, two things can happen: Either the CNN model becomes very good at generalizing as it learns many different representations of one environment, or it becomes very poor at predicting anything because it cannot find a mapping between the differing inputs which are all claiming to result in the same output. More effort should go into understanding the differences in frequency responses and how this can affect the environment representations and the CNN models.

With that said, this project is only aiming to develop a prototype. The optimal solution would incorporate audio streaming from the hearing aids to the smartphone and use this audio for analyzing the environment. Using the hearing aids as recording devices also limits the frequency response issue. Assuming that all users of the solution are wearing the same hearing aid, the audio representations are now all the same. Audio used to train the initial CNN model should also be recorded using the hearing aids. This ensures that frequency ranges and sample rates are kept consistent resulting in a usable CNN model from the outset and throughout.

Additional environments and hearing adjustment targets

The project is focus on three high-level environments: All-Around, Restaurant, and Outdoor. With a personalized solution, three general environments might not be satisfactory for the user. As an example, Restaurant can range from the canteen at the user's workplace to a fancy steakhouse restaurant. Even when assuming the user's intent is the same (e.g. have a conversation), the adjustments might need to be different because the level of noise might be different. Fortunately, the testing did show good performance in the prototype's ability to only make slight adjustments for clear speech and more significant adjustments for very noise environments. However, office space, forest, and beach might all be classified as All-Around. In the office, the user might like a lower volume in order to focus. Meanwhile, in the forest, the focus should be on bird song, and on the beach, the sound of the waves should be audible. This can be seen as different intent in the same general environment or as three unique low-level environments.

The prototype is already generating high-level environment class predictions based on multiple low-level classes. The accuracy testing shows good results even for low-level classes. It might be possible to add more which can give the user a solution that can differentiate between e.g. office space, forest, and beach. To improve the classification, equally many data samples should be provided from each low-level class to give the CNN model the best chance of learning without becoming biased. Making the CNN model able to differentiate between more classes might put more strain on the regression part as well. Depending on how the CNN model reacts, architectural changes might be necessary on the regression branch.

The prototype developed in this project has a coarse granularity. Sound profiles presented to the user while performing training session in the smartphone application differ significantly. Having finer granularity puts more responsibility on the reinforcement learning algorithm (Sarsa(λ)) and its ability

to determine what the user is trying to achieve. It is a compromise between number of training session iterations needed and granularity. Effort should be put into determining if the current is sufficient for the users, or if finer granularity is needed for true personalization.

Environment analysis processing time

The process of creating a Short-Time Fourier Transformation (STFT) spectrogram takes the smartphone application about 20 seconds with the current implementation. Focus has been on replicating the behavior of the LibROSA Python library, not on efficiency. However, over time, this might impact the battery life of the user's smartphone. Improving the efficiency can be a necessity for the solution to be viable in the long run. Furthermore, improving the required processing time results in more accurate predictions. Being more efficient makes it possible to analyze more than one sample. As a result, a final prediction can be based on the average of multiple predictions. This reduces occasional misclassifications and provides the user with hearing aid adjustment suggestions that represent the current environment and the user's preferences better.

Apart from efficiency and accuracy, another reason to improve processing time is to ensure that more users can benefit from the solution. Many different smartphones exist, and they are not all created equally. It can range from slow and outdated to newest flagship model on the market. The new and powerful smartphone might have the resources required by this solution, but older or less high-end devices might not. Apart from processing time, storage is needed. The reinforcement learning is reasonably lightweight, but the CNN models require about 100 megabytes of storage. It might be possible that the CNN architecture can be tweaked to reduce the size of the model.

As an alternative, smartphones with limited resources could just rely on the online prediction (see Figure 7.9 Section 7.2.1, page 109). This requires the user to be connected to the internet when they want to use the personalization functionality, but that might be an acceptable price to pay compared to not having the personalization at all. Issues related to data usage then arise as audio files would have to be moved from the smartphone to the Web service for prediction by the Training entity. According to **NFR14**, **NFR15**, and **NFR16**, one prediction is based on six recordings, a recording is one second long, and a new recording is taken every five seconds. This gives 12 seconds of audio per minute. An audio recording is mono and done with a sample rate of 48 kHz using 16 bits to describe each sample. This results in just over one megabyte of raw audio being recorded each minute. Over the course of a day, this adds up and a way to limit data usage must be developed. One way would be to compress the audio, and another would be to reduce the observation frequency. Observing every five seconds might not be necessary for stable environments.

Ability to learn and recall

As proven by the testing, the prototype is having issues recalling user preferences. Furthermore, learning new preferences also proves to be a challenge. Improvements to both the CNN architecture and the reinforcement learning component are need. The reinforcement learning component consists of the Temporal-Difference (TD) Prediction process and the Sarsa(λ) algorithm.

The risk and assumptions (Section 5.3, page 65) define **R1** and **R2** related to time-consuming training sessions and an incapability of capturing user preferences. The prototype behavior is a case of materialization of these two risks. The mitigation plans presented in Table 5.5 (page 69) suggest limiting the

state-space. That is a possibility, however, from the observed prototype behavior, the issue seems to be related to configuration rather than complexity.

The learnability of Sarsa(λ) is controlled by the reward strategy as well as a learning rate (α), decay factor (λ), and discounting factor (γ) (see Equation A.2-A.4, Appendix A.3). These values are used to update the eligibility traces which affect the update of the Q-table (i.e. the state-space). Increasing γ makes expected future rewards more important to the algorithm [80], increasing λ makes earlier steps leading to the current reward more important [96], and increasing α results in bigger adjustments of the current state-space. Figure 8.8 (Section 8.3.2, page 133) from the Testing chapter shows, decreasing γ gave an immediate learning improvement for a few initial training sessions. This means the algorithm benefits from not taking future rewards into account as much. Specifically for Sarsa(λ), γ also affects the decay of the eligibility traces. Therefore, an improvement could be to perform a detailed analysis on the eligibility traces, and evaluate whether they are behaving as expected. Initially, one might not expect these to be the problem as the eligibility trace matrix is reset when a new training session is initiated. However, the matrix affects the Q-table during the training session, and the Q-table is persisted. The eligibility traces ensure that previously visited state in this training session are affected by a current reward. As a result, negative and positive values are propagated throughout the Q-table by the eligibility traces. As a result, any mistakes made by the user, or any bad suggestions made by the reinforcement learning also contributes with a negative effect to the Q-table. By design, each training iteration results in a positive reward (for the preferred sound profile) and a negative reward (for the sound profile that was not chosen). As the granularity is coarse ($\pm$3 dB from one training session iteration to the other), the perception of what is good and bad might become very muddy. After accumulating, Sarsa(λ) becomes unable to navigate intelligently in the state-space resulting in some long training sessions as depicted by the fluctuating behavior in Figure 8.8 (Section 8.3.2, page 133). The original reason for choosing Sarsa(λ) was due to its ability to utilize input data well such that few inputs would be required (see Section 5.1.2, page 42). However, the prototype might have overdone it. Making Sarsa(λ) more focused on the current state, not the past or the future, might alleviate the problem. Furthermore, it can be considered if the reward strategy is appropriate and whether it is necessary to give negative rewards to the sound profile not chosen by the user.

The TD Prediction routine has been designed to excel at precision, and has been configured with a very high learning rate, but very little focus on future states. As it has to find a trajectory in the state-space towards the user's preference, some emphasis on future states might be beneficial. Configuring the TD Prediction routine to look ahead and possible letting it search for longer might improve the prototype's ability to recall user preferences.

Finally, the CNN model's ability to recall seemed excellent, however, it lacks learnability. Section 8.3.1 (page 131) did show that adjusting the learning rate parameter could resolve the issue but created a new problem. Learning too quickly resulted in the CNN model being incapable of making good decision in other environments; it seemed to overfit. Depending on the cause of this behavior, different approaches can be taken:

– The convolutional layers of the CNN architecture are capable of understanding the spectrograms. However, a high learning rate might cause them to become too specialized at recognizing the specific samples that are being shown currently and forget earlier learnings. In order to avoid this,

the convolutional layers could be frozen [116]. Freezing the layers refers to keeping them constant such that the weights are not updated by new training data. Hence, no learning occurs. This is desirable if the frozen layers have already learned to do their tasks, but other layers needs to be adjusted. In case of this project, these "other layers" are the fully connected layers creating the mapping from spectrogram features to environment classes and hearing aid adjustments.

– If the overfit is caused by an overweight of samples in a specific class, the CNN model adjustment can be halted until samples from other environments have been collected as well. However, this creates an unwanted delay.

– If the issue is caused by high complexity, the CNN architecture might not be able to capture it. Asking too much of too simple of an architecture can cause inaccurate results. If this is the case, more convolutional layers can be added, or the level of details which they examine the input with can be increased. If the complexity problem is in the fully connected layers, more neurons or layers can be added to increase the complexity in the function that describes the mapping between input and output.

The problem observed in this project seems to be related to overfitting. To resolve the issue, a good start would be to collect more data, ensure that each class has somewhat the same number of samples, and then train the initial model. When the initial CNN model is ready, freeze the convolutional layers such that the user specific data is only allowed to modify the fully connected layers. This way, adjusting CNN models becomes faster (as fewer neurons needs updating), and the adjusted models are still capable of learning user specific environments and hearing aid adjustment parameters.

Account for user intent

Both the environment and the user intent must be predicted in order to provide a hearing aid suggestion which for sure fits the current context. The solution is capable of predicting the environment, but does not include the aspect of the user intent. Essentially, the current prototype assumes that the user's intent is the same now as it was last time the user performed a training session. As this is not necessarily the case, the intent should be taken into account to provide the optimal solution.

This creates the issue of how to determine the intent. On way is to simply ask the user what the intent is e.g. by choosing from a list of options. However, this becomes problematic if the intent is not listed. Text or speech analysis techniques could be utilized to allow the user to write and speak their intent freely. However, in order to provide a fully automated system, the intent should be predicted without having to require attention from the user.

Using information such as day of the week, time of the day, and location might make it possible to predict the user's intent based on patterns. As an example, it is Friday evening and the user is at home. Patterns might show that the user tends to want to relax in this context. However, this makes it difficult to predict intents for situations that are not encountered very often. Adding information about the type of location (cinema, bar, shopping center etc.) can be beneficial. As an example, earlier learnings about a user's intent in one shopping center can then be used in another shopping center as well. It might be possible to extract more patterns if data related to speed and movement is added. High speed can indicate being in a vehicle whereas low speed might be a walk in the forest. In the first case, the user

wants to reduce noise, yet still be aware of traffic. In the second scenario, nature sounds should be emphasized, and enjoyment is the highest priority.

Furthermore, by using voice detection and analysis, it can be possible to detect conversations. If the user is facing the conversation and the user's own voice is also detected, this indicates an intent of wanting to have a conversation. This could be further developed by having a list of known voices. These voices could include family members and colleagues such that when a known voice is detected, the hearing aids can allow the user to better follow the conversation.

The parameters mentioned so far are similar to what Korzepa et at. [55] have listed as possible inputs for intent prediction. However, it might be possible to go even further than this. Entries in a calendar application on the smartphone could give clues to where a user will be and what the user will do in the future. Being able to know it in advance might not be important, but calendar entries can be used to support some of the earlier mentioned intent indicators. Activity on a smartphone might also indicate an intent. Hearing aid adjustments might need to be different if the user is reading an ebook compared to if the user is using social media applications.

If the solution is uncertain about the intent, data collected from many users could be used to provide a prediction. This way users can collaborate in providing the best experience at all time in any environment for each other. Users that generally like the same intents can benefit from each other when in new environments.

Chapter 10

Discussion

This chapter is divided into three parts. First part discusses the readiness of the solution designed in this project. This discussion assumes that any issues are resolved and evaluates if the solution would be sufficient. This is followed with a section that discusses the user aspect and whether such a solution is actually desired. This is based on four interviews done with participants from a test panel usually used by the User Experience and Audiology departments at GN Hearing. Finally, the last section of the chapter presents advancements within the hearing aids industry in the near as well as distant future.

10.1 Solution readiness

This section discusses the readiness of the prototype and what the solution is capable of. Based on the testing and the list of improvements that have been extracted as a result, it can be concluded that the prototype needs more development before it gets integrated into a hearing care solution. However, when these issues are resolved, the user will have a personalized hearing care solution which learns and adjusts according to their preferences. Based on interviews with real users presented in Section 10.2, this is likely to impress the user. However, a curious user might ask what the limitations are. Given that the current prototype issues are resolved, it is able to perform the personalization in three general environments: All-Around, Restaurant, and Outdoor. On top of this, as the reinforcement learning state-space contains all adjustment possibilities, recalling preferences should not be limited. As specified in the requirements section (text below Table 5.9 (page 74) and Table 5.12 (page 77)), this is likely sufficient for a full-scale solution. However, since such high-level environments are used, one might question if detailed personalization (e.g. specific adjustments for walks in the forest) is possible. As the Convolutional Neural Network (CNN) model actually classifies the environment into seven low-level classes, and the state-space can represent any adjustment, detailed personalization should be possible. As long as an environment can be recognized based on a spectrogram of the soundscape, it can be associated with user preferences.

Recall that the CNN model predicts initial hearing aid adjustments which are then used to find the user's latest preference in the state-space. In order to predict these initial values, regression is used. As a result, the CNN model can always make an initial hearing adjustment suggestion. A spectrogram that is a mix of two classes, results in a prediction which accounts for both scenarios. As an example, the user might be eating at a restaurant with tables outdoors on a windy day. Such a place is more open than

a regular restaurant and background noise might be less intense but mix with wind. The CNN model might classify this as an Outdoor environment, however, it has still picked up the restaurant background noise. As a result, the hearing aid parameters predicted by the model will try to lower wind noise while improving speech intelligibility. This initial prediction by the CNN is then used to search the state-space and find the user's closest preference. As the state-space is divided into environments as well, the final hearing aid suggestion might be more biased towards reducing wind noise as that is usually what the user prefers in the Outdoor environment. However, if the user trains the smartphone application for this specific environment (windy outdoor restaurant), the state-space will accommodate the new preference. To exemplify, consider that the CNN model is predicting {Bass: -5 dB, Midrange: -3 dB, Treble: 0 dB} for the current environment. Before training the solution for this specific environment, the strongest path in the state-space from this prediction might resolve in {Bass: -5 dB, Midrange: -5, Treble: -5 dB} (the default "Noise filter" for Outdoor in the ReSound Smart 3D application as defined in Section 2.6 (page 15) which might be what the user usually prefers) being suggested to the user. If the user finds this too noise reducing, the user will perform a training session e.g. resulting in {Bass: -4, Midrange: -2, Treble 0 dB}. The path through the state-space from the point initially predicted by the CNN model is now favoring the user's latest preference. Furthermore, this is remembered for next time when the user is at the windy outdoor restaurant. Meanwhile, other windy situations (e.g. a walk on the beach) will still work as intended. For such an environment the CNN model might predict: {Bass: -4 dB, Midrange: -4 dB, Treble: -3 dB}. This is a new point in the state-space which has not been associated with any specific user preferences. As a result, the state-space is searched and the final outcome is likely to be {Bass: -5 dB, Midrange: -5, Treble: -5 dB} as that is what the user usually prefers in the Outdoor environment.

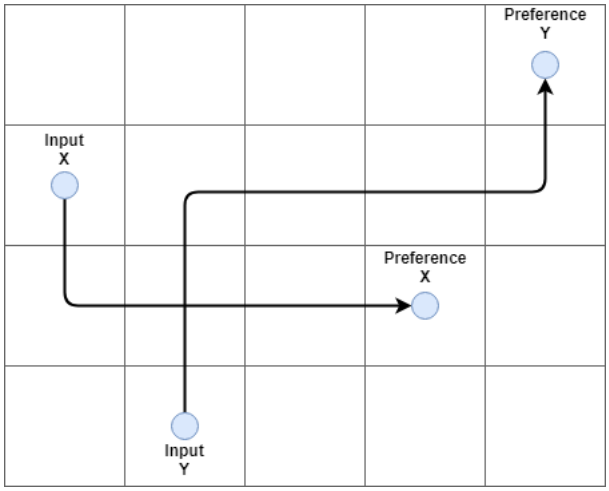


Figure 10.1: Illustration of crossing paths in the state-space. This can be a result of disagreements between predicted and actual user preferences. Made with draw.io online tool.

The limitation of this functionality becomes apparent if the user has too many situation specific preferences causing paths in the state-space to cross each other (depicted by Figure 10.1). If the path from input_X to preference_X is crossed by the path from input_Y to preference_Y, it might result in both input_X and input_Y ending up in e.g. preference_X. If the user strongly disagrees with the initial predictions made by the CNN model, this scenario could happen. However, since the CNN model, over time, adjust

to the user's preferences, this problem should be reduced. With that said, user intent is not something this solution models. Different user intents in the same environment might also result in incorrectly predicted hearing aid adjustments and crossing paths in the state-space.

From this discussion it can be concluded that the prototype should be capable of providing detailed personalization assuming that the current issues are resolved. The focus on high-level environments does not necessarily limit the solution's ability to personalize.

With the personalization aspect discussed it is natural to also consider if the solution will be able to adjust sufficiently. The developed prototype makes it possible to adjust ±6 dB for all three parameters. According to the user interviews (see Section 10.2), the adjustability available in the current ReSound Smart 3D application is sufficient for their needs. Just like the prototype, the Smart 3D application also allows for ±6 dB of adjustments. Hence, it must be concluded that the prototype should be capable of satisfying the needs of the users and provide sufficient adjustment. Furthermore, there is a safety aspect in it as well. It is undesirable to create a solution that might be able to adjust to a level that is unhealthy and can cause further damage to the hearing. Moreover, this solution is not supposed to correct a bad fitting. It simply provides the user with a tool to better manage challenging situations.

As a final note it should be mentioned that this project has been working with personalizing hearing aids. However, the solution developed is not necessarily limited to hearing aids. As an example, Jabra has developed the Jabra Sound+ smartphone application which supports a range of their headphones. The application contains an equalizer that allows the user to adjust the music to their liking [45]. The solution developed in this project might be adapted to predicting equalizer preferences for music instead. Spectrograms might not be the appropriate representation of music, and other adjustments to the solution might be needed as well. However, this is just to outline that the research and thoughts presented throughout the project applies to more than just hearing aids.

10.2 User need and interest in solution

In order to understand common needs for hearing aid users, and to determine whether a solution like the one developed in this project is likely to be used, four users were interviewed. All the user are members of a test panel established by GN Hearing to get feedback on their current solutions and new features. As members of the test panel, they have been given ReSound LiNX Quattro hearing aids and introduced to the ReSound Smart 3D application. As a result, the answers provided by the users stem from their experience with this hearing aid and smartphone application combination.

The User Experience department at GN Hearing pointed out these four test participant as good candidates for receiving input for project like this. Furthermore, the participants have been chosen such that two are female and two are male, and they are equally distributed in the two user types outlined in Section 5.2.2 (page 59-61). The two female interviewees are classified as care seekers, and the two male interviewees are categorized as proactive users. Detailed summaries of the interviews including age and hearing loss characteristics can be found in Appendix H. In the following text, they are denoted as User_A (male, 68 years old), User_B (male, 49 years old), User_C (female, 69 years old), and User_D (female, 67 years old).

The majority of the users express great satisfaction with their current hearing care solution. Both the ease of use (application and hearing aids) and the sound quality is mentioned as satisfying. However,

User_B has a different opinion and explains that he does not use the application much. He feels it provides little value to him. However, he does classify himself as an unusual hearing aid user as he has been hearing impaired his entire life without treating it. As a result, he has learned to live with it and compensate for it. He thinks the hearing aids should automatically adjust to the sound environment. He does not want to be the one to do it.

All the interviewees were asked to mention some improvement they would like from the top of their head. The following list outlines their wishes:

- User_A would like to be able to turn off all noise reduction such that he can hear e.g. the hissing sound from a punctured bike tire while trying to fix it. Furthermore, he would like even more adjustability in the Sound Enhance (see footnote on page 58 in Section 5.2.1) both on the range and the number of parameters.

- User_B asks for a solution that allows him to push a button which will cause some change in the hearing aids. The solution will then ask him if the sound is better as a result of the adjustments. He can then reply yes or no. This idea was proposed by the interviewee unprovoked by the interviewer and before any mentioning of personalization had been made.

- User_C would like automated volume control such that she does not have to adjust the volume according to her environment by herself. However, she does not see how this can be done as all people are different, and even she prefers different adjustments in the same context.

- User_D does not see what can be made better, and automation is not needed as the solution is already so easy and quick to use. However, she does express interest when presented for the idea of personalized hearing aids that adjust to the environment.

All the users express positive feedback about the solution developed in this project. However, User_C does not see how such a solution would be accomplished. User_D specifies that it is very important that such a solution does not affect the size of the hearing aid, and that it is very easy to use. It is important to her that the hearing aids do not get any bigger (due to comfort), and that the application is very clear in its instructions. In terms of training the solution, the users claim to be willing to spend the time that is required. However, they all seem to compare this with the time it takes to make adjustments today. The time they are willing to spend ranges from 30 seconds to two minutes when asked to be specific.

It is difficult to generalize to all hearing aid users based on four interviews, but it can be concluded that these four people do have an interest in what is being developed in this project. However, it is a must that the solution is quick and easy to use. The youngest and most tech savvy user (User_B) even requested such a solution by himself. He does not see much value in the current hearing care solution as it is. It can be speculated that he must expect that a trainable and personalized solution will provide him the missing value.

The findings based on the interviews also makes it possible to further discuss the risks and assumptions outline in Section 5.3 (page 65):

- **R2** is related to time-consuming training session. The interviews have now outlined that it should take somewhere between 30 seconds and two minutes. The prototype has to be significantly improved to achieve this, but a target has now been defined. This risk has earlier in the report been

classified as "Unlikely" with "Major" impact. Based on the testing and the interviews, it seems reasonable to increase the probability to "Likely".

- **R5** is related to frequently changing user intents. This was not explicitly answered by all interviewees. However, User_A has created customized listening programs e.g. for when he is in a shopping center. Furthermore, he has defined specific adjustments for listening to music as well. This indicates that his intent does not change that often. On the other hand, User_D adjusts the volume multiple times during an evening at social events. However, this seemed to be related to the level of noise. Hence, the solution should be able to take this into account and provide a personalized and automated experience for her. The probability of **R5** is prioritized as "Likely". This is not reflected in the interviews. One might consider adjusting this to "Unlikely". However, this is based on just four interviews.
- **R6** lists the parameters that must be adjusted and the risk of them not being sufficient. User_A would like more adjustability, but the interviewees are all generally happy with the adjustability. This risk is categorized as "Unlike" with "Moderate" impact and should be kept that way.
- **R8** talks about the user becoming uncomfortable with the solution as a result of the added personalization functionality. This seems unlikely to happen. Currently, the users seem to either explore or ignore features. Curious users will explore and try new functionality. Adding additional functionality will likely not make them uncomfortable. The other type of user stick to what they know and ignore the rest. If the personalization feature is ignored, it is also unlikely to cause discomfort for the user. However, if a care seeker accidentally enters the feature, the user might get confused. In this situation, a clear and easy-to-use layout as request by User_D, might be sufficient. The risk is currently evaluated as "Unlikely" and with "Moderate" impact. This should be kept the same or possibly lowered to "Minor" impact. This also links to assumption **A1** which specifying that one solution can be made to fit both user types. Even the tech savvy User_B requests a very simply design. Hence, the assumption that one design can fit both user types seems reasonable.
- **A2** and **A4** assumes that the users are carrying their smartphone at all time, and that it is connected to the internet occasionally to allow data to be synchronized. None of the interviewed users found this as a problem. As a result, the project can safely continue with these assumptions.

The remaining risks and assumptions still exist, but these are either focused on the technical aspect of the solution or not possible to reevaluate based on the interviews.

All the interviews ended with questions related to Over-The-Counter hearing aids (described in Section 10.3, page 147). Both User_C and User_D do not find this sort of hearing aid solution appealing to them. They prefer proper hearing loss assessments and the guidance and care a professional can provide. User_A does not see the value in such hearing aids either. However, if it allows him to sit down and adjust the hearing aids just like the audiologist would, it does sound appealing. User_B is not interested either, however, for a different reason. He appreciates his spot in the test panel, but probably would not get a hearing aid if he had to pay for it. For any type of hearing aid to be interesting to him, it should become more like a gadget. He suggests incorporating digital assistants into the hearing aids. Furthermore, he describes how other companies like Apple and Amazon might soon enter the market with new services like digital assistants.

10.3 Future of hearing aids

This section presents recent advances in how to power a hearing aid to avoid the use of batteries. This is followed with a presentation of how the brain might become an input to hearing aids in the future. Lastly, the section discusses the concept of Over-The-Count (OTC) hearing aids and how they are viewed by the current hearing care industry.

Fuel cells and brain waves

At CES[1], Widex presented their Widex Evoke hearing aid powered by the Widex Energy Cell [110]. The Widex Energy Cell is a fuel cell running on oxygen and methanol and replaces the battery which would normally provide the hearing aid with power [110]. Widex claims that the user can easily supply the fuel cell with methanol in 20 seconds which can then power the hearing aid for 24 hours [109]. The ReSound LiNX 3D hearing aids available for the project are traditional battery driven hearing aids with small button cell batteries. The experience gained with these throughput the prototype development does show that changing the batteries can be a fiddly process. With a fuel cell, the user no longer has to go through this process. Time will tell if the fuel cell is just as fiddly to refill. However, Widex themselves claims this new technology is "hassel free" for the user [110]. This technology is supposed to be available in hearing aids by summer 2019 [110].

Something that can also prove useful for hearing aid users, yet lies a bit further into the future, is electroencephalography (EEG) performed in the ear (ear-EEG). Kappel [51] has researched this topic in collaboration with Widex in order to develop an ear-EEG prototype and verify if the method can be used to detect insulin shock. Insulin shock refers to the event of a low blood sugar levels [5]. Kappel [51] found promising results and managed to develop a prototype with similar size and shape as the earpiece of some hearing aids. Furthermore, Kappel [51] also suggested that ear-EEG could be utilized for monitoring epilepsy as well as sleep.

Already in 2013, Lee et al. [57] talked about using EEG for detecting and monitoring disorders. Furthermore, they also presented the idea of using it to determine the user's intent. In a laboratory experiment they showed a correlation between brain activity and which of two parallel audio streams coming from different sources a user was focusing on [57]. Back in 2013, Lee et al. [57] explained that the technology was not ready yet, but that within 25 years, EEG systems for user intent might be a reality.

Over-The-Counter hearing aids

The solution presented in this project aims to provide a personalized experience to the hearing aid user. As explained shortly, future hearing aids might need such intelligence to function. Kollmeier and Kiessling [53] reckon that self-fitting is one of the elements that might be found in future hearing aid solutions. Providing a self-fitting solution removes the dependency on a professional [53]. Such a solution is likely a bit more elaborate than the prototype designed and developed in this project. However, this project can be seen as initial steps into automation, intelligence, and self-fitting.

Self-fitting is likely to become a requirement for some future solutions. The US Food and Drug Administration (FDA) has decided to establish a category of Over-The-Counter (OTC) hearing aids

[1]CES (short for Consumer Electronics Show) is an annually occurring event allowing companies to come and show case new technology and products. An event with more than 182000 visitors in 2018 which covers everything from fitness to robotics [16].

[106]. OTC hearing aids are supposed to help with self-perceived mild to moderate hearing loss and can be purchased without going to a dedicated hearing aid dispenser [108]. As dispenser might only have a subset of the hearing aids on the market, and they are often sold as a bundle including fitting and follow-up services with a professional [108]. All this makes hearing aid solutions expensive and difficult for people to compare costs and products [108]. However, critique of OTC hearing aids is related to how people can know if they have a mild hearing loss or if they need special care which an OTC hearing aid cannot provide. Furthermore, adjusting the hearing aid is done by a professional today and might be a complicated task for the average user [108]. This is where self-fitting might become a necessity. A meeting with Michael Wörösch, Director of System Engineering and Software Architecture at GN Hearing, revealed that even though the expected user of OTC hearing aids is tech savvy, interfaces providing easy setup in a few steps is required. One way to achieve this might be with the use of intelligent solutions.

According to Michael Wörösch, OTC hearing aids are expected to be cheaper and allow more competitors into the market. Cheaper hearing aids are possible as the audiologists and dispensers are now removed from the equation. Furthermore, less strict regulations are expected (however, FDA has not finalized the regulations yet [106]) which makes it cheaper to develop the devices. This is supplemented with increased production quantities and hardware cost optimization which further reducing the price, according to Michael Wörösch. In the Widex booth at the DSE career fair it was mentioned that a reduced feature set, less advanced signal processing, and a reduction in battery life could also be some of the trade-offs made in order to provide a cheaper solution. From the information picked up at the booth, Widex expects most big hearing care companies to provide an OTC solution, but not keep it as core business.

Michael Wörösch explains that it is difficult to predict if the OTC category is good or bad for the current hearing care industry. It can both be seen as a threat and an opportunity. On one hand, it gives companies that are currently not in the hearing care industry an easy entry to market resulting in more competitor. These companies can now easily develop and sell products for addressing hearing loss. On the other hand, it also enables current companies in the industry to expand their line of products and possible grow the market as more people can afford hearing aids and people might be more likely to try the out. However, this might also cause a negative effect where people who would have bought an expensive hearing aid finds the cheaper solution sufficient. This can cause a self-cannibalizing process to start.

Solutions claiming to improve hearing (however not labeled as hearing aids) is nothing new. Personal Sound Amplification Product (PSAP) is a classification for devices that claim to improve hearing ability which can be found on the market today as any other electronic product [104]. However, these differ from the OTC hearing aid category as they are not allowed to be targeted towards hearing impairment [108]. The OTC hearing aid category is still under development, and no products can legally be sold as an OTC hearing aid just yet [106]. How this new type of hearing aids will affect the market is hard to say without a thorough analysis which is beyond the scope of this project. Today, barriers of entry are high for companies as well as for the consumer. Hopefully, the new OTC category can bring innovative, and possibly disruptive, solutions to the market while encouraging consumers that could benefit from a hearing aid to obtain one.

Chapter 11

Conclusion

The project set out to design and develop a solution for providing hearing aid users with a personalized experience based on the sounds in the current environment. Hearing care solutions that allow users to fine-tune their listening experience already exist. However, there is a lack of learning as the user's preferences are not generalized and only reusable to a certain degree if at all. Furthermore, some users cannot comprehend how to make the appropriate adjustments. This project has developed a simple prototype that abstracts the complexity such that the user only has to make a decision whether one hearing aid adjustment sounds better than the other.

The project has taken a starting point in the ReSound Smart 3D application which provides various functionality in order to allow the user to reduce noise, focus on speech, or fine-tune the amplification of three frequency bands: Bass, midrange, and treble. The aim was to provide a solution which would be capable of learning a user's preferred adjustment of all these parameters and determine the characteristics of the sound environment leading to this specific adjustment. Using this knowledge, it should be possible to later recall the user's preferences next time the user is in an environment which consist of the same acoustic features. To achieve this, the following problem formulation was created:

– How can machine learning be utilized to learn and predict a user's preferred hearing aid configurations for a given environment?

To determine this, the following sub-questions must be answered:

– How can a machine learning architecture be designed to accommodate prediction of multiple values from one input?

– How can personalized machine learning models be created and made accessible to a user?

Finding the appropriate solution to answer these questions required research of multiple machine learning techniques. It was found that a combination of a multi-task branched Convolutional Neural Network (CNN) and the reinforcement learning algorithm Sarsa(λ) seemed appropriate for solving the problem. These two techniques complements each other by combining simplicity and good data utilization from Sarsa(λ) with the strong generalization power of the CNN architecture. Good data utilization was important for the project as the data is collected from interactions with hearing aid users. For this reason, short interactions are desirable, however, this also results in less data.

These two techniques were packaged into a solution consisting of a smartphone application and a cloud entity. The solution was designed such that all the environment classification and hearing aid adjustment predictions can be performed by a smartphone. An instance of a CNN model running on the smartphone takes a Short-Time Fourier Transformation (STFT) spectrogram of the audio in the environment as input. The CNN model then classifies the environment and makes a general prediction of hearing aid adjustments. The outcome of this analysis is then used to search a state-space defined by the user interactions with the Sarsa(λ) algorithms to recall environment dependent preferences. Training of CNN models is only feasible if performed by the cloud entity while the Sarsa(λ) training can be performed on the smartphone.

A prototype of this solution was implemented, however, only with a subset of the functionality. Due to complexity, it was decided that the prototype should be limited to only predicting bass, midrange, and treble preferences and generalize to three high-level environments: All-Around, Restaurant, and Outdoor. Testing showed mixed results with one of the problems being the environment representation. A slight inconsistency in the spectrograms used for training CNN models compared to the spectrograms used for environment analysis were found to cause inaccurate predictions. With that said, the CNN model showed high accuracy when analyzing spectrograms with no inconsistencies. The ability to adjust for changes in user preferences seemed to be challenging for both the CNN architecture and the Sarsa(λ) algorithm. Furthermore, recalling preferences based on the state-space also needs more work. However, it is believed that the design is still valid and that most of the problems can be addressed by different implementation specific configurations. A deeper analysis is needed to determine what these specific configurations should then be.

It can be concluded that the prototype is not yet fully developed and needs more work before the design can be fully proven. However, it is still believed that the problem formulation has been answered. Both the machine learning components have been included into the design as learning entities. Sarsa(λ) is oriented towards short-term learnings and changing preferences whereas the CNN architecture is oriented towards long-term preferences and generalization. This is packaged into a solution that is predominantly running on the user's smartphone and only occasionally needs internet access. The solution is capable of classifying an environment as well as suggesting hearing aid adjustments. Furthermore, both the state-space and the CNN architecture provides a link between the environment classes and the adjustments making it possible to predict the user specific hearing aid configurations based on the environment.

Bibliography

[1] Accord.NET Framework. Framework modules. http://accord-framework.net/docs/html/R_Project_Accord_NET.htm. [Online; accessed 26/05/2019].

[2] Agile Business Consortium. MoSCoW Prioritisation, October 2014. https://www.agilebusiness.org/content/moscow-prioritisation [Online; accessed 17/05/2019].

[3] Agresti, Alan; Finlay, Barbara. *Statistical Methods for the Social Science*, chapter 3.4 Measure of Position, pages 51–55. Pearson Education Limited, 2014.

[4] Aldaz, Gabriel; Puria, Sunil; Leifer, Larry J. Smartphone-Based System for Learning and Inferring Hearing Aid Settings. *Journal of the American Academy of Audiology*, 27(9):732–749, October 2016. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5266590/ [Online; accessed 12/02/2019].

[5] American Diabetes Association. Hypoglycemia (Low Blood Glucose), 11/02/2019. http://www.diabetes.org/living-with-diabetes/treatment-and-care/blood-glucose-control/hypoglycemia-low-blood.html [Online; accessed 31/05/2019].

[6] Anderson, David J.; Carmichael, Andy. Essential Kanban Condensed, 2016. http://leankanban.com/wp-content/uploads/2016/06/Essential-Kanban-Condensed.pdf [Online; accessed 29/05/2019].

[7] Apple Inc. Core ML. https://developer.apple.com/documentation/coreml. [Online; accessed 05/04/2019].

[8] Apple Inc. Siri does more than ever. Even before you ask. https://www.apple.com/siri/. [Online; accessed 12/02/2019].

[9] Banerjee, Shilpi. The Compression Handbook, 2011. https://uk.starkeypro.com/pdfs/quicktips/Compression_Handbook.pdf [Online; accessed 02/06/2019].

[10] Brusey, James; Hintea, Diana; Gaura, Elena; Beloe, Neil. Reinforcement learning-based thermal comfort control for vehicle cabins. *Mechatronics*, 50:413–421, April 2018. https://www.sciencedirect.com/science/article/pii/S0957415817300570 [Online; accessed 22/03/2019].

[11] Caffe2. Integrating Caffe2 on iOS/Android. https://caffe2.ai/docs/mobile-integration.html, 31/03/2018. [Online; accessed 05/04/2019].

[12] Caruana, Rich. Multitask learning. *Machine Learning*, 28:41–75, 1997. https://link.springer.com/content/pdf/10.1023/A:1007379606734.pdf [Online; accessed 24/02/2019].

[13] Cheng, Zhijin; Zhao, Qianchuan; Wang, Fulin; Jiang, Yi; Ding, Jinlei. Satisfaction based Q-learning for integrated lighting and blind control. *Energy and Buildings*, 127:43–55, September 2016. https://www.sciencedirect.com/science/article/pii/S0378778816304364 [Online; accessed 22/03/2019].

[14] Chervine. The Data Science Process with Azure Machine Learning. https://blogs.msdn.microsoft.com/azuredev/2017/02/19/the-data-science-process-with-azure-machine-learning/, 19/02/2017. [Online; accessed 14/02/2019].

[15] François Chollet et al. Keras. https://keras.io, 2015. [Online; accessed 27/05/2019].

[16] Consumer Technology Association. About CES. https://www.ces.tech/About-CES.aspx [Online; accessed 31/05/2019].

[17] CS231n Stanford University. CS231n Convolutional Neural Networks for Visual Recognition - Quick intro. http://cs231n.github.io/neural-networks-1/, 2017. [Online; accessed 27/02/2019].

[18] CS231n Stanford University. CS231n Convolutional Neural Networks for Visual Recognition - Convolutional Neural Networks (CNNs / ConvNets). http://cs231n.github.io/convolutional-networks/, 2018. [Online; accessed 07/02/2019].

[19] Dennis, Jonathan William. *Sound Event Recognition in Unstructured Environments using Spectrogram Image Processing*. Ph.d. thesis, Nanyang Technological University, January 2014. http://www.ntu.edu.sg/home/aseschng/Thesis/JohnDennis_PhDThesis2014.pdf [Online; accessed 07/02/2019].

[20] Dillon, Harvey; Zakis, Justin A.; McDermott, Hugh; Keidser, Gitte; Dreschler, Wouter; Convery, Elizabeth. The trainable hearing aid: What will it do for clients and clinicians? *The Hearing Journal*, 59(4):30–36, April 2006. https://journals.lww.com/thehearingjournal/Fulltext/2006/04000/The_trainable_hearing_aid__What_will_it_do_for.5.aspx [Online; accessed 12/02/2019].

[21] DSE. About DSE. https://studerende.dk/om-dse?lang=en. [Online; accessed 12/04/2019].

[22] Du, Jun; Xu, Yong. Hierarchical deep neural network for multivariate regression. *Pattern Recognition*, 63:149–157, March 2017. https://www.sciencedirect.com/science/article/pii/S0031320316303090 [Online; accessed 24/02/2019].

[23] Dyssegard, Christoffer. Environment dependent adaptive sound experience for hearing aid users. A prestudy, January 2019.

[24] Edwards, Rosalind; Holland, Janet. *What is qualitative interviewing?*, chapter 3 What forms can qualitative interviews take?, pages 29–42. Bloomsbury, 2013. http://eprints.ncrm.ac.uk/3276/1/complete_proofs.pdf [Online; accessed 29/05/2019].

[25] Edwards, Rosalind; Holland, Janet. *What is qualitative interviewing?*, chapter 1 What do the key terms used about qualitative interviews mean?, pages 1–9. Bloomsbury, 2013. http://eprints.ncrm.ac.uk/3276/1/complete_proofs.pdf [Online; accessed 29/05/2019].

[26] Edwards, Rosalind; Holland, Janet. *What is qualitative interviewing?*, chapter 4 Where can qualitative interviews take place?, pages 43–52. Bloomsbury, 2013. http://eprints.ncrm.ac.uk/3276/1/complete_proofs.pdf [Online; accessed 29/05/2019].

[27] Eeles, Peter. Capturing Architectural Requirements, 15/11/2005. https://www.ibm.com/developerworks/rational/library/4706.html [Online; accessed 17/05/2019].

[28] Eriksholm Research Centre. About Eriksholm. https://www.eriksholm.com/about-us. [Online; accessed 11/04/2019].

[29] Gemmeke, Jort F.; Ellis, Daniel P. W.; Freedman, Dylan; Jansen, Aren; Lawrence, Wade; Moore, R. Channing; Plakal, Manoj; Ritter, Marvin. Audio Set: An Ontology and Human-Labeled Dataset for Audio Events, 2017. https://ai.google/research/pubs/pub45857 [Online; accessed 20/04/2019].

[30] Ghosh, Aritra; Kumar, Himanshu; Sastry, P. S. Robust Loss Functions under Label Noise for Deep Neural Networks. https://arxiv.org/pdf/1712.09482.pdf, 27/12/2017. [Online; accessed 28/02/2019].

[31] Gibbons, Sarah. Empathy Mapping: The First Step in Design Thinking, 14/01/2018. https://www.nngroup.com/articles/empathy-mapping/ [Online; accessed 17/05/2019].

[32] Gibbons, Sarah. UX Stories Communicate Designs, 15/01/2017. https://www.nngroup.com/articles/ux-stories/ [Online; accessed 17/05/2019].

[33] Glauner, Patrick Oliver. Comparison of Training Methods for Deep Neural Networks. Master's thesis, Imperial College London, April 2015. https://arxiv.org/pdf/1504.06825.pdf [Online; accessed 26/02/2019].

[34] Glavin, Frank G.; Madden, Michael G. DRE-Bot: A Hierarchical First Person Shooter Bot Using Multiple Sarsa(λ) Reinforcement Learners. In *2012 17th International Conference on Computer Games (CGAMES)*. IEEE, July 2012. https://ieeexplore.ieee.org/document/6314567 [Online; accessed 21/03/2019].

[35] GN Group. GN enhances headset and hearing solutions with state of the art Artificial Intelligence. https://www.gn.com/Newsroom/News/2019/January/GN-enhances-headset-and-hearing-solutions-with-state-of-the-art-Artificial-Intelligence, 07/01/2019. [Online; accessed 04/02/2019].

[36] GN Hearing. GN Hearing launches the most intuitive and personalized hearing solution that adapts to the user's daily life. https://www.resound.com/en/press/ces-ai-2019, 08/01/2019. [Online; accessed 12/02/2019].

[37] Google Developers. Android 8.0 Behavior Changes. https://developer.android.com/about/versions/oreo/android-8.0-changes. [Online; accessed 26/05/2019].

[38] Google Developers. Generalization: Peril of Overfitting. https://developers.google.com/machine-learning/crash-course/generalization/peril-of-overfitting, 01/10/2018. [Online; accessed 24/02/2019].

[39] Gupta, Karan M. Performance Comparison of Sarsa(λ) and Watkin's Q(λ) Algorithms. http://www.karanmg.net/Computers/reinforcementLearning/finalProject/KaranComparisonOfSarsaWatkins.pdf. [Online; accessed 21/03/2019].

[40] Hansen, Colin H. Fundamentals of acoustics. https://www.researchgate.net/publication/228726743_Fundamentals_of_acoustics. [Online; accessed 17/04/2019].

[41] Harley, Aurora. Personas Make Users Memorable for Product Team Members, 16/02/2015. https://www.nngroup.com/articles/persona/ [Online; accessed 17/05/2019].

[42] Hershey, Shawn; Chaudhuri, Sourish; Ellis, Daniel P. W.; Gemmeke, Jort F.; Jansen, Aren; Moore, R. Channing; Plakal, Manoj; Platt, Devin; Saurous, Rif A.; Seybold, Bryan; Slaney, Malcolm; Weiss, Ron J.; Wilson, Kevin. CNN Architectures For Large-Scale Audio Classification. https://arxiv.org/pdf/1609.09430.pdf, 10/01/2017. [Online; accessed 20/04/2019].

[43] Hunter, John; Dale, Darren; Firing, Eric; Droettboom, Michael. Matplotlib - History. https://matplotlib.org/users/history.html, 18/05/2019. [Online; accessed 26/05/2019].

[44] IBM. Enterprise Design Thinking Field Guide, 2018. https://www.ibm.com/cloud/garage/files/design-thinking-field-guide.pdf [Online; accessed 20/04/2019].

[45] Jabra. Jabra Sound+, 14/05/2019. https://play.google.com/store/apps/details?id=com.jabra.moments&hl=en [Online; accessed 01/06/2019].

[46] Jensen, Niels Søgaard. Real-life hearing - part 2: Assessment and solutions. *Widexpress*, (40):1–10, May 2018. http://webfiles.widex.com/WebFiles/9%20502%204870%20001%2001.pdf [Online; accessed 12/02/2019].

[47] Johansen, Benjamin; Petersen, Michael Kai; Korzepa, Maciej Jan; Larsen, Jan; Pontoppidan, Niels Henrik; Larsen, Jakob Eg. Personalizing the Fitting of Hearing Aids by Learning Contextual Preferences From Internet of Things Data. *Computers*, 7(1):1–21, March 2018. http://www.mdpi.com/2073-431X/7/1/1 [Online; accessed 10/02/2019].

[48] Jones, Toni Stokes; Richey, Rita C. Rapid Prototyping Methodology in Action: A Developmental Study, 2000. http://www.uky.edu/~gmswan3/609/Jones_Richey_2000.pdf [Online; accessed 29/05/2019].

[49] Joshi, Prateek. *Artificial Intelligence with Python*, chapter 15 Reinforcement Learning, pages 385–398. Packt Publishing, 2017.

[50] Kabal, Peter. Audio File Format Specifications. http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html, 02/05/2017. [Online; accessed 26/05/2019].

[51] Kappel, Simon Lind. *Development and Characterization of Ear-EEG for Real-Life Brain-Monitoring*. Ph.d. thesis, Department of Engineering at Aarhus University, June 2018. https://ebooks.au.dk/index.php/aul/catalog/download/260/183/799-2?inline=1 [Online; accessed 31/05/2019].

[52] Keidser, Gitte; Dillon, Harvey. Siemens Expert Series: NAL-NL2 - Principles, Background Data, and Comparison to Other Procedures, 22/10/2018. https://www.audiologyonline.com/articles/siemens-expert-series-nal-nl2-11355 [Online; accessed 02/06/2019].

[53] Kollmeier, Birger; Kiessling, Jürgen. Functionality of hearing aids: state-of-the-art and future model-based solutions. *International Journal of Audiology*, 57(sup3):3–28, December 2016. https://www.tandfonline.com/doi/abs/10.1080/14992027.2016.1256504 [Online; accessed 31/05/2019].

[54] Korzepa, Maciej Jan; Johansen, Benjamin; Petersen, Michael Kai; Larsen, Jan; Larsen, Jakob Eg; Pontoppidan, Niels Henrik. Learning preferences and soundscapes for augmented hearing. In *Joint Proceedings of the ACM IUI 2018 Workshops*, volume 2068, Tokyo, Japan, March 2018. CEUR. http://ceur-ws.org/Vol-2068/humanize6.pdf [Online; accessed 08/02/2019].

[55] Korzepa, Maciej Jan; Johansen, Benjamin; Petersen, Michael Kai; Larsen, Jan; Larsen, Jakob Eg; Pontoppidan, Niels Henrik. Modeling User Intents as Context in Smartphone-connected Hearing Aids. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 151–155, Singapore, Singapore, July 2018. UMAP. https://dl.acm.org/citation.cfm?id=3226211 [Online; accessed 11/04/2019].

[56] Käding, Christoph; Rodner, Erik; Freytag, Alexander; Denzler, Joachim. Fine-tuning Deep Neural Networks in Continuous Learning Scenarios. In *Computer Vision - ACCV 2016 Workshop*, pages 588–605, Taipei, Taiwan, March 2017. Springer. https://link.springer.com/chapter/10.1007%2F978-3-319-54526-4_43 [Online; accessed 05/04/2019].

[57] Lee, Adrian K.C.; Drews, Michelle K.; Maddox, Ross K.; Larson, Eric. Brain Imaging, Neural Engineering Research, and Next-Generation Hearing Aid Design. *Audiology Today*, pages 40–47, January 2013. http://faculty.washington.edu/akclee/PDF/ATKCJan13.pdf [Online; accessed 31/05/2019].

[58] Levitt, Harry. Noise reduction in hearing aids: a review. *Journal of Rehabilitation Research and Development*, 38(1):111–121, January 2001. https://pdfs.semanticscholar.org/0822/0ecfbefa75baf369e82144af5c0aa696fc4e.pdf [Online; accessed 02/06/2019].

[59] Mackersie, Carol; Boothroyd, Arthur; Lithgow, Alexandra. A "Goldilocks" Approach to Hearing Aid Self-Fitting: Ear-Canal Output and Speech Intelligibility Index. *Ear and Hearing*, 40(1):107–115, January 2019. https://www.ncbi.nlm.nih.gov/pubmed/29894379 [Online; accessed 12/02/2019].

[60] Mackersie, Carol; Boothroyd, Arthur; Lithgow, Alexandra. A "Goldilocks" Approach to Hearing Aid Self-Fitting: Ear-Canal Output and Speech Intelligibility Index. *Ear and Hearing*,

40(1):107–115, January 2019. https://journals.lww.com/ear-hearing/Abstract/2019/01000/A__Goldilocks__Approach_to_Hearing_Aid.12.aspx [Online; accessed 16/04/2019].

[61] McFee, Brian; Raffel, Colin; Liang, Dawen; Ellis, Daniel P.W.; McVicar, Matt; Battenberg, Eric; Nieto, Oriol. librosa: Audio and Music Signal Analysis in Python. In Huff, Kathryn; Bergstra, James, editor, *Proceedings of the 14th Python in Science Conference*, pages 18–24, Austin, Texas, July 2015. http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf [Online; accessed 26/05/2019].

[62] Microsoft; IBM. *Multimedia Programming Interface and Data Specifications 1.0*, August 1991. http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/Docs/riffmci.pdf [Online; accessed 26/05/2019].

[63] Mohr, Niko; Hürtgen, Holger. Achieving business impact with data. White paper, McKinsey, April 2018. https://www.mckinsey.com/~/media/mckinsey/business%20functions/mckinsey%20analytics/our%20insights/achieving%20business%20impact%20with%20data/achieving-business-impact-with-data_final.ashx [Online; accessed 14/02/2019].

[64] Neural. About Neural. https://www.neural-ai.dk/about/. [Online; accessed 12/04/2019].

[65] Nielsen, Jakob. *Usability Engineering*, chapter 4 The Usability Engineering Lifecycle, pages 71–114. Elsevier Science & Technology, 1994.

[66] Nielsen, Jens Brehm. *Systems for Personalization of Hearing Instruments: A Machine Learning Approach.* Ph.d. thesis, Technical University of Denmark, 2015. http://orbit.dtu.dk/en/publications/systems-for-personalization-of-hearing-instruments(4240f1d7-0e1b-48ed-9765-a5846e2c1cc5).html [Online; accessed 12/04/2019].

[67] NIST. *e-Handbook of Statistical Methods*, chapter 1.3.5.17 Detection of Outliers. National Institute of Standards and Technology, 2013. https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h.htm [Online; accessed 16/04/2019].

[68] NumPy. NumPy. https://www.numpy.org/. [Online; accessed 26/05/2019].

[69] Nwankpa, Chigozie Enyinna; Ijomah, Winifred; Gachagan, Anthony; Marshall, Stephen. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. https://arxiv.org/pdf/1811.03378.pdf, 08/11/2018. [Online; accessed 26/02/2019].

[70] Object Management Group. *OMG Unified Modeling Language*, chapter 18.1 Use Cases, pages 637–644. Object Management Group, March 2015. https://www.omg.org/spec/UML/2.5/PDF [Online; accessed 01/06/2019].

[71] Object Management Group. *OMG Unified Modeling Language*, chapter 17.8 Sequence Diagrams, pages 593–597. Object Management Group, March 2015. https://www.omg.org/spec/UML/2.5/PDF [Online; accessed 01/06/2019].

[72] Object Management Group. *OMG Unified Modeling Language*, chapter 11.2 Structured Classifiers, pages 181–187. Object Management Group, March 2015. https://www.omg.org/spec/UML/2.5/PDF [Online; accessed 01/06/2019].

[73] Object Management Group. *OMG Unified Modeling Language*, chapter 15.2 Activities, pages 371–385. Object Management Group, March 2015. https://www.omg.org/spec/UML/2.5/PDF [Online; accessed 01/06/2019].

[74] One.com. One.com Plan Specifications. https://one-docs.com/guides/en/web-space-comparison-table/. [Online; accessed 26/05/2019].

[75] Oticon. AI-powered tool learns to automate hearing aid settings based on wearer's preferences and behavior. https://www.oticon.com/inside-oticon/news/News/2018/nov-9-kaizn-win-ces-awards, 09/11/2018. [Online; accessed 04/02/2019].

[76] Piskosz, Michael; Dyrlund, Ole. ReSound TS: Flexible and Personalized Sound Therapy. White paper, GN ReSound Group, 2015. https://www.resoundpro.com/en-US/support/linx2support [Online; accessed 08/02/2019].

[77] Ran, Xukan; Chen, Haoliang; Zhu, Xiaodan; Liu, Zhenming; Chen, Jiasi. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1421–1429, Honolulu, USA, April 2018. IEEE. https://ieeexplore.ieee.org/document/8485905 [Online; accessed 05/04/2019].

[78] Rasmussen, Carl Edward; Williams, Christopher K. I. *Gaussian Processes for Machine Learning*, chapter 2.2 Function-space View, pages 13–19. MIT Press, 2006. http://www.gaussianprocess.org/gpml/chapters/ [Online; accessed 13/04/2019].

[79] Rekleitis, Ioannis. Simultaneous localization and uncertainty reduction on maps (SLURM): Ear based exploration. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 501–507, Guangzhou, China, December 2012. IEEE. https://ieeexplore.ieee.org/abstract/document/6491016 [Online; accessed 23/04/2019].

[80] Rieser, Verena; Lemon, Oliver. *Reinforcement Learning for Adaptive Dialogue Systems*, chapter 3 Reinforcement Learning, pages 29–52. Springer, 2011.

[81] Rojanavasu, Pornthep; Srinil, Phaitoon; Pinngern, Ouen. New Recommendation System Using Reinforcement Learning. In *Fourth International Conference on eBusiness*, Bangkok, Thailand, November 2005. http://www.ijcim.th.org/SpecialEditions/v13nSP3/pdf/p23-1-5-New%20Recommendation%20System.pdf [Online; accessed 22/03/2019].

[82] Ruder, Sebastian. An Overview of Multi-Task Learning in Deep Neural Networks. https://arxiv.org/abs/1706.05098, 15/06/2017. [Online; accessed 24/02/2019].

[83] Schoukens, Johan; Rolain, Yves; Pintelon, Rik. Analysis of windowing/leakage effects in frequency response function measurements. *Automatica*, 42(1):27–38, January 2006. https://www.sciencedirect.com/science/article/pii/S0005109805002803 [Online; accessed 26/05/2019].

[84] Schwaber, Ken; Sutherland, Jeff. The Scrum Guide, November 2017. https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf [Online; accessed 29/05/2019].

[85] SciPy. SciPy library. https://www.scipy.org/scipylib/index.html. [Online; accessed 26/05/2019].

[86] Searchfield, Grant D.; Linford, Tania; Kobayashi, Kei; Crowhen, David; Latzel, Matthias. The performance of an automatic acoustic-based program classifier compared to hearing aid users' manual selection of listening programs. *International Journal of Audiology*, 57(3):201–212, 2018. https://www.tandfonline.com/doi/abs/10.1080/14992027.2017.1392048 [Online; accessed 04/02/2019].

[87] Sennheiser. Sennheiser AMBEO Smart Headset. https://en-us.sennheiser.com/finalstop [Online; accessed 30/05/2019].

[88] Shimobaba, Tomoyoshi; Kakue, Takashi; Ito, Tomoyoshi. Convolutional neural network-based regression for depth prediction in digital holography. https://arxiv.org/pdf/1802.00664.pdf, 02/02/2018. [Online; accessed 27/02/2019].

[89] Skagerstrand, Åsa; Stenfelt, Stefan; Arlinger, Stig; Wikström, Joel. Sounds perceived as annoying by hearing-aid users in their daily soundscape. *International Journal of Audiology*, 53(4):259–269, February 2014. https://www.researchgate.net/publication/260093127_Sounds_perceived_as_annoying_by_hearing-aid_users_in_their_daily_soundscape [Online; accessed 07/02/2019].

[90] Skymind. Using Deep Learning & Neural Networks in Android Applications. https://deeplearning4j.org/docs/latest/deeplearning4j-android. [Online; accessed 05/04/2019].

[91] Sokolova, Marina; Lapalme, Guy. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437, July 2009. https://www.sciencedirect.com/science/article/abs/pii/S0306457309000259 [Online; accessed 24/05/2019].

[92] Stalph, Patrick. *Analysis and Design of Machine Learning Techniques*, chapter 2 Introduction to Function Approximation and Regression, pages 11–28. Springer Vieweg, Wiesbaden, 2014. https://link.springer.com/book/10.1007%2F978-3-658-04937-9 [Online; accessed 28/02/2019].

[93] Sutton, Richard S.; Barto, Andrew G. *Reinforcement Learning: An Introduction*, chapter 1 Introduction, pages 1–22. The MIT Press, 2018.

[94] Sutton, Richard S.; Barto, Andrew G. *Reinforcement Learning: An Introduction*, chapter 6 Temporal-Difference Learning, pages 119–140. The MIT Press, 2018.

[95] Sutton, Richard S.; Barto, Andrew G. *Reinforcement Learning: An Introduction*, chapter 5 Monte Carlo Methods, pages 91–118. The MIT Press, 2018.

[96] Sutton, Richard S.; Barto, Andrew G. *Reinforcement Learning: An Introduction*, chapter 12 Eligibility Traces, pages 287–320. The MIT Press, 2018.

[97] Sutton, Richard S.; Barto, Andrew G. *Reinforcement Learning: An Introduction*, chapter 8 Planning and Learning with Tabular Methods, pages 159–193. The MIT Press, 2018.

[98] Taal, Cees H.; Jensen, Jesper. SII-Based Speech Prepocessing for Intelligibility Improvement in Noise. In *Proceedings of the International Conference on Spoken Language Processing*, pages 3582–3586, Lyon, France, August 2013. Interspeech 2013. https://vbn.aau.dk/en/publications/sii-based-speech-prepocessing-for-intelligibility-improvement-in- [Online; accessed 20/04/2019].

[99] TensorFlow. Get started with TensorFlow Lite. https://www.tensorflow.org/lite/guide/get_started#3_use_the_tensorflow_lite_model_for_inference_in_a_mobile_app. [Online; accessed 05/04/2019].

[100] The HDF Group. *High Level Introduction to HDF5*, September 2016. https://support.hdfgroup.org/HDF5/Tutor/HDF5Intro.pdf [Online; accessed 28/05/2019].

[101] Tokic, Michel. Adaptive ϵ-Greedy Exploration in Reinforcement Learning Based on Value Differences. In *KI 2010: Advances in Artificial Intelligence*, pages 203–210, Karlsruhe, Germany, September 2010. 33rd Annual German Conference on AI. https://link.springer.com/chapter/10.1007/978-3-642-16111-7_23 [Online; accessed 23/04/2019].

[102] Townend, Oliver; Nielsen, Jens Brehm; Ramsgaard, Jesper. Real-life Applications of Machine Learning in Hearing Aids. http://www.hearingreview.com/2018/03/real-life-applications-machine-learning-hearing-aids/, 26/03/2018. [Online; accessed 06/02/2019].

[103] Tripp, Steven D.; Bichelmeyer, Barbara. Rapid prototyping: an alternative instructional design strategy. *Educational Technology Research and Development*, 38(1):31–44, January 1990. https://www.researchgate.net/publication/225344439_Rapid_Prototyping_an_Alternative_Instructional_Design_Strategy [Online; accessed 29/05/2019].

[104] U.S. Food and Drug Administration. Regulatory Requirements for Hearing Aid Devices and Personal Sound Amplification Products - Draft Guidance for Industry and Food and Drug Administration Staff, 13/07/2018. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/regulatory-requirements-hearing-aid-devices-and-personal-sound-amplification-products-draft-guidance [Online; accessed 31/05/2019].

[105] U.S. Food and Drug Administration. How to get Hearing Aids, 16/01/2018. https://www.fda.gov/medical-devices/hearing-aids/how-get-hearing-aids [Online; accessed 02/06/2019].

[106] U.S. Food and Drug Administration. Hearing Aids, 30/08/2018. https://www.fda.gov/medical-devices/consumer-products/hearing-aids [Online; accessed 31/05/2019].

[107] Vázquez-Canteli, José R.; Nagy, Zoltán. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied Energy*, 235:1072–1089, February 2019. https://www.sciencedirect.com/science/article/pii/S0306261918317082 [Online; accessed 22/03/2019].

[108] Warren, Elizabeth; Grassley, Chuck. Over-the-counter hearing aids: The path forward. *JAMA Internal Medicine*, 177(5):609–610, May 2017. https://jamanetwork.com/journals/jamainternalmedicine/fullarticle/2606426 [Online; accessed 31/05/2019].

[109] Widex. WIDEX ENERGY CELL - fuel cell technology. https://www.widex.pro/en/evidence-technology/technological-excellence/fuel-cell-technology [Online; accessed 31/05/2019].

[110] Widex. Widex named as CES 2019 Best of Innovation Awards Honoree for the worlds's first battery-free hearing aid, 07/01/2019. https://global.widex.com/en/news/ces-2019-best-of-innovation-awards-honoree [Online; accessed 31/05/2019].

[111] Widex. New insights from WIDEX EVOKE^{TM} data bring the benefits of Artificial Intelligence to hearing impaired. https://global.widex.com/en/news/new-insights-from-widex-evoke, 28/08/2018. [Online; accessed 04/02/2019].

[112] Willmott, Cort J.; Matsuura, Kenji. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30:79–82, December 2005. https://www.int-res.com/articles/cr2005/30/c030p079.pdf [Online; accessed 28/02/2019].

[113] Wirth, Rüdiger; Hipp, Jochen. CRISP-DM: Towards a standard process model for data mining. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.198.5133, 2000. [Online; accessed 15/02/2019].

[114] Xu, Xin; Zuo, Lei; Huang, Zhenhua. Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences*, 261:1–31, March 2014. https://www.sciencedirect.com/science/article/pii/S0020025513005975 [Online; accessed 22/03/2019].

[115] Yang, Qi; Hahn, Shira; Chang, Bill; van den Berg, Almer, Olsen, Greg. Affordance of Real-Time Personalization and Adaptation of Hearing Aid Settings. In *HCI International 2017 - Posters' Extended Abstracts*, pages 309–316, Vancouver Canada, July 2017. HCI International. https://link.springer.com/chapter/10.1007%2F978-3-319-58753-0_46 [Online; accessed 12/04/2019].

[116] Yosinski, Jason; Clune, Jeff; Bengio, Yoshua; Lipson, Hod. How transferable are features in deep neural networks? In Ghahramani, Zoubin; Welling, Max; Cortes, Corinna; Lawrence, Neil D.; Weinberger, Kilian Q., editor, *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3320–3328, 2014. http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-n%E2%80%A6 [Online; accessed 30/05/2019].

Acronyms

A# - Assumption number #

CNN - Convolutional Neural Network

CTO - Cheif Technology Officer

EEG - Electroencephalography

FR# - Functional Requirement number #

GDPR - General Data Protection Regulation

MAE - Mean Absolute Error

MSE - Mean Squared Error

MVP - Minimum Viable Product

NFR# - Non-functional Requirement #

NR - Noise reduction

OTC - Over-The-Counter

OTE - Online training entity

PSAP - Personal Sound Amplification Product

R# - Risk number #

ReLU - Rectified Linear Unit

RMSE - Root Mean Squared Error

SDK - Software Development Kit

SP - Speech focus

STFT - Short-Time Fourier Transformation

TD - Temporal-Difference

WNR - Wind noise reduction

Glossary

Activation function - A function contained within a neuron to transform the output of the neuron. It makes it possible to achieve non-linear mappings.

Average Accuracy - An accuracy measure across multiple classes that increases as the number of true positives and true negatives increases.

Branched network - A neural network architecture that splits a network in multiple branches such that each branch can creates its own mapping from input to output.

Care seeker - Describes one of the two hearing aid user types which the project deals with. The care seeker is careful and can be uncomfortable around new technology. This type of user is reluctant to try new things without guidance.

Control problem - Used within the field of reinforcement learning to describe the problem of determining the optimal policy.

Convolutional layer - A type of layer used in CNNs responsible for scanning an input and detecting features.

Convolutional Neural Networks - A specific neural network architecture including convolutional and pooling layers that aim to extract features from an input.

Cost function - Quantifies the prediction error of neural network such that optimization can be performed.

Decay factor - A configurable parameter specific to reinforcement learning algorithms that utilizes eligibility traces. It defines how quickly values in the trace matrix decay. A quick decay (low decay factor) results in past states being less affected by the current reward. Denoted by λ.

Discounting factor - A configurable parameter utilized in reinforcement learning algorithms to specify how much emphasis should be put on expected future rewards. Denoted by γ.

Eligibility traces - A specific component of the Sarsa(λ) algorithm which keeps track of states and actions. The information is to update multiple state-action values when receiving a reward.

Evaluation problem - Used within the field of reinforcement learning to refer to the problem of evaluating a specific policy.

Fine-tuning - The act of adjusting individual hearing aid configurations to achieve a desired listening experience. This can be needed in challenging sound environments.

Fitting - Describes the procedure of ensuring that a hearing aid is configured to treat the specific hearing loss of an individual. The process is carried out by a professional.

Frequency response - The characteristics of how a microphone responds to a range of frequencies.

Fully connected layer - A layer that has connections from all the neurons in the previous layer to each and every neuron in this layer.

Learning rate - Used to describe how quickly a machine learning algorithm adjusts. A general term, but specifically denoted by α within the field of reinforcement learning.

Listening program - A predefined set of hearing aid configurations designed to cope with specific sound environments.

Macro F Score - An accuracy measure that combines Macro Precision and Macro Recall. Weighting can be applied to emphasize Precision or Recall.

Macro Precision - The ratio between correctly predicted samples for certain class compared to all samples predicted to fit in that class. The measure is calculated across multiple classes.

Macro Recall - A ratio between correctly predicted samples for certain class compared to all samples that should have been predicted to fit in the class. The measure is calculated across multiple classes.

Multi-task learning network - A neural network architecture for solving multiple problems using a single network.

NAL-NL2 - A generic fitting formula for improving speech intelligibility.

Sound environment - The project defines the sound environment as the sounds surrounding a user. This includes any audio (e.g. background noise, conversations, wind noise etc.) collected from the user's location.

Off-policy - Describes reinforcement learning algorithms that utilizes a behavior policy to learn a target policy.

On-policy - Describes reinforcement learning algorithms that acts based on the same policy as the one being optimized.

Over-The-Counter hearing aids - A new category of hearing aids that will make it possible for people to obtain a hearing aid without seeing a professional.

Personal Sound Amplification Product - Electronic equipment sold to improve hearing. However, specifically not allowed to target hearing impairment.

Policy - In this project it refers to the rules that define how a reinforcement learning agent should behave.

Pooling layer - A layer utilized in the CNN architecture to reduce the size of an input while keeping the important features.

Proactive user - One of the two hearing aid user types this project is dealing with. The proactive user is comfortable around new technology and likes to explore.

Q-table - A matrix containing all the state-action values for the entire state-space.

Reinforcement learning - A specific branch of machine learning where an agent interacts with an environment and learns how to achieve a goal by receiving rewards from the environment.

Sample rate - Describes how frequently a sample is taken. Used in the context of audio recording in this project.

Sarsa(λ) - The reinforcement learning algorithm chosen for the solution developed in the project. It is a variation of the Sarsa algorithm and differs by utilizing eligibility traces.

Short-Time Fourier Transformation spectrograms - A specific unbiased type of spectrogram used for representing the sound environments.

Single-task learning network - A neural network architecture that utilizes one network for solving only one problem.

Smart Buttons - Buttons from the ReSound Smart 3D application that are easily accessible. When enabled, they apply predefined hearing aid adjustments allowing the user to quickly achieve a certain listening experience.

Sound Enhancer - A feature of the ReSound Smart 3D application that provides the user with fine-tuning capabilities to achieve a desired listening experience. The Sound Enhancer makes it possible to adjust bass, midrange, treble as well as noise reduction and speech focus features depending on the chosen of listening program.

Sound profile - A collection of hearing aid configurations presented to a user as a pairwise comparison during training sessions.

Soundscape - Soundscape is used to describe the sounds that make up the acoustic scene and is used interchangeably with ”sound environment” in this project.

SoundSense Learn - An intelligent solution developed by Widex that allows a user to achieve personalized hearing aid adjustments through pairwise comparison of multiple sound profiles.

State-action value - Used to describe the value associated with taking a specific action in a specific state of a state-space.

State-space - Used to refer to all the states in a reinforcement learning agent's environment. In this project, it is used to denote the possible hearing aid configuration options for the considered environments.

Temporal-Difference Prediction - An approach to solving the policy evaluation problem. Used in this project to contribute to the decision of when to terminate a training session.

Training session - Used to refer to the act of training a machine learning model. Mostly used to refer to the user iteratively performing pairwise comparisons of sound profiles to teach USense SoundLearner about user preferences.

USense SoundLearner - The name of the solution developed in this project. Mostly used to refer to the smartphone application component.

Appendices

Appendix A

Reinforcement learning

A.1 Q-learning - Numerical example

The section walks though a numerical example of how the action-value function is estimated in Q-learning. Improving the estimates allows an agent to act optimally in an environment. The following example is very simple and only goes through a few updates. Nevertheless, it should suffice to understand how the value of a state-action pair is estimated. The example uses Q-learning, but the same process can be applied to other algorithms as well such as Sarsa.

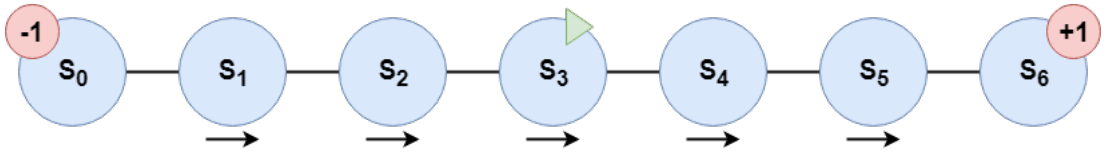


Figure A.1: Simple environment to exemplify how a Q-learning agent learns how to act. Made with draw.io online tool.

Figure A.1 presents a simple environment consisting of seven states (the blue circles). S_0 and S_6 are terminal states, and the agent receives a reward of -1 and 1, respectively, when reaching the states. The goal of the agent is to maximize the return meaning the agent should try to reach S_6. The green triangle at S_3 indicates that this is the start state. When the agent reaches a terminal state, its position is reset to S_3. To simplify the example, the agent starts in the same state on each run. Furthermore, none of the intermediate states give the agent any reward. The agent can choose to go either left or right, and the black arrows underneath the states indicates the optimal action in that state: The agent should always choose to go right. Of course, the agent does not have this information. This needs to be learned by the agent through an iterative trial and error process.

The action space and state space can be defined as follows:

$Action = \{\leftarrow, \rightarrow\}$

$States = \{0, 1, 2, 3, 4, 5, 6\}$, 0 and 6 are terminal, 3 is start

The following learning rate and discounting factor have been chosen for completeness:

Learning rate: $\alpha = 0.1$

Discounting factor: $\gamma = 1$

The Q-table has one entry for every action in every state. All entries are initialized as zero, and each row is a state and the columns represent the actions. This example has defined two possible actions for seven states. For this reason, the following matrix has seven rows and two columns:

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

As the example is based on Q-learning, the Q-table will be updated using Equation A.1. $Q(S, A)$ denotes a specific entry in the Q-table, α and γ are as defined above, and $max_a Q(S', a)$ describes choosing the action with the highest value for a given state S'.

$$Q(S, A) \leftarrow Q(S, A) + \alpha * (R + \gamma * max_a Q(S', a) - Q(S, A)) \tag{A.1}$$

First episode

The agent starts in S_3 and uses the Q-table to look up which actions is the best. However, all actions are rated equally currently. The agent chooses randomly and picks $\rightarrow$ (denoted as 1 because A is zero-indexed). This bring the agent to state to S_4 which does not bring a reward. The numbers are entered into Equation A.1 to update the Q-table.

$S = 3, A = 1, R = 0, S' = 4$
$Q(3, 1) \leftarrow 0 + 0.1 * (0 + 1 * 0 - 0), Q(3, 1) = 0$

The agent is now in S_4 and has to chose an action. Once again, the agent has to choose randomly as the value for all actions in S_4 are estimated to be zero. The agent picks $\rightarrow$ and gets to S_5 which does not return a reward.

$S = 4, A = 1, R = 0, S' = 5$
$Q(4, 1) \leftarrow 0 + 0.1 * (0 + 1 * 0 - 0),\ Q(4, 1) = 0$

In S_5 the agent once again has to choose a random action. It decides to go right ($\rightarrow$) and ends up in S_6 which returns a reward of 1. Since a reward has now been returned, the Q-table update is non-zero. Due to the learning rate (α) specified earlier, going right in S_5 is not valued at 1, but 0.1. Having a low learning rate means that the agent will have to perform the same actions over and over again. However, it also ensures that the agent does not get biased towards specific actions that are only locally good or non-static.

$S = 5, A = 1, R = 1, S' = 6$
$Q(5,1) \leftarrow 0 + 0.1 * (1 + 1 * 0 - 0)$, $Q(5,1) = 0.1$

The agent has now finished an episode and the Q-table has been updated and now looks as follows:

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.1 \\ 0 & 0 \end{bmatrix}$$

Second episode

The agent is reset back to S_3 and the second episode starts. The Q-table is carried from the first episode into the second episode meaning the agent can make use of what it learned in the first episode to further update the Q-table.

The action entries in the Q-table for S_3 are still only zeros. The agent chooses randomly and picks $\rightarrow$. The Q-table is updated accordingly.

$S = 3, A = 1, R = 0, S' = 4$
$Q(3,1) \leftarrow 0 + 0.1 * (0 + 1 * 0 - 0), Q(3,1) = 0$

The agent is in S_4 and once again has to choose randomly among the actions. Once again it decides to go right and ends up in S_5. Last episode, the agent learned that being in S_5 and taking action $\rightarrow$ returned a reward. That specific state-action pair has a value of 0.1 which is now used when updating the Q-table entry for $\{S_4, \rightarrow\}$. As a result, the Q-table entry for going right in S_4 is valued 0.01. It is not valued as much as going right in S_5 because there is no direct reward to find in S_5. However, it does have a non-zero value because the agent now knows that $\{S_4, \rightarrow\}$ will get it to S_5 where it can receive a reward if the appropriate action is taken.

$S = 4, A = 1, R = 0, S' = 5$
$Q(4,1) \leftarrow 0 + 0.1 * (0 + 1 * 0.1 - 0)$, $Q(4,1) = 0.01$

Being in S_5, the agent learned in the first episode that going right returns a reward. Therefore, the agent decides to go right again. This time it also returns a reward. The agent becomes increasingly more sure that going right in S_5 is a good choice. Hence, $\{S_5, \rightarrow\}$ state-action pair value is increased.

$S = 5, A = 1, R = 1, S' = 6$
$Q(5,1) \leftarrow 0.1 + 0.1 * (1 + 1 * 0 - 0.1)$, $Q(5,1) = 0.19$

At the end of the second episode, the Q-table looks as follows:

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.01 \\ 0 & 0.19 \\ 0 & 0 \end{bmatrix}$$

As it wants to maximize its reward, the agent has now learned, after two episodes, that action 1 ($\rightarrow$) is the smartest action to take in S_4 and state S_5. To learn the optimal behavior for the remaining states (the one presented with black arrows in Figure A.1), many more episodes must be iterated through.

This example, on purpose, chose to only go right to illustrate in as few steps as possible how the Q-table evolves. In a real scenario, the agent will have to learn that going right is the best choice. When the Q-table starts to be populated, this will become clear for the agent. However, till then, it must rely on luck to discover which terminal state returns a positive reward and which returns a negative reward.

A.2 Dyna-Q

The Research chapter presented that Q-learning (Section 4.3.1, page 34) and Sarsa (Section 4.3.2, page 36) do not require a model to operate. Dyna-Q is a variation of Q-learning that incorporates a model and a planning sequence [97]. The algorithm learns a model of the environment as it gains experience (interacts with the environment). This model can then be used to plan how to act and improve the action-value function [97].

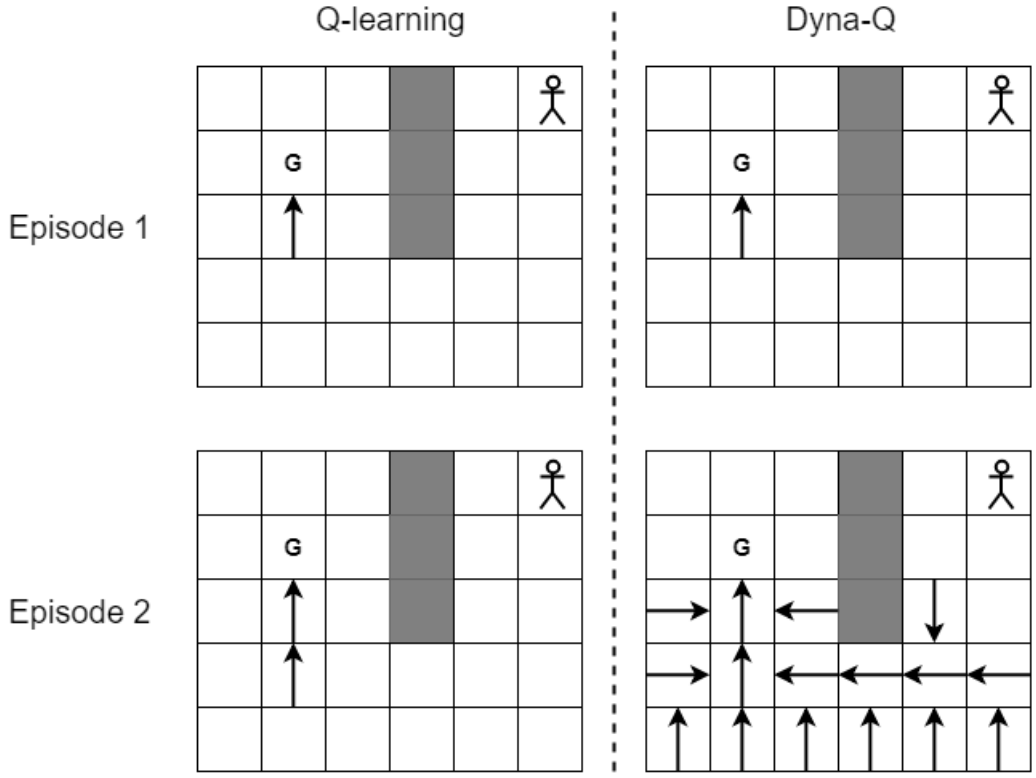


Figure A.2: Illustration of policy improvement over time for Q-learning and Dyna-Q. Inspired by: [97]. Made with draw.io online tool.

To understand the differences between the regular Q-learning method and Dyna-Q, consider Figure A.2. The arrows depict what the policy specify as being the best action in a specific state. Ordinary Q-learning is on the left and Dyna-Q is on the right. The top example shows the policy after one episode and the bottom example shows the policy after two episodes. An episode starts when the agent is placed in the starting location and ends when the goal is reached. The figure describes that after one episode, both the Q-learning agent and the Dyna-Q agent are aware of the best action to take in the state leading up to the goal. This is known as the first episode has elapsed and the agent received a reward when reaching the goal from that specific state. So far, both algorithms are performing the same. The difference between the approaches is apparent after the second episode. After the second episode, the Q-learning agent knows how to act in two states [97]. In the first episode it learned what state and action immediately precedes the goal. Now, after the second episode, it has learned what state and action precedes the state and action learned in the first episode. Dyna-Q, however, has learned a partial model of what state transitions are allowed, what action can be taken to reach the state, and which state was followed by a reward. By remembering the observations, Dyna-Q can iterate over these observations and create a much richer policy that describes what actions to take in many more states compared to Q-learning after this few episodes [97]. This difference is illustrated in the lower part of Figure A.2 where only two arrows are present for Q-learning compared to significantly more in the Dyna-Q example. The full Dyna-Q algorithm presented by Sutton and Barto in chapter 8 of their reinforcement learning book

[97] can be found in Appendix A.6.

The benefit of using an algorithm like Dyna-Q which incorporates planning is its ability to learn from limited experience with a real environment [97]. The limited real experience can be supported by simulated experiences created based on previous observations and the model of the environment. However, even though there are advantages in terms of the limited amount of real experience required and the quick convergence, this sort of methods also have disadvantages. A model must be implemented and updated, and the algorithm must keep track of already observed states and actions. These two elements make for a more complex solution. Furthermore, the environment is assumed to be deterministic and problems can occur if it changes [97]. As a result of a changing environment, the model can become incorrect. If the changes in the environment does not affect the optimal path from a starting state to the goal, the changes will not be discovered even though the changed environment might allow for a more optimal policy. This is the problem of the exploration and exploitation trade-off. A version of Dyna-Q named Dyna-Q+ tries to make it possible to detect and correct for changes in the environment [97]. This is done by keeping track of how much time has passed since a specific state-action has been observed. In Dyna-Q+, the algorithm gets rewarded during the planning iterations for taking an action in a specific state proportionally to how much time has passed since such an observation was made while interacting with the real environment. This encourages the algorithm to try state-action pairs that have not been observed in a while [97].

The fact that limited real environment interaction is needed and that it quickly converges makes Dyna-Q an interesting approach for this project. However, it is not the only method that is capable of learning the value of multiple steps after just one episode. The version of Q-learning depicted in Figure A.2 is a one-step approach. This one-step idea can be expanded to n-step approaches which updates n steps preceding the goal [96]. This can be further developed into methods such as Sarsa(λ) which can adjust all steps taken weighted by the distance to the goal [96]. Sarsa(λ) is presented in more details in Appendix A.3.

A.3 Sarsa(λ)

The Dyna-Q method for reinforcement learning is presented in Appendix A.2 and is a more complex method compared to Q-learning and Sarsa presented in Section 4.3.1 and Section 4.3.2 of the main report respectively. The added complexity, however, makes it possible to converge fast and learn more from less data [97]. Alternatively, Sarsa(λ) can be considered. The following text introduces Sarsa(λ) and presents how it is different and what advantages it provides.

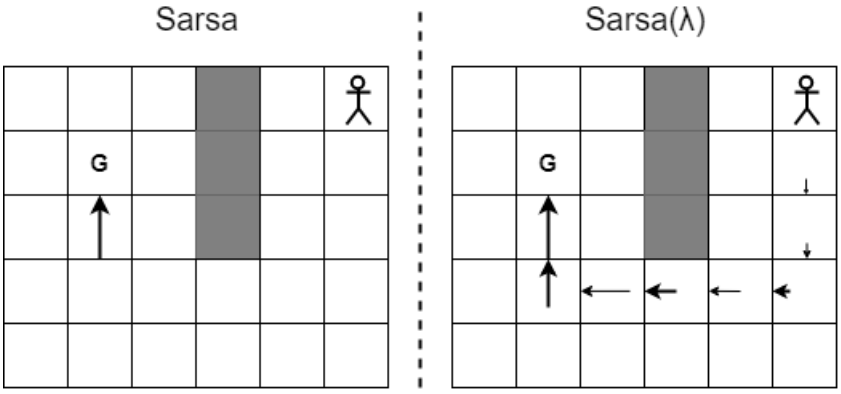


Figure A.3: Illustration of Sarsa and Sarsa(λ) and their learning ability over one episode. Larger and more bold arrows indicate a higher value. Inspired by [96]. Made with draw.io online tool.

Figure A.3 illustrates the different learning abilities for Sarsa (on the left) and Sarsa(λ) (on the right) when an agent moves towards the goal, "G". Sarsa(λ) can estimate the value of multiple states and actions during an episode while the simpler Sarsa only learns what action to take in one state [96]. Apart from estimating a value for more than just on state-action pair, Sarsa(λ) keeps track of how much each of the states and the actions taken contributes to reaching the goal [96]. Figure A.3 shows the contribution as the size and thickness of the arrow in each state. A larger and thicker arrow means that this state-action pair contributed more to reaching the goal, and a smaller arrow means it contributed less. The estimated value of the state-action pair is adjusted accordingly [96]. The full Sarsa(λ) algorithm presented by Glavin and Madden [34] can be found in Appendix A.5.

This gradual weighting is done using eligibility traces which are a sort of short-term memory that lasts no longer than a single episode [96]. In order to understand how these eligibility traces work, consider the following snippets from the Sarsa(λ) algorithm [34]:

$$\delta \leftarrow R + \gamma * Q(S', A') - Q(S, A) \tag{A.2}$$

$$e(S, A) \leftarrow \gamma * \lambda * e(S, A) \tag{A.3}$$

$$Q(S, A) \leftarrow Q(S, A) + \alpha * \delta * e(S, A) \tag{A.4}$$

As can be seen, the Sarsa(λ) algorithm (Appendix A.7) differs from basic Sarsa (see algorithm in Appendix A.5). δ serves as the error measure and is defined by Equation A.2 [34]. It is similar to the error measure used in the basic Sarsa algorithm [94], however, the algorithm presented in Appendix A.5 does not extract the error measure into its own variable. e is the eligibility trace matrix which has the same dimensions as Q. It holds the eligibility value for a specific state-action pair like Q holds the estimated value of taking a specific action in a specific state [34]. Updating e is based on prior eligibility values for this state-action pair, a discounting factor denoted by γ, and a trace decay value denoted by λ [96] as defined by Equation A.3 [34]. At the beginning of each episode, e contains only zeros. Whenever as state-action pair is visited, the entry in the eligibility matrix is increased. Hereafter, the estimated

state-action pair value is updated using Equation A.4 [34]. By substituting the definition of δ from Equation A.2 into Equation A.4, the reader can find great similarity with the state-action pair update done in the basic Sarsa algorithm. However, Equation A.4 makes use of the eligibility trace value which impacts the learned estimate of the state-action pair.

Every time an agent takes a step (makes a transition from S to S' by action A), Sarsa(λ) iterates over the eligibility trace (e) and the estimated state-action values (Q) using Equation A.3 and Equation A.4 respectively [34]. Neither of the equations do much for states and actions that have not been visited yet as $e(S, A)$ is zero in those cases. However, whenever the agent takes an action, one entry in the eligibility matrix becomes non-zero. At this point, the agent has taken a step and both e and Q is updated. As time passes, and the agent takes more steps, the prior eligibility values should gradually become less weighted as depicted in Figure A.3. This is what Equation A.3 is achieving using γ and λ [96]. If the state-action pair is not visited again (or does not lead to the goal), it slowly decays and, hence, ends up having less importance in the final path from start to goal.

Sarsa(λ) is a more complex approach compared to other algorithms presented in this report, but it provides a number of benefits in return. This is an approach to utilize if only limited data is available [96]. Since Sarsa(λ) learns multiple state-action value estimates for every step, it makes an effort to extract as much information as possible to improve the behavior of the agent. Dyna-Q (Appendix A.2) also utilizes the data well as it creates a model of the environment which is used to generate more data which the agent can learn from. However, Sarsa(λ) does not require a model as Dyna-Q does and therefore is less complex. The one-step approach presented in Figure A.3 requires more episodes and data before the entire path from start to goal can be estimated. Other approaches like n-step approaches (not covered in this report) are also able to create estimates for multiple steps before a goal, however, λ approaches tends to generalize better [96]. As an alternative to Sarsa(λ), Q-learning also exist in an eligibility trace variation. Research has shown that Q(λ) converges to a more optimal policy compared to Sarsa(λ), but Sarsa(λ) converges faster [39].

A.4 Q-learning algorithm pseudocode

Initialize $Q(s, a)$ arbitrarily
for *each episode* **do**
  Initialize S
  for *each step in episode* **do**
    Choose A from S using policy derived from Q
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha * (R + \gamma * max_a Q(S', a) - Q(S, A))$
    $S \leftarrow S'$
    Exit loop when S is terminal
  end
end

Algorithm 1: Q-learning algorithm pseudocode. Source: [94].

A.5 Sarsa algorithm pseudocode

Initialize $Q(s, a)$ arbitrarily
for *each episode* **do**
  Initialize S
  Choose A from S using policy derived from Q
  for *each step in episode* **do**
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q
    $Q(S, A) \leftarrow Q(S, A) + \alpha * (R + \gamma * Q(S', A') - Q(S, A))$
    $S \leftarrow S'$
    $A \leftarrow A'$
    Exit loop when S is terminal
  end
end

Algorithm 2: Sarsa algorithm pseudocode. Source: [94].

A.6 Dyna-Q algorithm pseudocode

Initialize $Q(s, a)$ and $Model(s, a)$
for *each episode* **do**
  Initialize S
  for *each step in episode* **do**
    Choose A from S using policy derived from Q
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha * (R + \gamma * max_a Q(S', a) - Q(S, A))$
    $Model(S, A) \leftarrow R, S'$
    for *each planning iteration* **do**
      $S_p \leftarrow$ Select randomly a previously observed state
      $A_p \leftarrow$ Select randomly a previously taken action in S_p
      $R_p, S'_p \leftarrow Model(S_p, A_p)$
      $Q(S_p, A_p) \leftarrow Q(S_p, A_p) + \alpha * (R_p + \gamma * max_a Q(S'_p, a) - Q(S_p, A_p))$
    end
    $S \leftarrow S'$
    Exit loop when S is terminal
  end
end

Algorithm 3: Dyna-Q algorithm pseudocode. Source: [97]. Slightly modified for clarification.

A.7 Sarsa(λ) algorithm pseudocode

Initialize $Q(s, a)$ and $e(s, a)$
for *each episode* **do**
  Initialize S
  Choose A from S using policy derived from Q
  for *each step in episode* **do**
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q
    $\delta \leftarrow R + \gamma * Q(S', A') - Q(S, A)$
    $e(S, A) \leftarrow 1$
    for *all* S_e *and* A_e **do**
      $Q(S_e, A_e) \leftarrow Q(S_e, A_e) + \alpha * \delta * e(S_e, A_e)$
      $e(S_e, A_e) \leftarrow \gamma * \lambda * e(S_e, A_e)$
    end
    $S \leftarrow S'$
    $A \leftarrow A'$
    Exit loop when S is terminal
  end
end

Algorithm 4: Sarsa(λ) algorithm pseudocode. Source: [34].

Appendix B

Exploratory prototype

This appendix describes the implementation of an exploratory prototype. The prototype is developed to gain a deeper understanding of Q-learning, Sarsa, Sarsa(λ), and Dyna-Q. This makes it possible to evaluate if there is any specific characteristics in how they behave in an implementation compared to each other. Furthermore, such a prototype also makes it possible to validate if it is viable to implement the mentioned algorithms as part of a smartphone application.

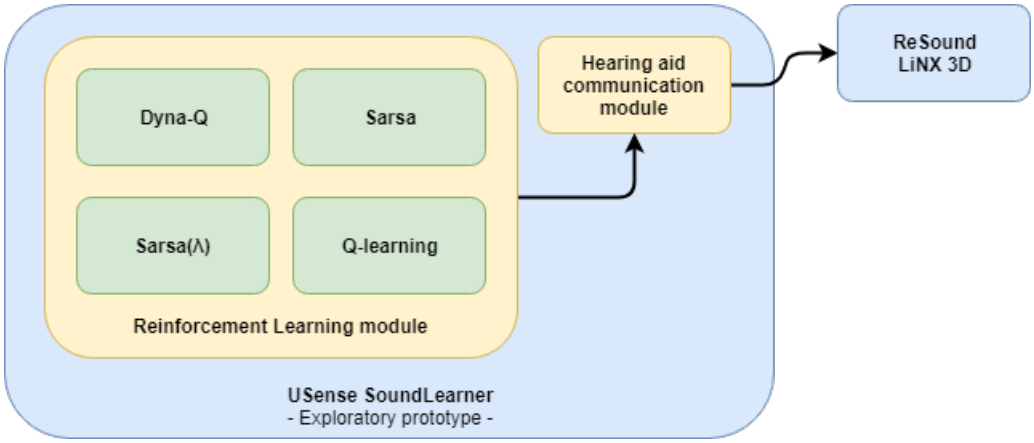


Figure B.1: Exploratory prototype implementing the four reinforcement learning algorithms considered in this project as well as a module for hearing aid communication. Made with draw.io online tool.

Figure B.1 depicts the architecture of the exploratory prototype. As the main focus of this prototype is on the reinforcement learning aspect, the prototype only implements minimal functionality. The prototype is developed as a smartphone application containing a reinforcement learning module and a communication module for the hearing aids. The reinforcement learning module contains the four algorithms which the prototype aims to cover. Furthermore, the hearing aid communication module makes it possible to apply changes to a set of ReSound LiNX 3D hearing aids such that auditory feedback is present. If no auditory feedback is present, it does not make sense to assess the reinforcement learning methods as there is no indication whether they are performing good or bad. Moreover, input to the algorithms is random if there is no feedback to base the input on. Random input is unlikely to make

sense to the algorithms. As a result, they cannot determine what to do and no conclusions can be made in relation to their performance.

Exploratory prototype findings

It can be confirmed that all four reinforcement learning algorithms (Q-learning, Sarsa, Sarsa(λ), and Dyna-Q) run just fine on a smartphone. The smartphone used for the trail runs is a OnePlus 6T from late 2018 making it a rather new smartphone. This can result in biased result as it might not represent the average smartphone on the market. However, it does show that these algorithms can run on a smartphone. With this knowledge, the project can continue as planned and implement a more complete prototypes as smartphone applications as well. The aim of providing a solution which a user can bring with them as a smartphone application still seems feasible.

In terms of finding user preferences, no obvious performance difference among the four algorithms is found. Running automated test trying to achieve a gain of -6 dB for bass, midrange, and treble shows that all four algorithms require between 61 and 101 iterations on the very first training session to reach this state (each iteration moved ± 2 dB for a single parameter). By automating the tests, user inconsistency does not interfere with the testing process. However, as the test is trying to achieve a specific state and not a specific auditory experience, it might not simulate real-life usage perfectly. Furthermore, this testing is not performed as a pairwise comparison. A positive reward of one is given if the new states is closer to the target state, otherwise a reward of negative one.

All the test findings are documented below. It can be seen that none of the algorithms seems to perform particularly well or consistently. Using any of them in a final solution which require user interaction might seem unacceptable. However, not much work has been put into optimizing hyper-parameters to achieve good learning results just yet. Furthermore, the task of finding a particular state, as mentioned, might not simulate real-life. An actual user is not aiming for particular values, but for a certain auditory experience which might be achieve to a satisfactory level by more than a single state.

Q-learning

Figure B.2 and Figure B.3 indicate how many iterations are needed to find a desired state using the Q-learning algorithm. Figure B.2 starts over between tests meaning all learnings are cleared. Hence, this figure shows how many iterations are needed the first time a user is using the solution. Figure B.3 shows how many iterations are needed when learnings are kept. The first rerun is based off Test10 from Figure B.2 and shows how the algorithm becomes better over time to find the same state.

It can be concluded that Q-learning needs an average of 61 iterations to find a desired states at first time usage. However, this varies a lot and anything from 22 iterations to 110 can be expected. Furthermore, it can be concluded that Q-learning quickly finds the desired state if it can apply prior learnings. After a seeking the same state three times, it is capable of finding it in 8 iterations.

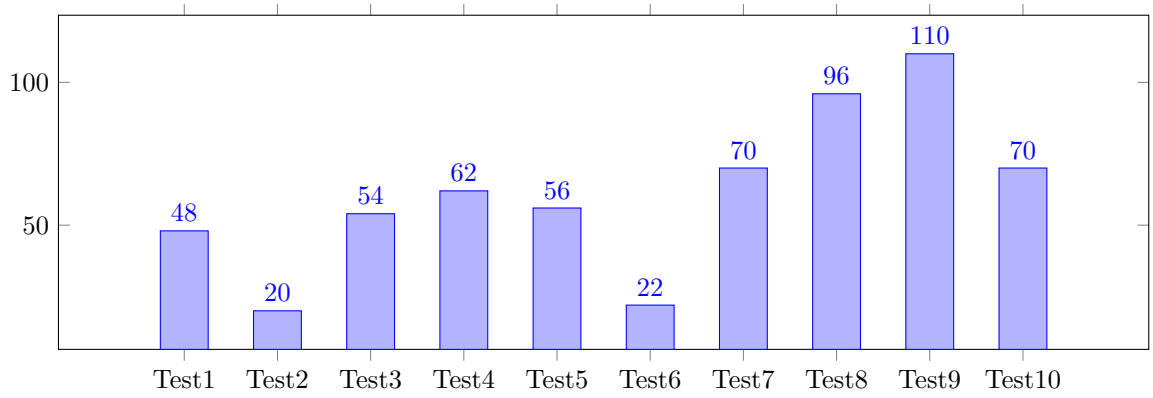


Figure B.2: 10 independent runs of the Q-learning algorithm. Each bar indicates the number of iterations to reach a specific desired states. All learnings are cleared between tests.

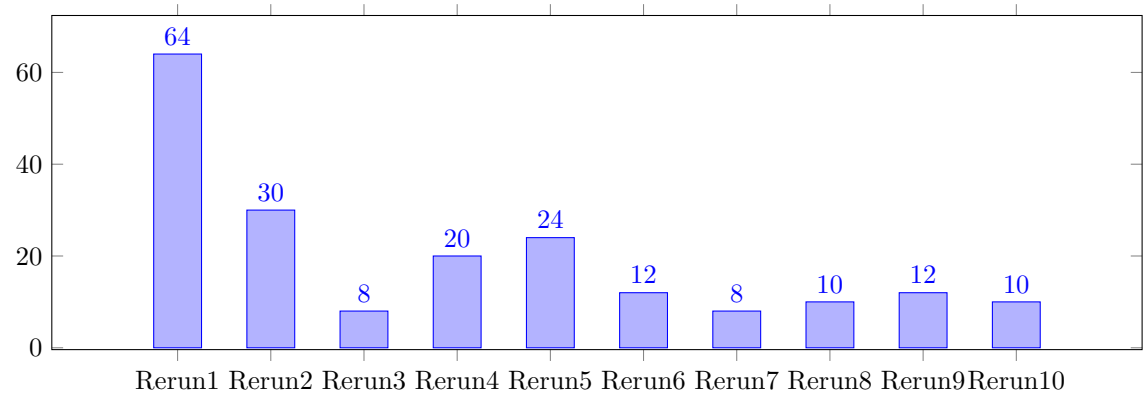


Figure B.3: 10 dependent reruns based on Test10 from B.2. The bars indicate how many iterations are needed for the Q-learning algorithms to reach a desired states. The state is kept the same for all reruns and learnings are kept.

Sarsa

The test for Sarsa is setup similarly to the Q-learning algorithm test. It can be found that Sarsa needs an average of 99 iterations to find a desired state, however, this can differ from 38 to 204 iterations (see Figure B.4). Compared to Q-learning, Sarsa is more reliable in quickly finding the same state again. After seeking the same state twice, it is capable of finding it in just 10 iterations (see Figure B.5).

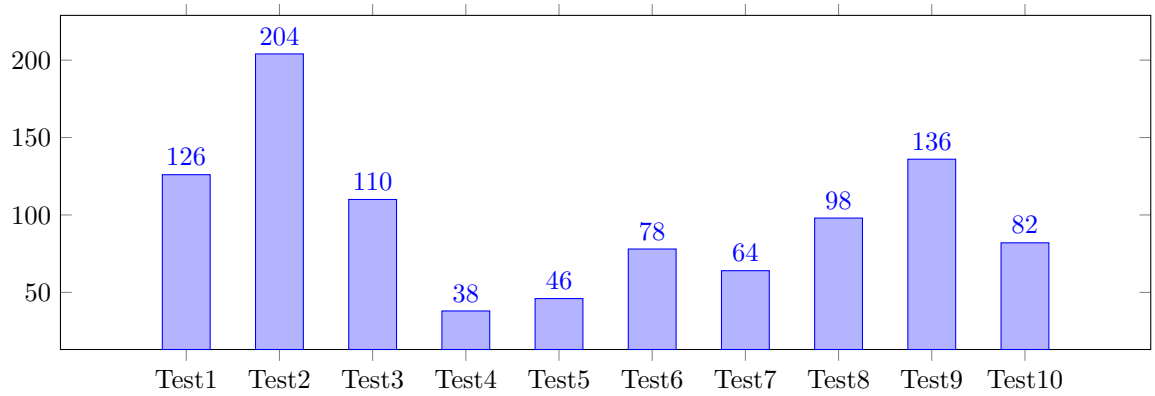


Figure B.4: 10 independent runs of the Sarsa algorithm. Each bar indicates the number of iterations to reach a specific desired states. All learnings are cleared between tests.

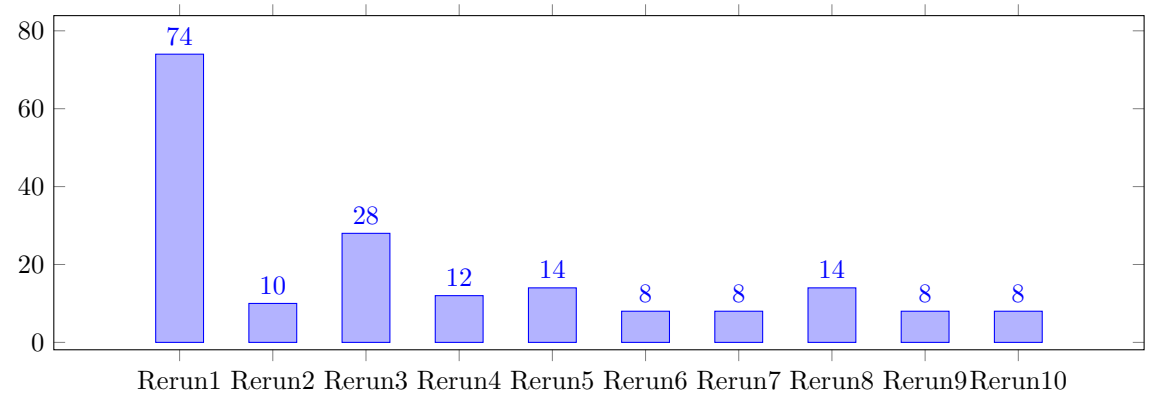


Figure B.5: 10 dependent reruns based on Test10 from B.4. The bars indicate how many iterations are needed for the Sarsa algorithms to reach a desired states. The state is kept the same for all reruns and learnings are kept.

Sarsa(λ)

Saras(λ) is tested in the same manner as Q-learning and Sarsa. It can be found that Sarsa(λ) requires an average of 101 iterations to find a desired state the first time it is being run. However, the number of iterations can vary from 54 to 214 iterations (see Figure B.6). When some knowledge has been gain, and the same state needs to be found again, Sarsa(λ) is capable of doing this in just 26 iterations based on only one prior successful search (see Figure B.5). The reruns mainly require a low number of iterations to find a specific state again, however, a spike is observed to occur with Sarsa(λ) which might be due to some unlucky explorations.

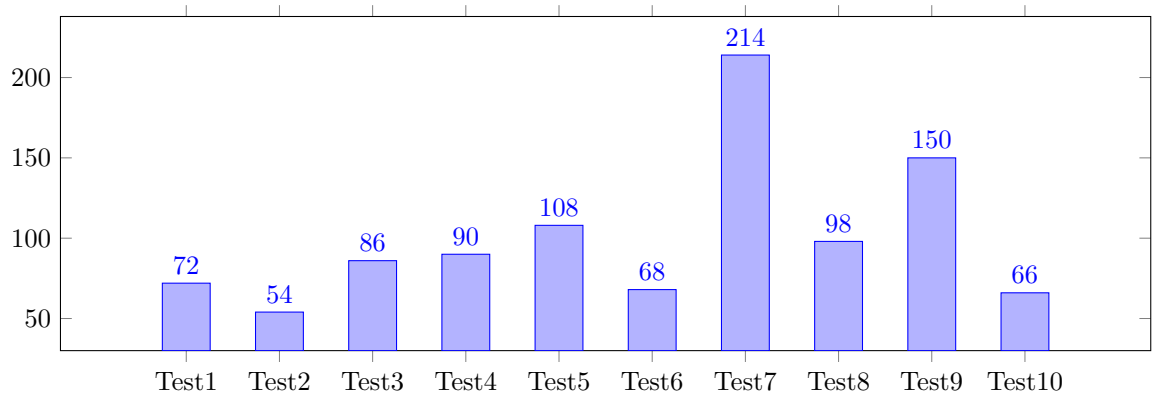


Figure B.6: 10 independent runs of the Sarsa(λ) algorithm. Each bar indicates the number of iterations to reach a specific desired states. All learnings are cleared between tests.

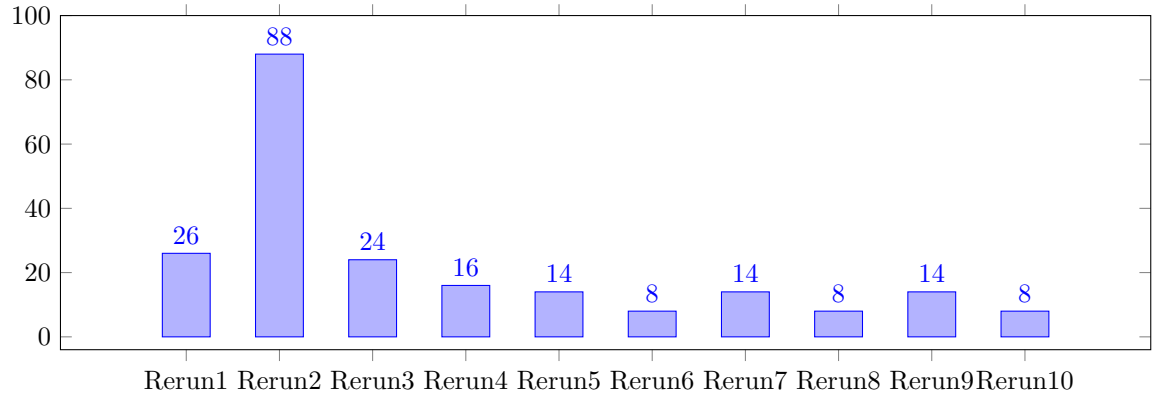


Figure B.7: 10 dependent reruns based on Test10 from B.6. The bars indicate how many iterations are needed for the Sarsa(λ) algorithms to reach a desired states. The state is kept the same for all reruns and learnings are kept.

Dyna-Q

Based on the same test setup as used for the other algorithms, Dyna-Q is tested and found to perform similarly. Figure B.8 shows that Dyna-Q requires an average of 80 iterations to find a desired state. However, anything from 24 to 180 iterations can be expected. Like Sarsa(λ), Dyna-Q has good utilization of learnings meaning the task of finding states that have previously been searched for can be done in few iterations. When searching the same state and utilizing the knowledge gained from one previous run, it is found that Dyna-Q can finish the task in 20 iterations as depicted in Figure B.9. However, like Sarsa(λ), Dyna-Q also experiences spikes where knowledge utilization seems to fail. This is likely due to exploration bringing the Dyna-Q agent to unknown states.

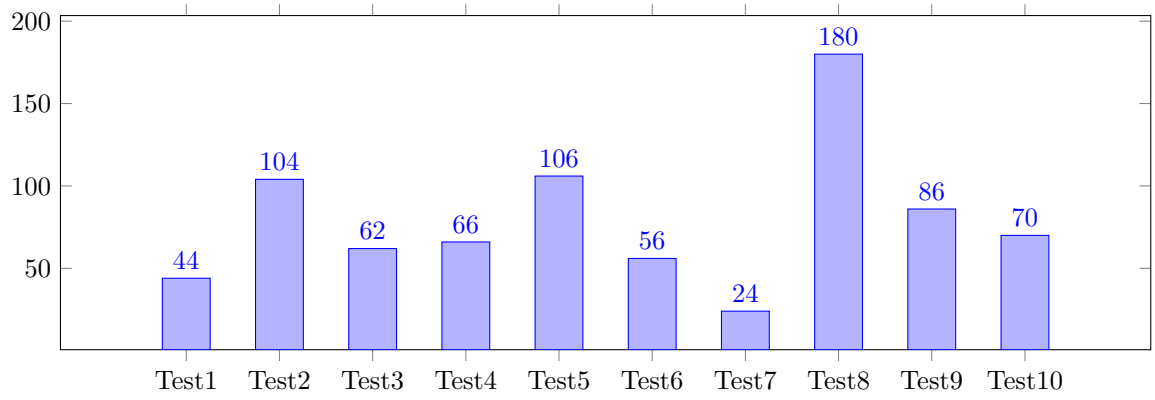


Figure B.8: 10 independent runs of the Dyna-Q algorithm. Each bar indicates the number of iterations to reach a specific desired states. All learnings are cleared between tests.

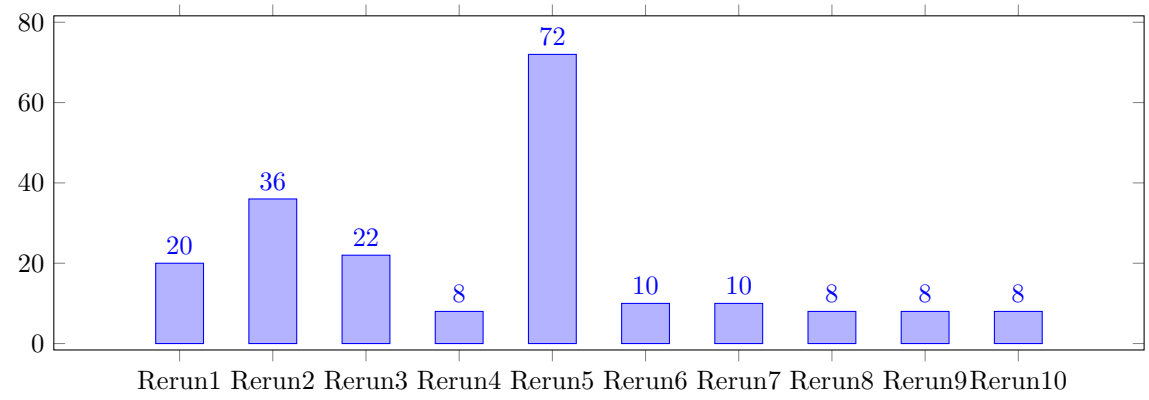


Figure B.9: 10 dependent reruns based on Test10 from B.8. The bars indicate how many iterations are needed for the Dyna-Q algorithms to reach a desired states. The state is kept the same for all reruns and learnings are kept.

Appendix C

SoundSense Learn testing - Sound profile preference plots

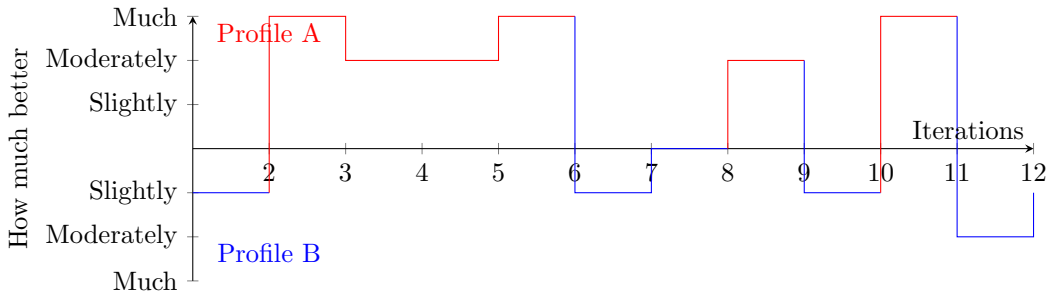


Figure C.1: Sound profile rating at each iteration during the SoundSense Learn test in the "Noisy venue" context with a "Conversation" and "Reduce noise" intent.

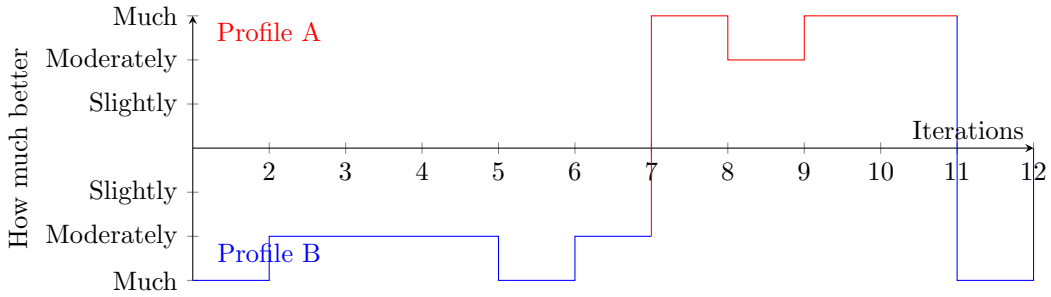


Figure C.2: Sound profile rating at each iteration during the SoundSense Learn test in the "Nature" context with "Relax" intent.

Appendix D

Requirements

D.1 Functional requirements

FUNCTIONAL REQUIREMENT	**FR1** The solution must be able to predict the user's preferred adjustments of bass, midrange, and treble in environments similar to previous training sessions
SUMMARY	Based on earlier training sessions, the machine learning components of the solution (Sarsa(λ) and the CNN model) have learned the user's preferences. This information must be recalled at a later time when the user is in a similar environment.
RATIONALE	If the solution is not able to predict bass, midrange, and treble based on previous training sessions, it is incapable of learning the user's preferences. As a result, the user has to go through a training session to adjust hearing aid settings even if the environment has been visited before. This is undesirable as it requires the user to interact with the smartphone application more often.
COMMENT	A full solution could also include Speech focus, Noise reduction, and Wind noise reduction. However, to simplify the prototype implementation this has been limited to bass, midrange, and treble.

FUNCTIONAL REQUIREMENT	**FR2** The smartphone application must collect audio of the user's soundscape
SUMMARY	Hearing aid adjustments are predicted based the audio in the user's environment. The smartphone application must collect this audio such that it can be processed and used to perform environment classification. This classification can be used to infer new adjustments or as input to a training session.
RATIONALE	If the smartphone application is incapable of collecting the audio, some other device must perform this task. However, the solution consist of very few components making the smartphone application the only viable option. Hence, if this is not possible, no environments can be classified. As a result, the user's preferences in one environment (e.g. Restaurant) will be mixed with preferences from a different environment (e.g. Outdoor).
COMMENT	In a full solution, collecting the audio should be performed by the hearing aids and streamed to the smartphone application. However, according to Brian Dam Pedersen, the CTO of GN Hearing, the hearing aids available for prototype development in this project do not have this capability.

FUNCTIONAL REQUIREMENT	**FR3** The solution must perform training of CNN models for environment classification and hearing aid adjustment regression in the cloud
SUMMARY	There is a need to assess the user's environment based on the surrounding sounds and the learn general user preferences. The models created by the Convolutional Neural Networks are used for those two tasks. New models must be trained when new information is available.
RATIONALE	Training Convolutional Neural Networks on smartphones is slow and likely not a feasible solution. Furthermore, it puts additional stress on the user's smartphone which might not be able to provide the computing power needed. Instead, CNN models must be trained on a server with the necessary resources (here named "the cloud").

FUNCTIONAL REQUIREMENT	**FR4** The CNN models must be able to classify any environments into one of the following classes: All-Around, Restaurant, and Outdoor
SUMMARY	The environment classification is based on a finite set of labels. The three classes mentioned in this requirement corresponds to listening programs in the ReSound Smart 3D application. Each listening program has been created to reduce specific acoustic annoyances. It is assumed that the user has specific preferences in environments with these annoyances.
RATIONALE	If the solution is not able to classify into the three mentioned classes, it might not succeed in recognizing significantly different environments that all require unique adjustments. As a result, the solution is not able to predict optimal and personalized hearing aid adjustments.
COMMENT	A full solution should possibly contain more environments to give a more fine-grained classification. However, for simplicity, the number of environments has been scoped as listed. The three environments have been adopted from the prestudy [23] which showed promising classification abilities in these environments.

FUNCTIONAL REQUIREMENT	**FR5** The CNN models must be able to predict the user's environment and general hearing aid adjustments for that environment
SUMMARY	The models created by the Convolutional Neural Network are used for environment classification and to learn long-term general user preferences. This is used as input to the Sarsa(λ) learning algorithm which then tries to optimize the hearing aid adjustment suggestion.
RATIONALE	The CNN model enables the solution to perform the crucial task of environment classification. Furthermore, it makes it possible to get a long-term and more general understanding of the user preferences which are used as a starting point for further personalization. Not having these two functionalities would force the user to run a training session much more often in order to ensure optimal hearing aid adjustments.

FUNCTIONAL REQUIREMENT	**FR6** The smartphone application must be able to download and implement newly trained CNN models
SUMMARY	The models created by the Convolutional Neural Network are trained in the cloud. The models must be downloaded to the smartphone and loaded into the application to be used.
RATIONALE	If it is not possible to download and utilize the pretrained models locally, the solution might not be able to provide environment classification or general user preference predictions. Alternatively, the environment and user preference predictions could be made in the cloud. However, this requires users to be connected to the internet constantly which is undesirable.

FUNCTIONAL REQUIREMENT	**FR7** The cloud must provide trained models which can be used for inference on the smartphone
SUMMARY	**FR6** mentions that the smartphone application must be able to download pretrained models. To enable this, the cloud solution must make them available for download. When a new model has been trained, it must be possible for the smartphone application to request it.
RATIONALE	If the cloud does not make it possible for the smartphone application to download models for offline use, internet connectivity might be required to use the solution. Alternatively, assuming a model might already be available offline, this will become outdated over time and provide incorrect predictions leading to suboptimal personalization.

FUNCTIONAL REQUIREMENT	**FR8** The smartphone application must be able to process the collected audio into STFT spectrograms
SUMMARY	A pretrained CNN model is running as part of the smartphone application and so is the audio collection (as prescribed by **FR2**). As the CNN model expects Short-Time Fourier Transformation spectrograms, these must also be created locally.
RATIONALE	Failure to create the Short-Time Fourier Transformation spectrograms will make it impossible to rely on the predictions provided by the CNN model. Data input must be as the model expects to provide reliable outputs.
COMMENT	This is based on the prestudy [23] which found that Short-Time Fourier spectrograms is a viable way of representing audio data for environment classification.

FUNCTIONAL REQUIREMENT	**FR9** The smartphone application must provide STFT spectrograms with user preferences and an environment label to the cloud as input for CNN model training
SUMMARY	When new information is available about the user's preferences, this must be included in the machine learning models. The cloud is responsible for creating an adjusted CNN model. Hence, the smartphone must provide this new information (user preferences in a specific environment) to the cloud.
RATIONALE	If the new user preference information is not provided to the cloud, it will not be able to adapt the CNN model. As a result, the model becomes outdated over time. Furthermore, failure to provide spectrograms, user preference information, or an environment label makes it impossible to create an updated model as the machine learning will not know the input and output relationship.

FUNCTIONAL REQUIREMENT	**FR10** The smartphone application must store STFT spectrograms with preference and environment information until it can establish a connection to the cloud solution
SUMMARY	The cloud solution is responsible for creating updated CNN models for predicting the environment and general user preferences. This is done based on the information listed in **FR9**. If the user is not connected to the internet when new preference information is captured, this must be saved for whenever connection can be established.
RATIONALE	If new user preference and environment information is not stored while the user is offline, this valuable information will be lost. If information is lost, the user has to go through additional training sessions leading to a solution that seems slow at adapting to changes.

FUNCTIONAL REQUIREMENT	**FR11** The cloud solution must start training a new CNN model when new STFT spectrograms with preference and environment information is received
SUMMARY	The cloud solution is responsible for creating up-to-date CNN models for offline use by the smartphone application. When new user information has been captured as the result of a training session and provided to the cloud, it must start adapting the CNN model immediately.
RATIONALE	User behavior can be dynamic and preferences might change over time. If the cloud does not start creating new CNN models when new information is received, CNN models might be outdated for longer than necessary. Models must constantly be up-to-date to provide the optimal experience.

FUNCTIONAL REQUIREMENT	**FR12** The smartphone application must ask for the user's intent when starting a new training session
SUMMARY	User intent can affect how they would like the hearing aids to be adjusted. Upon starting a training session, the user should select an intent from a predefined list. This intent can be used to set the trajectory of the training session as specified by **NFR7**.
RATIONALE	If the user's intent is not taken into account during a training session, the learning algorithm might require more iterations before getting an impression of the user's preferences. More iterations are undesired as it takes more time for the user to finish a training session.

FUNCTIONAL REQUIREMENT	**FR13** The training session must be a pairwise comparison of sound profiles
SUMMARY	Widex's SoundSense Learn solution uses a pairwise comparison of two sound profiles. The user is supposed to listen to both sound profiles and rate which is the best. Using this information, the optimal hearing aid configuration can be determined.
RATIONALE	A pairwise comparison is needed for this solution to ensure ease of use and facilitate user input consistency. Alternatively, the user could be asked if settings improve or become worse over training iterations, but this might be more difficult for the user to assess.
COMMENT	According to Nielsen [66], pairwise comparison is supposed to be easier for the users to interact with and makes them perform more consistently.

FUNCTIONAL REQUIREMENT	**FR14** The training session must terminate when the optimal state has been found
SUMMARY	The solution implements Sarsa(λ) which is used to search a state-space where each state represent a unique hearing aid adjustment. When the optimal state has been found, the training must terminate and the adjustment applied to the hearing aids.
RATIONALE	Not stopping the training session when an optimal state has been found will tie the user to the smartphone for longer than needed. Furthermore, it might result in termination in a suboptimal state if the user starts becoming inconsistent in their inputs.
COMMENT	What constitutes an optimal state is defined by **NFR9**.

FUNCTIONAL REQUIREMENT	**FR15** The smartphone application must allow the user to terminate a training session at any time
SUMMARY	Training sessions are supposed to end whenever an optimal state has been found. However, users might not be interested in going through an entire training session, but simply achieve decent settings. Functionality must be implemented such that the users can terminate the training session when they are satisfied with the settings.
RATIONALE	Not providing functionality to terminate a training session removes empowerment from the users. Furthermore, a user might realize that time does not allow for a complete training session. In such a case the user must at least benefit from the training till now.

FUNCTIONAL REQUIREMENT	**FR16** When a training session is terminated, the best user preference prediction must be applied to the hearing aids
SUMMARY	Whether termination happens because an optimal state has been found or because a user prematurely decides to do so, the best prediction of hearing aid adjustments must be applied to the hearing aids. Even at premature termination, some user input has been received which indicates a preference.
RATIONALE	Not applying adjustments to the hearing aids after a successful training session might create less incentive for the user to start the session in the first place. Furthermore, even if a training session is terminated by the user before an optimal state has been found, the information provided by the user so far can still be used to estimate the best possible adjustments.

FUNCTIONAL REQUIREMENT	**FR17** The solution must suggest new personalized hearing aid adjustments if the environment changes
SUMMARY	The user's preferences are linked to a specific environment. Therefore, when the environment changes, it is likely that the hearing aid adjustments should as well. A suggestion must be made by the solution asking the user to apply new adjustments.
RATIONALE	Failing to provide suggestions regarding new adjustments when the environment changes, may result in suboptimal adjustments for the environment. Furthermore, if no suggestions are made, the solution becomes less automated and it is up to the user to constantly ensure that optimal settings are applied.

FUNCTIONAL REQUIREMENT	**FR18** The user must receive a notification on their smartphone when new adjustments are being suggested
SUMMARY	When a change in environment is detected, the user is suggested new hearing aid adjustments. This suggestion must be made as a notification on their smartphone allowing the user to easily accept or reject the suggestion.
RATIONALE	If the suggestion is not made as a notificaiton on the smartphone, the user might not realize that a new adjustment suggestion has been made. As a result, the new adjustments will not take effect, and the user will not have the optimal experience.
COMMENT	A full solution could consider expanding this. In case a user has a smartwatch, this could be utilized for notifications too. See **FR19**.

FUNCTIONAL REQUIREMENT	**FR19** The user must receive a notification on their smartwatch when new adjustments are being suggested
SUMMARY	Like the user must receive a notification on their smartphone when new adjustments are suggested, users with a smartwatch must be able to receive these notifications on their smartwatch. This makes it possible to try, accept, and reject suggestion directly from their wrist.
RATIONALE	If notifications on smartwatch is not enabled, the user is forced to interact with the smartphone each time a new notification comes in. Notifications are made to provide a better experience for the user and to collect information about their preferences. Enabling smartwatch compatibility makes it easier for the user to respond to suggestions. As a result, more data can be collected which, over time, provides a better solution.

FUNCTIONAL REQUIREMENT	**FR20** All notifications can be rejected by the user
SUMMARY	Notifications with training session initiation suggestions or new hearing aid adjustments pop up as notifications on the users smartphone.
RATIONALE	If a user does not have the time to start a training session or is happy with the current hearing aid adjustments, it should be possible to reject the notification. Furthermore, a user rejecting a notification can also provide information about their preferences.
COMMENT	Adam Heleniak, Senior UX Designer at GN Hearing, suggested to test if such a button is actually needed. Do this by creating a "fake door". To the user, the button would be present but when pressed would explain that the feature is not yet fully implemented. Collect data about how many people click the button and how often to see if it should be implemented.

FUNCTIONAL REQUIREMENT	**FR21** The user can try the new hearing aid adjustment suggestion before accepting or rejecting it
SUMMARY	The user is notified about changes in the hearing aid adjustments if the environment changes. This suggestion must allow the user to listen to the suggestion first before actually applying it.
RATIONALE	The user will not know if they want the suggestion or not before they have tried it out. If it is not possible to listen to the suggestion before actually applying it, the user might never accept suggestions. The user might be fairly happy with the current adjustment and would not want to make it worse.

FUNCTIONAL REQUIREMENT	**FR22** Upon three consecutive notification rejections, the user must be asked if they would like notifications less often
SUMMARY	A user receives notifications when new hearing aid adjustments are available (due to environment change), or when the solution lacks information about the current environment. A user can decide to accept or decline notifications.
RATIONALE	If a user keeps rejecting notifications it might be because they are appearing too often. When a notification is received, it requires the user's attention. If the solution requires the user's attention too often, they might become displeased with it.
COMMENT	It should be verified through user testing if three is the correct number.

FUNCTIONAL REQUIREMENT	**FR23** First-time users must go through an onboarding session to determine who must initiate training sessions and how often notifications must come
SUMMARY	Different users might have different preferences in terms of how often the solution should notify about new adjustment or training session suggestions. Furthermore, some users might prefer that the solution suggests starting a training session while other users might like to initiate such a session themselves. The onboarding session allows the user to provide notification frequency and training session initiative information.
RATIONALE	Not having an onboarding session makes it difficult to design one solution that fits all users. The onboarding allows for some personalization and empowerment that might be needed for the user to have a good experience with the solution.

FUNCTIONAL REQUIREMENT	**FR24** The smartphone application must allow the user to start a training session
SUMMARY	A user might have disabled the solutions ability to suggest training sessions. In this case, functionality should be implemented to allow the user to take the initiative to start a training session.
RATIONALE	If the user cannot take the initiative to start a training session, the user will lose some empowerment. Furthermore, if the user has denied the solution the ability to suggest training sessions, no information will be learned about the user if the user cannot start a training session.

FUNCTIONAL REQUIREMENT	**FR25** The smartphone application must suggest starting a training session if the environment is new or if it is uncertain about the quality of the suggestion it can provide
SUMMARY	When the solution encounters a new environment, the user must be asked to provide preference details for this environment. Similarly, the solution must suggest starting a training session if no definitive optimal state can be found for the given environment.
RATIONALE	New environments and low quality suggestions are a result of lack of information. In order to improve the solution and make it possible to provide adjustment suggestions next time the user is in that particular environment, a training session must be completed. Failing to do so will result in a solution that is not able to provide an optimized and personalized experience in all environments.

FUNCTIONAL REQUIREMENT	**FR26** A consent message must be presented and accepted by the user before the machine learning components of the solution are enabled
SUMMARY	The user must be made aware what sort of data is being collected, how it is being used, and how it is being handled. Part of the solution lives in the cloud meaning data must leave the users device for analysis somewhere else.
RATIONALE	In order to ensure the user's trust in the solution, information about the solution should be sufficient and transparent. Furthermore, regulations might require the solution to ensure user consent before data can be collected and sent for analysis in the cloud.

FUNCTIONAL REQUIREMENT	**FR27** Adjustment suggestions can be applied automatically without confirmation from the user if the user has agreed to this during the onboarding session
SUMMARY	The user goes through an onboarding session such that the solution can get basic information about the user. The aim is to inform the user and collect information about how often the user want to interact with the solution.
RATIONALE	If hearing aid adjustments cannot be applied automatically, the user has to interact with the solution each time. This might only be in the form of accepting a notification. However, even that might not be possible in some situations. A user that is often busy and trusts the solution might prefer that adjustments are just continuously applied as needed without confirmation.

FUNCTIONAL REQUIREMENT	**FR28** The system must allow a user to overwrite the old adjustments if a training session resolves in an unexpected outcome dissimilar to previous knowledge
SUMMARY	The user can run training sessions in a given environment to teach the solution what they prefer in this setting. However, user intent and preferences can change even if the environment does not.
RATIONALE	If the user changes dramatically, a training session can be initiated (likely by the user). If the outcome of this training session is very dissimilar to what was predicted by the solution for that environment, the user should have a choice to overwrite the current knowledge of the solution. The outcome might be dissimilar because the user has genuinely changed. In that case, the solution should learn this as quickly as possible. Maybe the dissimilarity is a one-time occurrence, then the solution should not adapt. The user must be able to input this.

D.2 Non-functional requirements

NON-FUNCTIONAL REQUIREMENT	**NFR1** The solution must consist of a smartphone application and a machine learning cloud instance
FURPS CLASS	Performance
SUMMARY	The solution consist of a reinforcement learning algorithm (Sarsa(λ)) and a multi-task branched Convolutional Neural Network for classification and regression. The reinforcement learning must run on the smartphone along side a pretrained CNN model. However, the CNN models are to be trained in the cloud.
RATIONALE	There is a need to utilize a cloud instance for this project as it is not viable to train CNN models on a smartphone. However, the smartphone application is still needed to ensure the solution works offline. The entire solution cannot be cloud based.
COMMENT	The "cloud instance" (also named "the cloud" in order requirements) have been named like so to give room for interpretation. The cloud instance for this project is any internet connected server with the needed resources to train CNN models.

NON-FUNCTIONAL REQUIREMENT	**NFR2** The smartphone application must implement and use Sarsa(λ) to learn the user's preferences
FURPS CLASS	Performance
SUMMARY	There is a need to implement a learning algorithm which is sufficently advanced to understand the user preferences while also being able to run on a smartphone. Sarsa(λ) is the algorithm of choice as it provides better user input utilization compared to alternative algorithms.
RATIONALE	If a different learning algorithm than Sarsa(λ) is implemented, a number of disadvantages can follow: Firstly, the alternative algortihm might be too computationally expensive to run on a smartphone. Secondly, it might require more interaction with the user resulting in longer training sessions.

NON-FUNCTIONAL REQUIREMENT	**NFR3** The state-space must consist of the following dimensions: Bass, midrange, treble, and user environment (All-Around, Restaurant, and Outdoor)
FURPS CLASS	Reliability
SUMMARY	Sarsa(λ) requires a state-space to be defined. Each state is a representation of a unique hearing aid adjustment. The learning algorithm can navigate in this space to find the optimal state for the user. As context can affect what adjustments a user might want, environment is included as a dimension in the state-space.
RATIONALE	If bass, midrange, treble, and environment are not the dimensions of choice, the solution might not provide a satisfactory hearing aid adjustment. Widex's SoundSense Learn solution is adjusting bass, midrange, and treble too. Furthermore, these three parameters are found to be available to all users of the ReSound Smart 3D application.
COMMENT	It is possible that a full solution should include Speech focus, Noise reduction, and Wind noise reduction as dimensions as well. However, for simplicity, the scope of the prototype solution has been limited.

NON-FUNCTIONAL REQUIREMENT	**NFR4** The solution must determine the user's preferences in 12 (or less) iterations
FURPS CLASS	Usability
SUMMARY	Users must go through training sessions to teach the system about their preferences. A training session consist of multiple iterations where each iteration brings a bit more information about the user and what they prefer in the given situation.
RATIONALE	According to Anders Trier Poulsen, Machine Learning Specialist at Widex, the SoundSense Learn solution must go through at least 12 iterations to determine a user's preference. The solution developed in this project must be just as good or better. The fewer iterations the better such as not to keep the user busy with training session for too long.

NON-FUNCTIONAL REQUIREMENT	**NFR5** Sarsa(λ) is allowed to adjust hearing aid configurations $\pm$3 dB for each training iteration
FURPS CLASS	Usability
SUMMARY	Sarsa(λ) navigates the state-space where each state represents a hearing aid adjustment. The adjustments of bass, midrange, and treble are measured in decibel. Changes of 3 dB are just noticeable for the human ear.
RATIONALE	It is acceptable for Sarsa(λ) to take larger steps (3 dB of difference) as small changes are not noticeable. In order for the user to feel a difference, the change should be noticeable. Furthermore, it makes it possible to skip states and more quickly gain an understanding of the users preferences.
COMMENT	Specified according to Ditlev Munk Raboel, DSP Software Developer at GN Hearing, and Hansen [40].

NON-FUNCTIONAL REQUIREMENT	**NFR6** Sarsa(λ) must use the environment and user preference prediction from the CNN model as a starting point for the training session
FURPS CLASS	Performance
SUMMARY	The smartphone application implements both Sarsa(λ) for learning user preferences as well as a pretrained CNN model. The CNN model labels the environment and provides a prediction for general hearing aid adjustments for the environment. Sarsa(λ) uses this information as a starting point for a training session.
RATIONALE	Not using the information provided by the CNN models makes it impossible to determine what environment the user is in. This makes it much more difficult to predict a hearing aid adjustment that will fit the user. Furthermore, having a starting point for Sarsa(λ) should provide quicker convergence to an optimal hearing aid configuration.

NON-FUNCTIONAL REQUIREMENT	**NFR7** The user's intent information must bias the trajectory of Sarsa(λ)
FURPS CLASS	Performance
SUMMARY	User intent is important for the hearing aid adjustment. However, intent is not a dimension in the state-space. It should affect the training session by making a specific trajectory more likely.
RATIONALE	Biasing the trajectory makes for shorter training sessions as Sarsa(λ) knows where to search. However, it is only a bias meaning users can still get any state they would like. Therefore, it does not limit the user, but will provide quicker convergence in most cases.
COMMENT	The biasing information must be based on the Smart Buttons in the ReSound Smart 3D application. E.g. "Speech focus" should favor a positive gain in midrange and treble, but a negative gain in bass.

NON-FUNCTIONAL REQUIREMENT	**NFR8** All hearing aid adjustment suggestions must be predicted by a Temporal-Difference Prediction routine, not by the CNN model alone
FURPS CLASS	Reliability
SUMMARY	The CNN models provide environment labeling input as well as a prediction of general hearing aid adjustments for the environment. During user training sessions, Temporal-Difference Prediction is used to determine optimal states. It must be used to recall user preferences as well, and have the final word in suggesting new hearing aid settings.
RATIONALE	Relying only on the CNN model output might provide hearing aid adjustment suggestions that are too generalized. Sarsa(λ) is employed to adapt to the user's latest behavior and preferences. Sarsa(λ) modifies the Q-table used by Temporal-Difference Prediction to recall optimal states.

NON-FUNCTIONAL REQUIREMENT	**NFR9** An optimal state has been found when the following are aligned: A Temporal-Difference Prediction run, negative state rewards, and continuous user emphasis.
FURPS CLASS	Reliability
SUMMARY	An optimal state has been found when the following conditions are equal: At least one Temporal-Difference Prediction run has determined it as the most valuable state, that state returns negative rewards no matter what action is chosen when trying to leave it, and the user keeps increasing the value of that state.
RATIONALE	It is difficult to assess that an optimal state has been reached. Not using all three measures mentioned above might resolve in incorrect conclusion in regard to optimal states. As a result, suboptimal adjustments are applied to the hearing aids.

NON-FUNCTIONAL REQUIREMENT	**NFR10** The CNN architecture must be a multi-task network with one classification branch and one regression branch
FURPS CLASS	Performance
SUMMARY	The Convolutional Neural Network utilized in this solution needs to do both classification and regression. It utlizes the same input for both and can, by using branching, utilize the information from one task to ease the other task.
RATIONALE	Creating a branched network allows for information sharing across the tasks. If branching is not used, multiple networks have to be created. As a result, complexity will increase, two networks have to be trained with the same input data, and an additional knowledge sharing mechanism might need to be put in place.

NON-FUNCTIONAL REQUIREMENT	**NFR11** The CNN regression branch must use Mean Squared Error as cost function
FURPS CLASS	Reliability
SUMMARY	The CNN needs a cost function to optimize for when training new models. This cost function has been chosen to be Mean Squared Errors as it punishes big mistakes more than small mistakes.
RATIONALE	Punishing large mistakes more than small mistakes makes sense for this project. A configuration that is off by 1 dB might not be noticeable for the user compared to a configuration off by 3 dB. The latter case needs to be corrected and must be reflected much more strongly compared to the former case. Not using Mean Square Error might lead to a different weighting of errors and cause too much focus on details.

NON-FUNCTIONAL REQUIREMENT	**NFR12** The CNN regression branch must use ReLU in the hidden layers and the identity function in the output layer as activation functions
FURPS CLASS	Performance
SUMMARY	The regression branch of the CNN model needs to be able to predict positive as well as negative numbers. Adjustments for bass, midrange, and treble are measured in positive and negative gain.
RATIONALE	ReLU is a well proven activation function for the hidden layers. The identity function in the output layer makes it easy to predict positive as well as negative numbers. Using a different activation function in the output layer might require scaling.

NON-FUNCTIONAL REQUIREMENT	**NFR13** The STFT spectrograms must be used as input to the CNN model
FURPS CLASS	Reliability
SUMMARY	The Convolutional Neural Network must create models that are capable of classifying a users sound environment. Audio information can be transformed into Short-Time Fourier Transformation spectrograms. These can then serve as a basis for the environment analysis.
RATIONALE	If Short-Time Fourier Transformation spectrograms are not used, it might not be possible to create a mapping from input to output. Furthermore, all components of the solution must agree on an input format in order to be able to rely on the output.
COMMENT	The prestudy [23] found this to be a viable way for environment classification based on audio.

NON-FUNCTIONAL REQUIREMENT	**NFR14** The STFT spectrograms must cover 1 second of audio
FURPS CLASS	Reliability
SUMMARY	A Short-Time Fourier Transformation spectrogram covers a specific amount of time of the audio. One second has been found to be a good amount of time for classifying an environment.
RATIONALE	If the lengths of the spectrograms deviate from what this requirement specifies, it might capture too little information. On the other hand, if too much information is captured, it might include contradicting information. Both result in less reliable environment classification.
COMMENT	As specified in the prestudy [23] based on [29] and [42].

NON-FUNCTIONAL REQUIREMENT	**NFR15** The solution must sample the environment every 5 seconds
FURPS CLASS	Reliability
SUMMARY	The solution needs to collect an audio sample and classify the environment regularly to see whether it has changed or not. If the environment has changed, hearing aid adjustments might be needed.
RATIONALE	It is a trade-off between using resources on the user's smartphone and quickly adapting to the environment. Sampling too often wastes resources, and sampling to rarely makes the hearing aids slow at adapting to the situation.
COMMENT	Testing must be done to verify whether 5 seconds is an appropriate number.

NON-FUNCTIONAL REQUIREMENT	**NFR16** The solution must determine an environment change based on 6 samples
FURPS CLASS	Reliability
SUMMARY	The prestudy [23] found that the environment classification can benefit from observing the environment over a period of time before making an environment classification. Therefore, multiple samples are needed.
RATIONALE	Using too few samples to determine the environment might resolve in incorrect environment classification. As a result, incorrect adjustments will be applied to the user hearing aids and the solution will be the cause of a suboptimal experience.
COMMENT	Testing must be performed to determine if 6 samples are sufficient to reliably classify the environment.

NON-FUNCTIONAL REQUIREMENT	**NFR17** An accepted hearing aid adjustment suggestion provides +10 reward to that Sarsa(λ) state
FURPS CLASS	Performance
SUMMARY	When the solution suggests a new hearing aid adjustment which the user likes, the reinforcement learning algorithm must be rewarded. This will emphasize the state as a good state and make it likely to be suggested again.
RATIONALE	If no reward is given to the system, it will not learn anything. The reward input will teach the system what adjustments are preferable over others. Over time, it then becomes better at making suggestions.
COMMENT	Testing must be performed to evaluate if 10 reward points are appropriate.

NON-FUNCTIONAL REQUIREMENT	**NFR18** A rejected hearing aid adjustment suggestion provides -10 reward to the Sarsa(λ) state representing those adjustments and $+\frac{10}{2}$ reward for the state representing the current adjustments
FURPS CLASS	Performance
SUMMARY	If the user is suggested a hearing aid adjustment that is rejected, the system should be punished. The system will learn that this particular adjustment in that environment is not a good choice making it less likely to be suggested again. The current adjustments must then be better than the suggestion. Therefore, the state representing the current adjustment should be emphasized slightly.
RATIONALE	If the learning algorithm is not rewarded and punished, it will not learn. If the punishment and reward values differs from what is specified, the algorithm might be rewarded or punished too much. Too extreme rewards can bias the learning to much.
COMMENT	Testing must be performed to evaluate if negative 10 and positive 5 reward points are appropriate respectively.

NON-FUNCTIONAL REQUIREMENT	**NFR19** Each CNN model must be specific to the user
FURPS CLASS	Reliability
SUMMARY	CNN models are trained by the cloud based on Short-Time Fourier Transformation spectrograms that represent a specific environment the user has been in. Information about the user's preferences in this environment is provided with this spectrogram.
RATIONALE	Each CNN model must be user specific as they are trained based on user specific data. If data is mixed between users, or models are shared among users, each model does not represent the specific user's preferences. As a result, hearing aid adjustment suggestions might not be optimal.
COMMENT	This is a requirement specifically defined for the prototype development of this project. According to Anders Udahl, Senior Manager for the Connected Apps team at GN Hearing, one model per user does not scale for a company.

NON-FUNCTIONAL REQUIREMENT	**NFR20** The solution must run the environment observation as a background service
FURPS CLASS	Usability
SUMMARY	The solution constantly monitors the environment around the user to determine if it has changed. When an environment changes, new hearing aid adjustments are suggested to the user.
RATIONALE	The environment observer process must run as a background service. This makes it possible to continuously monitor the environment independently of what the user is using the smartphone for. If it is required to have the smartphone application in the foreground at all time, the automation aspect is lost.

Appendix E

Use case diagrams

This appendix supports the Analysis chapter and describes the system using use case diagrams. The system as a whole has two actors. The first one is the actual hearing aid user who is interacting with the USense SoundLearner application. The other actor is the USense SoundLearner application which invokes functionality in the Online training entity. As the diagrams depict, most of the functionality is within the USense SoundLearner. Each use case is defined by a name and also lists the requirements which it is responsible for fulfilling.

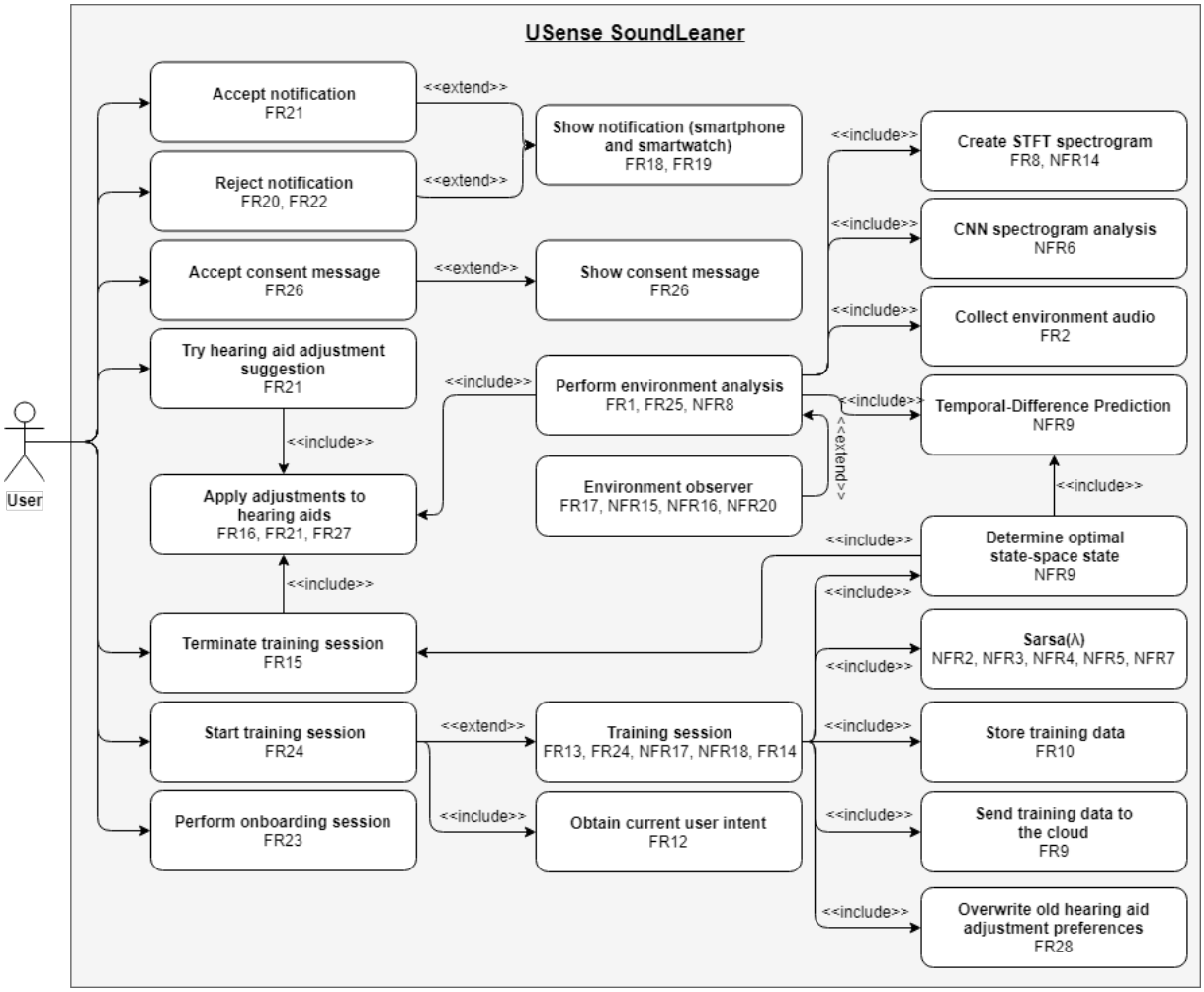


Figure E.1: Use case diagram describing the hearing aid user (actor) interacting with the USense SoundLearn smartphone application. Made with draw.io online tool.

Figure E.1 depicts the USense SoundLearner, its functionality, and how the hearing aid user can interact with it. It can be seen that the majority of the behavior is not directly invoked by the user. However, the functional behavior that is exposed relies on internal functionality. As a result, the USense SoundLearner is a complex system, however, the complexity is hidden from the user.

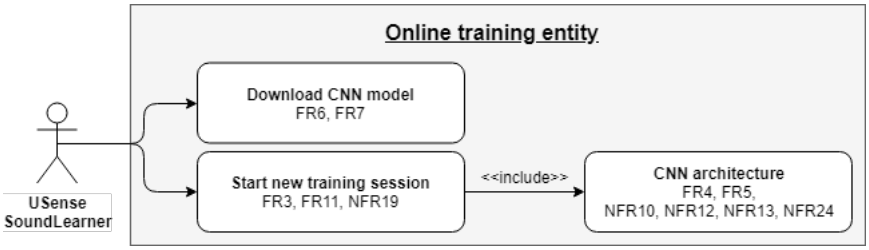


Figure E.2: Use case diagram describing the USense SoundLearner smartphone application (actor) interacting with the Online training entity. Made with draw.io online tool.

The USense SoundLearner is not capable of doing everything itself. It relies on the Online training entity to do the resource intensive process of training new Convolutional Neural Network (CNN) models. When a model is done training, it can be downloaded by the USense SoundLearner and made available to the hearing aid user.

Appendix F

Prototype implementation details

This appendix supplements the prototype implementation description found in Section 7 in the main report. The following text covers the online prediction request process and the Convolutional Neural Network model adjustment process. This is followed by a specification of the technical configurations of the Convolutional Neural Network architecture and the Sarsa(λ) algorithm.

F.1 Online prediction request process

Section 7.2.1 describes the process of creating and processing prediction requests. This process requires the USense SoundLearner application, the Web service, and the Training entity to all communicate. Section 7.2.1 describes the process using a sequence diagram. This appendix is added to provide the same description, but as process flows instead.

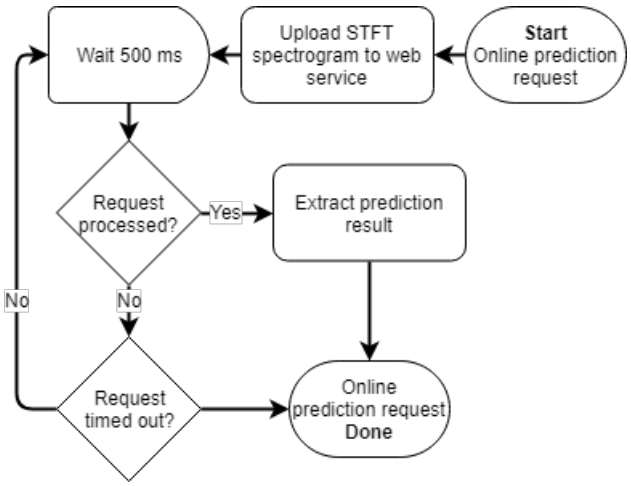


Figure F.1: Steps which the USense SoundLeaner application must go through in order to create an online prediction request and receive the answer. Made with draw.io online tool.

Figure F.1 depicts the steps of creating and receiving an online prediction request from the perspective of the USense SoundLeaner application. Firstly, the environment representation in the form of a Short-

Time Fourier Transformation (STFT) spectrogram is uploaded to the Web service. Then the application starts to wait for the request to be processed. It continuously checks if the request has been processed with 500 milliseconds of delay between requests. If the request has not been processed, it checks whether it should time out. If it has been waiting for 10 seconds, the application will move on and stop checking the prediction request. If the request has been processed, the environment class, bass, midrange, and treble values are extracted.

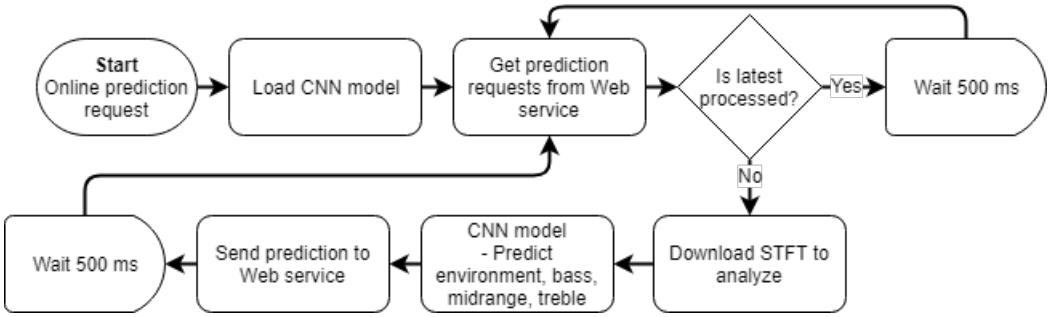


Figure F.2: Steps which the Training entity must go through to process an online prediction request. Made with draw.io online tool.

The Training entity must be constantly ready to respond to an online prediction request. As depicted in Figure F.2, when it starts up, it loads an instance of the current CNN model. Having the instance ready makes it possible to more quickly respond to requests. Hereafter, it starts checking if the latest request has been processed. If it has, no new prediction request has been made. The Training entity waits 500 milliseconds and asks again. When a new request come in, the associated STFT spectrogram is downloaded and processed by the CNN model instance. This results in an environment class, bass, midrange, and treble value prediction which is uploaded to the Web service such that the database can be updated. This process continues until the Training entity is stopped.

F.2 Convolutional Neural Network model adjustment process

When a training session is carried out by the user in the USense SoundLearner application, audio of the sound environment is recorded. The outcome of the training session can be used with the audio to adjust the Convolutional Neural Network (CNN) model which is doing the environment analysis in the smartphone application. Adjusting the CNN model requires four entities: USense SoundLearner application, Web service, Training entity, and Model conversion entity. The following text describes the process of updating the CNN model with the user's latest preferences from the perspective of the Training entity.

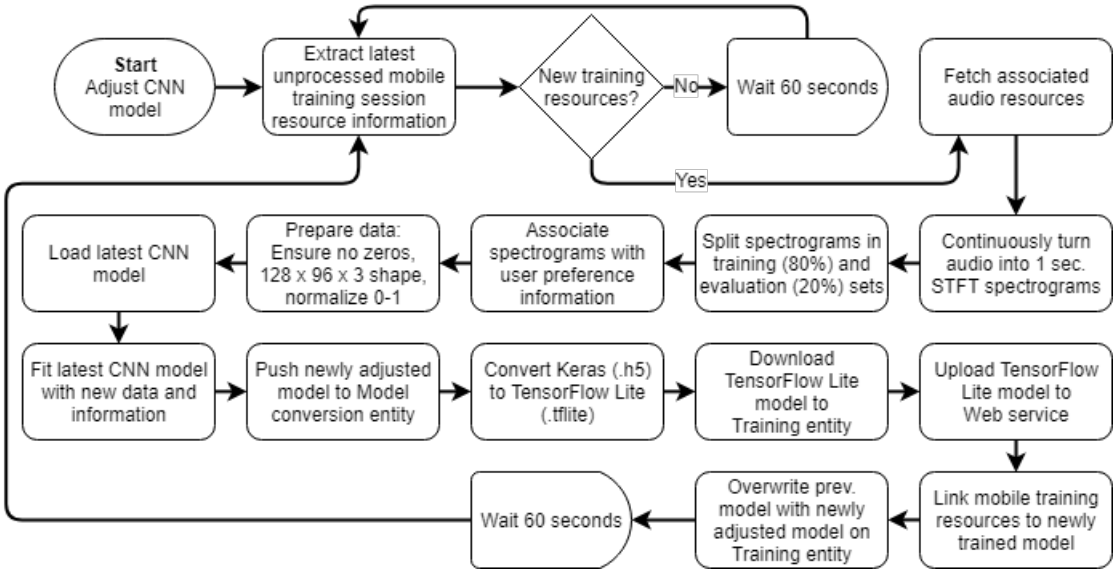


Figure F.3: The steps executed by the Training entity such that it is able to provide user specific CNN models. Made with draw.io online tool.

Figure F.3 describes the process of collecting user specific resources and information in order to return a customized CNN model. Initially, the Training entity sends a request to the Web service to determine if a new training session has been carried out by the user of the USense SoundLearner application. After each training session, information about the outcome (the determined user preference) and an audio file is uploaded. If no new information is available, the Training entity waits 60 seconds and checks again. If new information is available, the Training entity downloads the associated resources. The audio is then converted into Short-Time Fourier Transformation (STFT) spectrograms and divided into a training set (80% of the spectrograms) and an evaluation set (20% of the spectrograms). Each spectrogram is then given a label which will tell the CNN model that environment representations (STFT spectrograms) looking like this should result in the analysis outcome specified by the label. The labels are used to reflect the user's preference to the CNN model. Providing many spectrograms with these user specific labels will bias the CNN model towards the user's preference.

The spectrograms are then prepared through resizing and normalization such that the format is as the CNN architecture expects. The latest CNN model is then loaded and retrained with the new spectrograms and labels. This results in a new CNN model. However, this model is saved by Keras as a HDF5 formatted file [15, 100] and must be converted to a TensorFlow Lite model to run on the smartphone. This conversion process must be done on a UNIX-based system which the Training entity in this prototype is not. Therefore, a specific Model conversion entity has been put in place. The Keras model is uploaded to the Model conversion entity, converted, and a TensorFlow Lite model is then returned which is uploaded to the Web service. On the Web service, other data as a result of this process (number of samples, epochs, accuracy, supported environment classes and hearing aid adjustment parameters etc.) and the new CNN model are then linked to the resources that were used to create the new model. The Training entity then locally overwrites the old CNN model with the new adjusted CNN model. The new adjusted CNN model is now available for download from the Web service for the USense

SoundLearner application.

Next time the user decides to perform a training session in the USense SoundLearner application, this process starts over again in order to always provide the user with an updated and customized CNN model.

F.3 Convolutional Neural Network configurations

The following table outlines the specific configurations of each layer in the implemented Convolutional Neural Network (CNN) architecture. This table supplements the Implementation chapter (Chapter 7) and is a specific case of the CNN architecture presented in Section 6.3.1 (Figure 6.3, page 83).

Layer name	**Layer type**	**Connected to**	**Layer configurations**
stft_input	Input	-	Shape: 128x96x3
conv1	Convolutional	stft_input	Filters: 32 Receptive field: 5x5 Activation function: ReLU Padding: Yes
conv2	Convolutional	conv1	See conv1
pool1	Pooling	conv2	Pooling type: Max Pool size: 2x2 Stride: 2
conv3	Convolutional	pool1	Filters: 64 Receptive field: 5x5 Activation function: ReLU Padding: Yes
conv4	Convolutional	conv3	See conv3
pool2	Pooling	conv4	See pool1
conv5	Convolutional	pool2	See conv3
conv6	Convolutional	conv5	See conv3
pool3	Pooling	conv6	see pool1
model_base	Flatten	pool3	-
class_fc	Fully Connected	model_base	Number of neurons: 1024 Activation function: ReLU Dropout: 40%
regression_fc	Fully Connected	model_base	See class_fc
class_out	Output	class_fc	Activation function: Softmax Size: 7
regression_out	Output	regression_fc	Activation function: Identity Size: 3

Table F.1: Outlines the configurations of each layer of the implemented Convolutional Neural Network architecture.

The size specified in layer class_out and regression_out of Table F.1 is given by the number of environment classes and hearing aid adjustment parameters the architecture supports. The current implementation can classify into seven low-level environment classes and predict bass, midrange, and treble. Hence, the size is seven and three respectively.

F.4 Sarsa(λ) configurations

The following parameters specifies how the Sarsa(λ) implementation of this project has been configured. These parameters supplements the implementation documented in Chapter 7.

$\lambda = 0.5$

$\gamma = 0.9$ (later changed to 0.4 during testing)

$\alpha = 0.9$

$\epsilon = 0.4$

As found during testing, these parameters needs to be adjusted in order for the prototype to function properly. A deeper analysis is needed to determine specifically what needs to change and by how much.

Appendix G

Prototype testing

This appendix presents additional information in relation to the prototype testing presented in Chapter 8 of the main report. The appendix covers accuracy measures and confusion matrices which are used for the prototype accuracy assessment presented in Section 8.1 in the main report. Furthermore, a detailed description of the environments used for the real environment testing is presented in Section G.3 of this appendix.

G.1 Prototype accuracy assessment - Measures

Sokolova and Lapalme [91] list a range of measures which can be used for assessing multi-class classification problems. This projects adopts the Average Accuracy, Macro Precision, Macro Recall, and Macro F1 Score. These measure are found using Equation G.1, G.2, G.3, G.4 respectively [91]. The macro-variation of the measures have been chosen due to their robustness to deviation in sample sizes among the assessed classes.

$$Accuracy_{Average} = \frac{\sum_{i=1}^{l} \frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i}}{l} \tag{G.1}$$

$$Precision_{Macro} = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i+FP_i}}{l} \tag{G.2}$$

$$Recall_{Macro} = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i+FN_i}}{l} \tag{G.3}$$

$$FScore_{Macro} = \frac{(\beta^2+1) * Precision_{Macro} * Recall_{Macro}}{\beta^2 * Precision_{Macro} + Recall_{Macro}} \tag{G.4}$$

The accuracy measures use the following terminology [91]: TP denotes true positives. These are samples that have been correctly predicted as belonging to a specific class. True negatives are denoted by TN and describes samples that have correctly been classified to not belong in a specific class. FN and FP denotes false negatives and false positive respectively. False negatives are samples which have incorrectly been classified to not belong in a specific class. False positives are samples which have incorrectly been

classified to belong to a specific class [91]. The l in Equation G.1, G.2, and G.3 denotes the number of classes included in the analysis (three for the high-level environment classes and seven for the low-level environment classes used in this project).

The Macro F Score (Equation G.4) is an accuracy measure that combines the Macro Precision and Macro Recall in order to provide an accuracy measure that does not include the TN class. Choosing a values for β makes it possible to put more emphasis on either precision or recall if desired. This project is using the Macro F1 Score which is achieved by setting $\beta = 1$.

G.2 Prototype accuracy assessment - Confusion matrices

This section present the confusion matrices which are used as input to the four measures presented in Appendix G.1. The confusion matrices are created in order to assess the Convolutional Neural Network model's classification performance. Both the low-level and high-level environment classes are assessed. This section presents the confusion matrices for low-level as well as high-level classes.

A confusion matrix is a structured way of presenting true positives, true negatives, false positives, and false negatives. Each class being assessed is described by such a matrix. A confusion matrix is structured as follows:

	Predicted Yes	**Predicted No**
Actual Yes	True Positive (TP)	False Negative (FN)
Actual No	False Positive (FP)	True Negative (TN)

G.2.1 Confusion matrices - Low-level classes

City walk	**Predicted Yes**	**Predicted No**
Actual Yes	47	79
Actual No	39	571

Driving (car)	**Predicted Yes**	**Predicted No**
Actual Yes	57	21
Actual No	1	657

Forest (sit & walk)	**Predicted Yes**	**Predicted No**
Actual Yes	142	0
Actual No	23	571

Train station	**Predicted Yes**	**Predicted No**
Actual Yes	5	37
Actual No	22	672

Intersection	**Predicted Yes**	**Predicted No**
Actual Yes	42	18
Actual No	75	601

Wind	**Predicted Yes**	**Predicted No**
Actual Yes	66	0
Actual No	1	669

Canteen	**Predicted Yes**	**Predicted No**
Actual Yes	214	8
Actual No	2	512

G.2.2 Confusion matrices - High-level classes

All-Around	**Predicted Yes**	**Predicted No**
Actual Yes	445	3
Actual No	8	280

Restaurant	**Predicted Yes**	**Predicted No**
Actual Yes	241	8
Actual No	2	512

Outdoor	**Predicted Yes**	**Predicted No**
Actual Yes	66	0
Actual No	1	669

G.3 Real environment test - Environment descriptions

The prototype is tested in a number of real environments in order to verify its capability of generalizing learnings. This appendix describes each of these environments in detail. The results of the testing can be found in Section 8.2 (starting at page 124) in the main report. The test setup consists of a Oneplus 6T smartphone running the prototype application developed in this project. All training sessions are done while wearing the ReSound LiNX 3D hearing aids connected to the smartphone application.

Forest

Store Hareskov was visited on a sunny Sunday between 12:15 and 12:45. The forest has tracks of gravel and soft forest-type soil. Both types of tracks were utilized while performing the testing. The soundscape consisted of bird song from various birds as well as footsteps from walking on the gravel tracks. Furthermore, some of the testing was done close to a small lake introducing sounds from ducks and frogs. A small road goes through the forest introducing sporadic sounds from noisy cars in the distance. Furthermore, other people had decided to visit the forest as well that day resulting in noise from bikes, horses, and people chatting. A training session was done while walking on the gravel and soil tracks and aimed to focus on the sound of bird tweet.

Shopping centers

Two different shopping center environments where visited. First, Rødovre Centrum was visited on a Sunday between 13:50 and 14:50. Then environment was quite relaxed and not packed with people. Testing was done while walking through the center, while sitting on a bench in an open area, and while sitting at the fast-food restaurant "Carl's Junior" which has tables in an open area shared with the coffee shop "Espresso House". Sitting on a bench in the open area introduced sounds from people walking by as well as talk and baby cry from nearby benches. A brief conversation between family members took place next to the test setup for a short period of the testing time. Visiting the fast-food restaurant introduced a bit of speech from the tables owned by Espresso House. These tables were located next to the test setup with a head-high room separator in between. The talking was located in front, right next to, and behind the test setup. However, the talking was done in low voice and simply just added to the background noise. A training session was performed on the bench in the open area trying to achieve a pleasant sound experience. Furthermore, two training session were carried out at the fast-food restaurant in an effort to improve speech intelligibility.

In order to see if the training session performed in Rødovre Centrum had an effect on the Convolutional Neural Network (CNN) model, a new model was downloaded to the smartphone which was then brought to Field's on the same Sunday but in the evening between 19:00 and 19:30. This adjusted CNN model included changes based on the training sessions from both Rødovre Centrum and Store Hareskov.

The soundscape in Field's was similar to Rødovre Centrum. The testing was done in an open area with couches where people gathered. During the testing, three girls had a conversation behind the test setup and other people were walking by. Testing also took place just outside the telecommunications company "Telia" which had music playing in their store. Furthermore, this second test location was not far from an escalator which added to the overall background noise. A third test was done in the grocery store "Bilka One Stop". In here, sounds of store music, shopping carts, and refrigeration units were present.

Intersections

Road noise was tested out at three intersections: Toftegårds Allé and Lyshøjgårdsvej in Valby, Tårnvej and Rødovre Centrum in Rødovre, and Bådehavnsgade and Sjællandsbroen in Sydhavnen.

The intersection in Rødovre was visited on a Sunday between 13:30 and 13:50. The road Rødovre Centrum is an entry road to the shopping center "Rødovre Centrum" which was open at the time of testing. However, traffic was low and consisted of idle, accelerating, and passing cars.

The testing in Valby was done on a Monday morning at 8:25 resulting in a lot of traffic and high noise scenarios at times. The soundscape contained idle as well as moving cars and people on bikes. This is similar to the intersection in Sydhavnen which was visited twice. First, Monday morning at 8:45 to 9:00 which contained very busy and diverse traffic. The noise generated here came from bikes, scooters, cars, buses and lorries. Then, the same location was visited in the afternoon between 15:20 and 16.00. The sound was generated by the same sources, but it seemed less busy compared to the morning. Both in the morning and the afternoon, the prototype was tested right next to the road as well as on a parallel street to Sydhavnsgade making it possible to keep a bit of distance. The distance kept was about 10 meters in an effort to see if that would change the prototype's behavior.

Office space

The office space testing was done in the group room for ITCOM and ICTE students at Aalborg University Copenhagen. The testing was done on a Monday between 10:30 and 10:40. Only one other project group was present. As a result, the group area was mostly silent, with some discussion from this other group. The group was located to the right of the test setup on the other side of the group room with only a whiteboard in between. The group was not excessively loud, but as they were the only source of sound in the room, the conversation was noticeable. The testing was purposely performed while they had a discussion to see how the prototype would react. It is assumed that this represents a normal office space.

Canteen

The main canteen at Aalborg University Copenhagen was chosen as the testing location for the Restaurant environment. The data used for training the initial CNN model was collected in IBM's canteen during the prestudy [23]. The IBM canteen had more people and was more noisy when the recordings were made compared to the canteen at Aalborg University Copenhagen. With that said, the canteen was visited between 11:30 and 13:00 on a Monday. Plenty of people were present and the noise seemed to peak around 12:10. The test setup was located in the left side of the canteen with equally many rows of tables in front and behind it. The noises generated was predominantly by people talking and cutlery making contact with plates, but also sporadically contained the sound of a chair being dragged across the floor. At the start of the testing session, only few people were present in the canteen. Around 12:20, a group of four people sat down two chairs from the test setup and started talking while eating.

During the testing, three training sessions were performed. All three took place between 11:35 and 12:00 in an effort to create a new CNN model specifically adjusted to the environment. The new model was ready for test towards the end of the testing time frame.

Transportation – Car, bus, and train

Various different way of transportation was tried out to observe prototype behavior. First transportation method test was driving which was done for two different types of cars and road. The first car of choice was a gasoline powered Peugeot 206 RC driving from Smørum to Bryggeri Skovlyst. The radio was muted for the drive and no passengers were present. The speeds varied from 40 to 70 km/h and also included a stop at a red light. Traffic was low, and the drive was done on a Sunday between 11:55 and 12:10.

On the same day between 17:40 and 18:05, a drive in medium traffic from Smørum to Field's was done in a Peugeot 508 SW diesel. The speeds ranged from 50 to 130 km/h and the radio was turned off. The drive was done with one other person in the car, but no talking occurred when samples of the environment were taken. This testing was done with the adjusted CNN model based on the training sessions from the testing in the forest and in Rødovre Centrum.

Second form of transportation environment tested was train from Måløv station to Valby station. The test location was in the absolute middle of a coupé. The testing was done on a Monday morning from 7:48 to 8:10 with some talking in the coupé. Speech was isolated to a single source, however, the source changed over time. Talking was observed on the seat to the right of the test setup and also two rows ahead. Overall, the majority of the background noise came from the train itself and an open window two rows in front of the setup. Testing was done with the train moving, accelerating, decelerating, and stationary.

The third transportation methods test was bus 4A from Valby station to Bådehavnsgade in Sydhavnen. This was done on a Monday morning from 8:30 to 8:45. The sounds encountered during testing was the bus idling, accelerating, and driving on both smooth and bumpy road. Furthermore, the voice announcing the next stop was captured during the test. The bus was carrying plenty of people, but no talking occurred.

Quayside

The quayside at Aalborg University Copenhagen has been used to collect the data representing the winy environment (Outdoor). This data was collected during the prestudy [23]. During the prototype testing days, not much wind was to be found. The quayside seemed like the best option and did have some wind. However, nothing compared to when the initial data was collected. Furthermore, some work was being done at the quayside. As a result, the soundscape consisted of some wind, some talking, and some road noise in the distance from Sydhavnsgade.

Train station

Testing took place at Valby station from 8:10 to 8:25 on a Monday. The soundscape consisted mostly of trains arriving and leaving, train doors opening and closing as well as people moving. A diesel-powered train arrived at the station creating a high level of noise for a short period of time. Towards the end of the testing, a group consisting of about 20 young individuals arrived at the train station which added speech to the soundscape.

Dinner table

To test how dominant speech in an otherwise silent environment would be handled by the prototype, a brief test was carried out at the dinner table. Five people were present of which only on spoke at a time during testing. Only male voices were recorded during the testing. No other sounds were present.

Appendix H

User interviews

Four different users are interviewed for the project. This is done to learn more about the user and how they feel about their current cares solution. Furthermore, the aim is to draw conclusions in relation to whether they would actually use a solution as the one developed in this project.

A detailed summary of each of the interviews is provided in this appendix. The users are kept anonymous, but consists of two male subjects and two female subjects. Each user is classified according to the two user types presented in the main report (Section 5.2.2).

User$_A$ - The proactive user

The user is 68 years old and suffers from a symmetric moderate hearing loss. He is using the ReSound Smart 3D application very frequently; multiple times each day. The application is a necessity for him to use the hearing aids, and he compares it to a remote for a television.

He has thoroughly explored the features of the smartphone application and tries to customize it to fit his preferences. He has created four Favorites (user defined listening programs) which are GPS controlled. As an example, he mentions that one of the Favorites are for when he is at the grocery store where noise levels are high. The Favorite helps him reduce noise from sources such as shopping carts.

Wind noise also tend to bother him. As an example, this occurs when he is out biking. Therefore, he has a program for reducing wind noise. It was not clarified during the interview whether this was also a Favorite or a predefined listening program.

Depending on the situation, he might also use Smart Buttons (see footnote on page 57 in Section 5.2.1). Furthermore, he uses the Sound Enhancer (see footnote on page 58 in Section 5.2.1) when listening to music. If he does not make adjustments to bass, midrange, and treble in the Sound Enhancer, the music is unbearable. Usually, he fully increases the bass, turns down the midrange completely, and almost fully increases treble which results in a good sound experience. He has an open fitting allowing sound from the environment to enter his ear without being processed by the hearing aid first. If he is listening to music and puts his fingers in his ears, it sounds almost like if he was wearing a headset. The user has requested that the low frequencies are amplified even more, but his audiologist does not want to increase further. The user reports that the current adjustability is sufficient for his needs. However, more adjustability would be appreciated. He would prefer if he could adjust e.g. the bass to a point where he thinks it is too much. If possible, he would also like even more parameters to modify.

He is generally very happy with the entire solution and find it difficult to mention things that could be better. The hearing aids are almost like glasses to him. He will wear them and then forget about them. However, he does mention a problem related to noise filtering. If a vacuum cleaner or an angle grinder is started, the hearing aids will lower the sound from these after eight to ten seconds. He does not agree with this behavior in all cases. He should hear noise just as a person without hearing impairment would. As an example, this noise reduction is problematic when fixing bike punctures as the hissing noise is removed by the hearing aids. He suggests a listening program which turns off all the noise reduction. In situations similar to when fixing a punctured bike, he could select this specific program and be allowed to hear noise.

Even though he finds the solution close to perfect, he still likes the idea of a personalized solution like the one developed in this project when asked directly. He adds that it would be very helpful for a hearing aid user. He reckons that the application should have an "unlocked" program which would allow the user to sit down in front of an equalizer and adjust the frequency bands to his liking. This could then be sent off to the audiologist who would install it in the hearing aids. However, even if it is just a solution that allows him to adjust the current bass, midrange, and treble it is also helpful.

Having to train such a solution in the annoying environment does not bother him. He would continue training until it was just perfect for him. He did not want to specify how long such a training session should take, but he claimed it would not take long to find the optimal adjustment. He finds it easy to use the current Sound Enhancer interface implemented today.

When the solution has learned his preferences, he would prefer if the solution asked him if he would like the preferences to be applied next time a similar environment is encountered. He wants to make sure it is working as he expects. The fact that the user always has to carry the smartphone and that the smartphone must be connected to the internet once in a while does not bother him.

When discussing Over-The-Counter (OTC) hearing aids, he claims to have already used them. However, as explained in Section 10.3 (page 147), these hearing aids does not exist yet. What he has tried must have been a Personal Sound Amplification Product (also presented in Section 10.3 (page 147)). The functionality of such devices did not impress him as it simply amplified all sounds. This includes sounds that he can hear just fine already. However, he would be interested in an OTC solution if it gave him the possibility of fitting the hearing aids himself like the audiology usually does. This could possibly replace the solution he has today. However, he mentions that this only applies to people that are comfortable with smartphone, computers etc. Most of the people with hearing aid that he knows would not be able to use such a solution.

User$_B$ - The proactive user

The user is 49 years old and has an asymmetric hearing loss: Mild hearing loss on the right ear and a moderate severe loss on the left ear. He has been living with the left hearing loss for his entire life and has become used to compensate for it in every day life even without hearing aid.

He does have knowledge about the ReSound Smart 3D but rarely uses it. He used to explore the various functionality it offers, but does not find it provides value for him. As a result, he only rarely uses the application. He reckons that he probably uses it once every month. When he does use it, he finds himself on noisy or windy situations and he mainly just adjust the volume. He also uses noise reduction features and mentions the predefined "Noise filter" and "Speech clarify" Smart Buttons.

He has All-Around, Restaurant, and Outdoor listening programs available in the application but rarely uses them. If he does, it is to experiment. Furthermore, he does not find the need for the application just to change between programs. He can do that on the hearing aid itself. He mostly uses the All-Around program.

In his opinion the hearing aid should not require him to make various changes. It should simply work and provide him the best audio possible. If the hearing aid gets challenged too much and cannot handle the situation (he mentions wind noise as an example), the hearing aid should simply lower the volume.

This is actually very unlike him as he likes to tinker and make adjustments. However, he does not find that the features of the application benefit him very much. As there is no satisfying feedback, he does not using the features.

He mentions that the reason why he is so picky with the hearing aid is likely because does not strictly need the hearing aid. He is used to living with only one good ear and knows how to compensate for it. However, the hearing aid does help him in noisy situation. If someone is speaking to his bad ear, he can notice it now that he has the hearing aid. Then he can turn his good ear towards the person. According to him, he does not require the hearing aid to work optimally as he can do just fine without it.

For him to start using the ReSound Smart 3D application it should provide a semi-automated feature that helps him adjust the hearing aid. He finds the Sound Enhancer quite abstract and the wording is too specific to audio. He would like something that can help him in a current situation. As an example, he said that when he attends a presentation at work and someone is whispering behind him, he would like a button to push in the application. The hearing aids should then automatically make adjustments and ask him if it is better or worse. It should be clarified that this comment was unprovoked by the interviewer. The interviewee only knew that he was being interviewed for a project about improving hearing care solutions.

As he often finds himself on a bike or at social gatherings when he runs into problems where the above solution would be useful, training it should not take more than 30 seconds to a minute. He does not want to lose attention from what is going on. When he finds himself in the same environment at a later time, the solution should simply apply his preferences. No need to ask him first. The fact that he has to carry the smartphone at all time and allow it to connect to the internet once in a while does not bother him.

He has an interesting perspective on OTC hearing aids and is aware that the category exists. However, it is not something he would purchase for himself. In fact, he is not willing to pay for any hearing aid. He is currently involved in the test program at GN Hearing, but would not go out and buy a hearing aid if the program was closed. However, if the hearing aid became a gadget and not just a hearing aid, he would consider it. He believes the current hearing care industry is too focused on the immediate competition and not aware of what companies like Amazon and Apple might be capable of. He believes that companies like these will soon enter the market and provide hearing aids with digital assistants. They will use the same technology, but provide new services.

User$_C$ - The care seeker

The user is 69 years old and suffers from a symmetric moderate hearing loss. This is partly due to osteosclerosis on the left ear.

She is familiar with the ReSound Smart 3D application and uses it daily. She is very pleased with all the configurations that it provides her. Furthermore, she is enjoying the quality of the audio in the

hearing aids. Minor adjustments are sufficient for everyday use, and she can do this fine-tuning herself.

She has the All-Around, Restaurant, Music, and Outdoor programs available. Usually, she is using the All-Around program, and the Restaurant program is rarely used as the audio becomes too strong and makes some weird echoing. She finds that the solution gives her good spatial awareness. As an example, she can hear bikes coming up behind her when she is walking in the forest (what program is used in these situations was not mentioned). Furthermore, she has a device which she can connect to her television such that the audio can be streamed to her hearing aids which is working well, too.

She does not notice the hearing aids and it has become a habit. When asking if the hearing aids are like glasses (wear and forget), she exemplifies with a story about her on vacation. While on the vacation she went to the beach and wanted to go for a swim. However, she was still wearing her hearing aids when she was about to go in the water. The hearing aids have become so natural to her that she now has to remember to take them off in situations like this.

She uses Smart Buttons and the Sound Enhancer once in a while. The adjustments she actually uses depends on the situations. As an example, she likes to increase the treble adjustment to hear better (what she means by better was not specified). In noisy locations she reduces the audio (not specified what feature is used. Volume adjustments are assumed based on later answers). In locations where the sound is supposed to be loud (she mentions concerts and cinemas), she turns off her hearing aids as it is too overwhelming otherwise.

When asked if she would like something automated, she mentions the volume. However, she cannot see how this can be standardized. According to her, all people are different and even she has different preferences in the same context depending on the situations. It is not clear to her how such a functionality would be accomplished.

She explains that often she will enter an environment, determine that her hearing aids needs to be adjusted and then leave the environment in order to perform the adjustment. She seems enthusiastic when a personalized solution that can automatically adjust to the environment is suggested. She reckons that the solution should learn over time. Such that when she has been in the same environment three or four times, the solution should remember this. At a later time, it should then ask to apply these settings again if she is in the same environment. However, it is important that the solution asks as her preferences change. Furthermore, she adds that it would be great if the solution could distinguish between different voices.

She is not sure how long time she would be willing to spend training it. Based on how long it usually takes her to make an adjustment, she estimates 30 seconds. She follows up by expressing that such a solution sounds very science fiction like.

When asked if she thinks differently about the solution if it is required by her to always carry her smartphone and allow it to connect to the internet occasionally (once every day or every other day), she does not quite understand why this would be needed. When explained that it is to extract statistics from the data, she is fine with it.

In relation to OTC hearing aids, she has not heard about them and it does not sound like a solution she would like to purchase. She would rather have a professional adjust her hearing aids. The fitting quality is crucial to her. Furthermore, due to her osteosclerosis, she thinks it is better to contact a professional and get proper hearing care.

User$_D$ - The care seeker

The user is 67 years old and has an asymmetric hearing loss: A severe loss on the right ear (suffers from osteosclerosis) and a moderate loss on the left ear.

The user is familiar with the ReSound Smart 3D application and uses it frequently. At home she will use the All-Around listening program, and when she is with other people, she applies the Restaurant program. In noisy situations she will increase the volume of both ears, but the specific level fluctuates a bit. She says that during an evening with other people she adjusts the volume multiple times. When at home, the volume setting depends on how loud the television is and if the people in the television are mumbling or not.

She has become familiar with a limit set of the ReSound Smart 3D application and thinks the whole solution works very nicely and is easy to use. However, she does not explore new functionality. She is not aware of how to use these other features and have a mental filter which discourages her from trying. This was exemplified with the Sound Enhancer which is too complex to her. She is not comfortable with unfamiliar technology.

To achieve comfortable adjustments, she uses the different listening programs, the associated Smart Buttons, and the volume control. She also has external devices which can be used for watching television or attached to a speaker at lectures in order to stream the audio to the hearing aids. Unfortunately, she has not quite figured out how to use this yet and will have to talk to her audiologist about it. Furthermore, she has a program for tinnitus.

Common problematic situations for her were she must change the hearing aid adjustments are when she is with a lot of people and there is a lot of talk or music in the background. She can hear a person next to her just fine, but struggles to have a conversation with someone across the table. She finds the adjustability which she is comfortable using sufficient for her needs. However, she often gets new fittings from the audiologist applied to increase the amplification provided by the hearing aids. She mentions that her hearing loss is gradually getting worse (it was not directly confirmed that this is linked to the regular fittings). Thanks to the smartphone application, she can request the new fitting over the internet. The audiologist will then make the changes, and these will be sent back to the smartphone. The smartphone can then apply them to the hearing aids after the user has confirmed. She is happy with this functionality.

She does not see what could be made better. She answers yes when asked if the hearing aids are like glasses to her (wear and forget). Furthermore, she has the rechargeable hearing aids which she finds easy to use and keep charged.

The application is already quick and easy to use so she does not see the need for automation. She tries to guess where the interview is going and is explained that the purpose of the project is to develop a personalized hearing aid solution. To her it is important that this does not change the size of the hearing aid. This is not due to looks, but comfort. Having a bigger hearing aid might become uncomfortable. Putting everything into an application seems like a good idea to her. She is not bothered with the fact that it would force her to bring her smartphone with her at all time and connect it to the internet occasionally.

She is not really sure how much time she would spend training such a solution. She initially mentions one month and is then asked about the length of each training session. She would continue as long as required and is not bothered by taking out her smartphone in public. She will tell people it is to adjust her hearing aids to not come across as impolite. When asked to provide an actual training session time,

she reckons that she would spend one or two minutes.

OTC hearing aids are new to her. After the concept has been presented, she explains that it is not something for her. Before getting hearing aids, she was not really sure if she could hear or not. It was friends and family that kept reminding her about the problem. Furthermore, it can take time before someone accepts that they have a problem. She only trusts hearing test made by an audiologist. According to her, for a test to be trustworthy and accurately represent the hearing loss, it must be conducted in a sound proof room. Finally, her hearing is continuously getting worse and she would like the professional guidance and care.

Appendix I

Product Backlog

Period: 4/2/2019 – 6/6/2019
Usual sprint time frame: 2 weeks
Between sprints: Supervisor meetings and restructuring and reprioritization

Sprint 1 (4/2/2019 – 17/2/2019)

Initial report writing:

- Write the introduction and the problem field
  - Focus the problem field on the personalization aspect
  - Do research on other solutions (e.g. Widex's SoundSense Learn)
- Ensure the validity of the problem formulation
- Present the work done in the previous semester which acts as a prestudy

Meeting with David Landwehr (GN Hearing Software Architect) to learn how to programmatically interact with the hearing aids.

Ensure the thesis application documentation is filled out.

Sprint 2 (18/2/2019 – 3/3/2019)

Present the prestudy to GN Hearing and get input for the master's thesis.

Report writing:

- Scope and limitations

Start the research on machine learning architectures to use to solve the problem.

- It is likely to be a regression problem based on the user's environment. What should the environment representation be (spectrograms?)? Determine techniques which can do this.
  - New pre-processing or presentation of the audio might be needed. In that case, research that too.

Determine if the current audio data is sufficient or if more needs to be collected.

- Initiate the collection process if that is the case
- Furthermore, consider if the data is correctly ordered. The data might be sufficient, but should it be categorized differently?

Sprint 3 (4/3/2019 – 17/3/2019)

Begin prototype development. Early prototyping might make it possible to try different machine learning techniques and compare different solutions.

- Initially, just create an application that can work as a shell for the machine learning. Make it possible to communicate with the hearing aids and to collect sound from the smartphone microphones.

A way to handle user feedback should be initiated.

- Should the machine learning be adjusted (retrained) on the smartphone?
- Should the user data be forwarded to a server that can personalize a model?
- Should the user input be a very simple weighting system such that no retraining is needed?

Continue research and document it.

- Let all this feed into an analysis

Sprint 4 (18/3/2019 – 31/3/2019)

Ensure that all data is collected and ordered as it should be.
Research on reinforcement learning continued.
Plan meetings with:

- GN Hearing Algo team to gain an understanding of good audio features for representing an environment. This should help define the reinforcement learning states.
- GN Hearing User Experience team to understand the average user of the Smart 3D application. Define personas. Furthermore, get input on how to present a solution like this to a user and how to get user input.
    - Determine if it is possible to get in touch with actual hearing aid user's to get user input to the project.

Sprint 5 (1/4/2019 – 14/4/2019)

Conduct meeting with GN Hearing's User Experience and Algo team.
Write analysis:

- Use the research and any feedback from GN Hearing

Sprint 6 (15/4/2019 – 28/4/2019)

Make official requirements:

- Use the analysis, scenarios, empathy maps, and risk plotting
- Perform requirement verification session with GN Hearing

Make design of full prototype with all components:

- Hearing aids
- Sound collections and pre-processing
- Machine learning (both training and inference)
- Including user input
- Any other components as servers etc. that might be needed
- Document it

Start implementation:

- Create the machine learning model and train it (if initial training is needed). Iterate and adjust hyper-parameters until accuracy is sufficient.
- Implement user feedback in application
- Connect sound collection, machine learning, and user feedback in the smartphone application
- If research determines that a server is needed for re-training a model, implement that too

Sprint 7 (29/4/2019 – 12/5/2019)

Continue the implementation.

Plan and conduct (phone) meeting with actual hearing impaired people:

- How do they find the current solution?
- How could intelligence be implemented into the current solution?

Sprint 8 (13/5/2019 – 26/5/2019)

Finish the implementation.

Build a test framework for evaluating the accuracy of the machine learning.

Document the implementation:

- Make diagrams: Architectural, activity, use cases, and sequence diagrams and describe

Document the methodology.

Perform testing. This will show if changes are needed. Testing should be done in two parts:

- The structured test evaluating e.g. using the F1 Score
    - Consider how this can be done with the user input. Maybe it cannot.
- Bring the prototype into real environments
    - Ensure the solution can recognize the environments and that it can determine the appropriate user preferences (hearing aid adjustments).
        - It is unlikely to see an environment that is exactly the same (hence why regression is interesting to research). However, evaluate if the behavior is as expected.
- Iterate and adjust the solution if needed based on the test results

Sprint 9 (27/5/2019 – 6/6/2019)

Present prototype improvements.
Discuss the future of hearing aids:

- Over-The-Counter, EEG etc.
- Get input from GN Hearing if possible

Finishing touches:

- Ensure prototype is ready for hand-in and demo (add missing comments and clean up)
- Write: Discussion, conclusion, abstract, executive summary
- Proof read
- Official hand in

I.1 Gantt chart

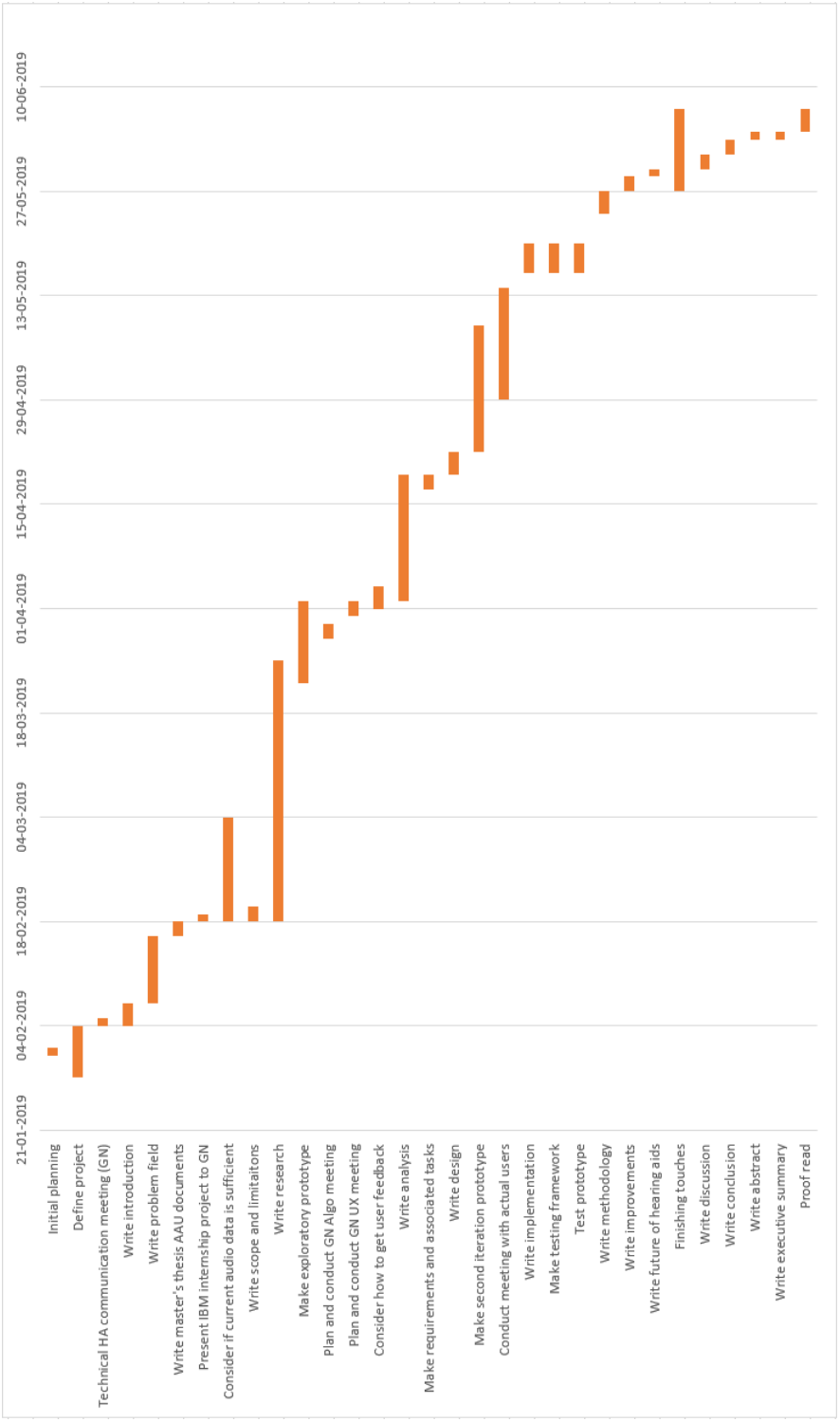


Figure I.1: Gantt chart (7th revision. At the end of the project)