TU WIEN Informatics

Design and Evaluation of a Sound Application using Acoustic Scene Classification

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

B.Sc. Tatiana Vladimirovna Rybnikova
Matrikelnummer 1267944

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Prof. Dr. Horst Eidenberger

Wien, 4. Oktober 2022

Tatiana Vladimirovna Rybnikova

Horst Eidenberger

TU WIEN Informatics

Design and Evaluation of a Sound Application using Acoustic Scene Classification

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Internet Computing

by

B.Sc. Tatiana Vladimirovna Rybnikova
Registration Number 1267944

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Dr. Horst Eidenberger

Vienna, 4th October, 2022

Tatiana Vladimirovna Rybnikova

Horst Eidenberger

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

B.Sc. Tatiana Vladimirovna Rybnikova

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. Oktober 2022

Tatiana Vladimirovna Rybnikova

Acknowledgements

At first, I would like to thank my thesis advisor, Prof. Dr. Horst Eidenberger, who gave me the opportunity to work on this interesting topic and made this work possible by offering his excellent advice and supervision. Furthermore, I would like to express my gratitude to Freya Schulz, who supported me the most by spending countless days and nights co-working together, and always found the right words of encouragement. Additionally, I would like to thank the participants of the case study for their time and effort.

Finally, I am grateful for the never-ending support I received from my family and friends, who dealt with my absence from many occasions. Last of all, considering the challenges I faced last year, I would like to thank myself for staying on course.

–Thank you.

Kurzfassung

Acoustic Scene Classification (ASC) verfolgt das Ziel, die Umgebung anhand von Audioaufnahmen zu identifizieren, in denen sie aufgenommen wurden. Die Entwicklung von Computerverfahren zur Erkennung der Umgebung ist Teil der Forschung in Bereichen wie maschinelles Lernen, Robotik und künstliche Intelligenz. Diese Arbeit untersucht die Verwendung von ASC im privaten Bereich, indem ein Prototyp entwickelt und eine Fallstudie durchgeführt wird, um zu evaluieren, ob mit dieser Technologie eine bestimmte psychologische Wirkung erzielt werden kann. Insbesondere untersuchen wir, ob eine akustische Illusion, sich in einer anderen Stadt zu befinden, erzeugt werden kann, wenn die Geräuschkulisse des Benutzers basierend auf dem Klassifikationsergebnis mit einer anderen Geräuschkulisse überlagert wird.

In dieser Arbeit erforschen wir den aktuellen Stand der Technik für ASC, indem wir die an die DCASE 2020 Challenge eingereichten Systeme untersuchen. Anhand der Auswertung wählen wir ein Basismodell, das ein ResNet ist und Log-Mel Spektrogramme ergänzt durch Log-Mel Deltas und Delta-Deltas als Eingabe akzeptiert. Wir modifizieren das Netzwerk, um 9 akustische Szenen zu klassifizieren. Der TAU Urban Acoustic Scenes 2020 Mobile Datensatz wird zum Trainieren des Modells verwendet. Um das Netz zu kalibrieren, experimentieren wir mit verschiedenen Hyperparametern, Verlustfunktionen und Data Augmentation Strategien und vergleichen die Ergebnisse anhand der Testgenauigkeit. Wir erreichen die beste Leistung mit einer Testgenauigkeit von $77,99\%$, indem wir Categorical Focal Loss und Mixup als Data Augmentation Methode verwenden.

Für den Prototypen wählen wir 4 Städte aus, die unterstützt werden sollen, und erstellen einen Geräuschkulissen-Datensatz mit Audioaufnahmen für jede Szene und Stadt, indem wir die Audiodaten aus Videos extrahieren, die den gewünschten Inhalt enthalten. Dann entwickeln wir eine Serveranwendung mit Flask, die eine API bereitstellt, um Vorhersagen aus dem Modell zu erhalten und Daten über die durchgeführten Klassifizierungsprozesse zu protokollieren. Schließlich entwerfen und implementieren wir eine mobile Anwendung mit dem Flutter SDK. Die Anwendung fordert vom Server in regelmäßigen Abständen Vorhersagen für Audiodaten an, die sie mit dem Mikrofon des Geräts aufgezeichnet hat. Basierend auf dem Klassifizierungsergebnis überlagert die mobile Anwendung die akustische Umgebung des Benutzers, indem sie eine Audioaufnahme für die erkannte Szene und die ausgewählte Stadt aus dem Geräuschkulissen-Datensatz abspielt.

Schließlich verwenden wir den Prototypen in einer Fallstudie, in der eine Gruppe von Teilnehmern die mobile Anwendung in einem vordefinierten Rahmen testet. In der Auswertung schätzen wir die Genauigkeit des Klassifizierers in einem realen Szenario mit 65%. Wir entdecken, dass die akustische Illusion, sich in einer anderen Stadt zu befinden, durch den Prototypen erzeugt wird. Außerdem zeigen wir, dass diese Illusion umso häufiger erlebt wird, je genauer die Klassifizierung wahrgenommen wird und je persönlicher die Beziehung zwischen dem Nutzer und der ausgewählten Stadt ist.

Abstract

Acoustic Scene Classification (ASC) aims to identify the environmental surroundings from audio recordings in which they were collected. Developing computational methods to recognize the environments is part of the research in fields such as machine learning, robotics and artificial intelligence. This work explores the usage of ASC in the private sector by developing a prototype and performing a case study in order to evaluate if a certain psychological effect can be achieved with this technology. In particular, we examine if an acoustic illusion of being in a different city can be created when the user's soundscape is overlaid with a different soundscape based on the classification result.

In this thesis, we explore the current state-of-the-art solutions for ASC by investigating the submissions of the DCASE 2020 challenge. Based on the evaluation we choose a baseline model, which is a ResNet that accepts log-mel spectrograms complemented by log-mel deltas and delta-deltas as input. We modify the network in order to classify 9 acoustic scenes. The TAU Urban Acoustic Scenes 2020 Mobile dataset is used to train the model. To calibrate the network, we experiment with different hyperparameters, loss functions and data augmentation strategies and compare the results based on the test accuracy. We achieve the best performance with a test accuracy of 77.99% by using categorical focal loss and Mixup as data augmentation technique.

For the prototype we select 4 cities to support and create a soundscape dataset including audio samples for each scene and city by extracting the audio data from videos that contain the desired content. Then, we develop a server application with Flask to provide an API to get predictions from the model and to log data about the performed classification processes. Eventually, we design and implement a mobile application using the Flutter SDK. The application requests at intervals predictions from the server for audio data, that it recorded using the device's microphone. Based on the classification result, the mobile application overlays the acoustic environment of the user by playing an audio sample for the recognized scene and the selected city from the soundscape dataset.

Finally, we use the prototype in a case study where a group of participants tests the mobile application within a predefined scope. In the evaluation, we estimate the classifier's accuracy in a real life scenario with 65%. We discover that the acoustic illusion of being in a different city is created by the prototype. Additionally, we show that this illusion is experienced more often the more accurate the classification is perceived and the closer the personal relationship between the user and the selected city is.

Contents

CHAPTER 1

Introduction

The first chapter gives an overview over this work. In Section 1.1 the subject of the thesis is introduced and reasoning behind why this was chosen is explained. Additionally, the associated challenges are laid out. The contribution of this work is stated in Section 1.2. Lastly, Section 1.3 gives an overview over the structure of the thesis.

1.1 Motivation and Problem Statement

Ever since the start of the *Digital Revolution*, computers have become the main tool for data processing. Over the past decades, the computing power and the storage capacities have increased tremendously, turning computers into more intelligent information processing tools that are able to recognize patterns and learn new information. Due to this development, and the increasing amount of available digital data, machine learning has become a growing research field and has gained importance in recent years. The field of machine learning is the study of algorithms that allow computer programs to automatically improve through experience [Mit97]. This is achieved by using algorithms that can analyze data, learn from them and then make a determination about new data. The ability to recognize unseen data is one of the reasons why current approaches to object recognition make essential use of machine learning methods. Machine learning systems are used to identify objects in images, to transcribe speech into text or to select relevant results of searches. Increasingly, these applications make use of a class of techniques inspired by the structure and function of the brain's neural networks called deep learning. Deep learning is a new field of machine learning that has been growing rapidly due to its huge success in a variety of application domains by outperforming the traditional machine learning approaches, such as for speech recognition [HDY+12] and image classification [KSH12]. Furthermore, with the increasing demand for applications that can benefit from context awareness, like smart devices and robotics, environmental sound processing has attained popularity [SAW95]. New fields such as computational

auditory scene analysis [WB06], soundscape cognition [RV13], sound event detection [YKM17] and acoustic scene classification [BGSP15] have emerged under the broad spectrum of machine hearing [Lyo10]. This work attempts to contribute to the research of acoustic scene classification.

Acoustic scene classification (ASC) allows the recognition of environments based on their general acoustic characteristics by selecting a semantic label (e.g. airport, metro station, urban park) to an audio recording. The first method appearing in the literature to specifically address the ASC problem was published about two decades ago [SM97]. Since then, significant research efforts have been put into this classification task due to its diverse applications: automatic audio surveillance [AMK06], robotics sensing [MAGH14], multimedia content analysis [WLH00] and machine listening [BGSP15]. This is not surprising, as audio information, in addition to images, often can be a good supplement for solving many real world problems. For example, during hours of darkness, ASC can keep the surveillance system running.

ASC systems can be described as a composition of two main parts: feature extraction and decision making based on pattern recognition. In order to recognise patterns in audio samples, the information describing the most meaningful characteristics of the signal has to be retrieved first. This is in order to obtain a compact and expressive description that is machine-processable. Audio feature extraction uses signal processing techniques and addresses the analysis and extraction of meaningful information from audio signals [CB05]. Features may capture audio properties, such as the fundamental frequency and the loudness of a signal [MZB10]. As of today a large number of audio features exists in audio retrieval for different purposes. Mel-Frequency Cepstral Coefficients (MFCCs) is a common approach for acoustic feature representation and was originally used in speech recognition [MZB10, PLV+19]. Due to the big success of deep learning in image classification, log-mel spectrograms became the dominant feature of deep learning based solutions for ASC [KSH12, PLV+19]. Once the features are extracted, a classification technique is used to catalogue the features [CS03]. It is common to combine several features into a feature vector. The goal of the classification is to predict the class membership of a pattern represented by a feature vector. The output of a classifier is the predicted class label of the feature vector. The recent research shows that classification systems based on Deep Neural Networks (DNNs) outperform the traditional classification approaches based on Gaussian mixture models and support vector machines [SGG20, MHB+18].

This thesis explores the usage of ASC in the private sector. Having lived abroad in several countries for most of my life, I know from personal experience how it is to be homesick. Hence, I quickly warmed up to the idea of creating a mobile application that could create an acoustic illusion of being back home. The aim of this work is to provide a cross platform mobile application that can classify the environment based on the surrounding sounds and play matching sound samples of the same scene of a different city according to the classification result. The focus of the thesis is to answer the following research questions:

- Which accuracy does the classifier have when applied to real life recordings?
- How does the user experience the overlay of a different soundscape? (How does it feel to walk on a public square in Vienna and experience the soundscape of a public square in Moscow?)
- Does the overlay have any effect on the user's perception of their surroundings?
- Will the app manage to provide an acoustic illusion of being in the selected city?

In order to answer these questions, several problems have to be solved. First, the state-of-the-art methods for ASC have to be researched and compared. Then, based on the outcome of the research, the most promising ASC classifier has to be chosen and trained. As for the training, a large enough dataset including recordings of several scenes in different cities has to be found or created. Afterwards, a mobile application has to be implemented which will record the surrounding sounds at regular intervals, classify those recordings and then play a sound sample of a chosen city based on the classification results. Once the prototype is implemented, a case study has to be designed and taken by a group of people to evaluate the classifier's performance in real world scenarios and the overall experience of the users. Eventually, the results of the case study have to be analysed and final conclusions have to be made.

1.2 Contribution

The main goal of this thesis is to provide a prototype that uses ASC and to use it for a case study to research the effect it has on human's perception. The author's contribution consisted of several parts that are explained below.

First, we did the literature research to gather background information which served as the theoretical basis of the classifier. Subsequently, we investigated a few chosen state-of-the-art methods for ASC and compared in details. For this purpose, we examined the submissions of the challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) in 2020. Based on the outcome of that analysis, we selected an ASC framework as the baseline for the planned classifier. We adapted, calibrated and trained the chosen framework using machine learning techniques which required appropriate training and test datasets being explored and modified according to the requirements. Once the model was ready to be used, we implemented a web application to provide a RESTful API to get predictions from the model. Then, we developed a cross platform mobile application with the following features:

- A simple UI with the possibility to choose a city and to start the classification process
- Classification of the environment at regular intervals by recording the surrounding sounds and sending them to the web-based classification service

- Audio player to play an audio sample of the chosen city accordingly to the prediction of the model

Of note is that we created the collection of audio samples to play in response to the classification results independently. For this purpose suitable video recordings were searched online, evaluated and processed. Eventually, in the last phase of this work we designed a case study focusing on the questions this thesis aimed to answer. Many aspects had to be taken into account, such as the composition of the test group, the testing conditions and the structure of the survey. Finally, we analysed and presented the results of the case study. Even though the presented results could not be counted as statistically representative due to the small number of participants, interesting correlations could be observed and corresponding conclusions drawn.

1.3 Thesis Structure

The overall ambition of this thesis is to develop a software prototype that uses ASC to create an acoustic illusion and to evaluate its effect on human's perception when used in real world scenarios. The steps to achieve this goal are mirrored by the structure of the thesis, which is organized as follows: In Chapter 2 the theoretical background of this work, including theory of audio signal processing, audio feature extraction and machine learning methods, is described. Chapter 3 gives an in-depth description of the state-of-the-art ASC frameworks and their comparison as well as the design and the implementation of the prototype. The evaluation of the prototype, consisting of the case study and the analysis of the case study results, is explained in Chapter 4. Finally, in Chapter 5 a summary of conclusions and future work is presented.

CHAPTER 2

Theoretical Basis

In this chapter, the background information about the methods used throughout this thesis is presented. First, Section 2.1 explains how an audio signal is processed and represented. Then, Section 2.2 gives an overview of feature extraction techniques that address the analysis and retrieval of meaningful information from audio signals. Afterwards, Section 2.3 describes the deep learning techniques used for classification tasks including corresponding artificial neural networks, their learning process and calibration. In Section 2.4 the data augmentation approaches used in Chapter 3 are explained. Lastly, Section 2.5 takes a look at related work.

2.1 Digital Signal Processing

Digital Signal Processing (DSP) is concerned both with obtaining discrete representations of signals, and with the theory, design, and implementation of numerical procedures for processing the discrete representation [RS78]. DSP is a core subject in many fields like electronics, communication and computer engineering. In the context of this work, this section focuses on audio signals. First, it explains what an audio signal is and with which techniques it is digitized. Then, it presents the mathematics and the algorithms used to represent an audio signal in different domains.

2.1.1 Audio Signal

One of our most important ways of sensing the world around us is sound. Imagine you are standing on a public square in a city with your eyes closed. What do you hear? Perhaps people talking, some cars and buses driving on the road, a bicycle passing by, and clunks from shopping bags and boxes. Your sense of hearing tells you what is happening around you. As humans we are able to distinguish individual sound sources from a complex mixture of sounds. We use this skill of listening everyday effortlessly. What we

usually take for granted, is a very challenging task for computers. *Machine hearing* is an emerging academic field with the focus on algorithms that can teach machines to analyze sounds in the same way that people do [Lyo10]. Before we can investigate how human hearing can nowadays be modeled artificially, we first have to understand what sound is and how it is represented to be machine readable.

Sound is a physical phenomenon [Chr19]. When an object vibrates, it creates a pressure wave that causes particles in the surrounding medium (air, water, or solid) to vibrate and by that to transmit the vibrational motion further through the medium. The human ear detects this wave when it enters the human eardrum causing the small parts within the ear to oscillate. Technically speaking what we perceive with our ears are changes in pressure over time. There are five main characteristics of sound waves: wavelength, amplitude, frequency, time period, and velocity [Dit17]. The wavelength is the distance between adjacent identical parts of a wave, parallel to the direction of propagation. The amplitude defines the maximum displacement of the particles disturbed by the sound wave as it passes through a medium, and determines the sound's relative loudness measured in decibels (dB). The frequency is the number of waves passing by a specific point per second measured in Hertz (Hz). The time period is the amount of time it takes for one wave cycle to complete. Each complete wave cycle begins with a trough and ends at the start of the next trough. Lastly, the velocity is the propagation velocity of a wave. Sound waves can be converted back into physical motion by the diaphragm of a microphone, and are generally represented as a waveform, where the y-axis represents the amplitude of the pressure and the x-axis represents time as shown in Figure 2.1.

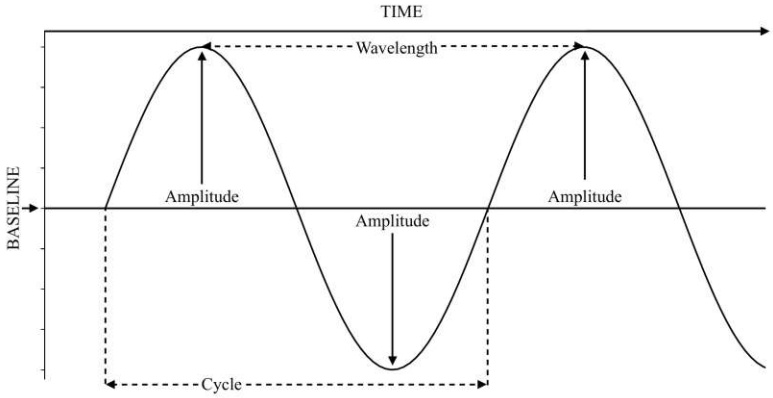
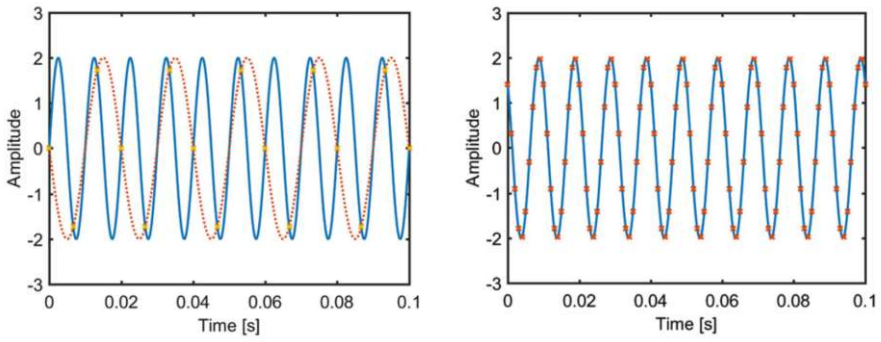


Figure 2.1: The five main characteristics of a sound wave.

In the context of this work, taking into account that a *signal*, represented as a function of one or more variables, may be defined as an observable change in a quantifiable entity [Cha18a], *Audio signals* are defined as electronic representations of sound waves in either digital or analog format [LH09]. Audio signals are audible sound signals, while audible sound consists of frequencies that are in the human hearing range of roughly 20 - 20 000 Hz [KS16]. Sound can be recorded with an electroacoustic transducer such as a

microphone, which translates the pressure wave into a voltage signal [Chr19]. This analog signal is a continuous signal, i.e., it can take on a value at any time, as is the pressure at a point in space. Since computers can only work with discrete and digital data, the analog signal has to be digitized first. This is done with the analog-to-digital conversion (AD conversion) that converts signals from analog to digital by sampling and quantization. The AD conversion includes the following steps [VPE18]:

- **Filtering**: An analog signal can include frequencies outside of the frequency range that is digitized. An effect called *aliasing* occurs when a frequency is sampled with a lower frequency rate than needed to reconstruct it resulting in an incorrectly lower frequency [Chr19]. In order to avoid this unwanted distortion of the original sound, a filter has to be applied to the signal to limit its frequency bandwidth by removing or attenuating all frequencies above the stopband frequency. This filter is called *low-pass filter* or *anti-aliasing filter* and is a mandatory step before the signal can be sampled [Chr19, VPE18].
- **Sampling**: Sampling refers to measuring the signal's amplitude (taking samples) at uniform time intervals in order to represent a continuous signal as a sequence of numbers. The *sampling rate* or *sampling frequency* defines the number of samples taken per second and is specified in Hz [Dit17, HR17]. Choosing the sampling rate too small can lead to loss of information and is called *undersampling* [OSB99]. On the contrary, choosing a too high sampling rate can result in a higher amount of data than needed for the reconstruction of the signal and is called *oversampling*. According to the Nyquist–Shannon *sampling theorem* [Sha49] to avoid aliasing, a signal with the frequency bandwidth $[0, f_{max}]$, where f_{max} is the highest frequency, needs to be sampled with the sampling rate f_s that is twice the highest frequency $f_s = 2f_{max}$. Figure 2.2 illustrates the sampling process with and without aliasing.



(a) (b)

Figure 2.2: Sampling of an audio signal where aliasing occurs (2.2a) and when the sampling theorem holds (2.2b). The *blue curve* illustrates the original signal and the *orange curve* illustrates the reconstructed signal [Chr19].

- **Quantization**: Before the samples can be stored on a digital storage medium their values have to be quantized into a digital code. Quantization is the process of mapping the samples (amplitude values from a continuous set) to predefined values [VPE18]. The *bit depth* defines the number of bits to encode each sample and corresponds to the resolution of the AD conversion [HR17]. The quantization process is depicted in Figure 2.3.

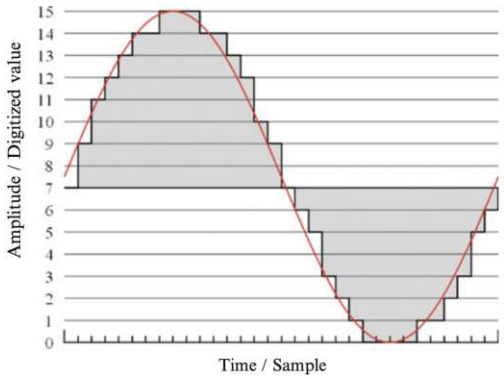


Figure 2.3: Quantization of an audio signal. The analog signal is represented by the *red curve*. Measurements are taken at fixed points in time (x-axis) and encoded using a range of 4 bits (y-axis) [KS16].

As mentioned earlier the human ear can perceive frequencies up to 20 kHz. Following the sampling theorem it is therefore standard that music and other sound waves are sampled with 40 kHz or greater. Typical values for compact disc quality (CD) are a sampling rate of $f_s = 44.1$ kHz and a 16 bits resolution leading to a bit rate of 705 600 kbit/s for a single channel audio signal. Sampling rates of 48, 96 or 192 kHz and quantization on 24 bits are used for higher quality standards [VPE18].

2.1.2 Audio Signal Representation

A sound signal can be represented in different domains. In the previous section the waveform was introduced which is a time-domain representation meaning voltage is expressed as a function of time. In most cases it is very difficult or nearly impossible to discriminate between sound scenes from a waveform. Therefore, sound signals are usually converted to the frequency-domain as this representation is more inline with the human perception [VPE18]. Nonetheless, this representation is usually too generic in order to identify the content of an audio signal that consists of different sound events at certain times. The time-frequency-domain reveals more information about a sound scene and it will be used in Section 2.2 to derive features from, which are more suitable for characterizing the semantic content of an audio sequence.

By using a pair of mathematical operators called a transform, a signal can be converted between the time- and frequency-domains. While the time-domain representation shows how a signal changes over time, the frequency-domain representation shows the distribution of a signal's energy over a range of frequencies [KS16]. A well-known and widely used linear operator for extracting the frequency behavior of signals is the Fourier Transform. The Fourier transform was named after the French mathematician and physicist Jean Baptiste Joseph Fourier and it states that any continuous and periodic function can be represented as the sum of sine and cosine waves oscillating at different frequencies [KS16]. The representation of the Fourier transform is called the *spectrum* of the signal [VPE18]. The Fourier transform of a signal is sometimes called the Forward Fourier transform and is given by Equation 2.1 [Boa16]:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft} \tag{2.1}$$

While the classical Fourier transform is suitable for continuous and stationary signals, for digital audio its discrete version the Discrete Fourier Transformation (DFT) is used [VPE18]. DFT decomposes a sampled signal into a set of DFT coefficients, which tell which frequency components are present in the signal as well as their relative intensity. Assuming that the signal has already been windowed, as explained later in this section, DFT is defined in Equation 2.2, where m is an integer ranging from 0 to $K-1$, K is the number of samples in the frame, and x_k represents the k^{th} input amplitude, i.e., the k^{th} sample in the frame [KS16]:

$$X_m = \sum_{k=0}^{K-1} x_k \cdot e^{\frac{-2\pi i}{K} \cdot k \cdot m} \tag{2.2}$$

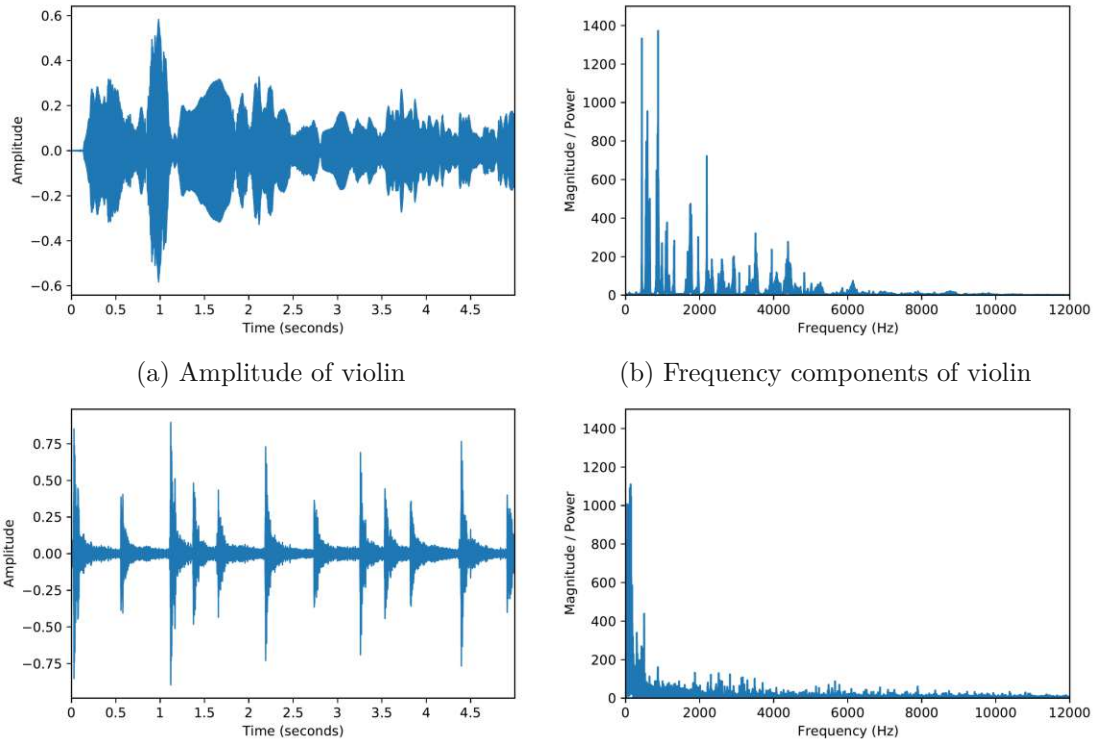
It can be seen that a direct evaluation of each value of X_m requires K^2 complex multiplications and $K(K-1)$ complex additions that have to be solved for the computation of an K-point DFT. Consequently, for large K, the computational complexity gets very high and can only be solved with appropriate computational power. In 1965, Cooley and Tukey proposed a more efficient algorithm called the Fast Fourier Transform (FFT) that reduces the computational complexity of an K-point DFT from $O(K^2)$ to $O(K \cdot logK)$ by decomposing the DFT into a sum of smaller DFTs [CT65]. As of today FFT stands for a number of efficient algorithms that have been developed for the computation of the DFT. The result of FFT is a sequence of complex numbers, in which the real-valued part $Re(X_m)$ and the imaginary part $Im(X_m)$ illustrate the importance of the cosine and the sine waves, respectively [KS16].

$$|X_m| = \sqrt{Re(X_m)^2 + Im(X_m)^2} \tag{2.3a}$$

$$\phi(X_m) = \tan^{-1} \frac{Im(X_m)}{Re(X_m)} \tag{2.3b}$$

The Equation 2.3a shows how the magnitude (power) can be obtained, which represents the amplitude of the combined sine and cosine waves. Hence, Equation 2.3b defines the phase, which indicates the relative proportion of sine and cosine.

To exemplify the result of the DFT, Figure 2.4 illustrates the representations of a violin sound and of a drum pattern sound in the time- and frequency-domains. Both audio segments[1] are 5 seconds long and were sampled with 44.1 kHz and a 16 bits resolution. The figures were generated using the Python libraries SciPy[2] and Librosa[3].



(a) Amplitude of violin

(b) Frequency components of violin

(c) Amplitude of drums

(d) Frequency components of drums

Figure 2.4: Representations of a violin sound and of a drum pattern sound in the time- and frequency-domains.

The figures on the left show the representation of the respective audio signal in the time-domain, plotting amplitude values over time. The figures on the right depict the Fourier-transformed signals, plotting magnitude information over frequencies. Looking at the time-domain representations, it can be observed that the drum sound shows

[1] Both examples taken from *Freesound* (`https://freesound.org`) a collaborative database of freely available audio snippets: violin sound sample (`https://freesound.org/s/92002` visited on 02.11.2021) licensed under Creative Commons Sampling Plus 1.0; drum pattern sample (`https://freesound.org/s/58866` visited on 02.11.2021) licensed under Creative Commons 0 1.0 Universal.

[2] `https://scipy.org` (visited on 02.11.2021)

[3] `https://librosa.org` (visited on 02.11.2021)

repetitive patterns, indicating a less melodic sound, in contrast to the violin. This observation gets reaffirmed by comparing the frequency-domain representations of the two sounds, which shows that the violin covers a larger frequency range than the drums. In particular, Figure 2.4b demonstrates that most of the energy in the violin sound is concentrated between 400 and 2500 Hz, while according to Figure 2.4d the main frequency component in the drum sound appears up to approximately 500 Hz. The comparison of the representations demonstrates the benefits of converting an audio signal to the frequency-domain since the distribution of power over frequencies present in a sound signal can be seen as its fingerprint. For some applications the analysis of the frequencies is sufficient. When it comes to the classification of acoustic scenes, it is important not only to analyse the present frequencies, but also the time frame when these occurred in the signal [VPE18]. To analyze the frequency content of signals, when it varies with time, a different transform, the Short Time Fourier Transform (STFT), is used [DRS18]. The STFT is a well-known technique in signal processing to represent a signal in the time-frequency domain and includes the following steps [VPE18]:

- **Framing**: To overcome the lack of time information the signal is not analysed as a whole, but is divided into chunks, i.e. short-time frames [KS16]. A frame is an aggregation of consecutive samples. The number of samples to include in a frame is chosen to be long enough to be perceivable by the human ear. Therefore, when deciding on a proper frame size, the sampling frequency and the time resolution of the human ear, which is around 10 ms, should be taken into account. Taking the sampling frequency of 44.1 kHz, where each amplitude value is valid for only 0.0227 ms, the frame size should be 442 samples or greater [KS16].

- **Windowing**: Abrupt changes at the frame boundaries can cause distortions in the spectrum, which is referred to as *spectral leakage* [KS16]. Windowing refers to applying a windowing function to each frame to avoid or to reduce spectral leakage. One of the most common windowing functions is the Hann function, named after the Austrian meteorologist Julius von Hann. The Hann function can be seen as a taper formed by using a weighted cosine where the end points touch zero. It is defined in Equation 2.4, where K is the frame size and $k = 1...K$ [KS16].

$$w(k) = 0.5\left(1 - \cos\left(\frac{2\pi k}{K-1}\right)\right) \tag{2.4}$$

  The windowing process using a Hann window is illustrated in Figure 2.5. As can be seen in the figure, the samples towards both ends of the frame are suppressed resulting in loss of information. To avoid this the window is shifted with an overlap ratio called the *hop size* which defines the amount of information that is shared by the consecutive window locations [KS16]. The hop size is typically selected so that the consecutive frames are overlapping at least 50% [AR77]. Since the windowing function is applied to a frame, in some literature the frame size is referred to as the window size.

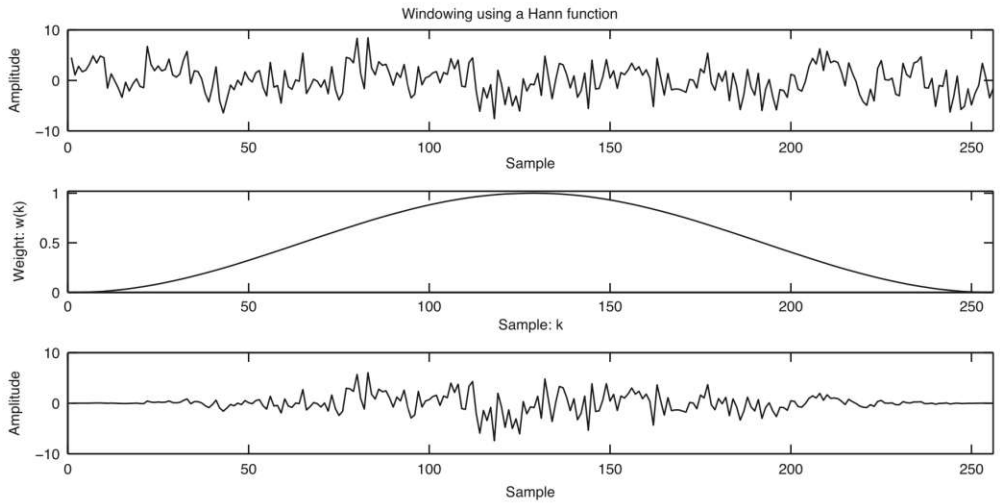


Figure 2.5: Windowing of a 256-sample frame using a Hann function [KS16].

- **FFT**: The window is incrementally shifted over the signal until it reaches the end of the signal. The FFT is computed for each windowed segment in temporal order and the resulting frequency domain representations of the individual FFTs are concatenated [KS16].

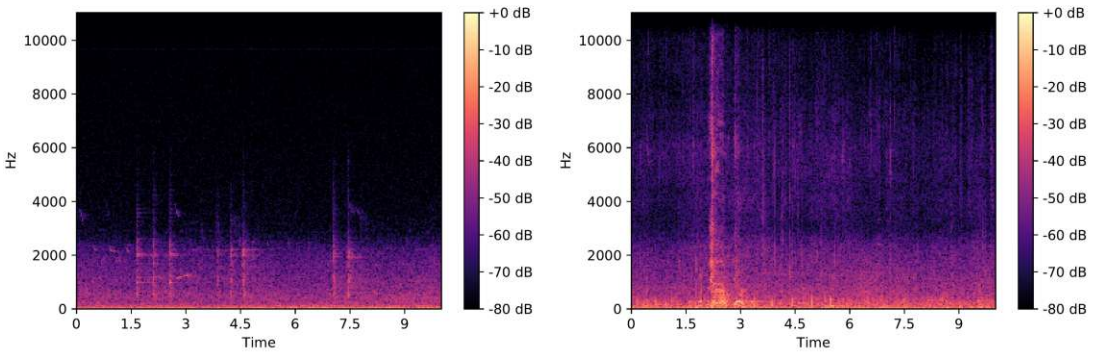
The STFT can be described as the approximation of the spectrum $X(f)$ by applying the DFT on a windowed frame of length N of the signal $x(n)$. The Equation 2.5 shows how the fth component of the DFT of the tth frame of $x(n)$ is computed where $w(k)$ is the window function and the hop size equals to the length of the frames (N) meaning there is no overlap between consecutive frames [VPE18].

$$X(t,f) = \sum_{k=0}^{N-1} w(k)x(tN+k)e^{\frac{-i2\pi kf}{N}} \tag{2.5}$$

The important parameters of the STFT are the frame size, the hop size and the windowing function. When a smaller frame size is chosen, high time resolution and low frequency resolution properties are achieved. In contrast, a larger frame size provides higher frequency resolutions and low time resolutions [EISA18]. Depending on the hop size some information gets lost or is analysed in several windows. The windowing function is chosen depending on the needs of the particular application and the frequency content of the signal as it impacts the frequency resolution and spectral leakage.

The result of STFT is a matrix which carries both the time and frequency information. The time information corresponds to the number of frames. Hence, the matrix includes for each frame the distribution of power over frequencies divided into frequency bins as a sequence of complex numbers. Calculating the magnitude of each frequency bin for each

frame results in a matrix of the same size as the STFT matrix consisting of real numbers. This matrix can be depicted along the time axis as a *spectrogram* [KS16]. Spectrogram is a popular 2D representation of the audio content in audio signal processing, where the x-axis represents time and the y-axis represents frequency [VPE18]. Each pixel of the spectrogram image represents the amplitude of the signal in that frequency (vertical axis), at that time (horizontal axis).



(a) Park (b) Airport

Figure 2.6: STFT spectrograms of sound signals recorded in Vienna.

Figure 2.6 illustrates STFT spectrograms of sound samples[4] that were recorded in Vienna in different scenes. The audio segments are 10 seconds long and were sampled with 44.1 kHz and a 24 bits resolution. The spectrograms were generated using the Librosa library which provides STFT as its core application. The sound recorded in the park consists mainly of birds' twittering and some background noise. Approximately at the seconds 2, 4 and 6 there is a sound of a hammer, probably from a construction nearby, that hits 3 times. It can be seen that the Subfigure 2.6a illustrates the content of the park sound pretty well, as it shows the presence of higher frequencies, those of the birds, and a change in those at the mentioned times. The audio recorded at the airport consists of different types of sounds like foot steps, people talking and various movement noises in the background. You can hear someone coughing loudly around second 2. Respectively to the content of the airport sound sample, the Subfigure 2.6b shows a larger frequency range with a peak around second 2. It can be observed that the spectrograms deliver a good description of the content of a sound and can be seen as the fingerprint of sound scenes. Which spectrograms were used in the context of this work will be explained in the next section.

[4]Both examples taken from *TAU Urban Acoustic Scenes 2020 Mobile Development dataset* [HMV20b]: park sound sample *park-vienna-104-2970-a.wav*; airport sound sample *airport-vienna-13-524-a.wav*.

2.2 Acoustic Feature Extraction

Digital audio signals are rarely used as input for classification systems, but rather their general acoustic characteristics (often called *features*). The acoustic features provide a numerical representation of the signal content relevant for machine learning, characterizing the signal with values which have connection to its physical properties, its distribution in frequency, and change over time. Acoustic feature extraction is the process of obtaining information that is sufficient for the classification of the target sounds, making the subsequent modeling stage computationally cheaper. This section gives an overview of the acoustic features commonly used for ASC and presents the theory of the acoustic features used in this thesis [VPE18].

2.2.1 What is a feature?

In machine learning, a classification task is about developing an algorithm that is able to specify to which category (class) some input belongs to [GBC16]. ASC refers to assigning an audio sample to one of the predefined scenes. A machine learning algorithm is designed to learn from *examples*, which are collections of features [GBC16]. A feature is a compact and expressive description of raw data that is machine-processable [MZB10]. A feature refers to a numerical (scalar, vector, or matrix) description and is the result of the feature extraction process [KS16]. The purpose of the feature extraction is to extract meaningful information that is significant for the particular task, and by that to reduce the amount of input data for the classification system. Furthermore, the feature design and parameters used in the extraction commonly rely on prior knowledge and assumptions about the content of the signal. Hence, audio features are usually designed in the context of a specific task and domain, and do not generalise well [VPE18]. Since some tasks are related, it is not unusual that features are employed for different tasks than they were originally developed for. A good example are cepstral coefficients, such as mel-frequency cepstral coefficients (MFCCs) [DM80]. MFCCs have originally been developed for speech recognition and were later used in other domains such as music information retrieval and ASC [TC02, MHB+18]. The necessary property of features is low variability for features for examples that are associated with the same class, and at the same time high variability allowing the distinction between features from examples that belong to different classes [VPE18]. When it comes to audio samples, it is important to note that their data type is a time series as they refer to a set of repeated measurements over time [DL18]. Time series are represented by an ordered vector $x = (x_1, x_2, ..., x_N)$, where N measurements have been taken at times $t = (0, \Delta t, 2\Delta t, ..., (N-1)\Delta t)$ with a constant sampling period Δt [DL18]. This allows the representation of real-valued sequential data in the same way, such as spectra (where measurements are ordered by frequency), and the recognition of patterns.

The level of abstraction of acoustic features can be described on a scale of three levels: low-level, intermediate-level and high-level [LWC98]. Low-level features are typically computed directly from the raw audio waveform and constitute of acoustic characteristics

such as loudness or bandwidth of an audio signal. Intermediate-level features capture aspects that are more meaningful providing audio signatures (e.g. MFCCs). On the highest level, the features represent semantic models of audio in different scene classes [LWC98].

2.2.2 Mel Scale

The characteristics of sound that were introduced in Section 2.1.1 such as frequency are objective, in a sense that they can be measured with an electronic device. Human perception studies have shown that sound has properties that are subjective as they are auditory perceptions in our heads. *Pitch* is a perceptual property which allows the ordering of sounds on a frequency-related scale extending from low to high [Kla06]. It defines the frequency of a sine wave that is matched to the target sound by human listeners [Har96]. The corresponding physical term is *fundamental frequency* which is defined for periodic or nearly periodic sounds only [Kla06]. Perceptual studies have shown that our perception of sound differs in each region of the spectrum and that the resolution of the human ear also varies along the frequency axis. In order to mimic human hearing and to provide a way to extract information from sound signals that is more inline with the human perception, non-linear-frequency scales have been developed. Stevens, Volkmann and Newman introduced the *mel scale* in 1936, which relates pitch to frequency. The unit name *Mel* was taken from the root of the word *melody* [SVN37]. The mel scale is defined by a logarithmic function that was determined by perceptual experiments designed to find a linear relation among perceived pitches. Several listening experiments were executed by different authors and led to different versions of formulas which are only approximations [SV40, O'S99]. The Equation 2.6 shows a common relation between the mel scale $mel(f)$ and the Hertz scale f that was given by Fant [Fan68].

$$mel(f) = \frac{1000}{\log 2} \log(1 + \frac{f}{1000}) \tag{2.6}$$

The formula is used to visualize the relationship between Hertz and Mel as shown in Figure 2.7.

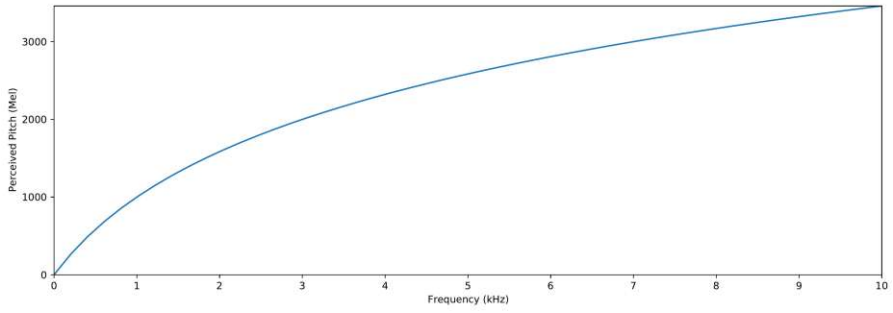


Figure 2.7: Relationship between Hertz and Mel.

The core idea is that sounds of equal distance on the mel scale are perceived to be of equal distance to humans. It can be seen in Figure 2.7 that the perception of pitch intervals is approximately linear for frequencies below 500 Hz, and above that threshold frequency ranges that are judged to yield equal pitch intervals become larger and larger. The scale's reference point is 1000, which means that 1000 Hz equals 1000 Mel. To give an example, the range [1000; 1150] Mel corresponds to [1000; 1220] Hz, whereas the interval [3310; 3460] Mel equals [8400; 10000] Hz. Even though the first range only covers a frequency interval of 220 Hz, the perceived pitch difference between 1000 and 1220 Hz equals the perceived pitch difference between 8400 and 10000 Hz, which covers 1600 Hz. In other words, the higher a frequency the larger the difference has to be in order to be perceived differently.

2.2.3 Mel Filter Bank

In signal processing, a filter bank is an array of bandpass filters that splits the input signal into multiple frequency ranges, which are referred to as frequency bands [PNJ19a, Smi11]. The mel filter bank consists of overlapping triangular filters where each filter is spaced according to a mel frequency scale. The cutoff frequencies of each filter are determined by the centre frequencies of the two adjacent filters [CB96]. An example of a mel filter bank with 18 filters is illustrated in Figure 2.8.

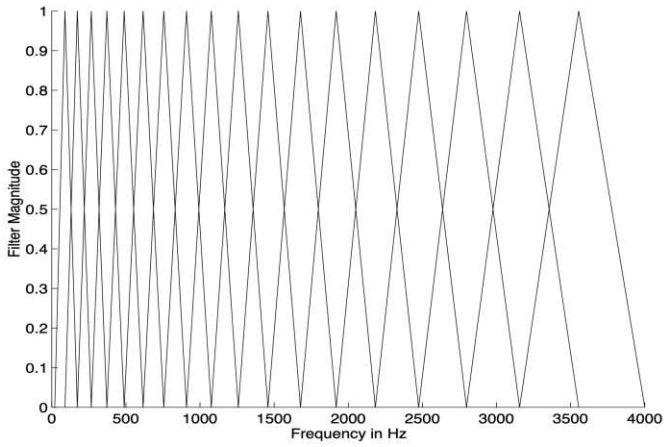


Figure 2.8: A mel filter bank containing 18 filters [CB96].

Given a windowed signal with a frame size N, the sampling rate f_s and $k = 0...N-1$ then k corresponds to the frequency $l_f(k) = kf_s/N$. The mel frequency ϕ_f and the linear frequency l_f are related by the definition of the mel scale as shown in Equation 2.6. A mel filter bank is defined by the number of mel filters F, the minimum frequency $l_{f_{min}}$ and the maximum frequency $l_{f_{max}}$ [KL10]. If $\phi_{f_{max}}$ and $\phi_{f_{min}}$ are the frequencies on the mel scale corresponding to the linear frequencies $l_{f_{max}}$ and $l_{f_{min}}$ respectively, a fixed frequency resolution in the mel scale is computed using $\delta\phi_f = (\phi_{f_{max}} - \phi_{f_{min}})/(F+1)$. The mel filter bank $M(m,k)$ is given by the Equation 2.7 where $m = 1...F$ and $l_{f_c}(m)$

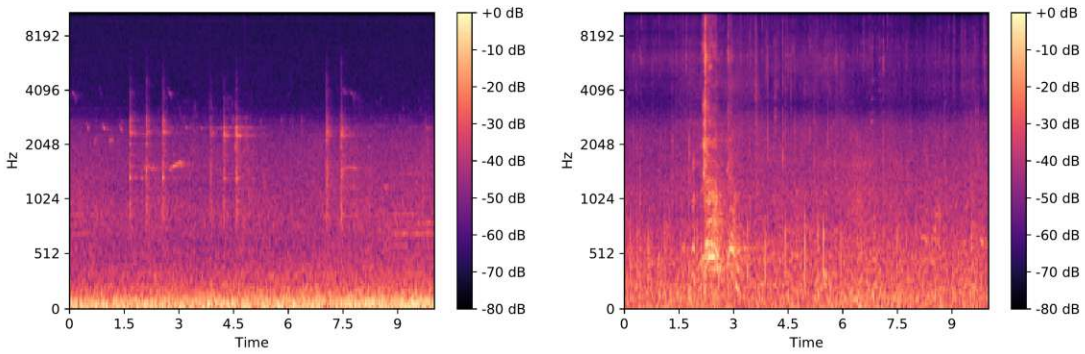
is the center frequency of mth filter [KL10, SPL06]. The mel filter bank $M(m,k)$ is an $F \times N$ matrix.

$$M(m,k) = \begin{cases} 0 & \text{for } l_f(k) < l_{f_c}(m-1) \\ \frac{l_f(k) - l_{f_c}(m-1)}{l_{f_c}(m) - l_{f_c}(m-1)} & \text{for } l_{f_c}(m-1) \leq l_f(k) < l_{f_c}(m) \\ \frac{l_f(k) - l_{f_c}(m+1)}{l_{f_c}(m) - l_{f_c}(m+1)} & \text{for } l_{f_c}(m) \leq l_f(k) < l_{f_c}(m+1) \\ 0 & \text{for } l_f(k) \geq l_{f_c}(m+1) \end{cases} \tag{2.7}$$

Mel-band energies and mel-frequency Cepstral Coefficients (MFCCs) are the most common acoustic features used to represent spectral content of audio signals and both include the mel filter bank in their design [VPE18]. Mel-band energies refer to the measurement of the collective energy output of the respective frequency bands by taking the logarithm of the magnitude of each resulting band. MFCCs are obtained by computing the discrete cosine transform (DCT) of the log energy in mel frequency bands. The mel filter bank can also be used to generate a mel-spectrogram which is explained in the next section.

2.2.4 Mel-Spectrogram

Spectrogram was explained in Section 2.1.2 and is a graphical representation of the spectrum of an audio signal. It is an image-based feature derived from the time-frequency content that is on a linear-frequency scale. Mel-spectrogram is a spectrogram that takes the human perception into account by applying the mel filter bank to convert the frequencies to the mel scale [VPE18]. Figure 2.9 illustrates mel-spectrograms of the same sound samples as shown in Figure 2.6 and were generated using the Librosa library and 128 Mel filters.



(a) Park (b) Airport

Figure 2.9: Mel-spectrograms of sound signals recorded in Vienna.

It can be seen that in comparison to the linear-frequency spectrograms the mel-spectrograms have a high resolution for low frequencies which decreases for the higher frequencies.

The changes in the frequencies are depicted more clearly making it easier to recognise patterns. Mel-spectrograms became popular features for ASC in recent years due to the success of deep learning in image classification [KSH12], such that best practice image classification methods can be applied [HMV20a, MHV19, MHV18a].

2.2.5 Derived Features

When it comes to the classification of sound scenes, where longer segments of sound are used, the temporal integration of features can be important as it provides information about the dynamics. While features like mel-band energies and MFCCs represent the local spectrum, they neglect temporal changes in the spectrum over time [VPE18]. New features can be generated from the previously extracted features [Ler12], therefore one of the most common ways to model the temporal information is to extend the feature set with their first and second order derivatives. The first derivatives (referred to as delta coefficients or delta features), can help detect changes of feature values as they may mark the boundaries of important segments. The second order derivatives (referred to as acceleration coefficients or delta-delta features), can be used to estimate the acceleration of these changes [VPE18, YEG+15]. The Equation 2.8 shows how the first order derivatives can be calculated, where d_t is the delta coefficient at time t, computed in terms of the corresponding static coefficients $c_{t-\theta}$ to $c_{t+\theta}$ with Θ defining the size of the delta window [YEG+15].

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{2.8}$$

The second order derivatives are calculated in the same way, but they are calculated from the deltas, not the static coefficients. Using delta features showed improvement of the classifier's performance in acoustic speech analysis [Pic15, MLM09, ZPB+16, AMJ+14]. This inspired authors to complement the static features with their derivatives for their ASC systems [HL16, MG20]. In particular, recent approaches for ASC used a combination of log-mel deltas and delta-deltas with a mel-spectrogram as input for deep neural networks [SPJL20, HYX+20, GM20]. Deltas of a spectrogram can be computed either on time or frequency [TLX+19]. Deltas along time axis represent the dynamic time features, while deltas along frequency axis reveal the dynamic change of the frequencies.

2.3 Deep Neural Networks

Machine learning algorithms that learn patterns from data are mathematical functions called *models* [LRM21]. For a classification task like ASC a model is asked to assign an input to a predefined category. In order to be able to do that, the model was previously asked to approximate a function through learning that is later used for the assignments [GBC16]. This section provides the background of the deep learning techniques used in this work. First, the terminology of machine learning basics is described. Afterwards,

the architectures of the neural networks relevant for this work as well as their calibration are explained.

2.3.1 Machine Learning Basics

Machine learning models are algorithms that learn from data. But what is the definition of this type of learning? Tom Mitchell provides a succinct definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [Mit97, p. 2]. According to Mitchell, for the definition of a learning problem, three features must be identified: the class of tasks, the measure of performance to be improved, and the source of experience. Many learning problems can be specified in this fashion. To give an example, the ASC problem can be defined as following:

- Task T: recognizing and classifying acoustic scenes within audio recordings
- Performance measure P: percentage of scenes correctly classified
- Training experience E: a database of acoustic scenes with given classifications

The way how the machine learning system should process an *example* describes the task. An example consists of a ground truth is a collection of features and is usually represented by a vector $\boldsymbol{x} \in R^n$ where each entry x_i of the vector is another feature [GBC16]. Besides classification, one of the most typical machine learning tasks is *regression*. Depending on the task, a quantitative performance measure must be designed to evaluate the abilities of the model. For a classification task it is common to measure the *accuracy* of the model [GBC16]. Accuracy is the percentage of examples for which the model produces the correct output. On the contrary, the proportion of examples for which the model makes an incorrect output is called the *error rate*. It is usually the goal to design a model that performs well in a real world scenario. Therefore, the performance measures are evaluated based on the model's outputs for data that it has not seen before. Such data is referred to as the *test set* and it does not include examples used for training. The training experience depends on the design of the learning algorithm.

Deep learning approaches can be broadly categorized as *supervised* and *unsupervised* [ATY+19, GBC16]. Supervised learning is the most common form of machine learning, deep or not, and is a common choice for problems like classification and regression [LBH15, LRM21]. This learning algorithm use a dataset where each example is associated with a ground truth *label* (also referred to as *target*) [GBC16]. Since for each input vector $\boldsymbol{x}$ the value of the desired output vector $\boldsymbol{y}$ is specified, the model learns through observation to predict $\boldsymbol{y}$ from $\boldsymbol{x}$ usually by estimating the probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$. The goal of unsupervised learning algorithms is to build a model that learns the structure of the data from only the data itself, meaning the model is trained with a dataset without labels. This type of learning is usually used for clustering or dimensionality reduction.

Hence, the model observes several examples of a random vector $\boldsymbol{x}$ and attempts to implicitly or explicitly learn the probability distribution $p(\boldsymbol{x})$. In the context of this work, we will focus on supervised learning. For training a model two datasets are needed: *training set* and *validation set* [Cha18b]. The training set is a collection of examples that are fed to the model during training to adjust its parameters. The validation set is used during training to test the model in order to improve it.

2.3.2 Neural Networks

Artificial neural networks (ANNs) are a type of machine learning algorithm that was inspired by the human brain. A single artificial neuron has many inputs, a body, and a single output [Cha18b]. Artificial neurons (also referred to as *units*) are simple computational models of neurons, each producing a sequence of real-valued activations. They are organized in groups (*layers*) and work in parallel. Each neuron in a layer is connected with every single neuron in the next layer. A neuron is a computational unit that calculates an output based on the given inputs (including the biases) and the weights of its incoming connections. Each layer sends its outputs to the next layer. The last layer produces a prediction as output. Neural networks with only an input and output layer are known as *linear models*, which use a linear function to represent the patterns they have learned from data [LRM21]. Layers other than the input and an output layer are called *hidden*, because their output is available only within the network. The amount of layers in a network gives the *depth* of the model. Therefore, neural networks with more than one hidden layer are classified as deep [LRM21]. The *width* of the model is determined by the dimensionality of the network's hidden layers which are typically vector valued [GBC16]. An example of a simple *feedforward neural network* (FNN) with one hidden layer is depicted in Figure 2.10. The network is called feedforward because the information flows in one direction and there are no feedback loops in the network. The weight on the link from units in the input layer i to units in the hidden layer j is denoted with w_{ij}. The subscript k indexes units in the output layer.

One of the properties of FNNs is that their outputs can be expressed as deterministic functions of the inputs [Bis96]. Hence, the whole network represents a multivariate non-linear functional mapping. The function of the illustrated network is presented in Equation 2.9 [Rip96]. Each unit calculates the output y_k by applying a function f_j to x_j which is a sum of the unit's inputs with a constant (the *bias*) added. The weights are the coefficients that are multiplied with the inputs before summing them up. The input units have $f \equiv 1$ as they only distribute the inputs to the next layer.

$$y_k = f_k\left(\alpha_k + \sum_{j \to k} w_{jk} f_j\left(\alpha_j + \sum_{i \to j} w_{ij} x_i\right)\right) \tag{2.9}$$

Considering that a neural network can have K layers and the input is a feature vector, the previous notation can be extended as shown in Equation 2.10.

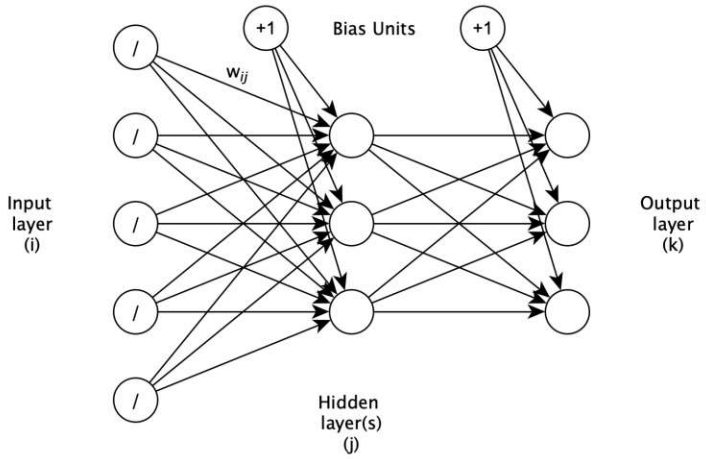


Figure 2.10: A generic feed-forward network with a single hidden layer [Rip96].

$$h^k = f^k(b^k + W^k h^{k-1}) \tag{2.10}$$

Given a vector of offsets b^k and a matrix of weights W^k, layer k computes an output vector h^k using the output h^{k-1} of the previous layer, starting with the input $x = h^0$ [Ben09]. In deep learning the functions f put between layers are called *activation functions*. Those functions are typically nonlinear and they define the output of a neuron given a set of inputs. The recommended activation function for modern neural networks is the *rectified linear unit* or ReLu, defined as shown in Equation 2.11 [GBC16, Cha18b].

$$p(x) = max(x, 0) \tag{2.11}$$

The ReLu function returns the value provided as input x directly for $x > 0$ and the value 0 for $x \leq 0$. It is a piecewise linear function with two linear pieces. This makes rectified linear units nearly linear and therefore many of the properties that make linear models easy to optimize with gradient-based methods are preserved [GBC16]. Equation 2.12 shows another commonly encountered function, the *logistic sigmoid* function.

$$S(x) = \frac{e^{-x}}{1 + e^{-x}} \tag{2.12}$$

The sigmoid function maps the interval $(-\infty, \infty)$ to $(0, 1)$. It is therefore often referred to as the squashing function [Mit97]. This function is useful where a real number needs to be converted to a probability. It is typical to place a sigmoid function as the last layer to convert the model's output into a probability score. Once the outputs are calculated, they have to be evaluated in order to understand how to adapt the model. A *loss function* (e.g. cross-entropy) is applied to the output to measure how accurate it is compared to

the label [Cha18b]. To minimize the loss, the parameters of the network's function have to be adjusted accordingly resulting in an optimization task. One of the most popular algorithms to perform optimization is *gradient descent* [Rud17]. The idea is to lower the loss (descend) by looking at the slope of the loss function (its gradient). Training a neural network means that its parameters (weights and biases) are learned through an iterative process by being updated after each *epoch* in the opposite direction of the gradient of the loss function. An epoch refers to one iteration through the entire training dataset. This learning method is referred to as *backpropagation* since the gradient is computed by going backward through the network. We denote the network's parameters with Φ. With $\phi_i \in \Phi$ being the ith parameter, every ϕ_i is updated by adding $\Delta\phi_i$ as defined by Equation 2.13 where L is standing for the loss function [Cha18b].

$$\Delta\phi_i = -l\frac{\partial L}{\partial \phi_i} \tag{2.13}$$

The *learning rate* l is a small constant that moderates the size of the update. Hence, it is multiplied with the partial derivative of the loss L with respect to the parameter that is adjusted. In order to move the weight vector in the direction that decreases L, the negative sign is present [Mit97]. Since the gradients for the whole dataset have to be calculated to perform one update, depending on the size of the training data this process can be very slow or exceed the memory. One common variation on gradient descent is called *stochastic gradient descent* (SGD). SGD updates the parameters incrementally every m examples where m is referred to as the *batch size* [Mit97]. The parameters of a model are distinguished between those that are learned during training and those that control the algorithm's behavior. The latter are called the *hyperparameters* and they include settings like the learning rate, the optimization algorithm, the loss function and the batch size [Cha18b].

Convolutional Neural Networks

In the previous section fully connected NNs were explained, which have the property that all linear units in a layer are connected to all the linear units in the next layer. As there is no requirement that NNs have this particular form, other network architectures were developed in order to train better models for specific domains. Convolutional neural networks (CNN) are partially connected NNs and were designed for visual pattern recognition [LBD+89]. The idea was to increase the probability of correct generalization by reducing the number of free parameters in the network. When it comes to pattern recognition in images, what matters is differences between nearby pixel values, not their absolute values. Local light differences describe what is going on in a scene. In computer vision many of the modern approaches focus on this key property of images by extracting *local features* that depend only on small subregions of the image [Bis06]. Since distinctive features of an object can appear at various locations on the input image, a model is considered to generalize well when it is invariant to certain transformation of the inputs. CNNs are neural networks that use a mathematical operation called *convolution* in place

of general matrix multiplication in at least one of their layers [GBC16]. Those layers are called convolutional layers. The learning in a typical CNN is hierarchical [LRM21]. The first layers learn to recognize local features like edges and shapes. The next layers merge semantically similar features into one in order to detect higher-order features like groups of edges. A CNN's final layers can then piece all the features together and yield information about the image as whole. A common CNN architecture is comprised of a series of convolutional layers interleaved with *pooling layers*, followed by one or more fully connected layers [PLV+19]. Figure 2.11 illustrates the architecture of the VGG-19 model, one of the first introduced deep CNNs that achieved state-of-the-art performance for large-scale image classification [SZ15]. The model consists of 19 layers of which 16 are convolutional layers grouped into 5 blocks followed by 3 fully connected layers in the end.

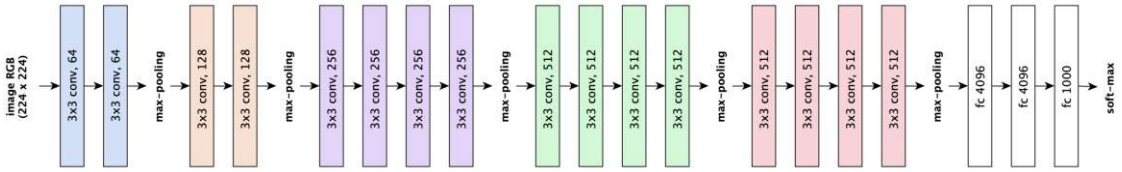


Figure 2.11: VGG-19 model as an example of a deep CNN for image classification.

Convolution is an operation on two functions of a real-valued argument. The units in a convolutional layer convolve their input function with a *kernel function* and output a function referred to as the *feature map* [GBC16]. A (filter) kernel is usually a multidimensional array of weights called filter bank, whereby the weights are learned during training. The size of the kernel is chosen smaller than the input. The kernel is convolved with a patch of an image by calculating the dot product of the kernel and a piece of the image of an equal size [Cha18b]. The result is passed to the corresponding unit in the feature map for that kernel. The kernel is then applied to the next neighbouring piece of the image and that process repeats until all values of the image are convolved. A parameter called *stride* defines how many pixels the kernel moves in every step along one direction. The kernel size and the stride are hyperparameters of the network. During this convolution process the weights of the kernel do not change. Hence, all units in a feature map share the same filter bank [LBH15]. Weight sharing reduces the number of weights that must be learned and leads to detection of features independently of their location in the image. In order to detect multiple features the convolutional layer performs several convolutions in parallel producing multiple feature maps, each from its corresponding kernel. Each feature map has its own set of weight and bias parameters [Bis06].

In the subsequent pooling layer for each feature map there is a plane of units. Each unit takes inputs from a small *receptive field* in the corresponding feature map of the convolutional layer. It applies a pooling function on its inputs, multiplies the result by a trainable weight, adds a trainable bias parameter and passes the result through a non-linear transfer function like ReLU [HKK+10]. The (typically rectangular) receptive fields are chosen to be contiguous and non-overlapping. Due to the use of local receptive fields, a CNN has a smaller number of weights than a fully connected network. Using

pooling is sometimes referred to as subsampling or downsampling as it reduces the size of the inputs for the next layer[Bis06, GBC16]. Therefore, the usage of pooling leads to fewer parameters in the network. It is a form of regularization to improve the robustness of the network. *Max pooling* is a commonly chosen pooling operation that takes the maximum value within a group of inputs [ZC88]. It can be interpreted as a softened version of a logical-OR, indicating whether a feature is present. Summarising the neighboring outputs makes this approach approximately invariant to small translations of the input [GBC16]. CNNs have much fewer connections and parameters compared to a fully connected network due to the presented mechanisms: local receptive fields, weight sharing, and pooling. Thus, compared to standard FNNs of a similar size, CNNs are more computationally efficient and easier to train while they achieve a performance that is only slightly worse [KSH12]. CNNs can be trained by error minimization using a slight modification of the usual backpropagation algorithm in order to take the shared-weight constraints into account [Bis06].

In 2012 deep CNNs have shown that they can outperform the previous state-of-the-art frameworks in image classification and later research gave evidence that they generalize well to other visual recognition tasks [KSH12, DJV+13]. This led to the trend, that lasts until today, to use deep CNNs in ASC frameworks by using spectrograms as input [MHB+18, MHV18a, MHV19, HMV20a].

Residual Neural Networks

Many works on the image recognition task revealed that the depth of a CNN is of crucial importance [SZ15, SWY+15]. By going deeper and using a higher number of stacked layers features of higher levels can be detected and therefore the accuracy of the model increases. This might lead to the assumption that learning better models might be as easy as stacking more layers. In 2016 the work of [HZRS16a] has empirically shown that making a suitably deep model deeper leads to a higher error rate. The problem that the network's training accuracy significantly degrades as its depth increases is called the *degradation* problem, which was also identified in other works [HS14, SGS15]. Considering the methods used to train the network, the authors argue that it is unlikely that the problem is caused by vanishing gradients [HZRS16a]. Furthermore, they conjecture that deep networks may have exponentially low convergence rates, which affect the reducing of the training error. To address the degradation problem by construction, a new network architecture was introduced, the *Residual Neural Network* (ResNet).

The key feature of a ResNet is the usage of *residual blocks*. A residual block consists of a couple of stacked layers and a *shortcut connection*. A shortcut connection is a layer that connects the network's input with its output in a deeper layer by skipping one or more layers [Bis06]. In a residual block the shortcut connection performs identity mapping and its output is added to the outputs of the stacked layers as shown in Figure 2.12.

The idea is that it should be easier to approximate the residual function than the underlying function. Given $H(x)$ as an underlying function of a few stacked layers with

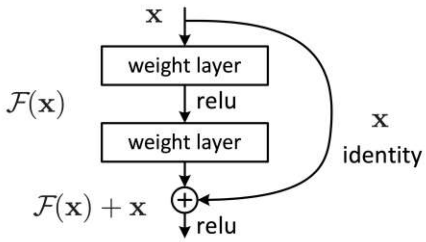


Figure 2.12: Residual learning; a building block [HZRS16a].

x denoting the inputs to the first of these layers, the residual function is then defined as $F(x) := H(x) - x$ [HZRS16a]. Thus, the original function becomes $F(x) + x$. The $F(x)$ acts like a residual, hence the naming. A residual block is defined by Equation 2.14.

$$y = F(x, \{W_i\}) + x \tag{2.14}$$

The output vector y of the stacked layers is the sum of the residual function $F(x, \{W_i\})$ and the input vector x with W_i denoting a set of weight matrices. Given a residual block with two layers as shown in Figure 2.12, the residual function is defined as $F = W_2\sigma(W_1x)$ with σ standing for a nonlinear activation function like ReLu (for simplifying notations the biases are omitted) [HZRS16a]. A shortcut connection calculates $F + x$ by performing element-wise addition. When the considered layers are convolutional, the element-wise addition is performed on two feature maps. The Equation 2.14 shows that the residual function neither introduce extra parameter nor computation complexity. The element-wise addition implies that the dimensions of x and F must be of equal size. As this might not always be the case, to match the dimensions the shortcut connections can perform a linear projection W_s (e.g. zero-padding) as shown in Equation 2.15.

$$y = F(x, \{W_i\}) + W_sx \tag{2.15}$$

Like deep CNNs, ResNets can be trained by SGD with backpropagation. Skipping layers allows to propagate larger gradients to initial layers, which leads to faster convergence at the early stage [SIVA16]. This gives the ResNets the ability to improve the accuracy through increased depth [HZRS16a]. The introduction of ResNets was a breakthrough in deep learning as it allowed to build deeper networks that achieved state-of-the-art performance for several challenging visual recognition tasks [HZRS16b]. In addition, ResNets showed to generalize well, meaning the features can be utilized in transfer learning with good efficiency. It is therefore not surprising that in the last years it became one of the most popular networks in ASC frameworks [MG20, LWL19, HMV20a].

2.3.3 Calibrating Deep Neural Networks

Deep neural networks are very expressive models due to their high depth and large number of parameters that can learn highly non-linear relationships between the input and the output. However, their high expressional capability may lead to a problem known as *overfitting* [Haw04]. Overfitting is the term used when the trained model make predictions with high accuracy for training examples, but does not generalize well such that it performs poorly for unseen data in the test set. On the one hand, this problem can occur when the training dataset is not large enough to learn the intra-class variations. On the other hand, it may be the result of bad calibration which means that the model has learned the peculiarities of the training examples instead of their general characteristics [VPE18]. The process of modifying a model in order to reduce overfitting is generally referred to as *regularization* [Cha18b]. The scenario where a network generates a high error rate on the training set and the test set is called *underfitting* [vRV+10]. Underfitting can be the result when the model is not complex enough or was regularized too much. High capacity models (such as DNNs) tend to overfit the training set [GPSW17]. Therefore, many regularization methods have been developed for reducing the risk of overfitting in the recent years. In the following the regularization techniques used in this work are presented.

Focal loss

The loss function was explained in Section 2.3.2 as a function that measures how bad the network's outcome is during training. This function is also known as the *cost function* and the goal of training a model is to minimize the loss (or the cost) [Cha18b]. To find the right weights to minimize the loss is an optimization problem. A popular loss function for optimization tasks like multi-class classification is *cross-entropy* (CE). The CE method was introduced by Rubinstein as an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks, which involves variance minimization [Rub97]. For multi-class classification tasks CE is defined as shown in Equation 2.16 with c denoting the number of classes and p_j being the estimated probability of the model for class j of a sample. The ground-truth class label (one-hot vector) y_j is the probability distribution that class j is the correct classification for the sample [NPK20a].

$$CE(p, y) = -\sum_{j=1}^{c} y_j log(p_j) \tag{2.16}$$

In other words, CE is the negative log-likelihood between the empirical distribution defined by the training dataset and the probability distribution defined by the network [GBC16]. Many works presented that this loss function can be successfully applied to a wide range of combinatorial and continuous optimisation problems [HBKK05]. It was also found that using CE for a classification problem with CNNs leads to faster training as well as improved generalization [SSP03]. *Focal loss* (FL) is a loss function that extends

CE by adding a modulating factor $(1 - p_j)^\gamma$ with tunable focusing parameter $\gamma \geq 0$ [LGG+17]. The definition of FL is given in the Equation 2.17 [NPK20a].

$$FL(p, y) = -\sum_{j=1}^{c}(1 - p_j)^\gamma y_j log(p_j) \tag{2.17}$$

In the case of a misclassified example, the p_j is small which leads to the modulating factor $(1 - p_j)$ to go near 1 and as the result the loss is unaffected [LGG+17]. On the contrary, for a well-classified example the scaling factor decays to zero which has the affect that the weights are down-weighted. The rate at which easy examples are down-weighted is controlled by the hyper-parameter γ. With $\gamma = 0$ FL is equivalent to CE. To put it simply, during training FL slows down the increase of weights for examples that are easily classified and increases the model's focus on hard, misclassified examples. This approach is simple and addresses highly effective the class imbalance. In practice the *categorical focal loss* is used, which has a balancing parameter α that scales the loss function linearly as presented in Equation 2.18 [LGG+17, SPJL20].

$$FL(p, y) = -\sum_{j=1}^{c}\alpha(1 - p_j)^\gamma y_j log(p_j) \tag{2.18}$$

The authors of FL have shown with experiments in object recognition how the loss of their model was affected by values of $\gamma \in [0, 5]$, which is visualised by Figure 2.13.

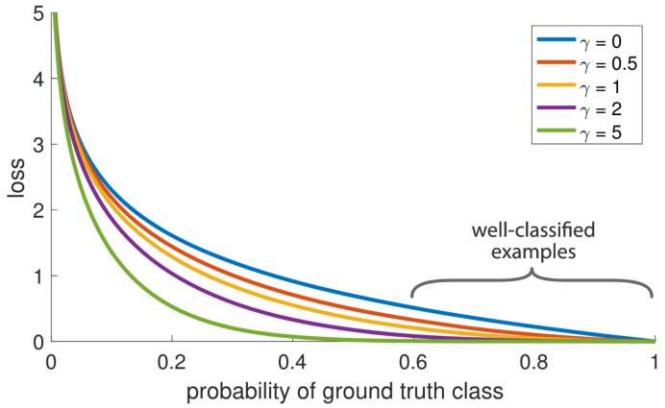


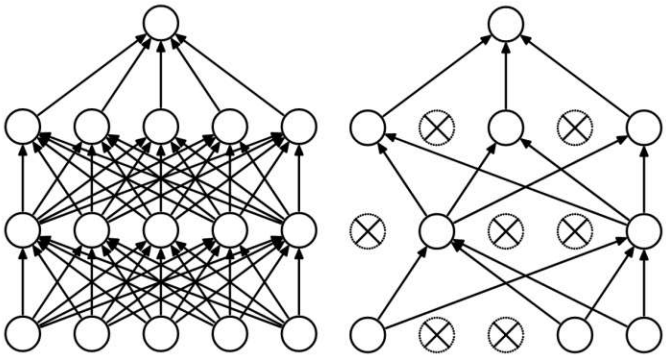
Figure 2.13: An illustration of the loss of a RetinaNet when using FL as the loss function with $\gamma \in [0, 5]$ [LGG+17].

The blue (top) curve in Figure 2.13 is CE, since $\gamma = 0$, and it shows that even well-classified examples with $(p_j > 0.5)$ have a loss with non-trivial magnitude. It can be observed that by dynamically scaling the CE loss, the range in which an example receives low loss is extended. Due to the implicit regularization of the weights, FL can be seen as a form of regularization that reduces overfitting and improves the calibration of DNNs

[MKS+20]. In the ASC domain, several recent works that presented new state-of-the-art systems trained their models with FL [YCT18, GM20, NPK20a, SPJL20].

Dropout

Dropout was introduced in 2014 and the term refers to randomly dropping out units (hidden and visible) from the neural network during training [SHK+14]. In particular, the dropped units are temporarily removed from the network including all their incoming and outgoing connections, as shown in Figure 2.14.



(a) Standard Neural Net (b) After applying dropout

Figure 2.14: An illustration of a neural network with 2 hidden layers (left) and an example of this network after applying dropout (right). The dropped units are shown as crossed [SHK+14].

For any layer, there is a vector of independent Bernoulli random variables each of which has probability p of being 1. The outputs of a layer are multiplied element-wise with that vector. The reduced number of outputs is then passed as input to the next layer. This process is executed at each layer of the network. Dropout can be seen as a technique that adds noise to the states of the hidden units which significantly reduces the gradients. Therefore, a lot of gradients tend to cancel each other. Due to this, when using dropout the previously used learning rate should be increased by 10-100 times [SHK+14]. The probability of retaining a unit is controlled by the dropout hyperparameter p where $p = 1$ means no dropout. Hence, the lower the p the more units will be dropped. A p in the range of 0.5 - 0.8 is a typical value for hidden units, whereas the choice for the input layers depends on the type of input. A $p = 0.8$ is usually chosen for real-valued inputs like image patches or speech frames [SHK+14]. Dropout neural networks can be trained using stochastic gradient descent and backpropagation. Noteworthy is, that during training forward and backpropagation are done only for the units that were not dropped for the training of one batch. The resulting neural network is used without dropout. Dropout is a popular regularization method that reduces overfitting. The recent research shows that this technique could improve the performance of the models in a wide variety of application domains including object classification, digit recognition, speech recognition,

document classification and analysis of computational biology data [SHK+14]. Dropout is also used to train ASC networks [HYX+20, FXM+18].

Batch normalization

One key aspect that makes training of neural networks complicated is that an update of the parameters of a layer affects all the following layer inputs. This effect gets amplified with each layer and is therefore especially problematic for deep networks. Additionally, it makes it particularly hard to train models with saturating nonlinearities. Due to this, the model's parameters have to be initialized very carefully and lower learning rates have to be used, which slows down the training. This problem is known as *internal covariate shift* and the *Batch Normalization* (BN) technique was introduced to address it [IS15].

BN is a regularization approach that is integrated into the network's architecture. The main idea is to reduce the internal covariate shift through the stabilization of the input distributions. This is achieved by introducing additional network layers that normalize the output of the previous layer. The normalization is performed for each mini-batch and the gradients are backpropagated through the normalization parameters [IS15]. The BN transform adds two extra parameters per activation, that are learned during training, to preserve the representation ability of the network. The algorithm of the BN transform is presented in Figure 2.15.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 2.15: The algorithm of the BN transform applied to activation x over a mini-batch. [IS15].

Given a mini-batch $\mathcal{B}$ of size m and an activation function x, each activation value x_i is normalized by subtracting the batch mean $\mu\mathcal{B}$ and dividing it by the batch standard deviation $\sigma^2_{\mathcal{B}}$ with ϵ being a constant that is added for numerical stability. The normalized activations $\hat{x}$ are scaled and shifted before they are passed to next layer. By fixing the means and variances of layer inputs, the amplifying effect of small changes in layer parameters is prevented. Additionally, the gradients become less dependent on the scale

of the parameters. BN allows to be less careful about initialization and to use much higher learning rates, which significantly speed up the training process [STIM19]. Furthermore, it prevents the network from getting stuck in the saturated modes making it possible to train networks with saturating nonlinearities. This approach reduces overfitting and in some cases eliminates the need for Dropout [IS15].

Domain adaptation

Classical machine learning assumes that models are trained and tested using data from the same domain. In practise this assumption does not always hold. The training set can be biased or out-dated and therefore not cover the distribution of the real-world data. In some cases there is a large amount of data from one *source* domain available, but little or no data from the *target* domain the trained model should be applied on. *Domain adaptation* is a sub-field within machine learning that focuses on the research of how to train a model using data from one domain that performs well for data from a different domain [FVRA20]. The main goal is to minimize the discrepancy between the training distribution and the target distribution [VPE18]. Domain adaptation methods are distinguished between supervised and unsupervised learning depending on whether labeled data for the target domain exists [Abe20]. Furthermore, existing domain adaptation approaches can be generally divided into methods with shallow and deep architectures [FVRA20]. Shallow domain adaptation approaches are based on instance-based and feature-based techniques in order to align the domain distributions. Deep domain adaptation approaches utilize neural networks by using convolutional or adversarial based networks to diminish the domain gap.

ASC systems usually face data that were recorded by different devices and under different environmental conditions than the training data. In order to increase the robustness of the classifiers, several recent works proposed different domain adaptation approaches for ASC [VPE18, Abe20]. To name a few, the first introduced method of unsupervised adversarial domain adaptation for ASC used an adversarial training approach to align the distributions of the intermediate feature mappings for both the source and target domain data [GDÇ+18]. In this approach a model was trained on the source domain data, which included the recordings of the major device, and in a later step adapted to the target domain data, which included the recordings of the other devices. The authors found that their method could increase the accuracy of their models for unseen data by approx. 10%. Another work added an auxiliary binary classifier to their network to learn whether the examples were recorded by the major device (as source domain) or by other devices (as target domain) [GM20]. During training the binary loss was backpropagated and used to weight the CE loss resulting in $Total\ Loss = CE\ Loss - 0.1 * Adaptation\ Loss$. However, the resulting network could not show any improvement in its performance. A similar experience was made by other researchers, who explored the use of two different domain adaptation objectives for a ResNet and did not achieve better results [KHEW20]. When analyzing the submissions of the DCASE 2020 challenge it can be concluded, that domain adaptation is not a popular approach as only a few systems used it [HMV20a]. Noteworthy

is, that a very large proportion of the submitted systems used data augmentation as their regularization method to improve the generalization ability of their models [HMV20a]. Data augmentation will be explained in the next section.

2.4 Data Augmentation

The performance of a machine learning model, in particular the ability to generalize, depends on the availability of large quantities of training data. In practice, the amount of available data is limited and varies enormously between the research domains. One of the most known datasets used for visual recognition tasks is the ImageNet dataset with over 14 million images and over 21 thousand object classes [RDS+15]. In the case of audio, the only comparable dataset is the AudioSet with over 2.1 million audio excerpts and 527 sound event classes [GEF+17]. When it comes to ASC, the dataset used in the DCASE 2020 challenge with 23 thousand audio segments and 10 scene classes can be seen as rather sparse [HMV20a]. One way to address the lack of data is to generate additional data by applying one or more deformations to the training samples. This approach is called *data augmentation* and it is particularly easy and effective for classification tasks [GBC16].

A key concept of data augmentation is that the applied transformations do not change the semantic meaning of the data. In object recognition, a rotated, translated, mirrored or scaled image would contain the same object and to train the network on these augmented data can make it invariant to these transformations. While data augmentation consistently leads to improved generalization [MOM12], it is important to note, that the type of transformations should be chosen with respect to the problem. In a task like optical character recognition, it is required to recognize the difference between "b" and "d", so mirroring the image would not be an appropriate way to augment data. Besides from object recognition, data augmentation showed to be an effective technique for speech recognition and many classification tasks including music, environmental sound and acoustic scene [KSGG19, SB17, Abe20]. In the following, an overview of the data augmentation techniques, that were investigated in this work, is given.

2.4.1 Mixup

Not all data augmentation strategies generate physical training data. *Mixup* is a simple domain-agnostic data augmentation technique that constructs virtual training examples [ZCDL18]. A key concept of mixup is that by following the Vicinal Risk Minimization (VRM) principle instead of the Empirical Risk Minimization (ERM) principle the model is encouraged to behave linearly in-between training examples which reduces undesirable behaviors such as memorization and increases the robustness to adversarial examples. Given (x_i, x_j) are two raw input vectors drawn at random from the training data with (y_i, y_j) being their corresponding one-hot label encodings and $\lambda \in [0, 1]$, the virtual feature-target pair $(\tilde{x}, \tilde{y})$ is defined by Equation 2.19 [ZCDL18].

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \tag{2.19a}$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{2.19b}$$

In accordance with its name this technique is mixing up the features and their corresponding labels, whereas the strength of interpolation between feature-label pairs is controlled by the mixup hyperparameter λ. When applied to different datasets the same λ can lead to different changes of the training error [ZCDL18]. Hence, currently there is no rule available for how to find the perfect λ except through experiments. The developers of mixup showed that their approach improves the model's generalization on speech and tabular data. Since mixup introduces minimal computation overhead and can be implemented in a few lines of code, it became a common data augmentation method in ASC frameworks. In the DCASE 2020 challenge 71 of 92 submissions used mixup [HMV20a].

2.4.2 Acoustic Scene Transformations

Additional training data can be generated by modifying the existing data with transformations that do not change the class. In ASC many techniques were explored to create extra recordings that would increase the robustness of the classifier. It is important that for each augmentation the deformation parameters are chosen such that the semantic validity of the label is maintained. In the following, an overview of the transformations used by state-of-the-art submissions of DCASE 2020 challenge is given:

- **Mixing audios**: Two recordings from the same acoustic scene class are randomly mixed [HYX+20].
- **Temporal cropping**: Each of the two samples combined using mixup are randomly cropped into a fixed-length along the time axis [SPJL20, MG20, HYX+20].
- **Pitch shiftig**: The pitch of the audio sample is raised or lowered while the duration is kept unchanged. This method is commonly chosen, but there is no rule available for how the pitch should be shifted. Therefore, the authors use different solutions. Some shift the pitch randomly based on a uniform distribution [HYX+20], in other cases predefined values (in semitones) are used for the shift [AMGL17], while others decided to only lower the shift [XLY+18].
- **Time stretching**: Similar to shifting the pitch, changing the audio speed of a recording to augment new data is a popular method that does not have one solution. New audio samples are generated by slowing down or speeding up an existing audio sample based on predefined values or a uniform distribution while keeping the same pitch. Usually the authors augment data by changing the audio speed in both directions [AMGL17, HYX+20], while in some cases the authors decided to speed

up only [XLY+18]. If the transformed waveform is longer than the original one, it is cut from the end. In case it is shorter, padding is applied till the original length is attained.

- **Adding noise**: To simulate recordings made under suboptimal conditions with low quality microphones, in many works additional data are created by adding noise to the existing audio samples. In come cases random Gaussian noise is used [HYX+20, XLY+18]. Other cases focus on adding some kind of background noise. One group mixed the audio files with environmental background noise using a random signal to noise ratio of a defined range [AMGL17].

- **Spectrum augmentation**: Spectrum augmentation (SpecAugment) was introduced in ASR as an augmentation technique that applies three kinds of deformations on the log mel spectrogram of the input audio [PCZ+19]. The three deformations are time warping, a deformation of the time-series in the time direction, as well as time and frequency masking, where a block of consecutive time steps or mel frequency channels is masked. SpecAugment could improve the performance of ASR networks, what motivated a team to use it for ASC networks [HYX+20].

- **Spectrum correction**: In ASC one of the challenges is that the model has to generalize well to samples recorded using devices with different frequency responses than those it was trained on. To address this problem, *spectrum correction* was introduced as a method to adjust the varying frequency response of the recording devices [Kos19]. Spectrum correction requires a dataset of aligned recordings from the same moment in time and the choice of a reference device. The method consists of two steps: the calculation of the correction coefficients from the spectrum of n aligned pairs of recordings using the reference device and the transformation of the recordings using the computed coefficients. In the later step, the transformation is performed by multiplying STFT of the signal with the correction coefficients on the frequency axis for each point in time. After the correction the spectra for all devices should look alike. To put it simply, spectrum correction aims at transforming a given input spectrum to that of a reference device, so that it appears as if it was recorded using the reference device. This method can be applied both in the frequency domain and in the time domain [Koś20].

  While some works [Kos19, NPK20b] used spectrum correction for device adaptation during training by applying the correction to the input before generating the spectrograms, one team employed spectrum correction as a data augmentation technique to obtain extra data [HYX+20]. The procedure was modified as follows: First, a reference device spectrum is calculated by averaging the spectrum from all training devices except that from the reference device. Then, the spectrum of all recordings collected with the reference device is corrected to generate extra recordings.

- **Reverberation + Dynamic Range Compression (DRC)**: The authors of the DCASE 2020 challenge provided datasets which included the recordings of simulated

devices [HMV20c, HMV20b]. To simulate a device, they first created a dataset of impulse responses (IR) measured for multiple angles using mobile devices other than the reference device [HMV20a]. Then, for every simulated device, recordings from the reference device were processed through convolution with the randomly selected device-specific IR, followed by a DRC with device-specific parameters. Some participants used the same procedure to generate more data [HYX+20].

Enlarging the dataset with transformed samples was found to be an effective method to improve the classification accuracy of the model [XLY+18, HYX+20, AMGL17, Abe20].

Task 1a of the DCASE 2020 challenge focused on ASC of audio signals recorded with multiple devices [HMV20a]. Therefore, reducing the device dependency of the models was especially important. Analyzing the results of the challenge it can be observed, that all systems in the top 10 were trained with mixup and at least one of the presented transformations as data augmentation methods[5].

2.5 Related Work

State-of-the-art ASC algorithms almost exclusively use deep CNNs [Abe20]. Especially the approach with choosing Mel energies as feature representation and using deep learning architectures based on e.g. ResNet and dilated convolution was very common between the submissions of the DCASE 2020 challenge (Task 1a) [HMV20a]. Nonetheless, other ASC frameworks were researched and can be summarized in those that use either deep feed forward networks or recurrent neural networks [Abe20]. In this section some of these works are presented.

Recurrent Neural Networks

Neural networks that have recurrent connections between hidden units are called *recurrent neural networks* (RNNs) [GBC16]. Connections from a neuron to either itself or a neuron in the previous layer are called *recurrent*. Therefore, in graph terminology an RNN is a directed acyclic graph. While DNNs and CNNs are not capable of modeling sequences, RNNs are powerful sequence learners [Gra12]. As with other neural networks, many varieties of RNN exist. An example of an RNN with recurrent connections between hidden units that produces an output at each time step is shown in Figure 2.16.

The illustrated network parameterizes its connections as following: from input to hidden layer with weight matrix U, recurrent hidden-to-hidden connections with weight matrix W and from hidden to output layer with weight matrix V. The network maps an input sequence of x values to a corresponding sequence of output values o. The hidden units calculate the outputs based on the input and the previous output. Hence, the network

[5]Table *General characteristics* of DCASE 2020 challenge results for Task 1A `https://dcase.community/challenge2020/task-acoustic-scene-classification-results-a` (visited on 08.05.2022)

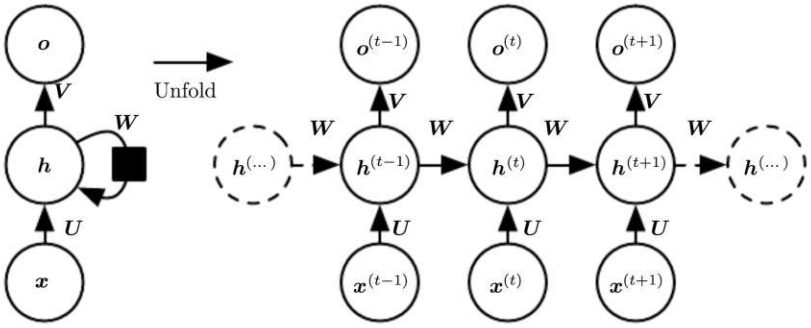


Figure 2.16: An example of an RNN with hidden-to-hidden recurrent connections. *(Left)* The network as a computational graph. *(Right)* The same network as a time-unfolded graph showing the time instance for each node [GBC16].

is recurrent because the state at time t depends on the state at time $t-1$. Providing feedback with delay, introduces memory to the network [MC01]. This allows an RNN in principle to map from the entire history of previous inputs to each output. The feedback within an RNN can be distinguished between local and global. The local feedback is produced within the hidden layer, whereas the global feedback is produced by the connection from the output layer to the input layer [MC01]. Since observations can be encoded and accumulated over temporal or spatial dimensions, RNNs can learn features and long term dependencies from sequential and time-series data [SSB+18]. Therefore, they are widely used for sequential problems like text analysis, speech recognition and video analysis [SSB+18]. Since the audio signal can be represented as a sequence of either frames of raw audio or human engineered feature vectors (e.g. log-mel), matrices (e.g. spectrograms), or tensors (e.g. stacked spectrograms), it can be analyzed by a recurrent network. Several ASC frameworks based on different types of RNNs have been proposed in the last years. Some ASC algorithms are based on convolutional recurrent neural networks (CRNN) [LHX+19]. In this case, local information is extracted by the convolutional layers and then combined by the recurrent layers over a longer temporal context [PLV+19]. Others use RNN variations that address the vanishing/exploding gradient problem like Gated Recurrent Unit (GRU) or Long Short Term Memory (LSTM) [PKK+17, DLC+20]. To mitigate the information flow and alleviate gradient problems LSTM utilizes a gating mechanism and memory cells [HS97]. Furthermore, LSTMs have been extended to model audio signals across both time and frequency domains [PLV+19]. GRU is a variant of the LSTM unit, but is simpler to compute and implement as it only has two gating units instead of four [CvG+14]. Even though GRU networks were proposed more recently, several works have shown that they can outperform LSTM networks in many sequence modeling tasks [CGCB14, JZS15].

In the 2017 paper *Audio Scene Classification with Deep Recurrent Neural Networks* by Phan et al. [PKK+17] the authors treat the ASC task as a sequential modeling problem and propose an approach where deep RNNs are trained on audio data presented as

sequences of label tree embedding (LTE) features. The motivation of their work to introduce an RNN framework for ASC was that RNNs have been shown to be inferior to CNN competitors in the past [MTX+16, VW16]. The goal was to find an efficient way of transforming audio scenes into sequences. Audio scenes typically consist of background noise mixed with rich foreground sounds. Since foreground sounds occur in an arbitrary order, it is hard to uncover the hidden sequential patterns. In the proposed approach, an audio scene instance is transformed into a sequence of different LTE feature vectors which is then divided into smaller subsequences to enhance the repeatability of patterns. A deep GRU-based RNN is trained and evaluated on the extracted subsequences. The predicted label for an entire sequence is obtained by aggregating classification outputs of its constituent subsequences. The low-level features used in this work were Gammatone cepstral coefficients, MFCCs and log-frequency filter bank coefficients. The authors investigated the influence of different related factors that affects the network's performance and used the LITIS Rouen dataset [RG15] for training. The best performance was achieved by *RNN-Fusion*, a network that concatenated the feature vectors of three deep RNNs, each trained on one of the three LTE channels (i.e. *RNN-Gam*, *RNN-MFCC*, and *RNN-Log*) at every time step and used a multiplicative probabilistic voting scheme as aggregation method. The final classification step was calibrated with a linear support vector machine (SVM) which showed a significant performance improvement in comparison to the softmax layer. Cross-entropy was used as loss function and dropout was applied on the output of the last layer for further regularization. With *RNN-Fusion* the authors achieved the state-of-the-art performance on the LITIS Rouen dataset and outperformed the best previously reported result (i.e. CNN-Fusion6 [PHM+17]) by 1.4%, 1.2%, and 1.2% absolute on precision, F1-score, and accuracy. Noteworthy is that the performance of the *RNN-Gam* alone achieved a better performance than that obtained with the best CNN with multiple LTE types reported in [PKH+17]. Therefore the authors conclude that with an appropriate sequential modeling a standalone RNN can outperform state-of-the-art CNNs in the same ASC benchmark.

In 2020 Dong et al. published the paper *At the Speed of Sound: Efficient Audio Scene Classification* in which the authors formulate ASC as a retrieval problem [DLC+20]. The goal of this work was to propose a retrieval-based scene classification architecture that can predict an audio scene with high accuracy after listening in for at most 300ms. Their approach combines bidirectional LSTM [SVL14] and attention mechanism [VSP+17] to learn high-level features (or embeddings) for short audio segments. In particular, the audio scenes are split into segments shorter than a second and the similarity and importance embeddings are learned for each segment. MFCCs, their first derivatives and 12 harmonic and percussive features were extracted as low-level features. While LSTM captures long-term temporal dependencies, attention is used to emphasize the important time steps in a segment. A test segment is classified in the same class as its nearest neighbor in embedding space. To reflect the class membership of embeddings, a similarity or distance function, such as the Euclidean distance, is used. The presented network was trained by optimizing a custom *audio loss function* that captures the importance of each segment in a scene and of each time step in a segment. The authors evaluated their

architecture with experiments using three datasets extracted from the DCASE 2016-2018 challenges [MHV16]. Each dataset was randomly split into training and validation set by 80% to 20%. The experiments showed that the model can predict the category (out of 15 possibilities) with high accuracy (around 70%) by listening to the first 300 ms of a 10 s audio scene. Additionally, the authors pointed out that processing the first few segments rather than the entire scene led to a decrease of the detection accuracy by only 7%. Furthermore, they demonstrated that a higher classification accuracy (over 95%) can be achieved for datasets with a smaller number of possible categories. The authors came to the conclusion that the category of a scene can be reliably detected after less than one second.

Feedforward Neural Networks

In ASC combining deep FNNs with generative learning methods or feature learning techniques is another common approach besides those using CNNs or RNNs that have proven to be viable alternatives [VPE18, Abe20]. Some works successfully proposed the use of unsupervised feature learning techniques such as non-negative matrix factorization (NMF) and it's variants [BLD12, BSER16a]. Others adapt approaches originally designed for speech recognition by combining a DNN and a Gaussian mixture model (GMM) [TYMO16]. Apart from this, there have been proposals to focus on DNNs only and to combine several DNNs into one system by using hierarchical learning [XHWP16] or late DNN score fusion [PG18]. In the following two approaches that extend DNNs with other learning techniques are presented.

In the 2017 paper *Nonnegative Feature Learning Methods for Acoustic Scene Classification* by Bisot et al. [BSER17] the authors propose an ASC framework that is a simple fusion of two systems trained on both channels of the scene stereo recordings. The systems used for the fusion are a DNN and a task-driven non-negative matrix factorization (TNMF) model. The DNN network (referred to as DNN-M) is composed of two branches that are merged by concatenation before the final layer of the network. This architecture allows the network to be jointly trained on NMF features and Constant-Q transforms (CQT) as time-frequency representations. All layers are simple fully-connected layers with ReLu activations and dropout is applied between each layer. Non-negative matrix factorization (NMF) [LS99] is an unsupervised data decomposition technique that has been found to be an effective feature learning approach in ASC [BSER16a, BSER16b]. NMF decomposes non-negative data into non-negative dictionary elements. Noteworthy is that most of the time-frequency representations for audio signals contain only non-negative coefficients. The TNMF system (referred to as TNMF-AS) used in the system fusion is a supervised variant of NMF which applies an adaptive scaling strategy to balance the contribution of each dictionary component when jointly learning the dictionary and classifier. For the dictionary Constant-Q transforms are extracted from the audio samples and averaged by slices of a fixed length. Then, for each example all the averaged slices are concatenated to build the data matrix. A square root compression is applied to the data and each feature dimension is scaled to unit variance. In deep learning terminology the TNMF model

can be seen as a one hidden layer network taking the data matrix as input, followed by the classification layer which is a fully connected layer with softmax activations acting as multinomial regression. The parameters of the network are trained to minimize a categorical cross-entropy loss. The authors use the DCASE 2017 challenge dataset for ASC [MHD+17] to evaluate their proposed framework and to investigate it's performance and that of it's subsystems. Their experiments showed that a DNN trained on the NMF largely outperforms those trained on CQTs or unsupervised NMF. By merging the branches of a DNN-NMF and a DNN-CQT to one DNN-M architecture the performance could be slightly increased. Nonetheless, it was shown that the proposed TNMF-AS provided better results than a DNN-M. Furthermore, the authors could show that TNMF-AS outperforms TNMF that does not use the adaptive scaling strategy. They therefore conclude that adding a scaling strategy can improve the model generalization capacity. Compared to the baseline system provided in the 2017 DCASE challenge with an accuracy of 77.2%, the proposed final system (TNMF-AS+DNN-M) could reach a significantly higher accuracy with 91.1%.

Takahashi et al. described their contribution to the ASC task of the DCASE 2016 challenge in the paper *Acoustic scene classification using deep neural network and frame-concatenated acoustic feature* where they propose to combine a DNN with multiple Gaussian mixture model (GMM) classifiers to model the individual acoustic scenes [TYMO16]. The introduced DNN-GMM is trained on high-dimensional acoustic features. To form these features, MFCCs and their first and second derivatives are computed for each frame for each of the left and right channels. Then, in each frame and channel the extracted features are concatenated with the features of a fixed number of other frames selected by a defined interval. Finally, the acoustic features of the left and right channels in each frame are concatenated to capture the spatial information. The training of the DNN-GMM consists of several steps. First, a model for each scene class is learned in a supervised manner using a conventional GMM. Next, training data for the DNN is composed of the features in each frame of each training data and the state number of the GMM. Then, the initial parameters of the DNN are determined by unsupervised pre-training [Hin12]. Finally, the DNN is trained based on generated training data with backpropagation and SGD. The authors evaluated several variants of the proposed framework using the development dataset and evaluation dataset provided by the DCASE 2016 challenge. The highest classification accuracy with 77.5% was achieved by the DNN-GMM with three hidden layers which concatenated 5 frames at 100 ms interval. The DNN-GMM could achieve a better performance by 8.4% on the evaluation dataset compared with that of a GMM alone, which was used as the baseline classifier in the DCASE 2016 challenge. The experimental results showed on the one hand that frame concatenation can boost the performance and might be further increased by using a higher number of concatenated frames. On the other hand, the frame concatenation interval is an important parameter worth researching. In 2017 the authors published a follow-up work where they compared the performance when using different numbers of concatenated frames and frame concatenation intervals [TYOM17]. Based on their experimental results they found that best results could be achieved when concatenating 5

frames or less using a concatenation interval for 500 ms or less. The highest classification accuracy was obtained by the system that concatenated 3 frames at an interval of 500 ms outperforming the previous method by 1.3%.

In this section we gave a broad overview of the current approaches for ASC that are not based on CNNs. Even though the presented works could achieve good performances while trying to solve a similar ASC problem as discussed in this thesis, the results of the DCASE 2020 challenge show that the top ranked positions are dominated by convolutional frameworks as they achieve significantly higher classification accuracies and better generalization [HMV20a]. In this work, we analyzed the top ranked systems of the DCASE 2020 challenge and tried to adopt the good aspects in the design and training of our classifier. The results of our analysis as well as the details of our approach of developing an ASC framework, will be discussed in the next chapter.

CHAPTER 3

Development of the Prototype

In this chapter we discuss the planning, the design and the implementation of the software prototype that was used in the case study that is discussed in Chapter 4. As mentioned in the first chapter, the aim for this work is to provide a cross platform mobile application that can classify the environment based on the surrounding sounds and play according to the classification result matching sound samples of the same scene of a different city. The development of the required application consisted of several steps, which are explained in separate sections. First, Section 3.1 introduces the idea and the requirements of the software. Then, Section 3.2 gives an overview of the researched state-of-the-art ASC systems and explains the choice of the baseline model used in the design of the prototype's classifier. In Section 3.3 the modification of the baseline model as well as its training and calibration are presented. The selection of the sound samples to play in response to the classification result is discussed in Section 3.4. Due to current technological limitations, the trained model can not be included in a mobile application. Hence, it was integrated in a server application that is described in Section 3.5. Finally, the details of the mobile application, which communicated with the server to classify recordings and was used by the case study participants, are laid out in Section 3.6.

3.1 Design

The main goal of this thesis is to provide a prototype that uses ASC to find a matching sound sample that was recorded in the same scene, but in a different city. The found sound sample is then played to the user to evaluate the psychological effect the overlay of a different soundscape might have on them. The development of the required mobile application consists of several parts, the requirements of which are defined as follows:

- **Classifier:** An ASC model that identifies 9 acoustic scenes from a 10 seconds recording has to be designed and implemented. Since during the case study it

will be used in real life scenarios, the overall goal is to develop a classifier that achieves sufficient performance when applied to unseen data. Therefore, the current state-of-the-art ASC solutions has to be researched to adopt well performing aspects in the architecture and training of the model.

- **Soundscape overlay:** The scope for the prototype is defined such that the user can choose a soundscape from one of four cities. This means that a set of sound samples for 9 scenes for each city has to be created to use for the soundscape overlay. The sound samples have to include city specific sounds and to be long enough for the user to be able to adapt to the new soundscape.
- **Server architecture:** The classification is executed on a server. Hence, an API has to be provided to get predictions from the model. The server application has to support the upload of a 10 seconds audio file, which is then passed to the classifier. The predicted scene or "unknown" (for predictions with low accuracy) has to be send as response to the client.
- **Mobile application:** A cross platform mobile application has to be designed and implemented with the goal to be used by the participants of the case study discussed in Chapter 4. The mobile application has to include the following features:
  - A simple UI with the possibility to choose a city, and to start and stop the classification process.
  - The recording of the surrounding sounds using the device's microphone at an interval of 5 minutes.
  - The classification of the recorded audio files through communication with the server.
  - An audio player that plays an audio sample of the chosen city accordingly to the classification result.
  - The process of the application can only be stopped by the user manually and continues to run when it is put in the background.

The listed requirements were mandatory, but no further limitations with regard to allowed technologies were defined. The rest of the chapter describes each part in more details, starting with the analysis of the current state-of-the-art ASC systems.

3.2 Overview of ASC Frameworks

One of the contributions of this thesis is the research of the effect the overlay of a different soundscape has on the user's perception. In particular, we are interested in the effect the user might experience, when they hear the soundscape of the scene in which they are in, but from a distant city. How does it feel to walk on a public square in Vienna and experience the soundscape of a public square in Moscow? Therefore, hearing the

soundscape of a different scene might cause a different effect and lead to false conclusions in the case study. Accordingly, the performance of the prototype's classifier plays an important role in the success of this work. This section gives an overview of the researched state-of-the-art ASC systems and explains the choice of the ASC framework that is used as the baseline model in Section 3.3.

A rapid increase of scientific publications in the research field of ASC has been observed in the last decade. This progress was mainly stimulated by recent advances in the field of deep learning as well as the release of various public datasets. Since 2013, the DCASE community plays a major role in this development by organizing annual competitions on several machine listening tasks beyond the domains of speech and music [SGB+15]. This challenge focuses mainly on tasks for ASC and sound event detection, and aims to benchmark the state of the art. In order to find the baseline model for our classifier, we decided to investigate the submissions of one of the recent DCASE challenges that included a task for a similar problem as discussed in this work. During our search we found that Task 1a of the DCASE 2020 challenge addressed a comparable problem. The main focus of the task was the device mismatch problem of ASC that is common for the real-world scenarios where data has to be classified recorded by various devices, many of which are not available at training stage. The objective of this task was to develop systems with very good generalisation properties across a large number of different devices [HMV20a]. For the training of the models and the evaluation of the submissions a new dataset (TAU Urban Acoustic Scenes 2020 Mobile) consisting of recordings from multiple devices was provided [HMV20b, HMV20c]. For this dataset audio data was recorded in multiple European cities at 10 different acoustic scenes. More details about this dataset will be presented in Section 3.3.2. As it can be assumed that the participants of the case study will use the prototype on different devices in an European city, we concluded that using one of the submissions for Task 1a as the baseline model for our ASC framework will deliver the classification accuracies needed for our task. Our approach to find the baseline model consisted of the following steps:

1. Investigate the top 10 ranked systems submitted for Task 1a
2. Compare the used technologies and the achieved performances
3. Choose one of the systems as the baseline model based on the insights gained from the previous steps

Following the first step, Table 3.1 gives an overview of the top 10 ranked systems of the 92 total submissions. The listed systems were developed by the teams Suh et al. [SPJL20], Hu et al. [HYX+20], Gao et al. [GM20] and Jie et al. [Jie20]. The submissions were ranked by accuracy calculated as macro-average accuracy (average of the class-wise accuracies) on the evaluation dataset. The values in the scopes show the accuracies of the model at the 95% confidence level. Multi-class cross-entropy (log-loss) was used to evaluate the uncertainty of the classifier. The complexity of each model is described by its number of learnable parameters (in millions).

#	Submission	Accuracy (95%)	Log-loss	Complexity	Classifier
1	Suh_ETRI_3	76.5% (75.8 - 77.3)	1.21	39M	ResNet ensemble
1	Suh_ETRI_4	76.5% (75.7 - 77.2)	1.21	39M	ResNet ensemble
2	Hu_GT_3	76.2% (75.4 - 77.0)	0.89	130M	CNN ResNet ensemble
3	Hu_GT_2	75.9% (75.1 - 76.7)	0.89	67M	CNN ResNet ensemble
4	Hu_GT_4	75.8% (75.0 - 76.5)	0.90	91M	CNN ResNet ensemble
5	Hu_GT_1	75.7% (74.9 - 76.4)	0.92	62M	CNN ResNet ensemble
6	Suh_ETRI_2	75.5% (74.7 - 76.2)	1.22	13M	ResNet
7	Gao_UNISA_4	75.2% (74.4 - 76.0)	1.23	12M	ResNet ensemble
8	Gao_UNISA_1	75.0% (74.3 - 75.8)	1.22	4M	ResNet
8	Jie_Maxvision_1	75.0% (74.3 - 75.8)	1.20	3M	ResNet

Table 3.1: Official systems ranking of the DCASE 2020 challenge showing the top 10 systems submitted for Task 1a[1].

The last column states the neural network used as the classifier and it can be observed that all systems used a ResNet, either as a single network or as an ensemble. Table 3.1 gives the first insights of the main characteristics of the systems, which can be summarized as follows:

- According to the log-loss results, the system with the highest accuracy by Suh et al. is more uncertain of its decisions than the systems by Hu et al. with slightly lower accuracies.

[1]Table *Systems ranking* of DCASE 2020 challenge results for Task 1A `http://dcase.community/challenge2020/task-acoustic-scene-classification-results-a#systems-ranking` (visited on 03.07.2022). The data for *Classifier* were updated by us according to the reports.

- With an accuracy range of $76.5\% - 75.0\%$ all systems outperformed the baseline system provided by the challenge which had an accuracy of 51.4% [HMV20a].
- A classification accuracy of $\geq 75.7\%$ could only be achieved by ensembles.
- Training an ensemble of classifiers involves the training of several networks which is reflected in the number of learnable parameters meaning there is a trade-off between model's accuracy and its complexity.
- A more complex model does not necessarily perform better. The submission with the highest accuracy by Hu et al. has a slightly worse performance than the top system by Suh et al. while having 4 times more parameters. Noteworthy is, that the smallest model by Jie et al. has 10% of the parameters compared to the top ranked system while achieving an accuracy that is only 1.5% less.

The comparison of the measured characteristics shows that all systems achieved similar performances, whereby ensembles performed slightly better in terms of accuracy for the price of a large number of learnable parameters. To get a better insight of the used technologies, Table 3.2 gives an overview of the features and data augmentation techniques used by the systems grouped by teams[2]

All solutions used log-mel energies as audio features with deltas and delta-deltas of the spectrum following the same approach. The audio waveforms were analyzed with a window size of 2048 samples and a hop-length of 1024 samples [SPJL20, HYX+20, GM20, Jie20]. A spectrum of 431 frames was yielded from the 10 seconds audio files, and each spectrum was compressed into 128 or 256 Mel frequency bins. Deltas and delta-deltas without padding were calculated from the spectrograms, which reduced the number of time samples to 423, and stacked into the channel axis. Hu et al. have scaled each feature value into $[0, 1]$ before feeding the input to the network. In order to improve the model's generalization and therefore to compensate for the device mismatch, all submissions used mixup and cropping for data augmentation. Hu et al. was the only group that investigated the usage of additional data augmentation strategies. Their experiments showed that the combination of several data augmentation techniques can increase the classification accuracy and minimize the loss [HYX+20]. As shown in Table 3.1, the submissions by Hu et al. achieved loss results of $0.89 - 0.92$, which are much smaller compared to the other submissions with a loss of $1.20 - 1.23$. It can be therefore concluded that the usage of multiple data augmentation techniques can increase the classifier's certainty of its decisions while accomplishing similar accuracies.

Without investigating the network architectures used in the presented submissions any further, we can conclude that all solutions followed the same or very similar approach

[2]Table *General characteristics* of DCASE 2020 challenge results for Task 1A `https://dcase.community/challenge2020/task-acoustic-scene-classification-results-a/#system-characteristics` (visited on 03.07.2022). The data for *Features* and *Data augmentation* were updated by us according to the reports.

#	Submission	Features	Input tensor	Data augmentation
1, 6	Suh_ETRI	log-mel energies deltas delta-deltas	[256 × 423 × 3]	temporal cropping mixup
2-5	Hu_GT	log-mel energies deltas delta-deltas	[128 x 423 × 3]	mixup random cropping SpecAugment spectrum correction reverberation-drc pitch shift speed change random noise mix audios
7-8	Gao_UNISA	log-mel energies deltas delta-deltas	[128 x 423 × 3]	temporal cropping mixup
8	Jie_Maxvision	log-mel energies deltas delta-deltas	[128 x 423 × 3]	temporal cropping mixup

Table 3.2: Overview of the features and data augmentation techniques used by the systems listed in Table 3.1 grouped by teams[2].

in regards of feature extraction, with data augmentation being the main tool used for the generalization problem. The highest accuracies could only be achieved by ensembles of classifiers, whereby the gain in performance with 1.5% compared to the best single network solution was rather small. Each of the submitted classifiers was trained with backpropagation and SGD for 126 [Jie20], 254 [SPJL20, HYX+20] or 310 [GM20] epochs. Considering the large amount of data used for training, increasing the model's complexity leads to longer training periods and requires more computational resources. Accordingly, it can be assumed that the training of multiple networks will exceed the resources of this work without a significant pay-off. We therefore decided to focus on the single network solutions, in particular on Gao_UNISA_1 and Jie_Maxvision_1 due to their low complexity.

In their previous submission for the DCASE 2019 challenge, Gao et al. introduced a residual network with two pathways: one for high frequencies and one for low frequencies, that were fused two convolutional layers prior to the network output [MG20]. The results of the challenge showed that this network could achieve excellent generalisation to unknown devices in the evaluation set [MHV19]. Their code for training models and running trained models was made available online and got awarded the "Reproducible

System Award" for DCASE 2019 Task 1 [MG20]. Noteworthy is, that the submission Gao_UNISA_1 is the model from the previous challenge without modifications and that Jie_Maxvision_1 used that model as their baseline. Jie et al. investigated different attention mechanisms in order to enhance the model's generalization. However, as it can be observed from the challenge results in Table 3.1, adding attention mechanism had no effect on the classification accuracy. Since Gao et al. have published[3] their code for training the models as part of the DCASE 2020 challenge, we decided to use Gao_UNISA_1 as the baseline model for our ASC framework. The technical details of the model as well as our modifications and training will be explained in the next section.

3.3 Implemented ASC Framework

In this section, we give a detailed overview over the architecture of the implemented classifier as well as its training and evaluation. First, we present the design of the network we used as the baseline model and how we adjusted it to our task. Afterwards, the training setup including the datasets used for training and evaluation are discussed. Since the network's generalization ability will play an important role during the case study, we investigated several approaches based on our observations in Section 3.2 by conducting experiments. Our experimental results and insights are laid out at the end of this section.

3.3.1 Baseline Model

The baseline model we used is the ResNet that was introduced by Gao et al. in 2019 [MG20]. In their paper the authors argument that spectrograms have different characteristics than images. While an object in an image can carry the same meaning regardless of its location, patterns of features may represent different physical origins at different frequencies. Consequently the two spatial axes in an image are not of the same nature as the two axes of a spectrogram. In order to take into account that the temporal and frequency axes in spectrograms represent fundamentally different information, the ResNet was designed with two pathways: one for high frequencies and one for low frequencies. The main motivation behind the approach of using two branches is the idea that by not applying the convolutional kernels at all frequencies in a spectrogram different frequency features can be learned and the overall learning improved.

Figure 3.1 illustrates the architecture of the model, whereby only the convolutional layers are depicted for simplification. The network is designed as a pre-activation ResNet, where the input to each convolutional layer is processed by a batch normalization layer and then a ReLU activation [HZRS16b, McD18]. The overall network input has 128 frequency dimensions, which are split in the network before being fed to the branches. This means that dimensions 0 to 63 are processed by one path and dimensions 64 to 127 by another. Each path is a residual network that consists of a convolutional layer followed by 4 stacks, where each stack is built of two residual blocks containing two convolutional layers. Each

[3] https://github.com/emilywg/DCASE2020-Task1 (visited on 03.07.2022)

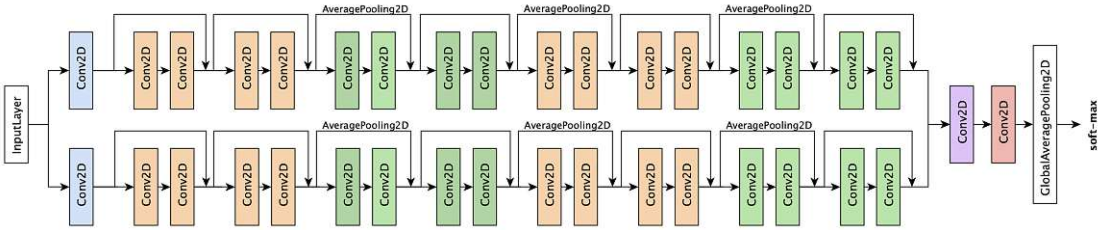


Figure 3.1: ResNet with two pathways used as baseline model.

branch has therefore 17 convolutional layers and all kernels are 3×3. The pathways are combined using late fusion by concatenation of frequency axes to form 128 frequency dimensions. The concatenated output is then processed by the final two 1×1 layers, which act as a two layer non-convolutional neural network that weights the contributions of each channel in each branch for classification of the scene at each frequency. The result is then passed to a batch normalization layer, a global average pooling layer, and a soft-max layer. The network's input is processed by 19 convolutional layers in total.

Since an important cue could happen with equal likelihood at any point in time, in order to learn global temporal information across all time samples, the authors followed a standard image classification practise by regularly downsampling in time using stride-2 convolutional layers. Downsampling is only done for the temporal axis, therefore the number of frequency dimensions in the feature maps for each path remains constant at 64 throughout the network. When the two branches are concatenated, each path has processed a frequency-axis receptive field of size $1 + 2 \times 17 = 35$ bins (from 17 convolutional layers each with kernel size of 3 bins). Consequently, the global average pooling layer averages over different feature maps in the frequency axis, while weighting them equally in the temporal axis.

In order to avoid the need for pre-normalization of input features and to reduce overfitting, the authors followed the idea of [McD18] and used a batch normalization layer with learned offset and scale parameters as the first layer of the network. Another regularization technique was using weight decay 5×10^{-4} on all convolutional layers. Additionally, mixup and temporal crop augmentation were applied. Based on our observation in Section 3.2 that multiple data augmentation strategies can improve the classifier's performance, we decided to try out some of the data augmentation techniques that were used by Hu et al. [HYX+20]. The techniques we used are explained in Section 3.3.2.

For the DCASE 2020 challenge Gao et al. investigated the use of focal loss and mild domain adaptation in order to improve the generalization of the model [GM20]. Their experimental results showed that the use of focal loss led to a minor increase of 0.1% in accuracy, whereby domain adaptation led to a decrease by 0.3%. Interestingly, in the DCASE 2020 challenge the system using domain adaptation performed slightly better than the one using the focal loss. We therefore decided to investigate the effect of these two approaches during the calibration phase, which will be discussed in Section 3.3.3. The

code for creating the network and running its training using Keras [Cho15] was published by the authors under the MIT License[3]. We used the code in our implementation without making any changes in the network's architecture besides from adopting the number of targets to 9. Our contribution was to train the model with a different dataset and to examine the effect the usage of focal loss, domain adaptation and data augmentation techniques has on the model's accuracy and generalization. Same as Gao et al. we used Tensorflow [AAB+15] with the Keras API to implement and train our neural network.

3.3.2 Model Training

Our classifier is a deep residual network that is trained with log-mel spectrograms complemented by log-mel deltas and delta-deltas. Following the requirements stated in Section 3.1, in order to train our model, we need a large number of audio files that were recorded in 9 different scenes and are at least 10 seconds long. Since Task 1a of the DCASE 2020 challenge is similar to our problem, we decided to work with the TAU Urban Acoustic Scenes 2020 Mobile dataset that was provided for that task. The dataset contains recordings from 12 European cities in 10 different acoustic scenes. Audio data consists of 10-seconds audio segments provided in single channel 44.1 kHz 24-bit format. Four devices were used to record simultaneously (A, B, C, and D), and additional synthetic devices S1-S11 were simulated by using audio recorded with device A. Simulated recordings were created using a dataset of impulse responses (IR) [HMV20a]. The dataset was split into development and evaluation sets. We used only the development set for training as the reference labels were not provided for the evaluation set.

The development dataset was provided with a training/test split in which 70% of the data for each device is included for training and 30% for testing whereby some devices appear only in one of the sets. In order to allow a comparison of systems on the development set the participants were required to report the performance of their system using the provided train/test setup. The distribution of devices among training and test subsets is shown in Table 3.3.

Device		**Dataset**		**Cross-validation setup**		
Name	**Type**	**Total length**	**Total seg**	**Train seg**	**Test seg**	**Unused seg**
A	Real	40h	14400	10215	330	3855
B,C	Real	3h*2	1080*2	750*2	330*2	
S1,S2,S3	Sim.	3h*3	1080*3	750*3	330*3	
S4,S5,S6	Sim.	3h*3	1080*3		330*3	750*3
Total		**64h**	**23040**	**13965**	**2970**	**6105**

Table 3.3: Distribution of devices among training and test subsets in the TAU Urban Acoustic Scenes 2020 Mobile dataset [HMV20a]

.

As it can be seen in Table 3.3 the training set consists mostly of recordings made with device A, while the majority of the data in the test set was recorded with other devices. Noteworthy is, that devices S4-S6 appear only in the test set. To get a better understanding of how much data was provided for each scene, Table 3.4 shows the distribution of audio segments per device and scene class.

	Train / Device		Test / Device		
Scene class	**A**	**B,C,S1-S3**	**A**	**B,C,S1-S3**	**S4-S6**
Airport	1019	75	33	33	33
Bus	1025	75	33	33	33
Metro	1007	75	33	33	33
Metro station	1005	75	33	33	33
Park	1054	75	33	33	33
Public square	1053	75	33	33	33
Shopping mall	999	75	33	33	33
Street, pedestrian	1011	75	33	33	33
Street, traffic	1038	75	33	33	33
Tram	1004	75	33	33	33
Total	**10215**	**750**	**330**	**1650**	**990**

Table 3.4: Distribution of audio segments per device and scene in the train / test setup provided by the TAU Urban Acoustic Scenes 2020 Mobile dataset [HMV20b]

We decided to use the same train/test setup to be able to compare the performance of our classifier with the reported performance of the baseline model. In order to fulfill our requirement of supporting 9 different scenes, we decided to remove one of the scenes from the TAU development dataset. We therefore investigated the class-wise performances of the baseline model and the top ranked DCASE 2020 submissions. First, we analyzed the performance of the baseline model as stated in the submission report [GM20]. Table 3.5 shows the class-wise performance of the models submitted by Gao et al. that were ranked in the top 10. The performance was measured using the test set. It can be observed that the scenes *airport*, *public square* and *street pedestrian* have the lowest accuracies. It is not surprising considering that these scenes share many sound events. This makes it difficult to distinguish them even for a human.

Submission	**Airport**	**Bus**	**Metro**	**Metro station**	**Park**	**Public square**	**Shop. mall**	**Street pedestr.**	**Street traffic**	**Tram**
Gao_UNISA_4	56,5%	82,4%	75%	73,7%	91,5%	51,8%	70,3%	56,5%	90,2%	77,4%
Gao_UNISA_1	56,5%	82,8%	77,7%	76%	90,9%	49,8%	63,9%	55,5%	88,8%	75%
Average	**56,5%**	82,6%	76,35%	74,85%	91,2%	**50,8%**	67,1%	**56%**	89,5%	76,2%

Table 3.5: The class-wise performance of the models submitted by Gao et al. that were ranked in the top 10 [GM20]

Submission	Airport	Bus	Metro	Metro station	Park	Public square	Shop. mall	Street pedestr.	Street traffic	Tram
Suh_ETRI_3	60.8%	88.8%	82.9%	76.6%	93.7%	52.9%	81.4%	**50.9%**	92.3%	84.8%
Suh_ETRI_4	60.7%	88.6%	83.2%	76.5%	93.7%	52.4%	81.2%	**51.2%**	92.3%	84.9%
Hu_GT_3	61.7%	92.1%	84.0%	74.5%	93.9%	53.8%	81.5%	**39.4%**	92.6%	88.6%
Hu_GT_2	60.7%	91.8%	83.2%	75.5%	93.8%	52.4%	81.1%	**39.4%**	92.3%	88.6%
Hu_GT_4	59.5%	91.6%	83.4%	75.0%	93.9%	52.4%	81.4%	**39.1%**	92.6%	88.7%
Hu_GT_1	62.3%	89.8%	82.1%	72.4%	92.8%	56.2%	83.4%	**40.4%**	91.5%	85.7%
Suh_ETRI_2	59.2%	88.0%	83.4%	76.0%	93.4%	**49.8%**	78.5%	51.3%	92.1%	82.7%
Gao_UNISA_4	58.5%	88.0%	79.4%	74.7%	92.8%	56.4%	74.2%	**53.4%**	90.5%	84.3%
Gao_UNISA_1	58.2%	87.8%	76.2%	75.6%	92.7%	57.5%	75.3%	**52.3%**	90.7%	84.3%
Jie_Maxvision_1	62.4%	88.6%	75.2%	70.4%	93.8%	58.4%	76.8%	**48.3%**	90.4%	86.1%
Average	**60.40%**	89.51%	81.30%	74.72%	93.45%	**54.22%**	79.48%	**46.57%**	91.73%	85.87%

Table 3.6: Official systems ranking of the DCASE 2020 challenge showing the class-wise performance of the top 10 systems submitted for Task 1a[4].

Table 3.6 summarizes the class-wise performance of the top 10 systems submitted for Task 1a. Noteworthy is, that the performance was measured using the evaluation set, which included devices that were not part of the training and test subsets. This means, that the results show the classifier's performance when applied to unseen data like in a real world scenario. After analyzing and comparing the two tables, we summarized our findings as follows:

- The systems submitted by Gao et al. classified *airport* and *public square* more accurately when applied on the evaluation set. On the contrary, in regards of the *street pedestrian* scene they showed a worse generalization than for the test subset.
- Similar to the experimental results by Gao et al., the scenes *airport*, *public square* and *street pedestrian* are recognized the least by all top ranked submissions.
- *Street pedestrian* has the lowest accuracy in 9 of the top 10 submissions.

Our investigation showed that the *street pedestrian* scene is the most difficult to classify when using the TAU Urban Acoustic Scenes 2020 Mobile dataset. In order to fulfill our requirement of supporting 9 scenes and to increase the overall classification performance, we decided to remove the *street pedestrian* scene from our development dataset. We made this decision also considering the idea, that when it comes to the planned case study, scenes like *airport* and *public square* of a known city are more likely to be recognized by the participants. We assume that pedestrian streets have a lower recognition value and supporting this scene might not conduce to the success of the case study.

We used Python and the Librosa library to generate the acoustic features from the audio files in our implementation. The spectrograms were generated for 128 log-mel energies using 2048 FFT points, the original sampling rate of 44.1 KHz, frequencies from

[4]Table *Class-wise performance* of DCASE 2020 challenge results for Task 1A `https://dcase.community/challenge2020/task-acoustic-scene-classification-results-a#class-wise-performance` (visited on 03.07.2022)

0 to half of the sampling rate, and a hop-length of 1024 samples. The log-mel deltas and delta-deltas were calculated without padding, which reduced the number of time samples from 431 to 423. The final input feature data consisted of spectrograms with 128 frequency bins, 423 time samples and 3 channels each representing log-mel spectrograms, its delta features and its delta-delta features respectively.

In some of our experiments we used the modified TAU development dataset combined with augmented data to train our model. To generate the data, we applied some of the data augmentation strategies investigated by Hu et al., who have published[5] their work as part of the DCASE 2020 challenge [HYX+20]. We used the published code to implement the following augmentation methods:

- Speed change: The audio speed is randomly changed based on the uniform distribution. The original input length is maintained by dropping extra samples from the end when the resulting waveform is longer, and by applying padding to attain the same input length when the resulting waveform is shorter. We used the *librosa.effects.time_stretch*[6] function from the Librosa library to time-stretch the audio. The stretch factor was in the range $[0.5; 2]$.
- Pitch shift: The pitch is shifted randomly based on the uniform distribution. We used the *librosa.effects.pitch_shift*[7] function to shift the pitch in many (fractional) steps, where a step is equal to a semitone. The number of steps was chosen randomly in the range $[-4; 4]$.
- Random noise: We used the *numpy.random.normal*[8] function from the Numpy library to draw random samples from a normal (Gaussian) distribution, which were then added to the sound samples to create noise.
- Mix audios: Two audio files from the same acoustic scene class are randomly mixed. In particular, the samples of each waveform are scaled by a random factor and then added together. The scaling factors were random floating numbers between 0.5 and 1.

The presented audio transformations were performed for each audio file in the training dataset to generate new data. In order to investigate the effect of the augmentation methods on the classifier's performance, we trained and evaluated our model with different combinations of training data in our experiments. Additionally, we used mixup with crop augmentation in the temporal axis: each of the two samples combined using mixup were first cropped independently and randomly from 423 dimensions down to 400.

[5] `https://github.com/MihawkHu/DCASE2020_task1` (visited on 03.07.2022)

[6] `https://librosa.org/doc/main/generated/librosa.effects.time_stretch.html` (visited on 07.07.2022)

[7] `https://librosa.org/doc/main/generated/librosa.effects.pitch_shift.html` (visited on 07.07.2022)

[8] `https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html` (visited on 07.07.2022)

As mentioned earlier, we used TensorFlow with Keras to build and train our model. To cope with the high computational effort required to train a neural network on a large number of training samples, we performed the training on a GPU cluster with *GeForce RTX 2080 Ti* GPUs. Even though it is possible with TensorFlow to distribute the training workload on multiple GPUs, each of our experiments was conducted on a single GPU. Our network was trained for 310 or 630 epochs with backpropagation and stochastic gradient descent and a batch size of 32. We used a learning rate schedule with warm restart that resets the learning rate to its maximum value of 0.1 after 10, 30, 70, 150, 310 and 630 epochs, and then decays according to a cosine pattern to 1×10^{-5}. Depending on the configuration of the experiment, the loss was calculated using cross-entropy, focal loss or domain adaptation.

3.3.3 Evaluation

In this section we discuss the conducted experiments and their results. We tested different hyperparameters, loss functions and augmentation strategies in order to find the best calibration for our network. It is important to point out, that due to a limited access to a GPU cluster and long training times, each experiment was performed only once. Therefore, the presented results refer to one possible outcome when measured with the same hardware. In total we performed 57 training runs, of which 42 finished successfully. The training time ranged from 9 hours to 7 days. In order to track, compare and visualize our models we used a performance visualization tool for machine learning called *Weights & Biases*[9]. Weights & Biases is a free Python library which is integrated with many popular libraries such as TensorFlow. It is easy to integrate and allows the tracking of custom metrics, images and terminal outputs.

Setup

In all our experiments we used the same model that was introduced in Section 3.3.1. The model was trained on the dataset we described in Section 3.3.2 with the official train/validation split of the DCASE 2020 challenge Task 1a. In some experiments we added extra data to the training dataset, which were generated using the chosen augmentation strategies (speed change, pitch shift, random noise, mix audios). In particular, we created 4 datasets for each augmentation strategy such that we could perform the experiments using the same augmented data. In the following we explain which parameters were configured differently for each experiment and the values we used:

- Epochs: In most of our experiments we trained our model for 310 epochs as suggested by Gao et al. [GM20]. In order to investigate the network's behaviour when it is trained for a high number of epochs, we decided to run some experiments for 630 epochs, which is the last epoch after which the learning rate is reset.

[9] `https://wandb.ai/site` (visited on 03.07.2022)

- Loss function: We explored 3 approaches: cross-entropy loss (CE), categorical focal loss (FL) and mild domain adaptation (DA). We recall that for their DCASE 2020 submission Gao et al. used $FL = \{\alpha = 0.3; \gamma = 1.0\}$ and $DA = \{\beta = 0.1\}$ [GM20]. We used the categorical focal loss with varying α and γ in most cases since this approach showed to slightly boost the performance of the baseline model [GM20].

- Augmented datasets: We took into account that Gao et al. might have found the best configuration for FL and decided to focus more on experiments that use additional training data. In most of our experiments we added at least one augmented dataset to the training data. Furthermore, we investigated a few combinations and used up to 3 additional datasets in one training. Unfortunately, the training with all 4 augmented datasets exceeded the available amount of memory and could not be carried out.

Besides from evaluating the model, our main goal for running the experiments was to investigate different configurations for FL and the usage of augmented data, because we assumed that extra data may help with the problem of device mismatching. Since performing experiments for all possible configurations would exceed the scope of this work, we decided to follow a simple approach. First, we started with experiments using the configurations as reported by Gao et al. [GM20]. Then, we focused on different configurations for FL. Lastly, we performed experiments with augmented datasets, where we also investigated different FL settings. Based on the results of the finished trainings, we decided how to configure the next ones.

Experimental Results

In the following we present our results starting with the first step of our approach. Table 3.7 shows the achieved performances when we used the same configurations as Gao et al. in their submissions [GM20]. Further we will refer to these configurations as the *default*. In the table we report the test accuracy as stated by Gao et al., the model's accuracies on the test and training datasets, the difference Δ between them and the loss after the training. Additionally, we note if FL or DA was used and for how many epochs the model was trained. As it can be seen, the model that was trained with FL slightly

#	Gao et al.	Test Acc.	Train. Acc.	Δ Acc.	Train. Loss	Loss f	# Epochs
1	71.8%	77.99%	91.83%	13.85%	0.0277	FL (α=0.3, γ=1)	310
2	71.7%	77.46%	91.40%	13.94%	1.1054	CE	310
3	71.4%	77.05%	91.97%	14.92%	1.0580	DA (β=0.1)	310

Table 3.7: Experimental results using different loss functions with the default configurations ranked by test accuracy.

outperformed the other models. It achieved the highest test accuracy and is the least overfitted. In this context we examine the difference between the achieved accuracies in order to evaluate the overfitting of the model. According to our results, the application of domain adaptation increased the overfitting, since in comparison to the model that was trained with CE the test accuracy has decreased while the training accuracy went up. Overall, it can be said that all models performed similarly well as the difference in the metrics is rather small. We achieved similar results as Gao et al. which is not surprising considering that we trained our model on the same data with the exception of using one scene less. We recall that we removed the scene, that was the most difficult to classify, from the dataset we used in our experiments. When we compare our test accuracies with those reported by Gao et al., we find that by removing the difficult scene, the overall accuracy could be increased meaning that our network is more confident about its predictions. Furthermore, compared to the results of Gao et al. it can be seen that the usage of FL in our experiment led to a higher increase in test accuracy compared to the CE model. We therefore concluded that FL can improve the classification accuracy and the generalization capabilities, and decided to perform further experiments with different FL settings.

#	**Test Acc.**	**Train. Acc.**	**Δ Acc.**	**Train. Loss**	**FL α**	**FL γ**	**# Epochs**
1	77.99%	91.83%	13.85%	0.0277	0.3	1	310
2	77.87%	91.62%	13.75%	0.0529	0.5	0.5	310
3	77.05%	91.89%	14.84%	0.0902	1	1	310
4	76.79%	90.63%	13.85%	0.0904	0.75	0.1	310
5	76.71%	91.93%	15.22%	0.0179	0.25	2	310

Table 3.8: Experimental results using different focal loss parameters. Top 5 models ranked by test accuracy.

In order to save resources for the experiments with augmented data, we decided to test only a couple of different FL settings to get a general idea of how a change of the FL parameters affects the model's performance. We chose the parameters α and γ randomly following the idea of covering a brighter spectrum. In Table 3.8 we summarize our results for the FL configurations we have investigated. By comparing the presented metrics we make the following observations:

- The model, that was trained with the default settings, outperformed the other models in terms of test accuracy.
- Test #2 shows that by increasing α and decreasing γ to 0.5 the metrics change only slightly. Noteworthy is, that even though the test accuracy decreased from 77.99% to 77.87%, the overfitting was reduced from 13.85% to 13.75%.

- According to the tests #1 and #3, a larger α in combination with the same γ decreases the model's test accuracy and generalization.
- The tests #4 and #5 show that by using a significantly lower/higher γ the test accuracy does not change as much as we assumed it might.
- According to the tests #1 and #5, where a similar α was chosen, using $\gamma > 1$ leads to a less accurate and more overfitted model.

Based on our findings, we concluded that we can achieve the best results with $\alpha \in \{0.1..1\}$ and $\gamma \leq 1$. We decided to perform the experiments with augmented data using the default settings first, and then to repeat them with a slightly increased $\alpha = 0.4$. Depending on the previous results, we kept repeating the experiments while linearly de-/increasing α by 0.1. We set $\gamma = 1$ and did not change it until we have not found the best α.

#	Test Acc.	Train. Acc.	Δ Acc.	Train. Loss	FL α	FL γ	Augmentation	# Epochs
1	77.57%	77.06%	-0.51%	0.0373	0.3	1	mix audios	630
2	76.75%	80.52%	3.77%	0.0323	0.3	1	mix audios	310
3	76.68%	78.46%	1.79%	0.0338	0.3	1	pitch shift	630
4	76.41%	82.05%	5.64%	0.0325	0.3	1	random noise	630
5	76.00%	77.54%	1.54%	0.0348	0.3	1	speed change	310

Table 3.9: Experimental results using different data augmentation methods. Top 5 models ranked by test accuracy.

Once we finished the evaluation of the FL configurations, we started using additional data in our experiments. First, we performed one experiment for each augmentation strategy by adding the according augmented data to the training dataset. We then evaluated the results and depending on the outcome we repeated the experiments with modified FL parameters or for a different amount of epochs. Since the usage of additional data slowed down the training, we could achieve better performance by training the model for 630 epochs. Table 3.9 shows the top 5 of our models that used augmentation strategies ranked by their test accuracy. The findings of our evaluation of the presented results are summarized as follows:

- The best performance was achieved by the model that was trained with additional data generated by mixing files.
- Due to the long training times and a limited access to a GPU cluster, we trained the models for 630 epochs only once using the default configuration. Unfortunately, we miss the data of the experiment that used the *speed change* dataset as it crashed before it finished. Based on our results we conclude that we could achieve better

performances by using 630 epochs than by modifying the FL settings for most of our augmentation strategies.

- It can be observed that the difference between the accuracies for each model is significantly smaller in comparison to our previous results as shown in Table 3.8. We therefore conclude that the chosen augmentation strategies can improve the generalisation capabilities of a model.

Besides from evaluating models that use one augmentation strategy, we additionally carried out several experiments where we combined the strategies *speed change*, *random noise* and *mix audios*. Since the experiment that used the *pitch shift* dataset performed the worst for the default settings, we decided against using it in the combinations. Nonetheless, our experiments showed that using more training data slows down the learning process and leads overall to a decrease in the test accuracy. In this context the highest test accuracy that we achieved was 75.14%. We therefore conclude that data augmentation can improve the generalization ability of an ASC network, but since enlarging the training dataset slows down the learning process and might require additional calibration augmentation techniques should be used with caution.

Lastly, in order to complete the evaluation process of our model and to make final conclusions, we put our best experimental results in relation to each other and present the final ranking as shown in Table 3.10. As it can be seen, we achieved the best performance with a test accuracy of 77.99% by using FL and the training setup as proposed by Gao et al. This outcome is not very surprising since we used the same baseline model and training data. Nonetheless, we assumed that by reducing the number of scene classes by 1 and modifying the development dataset accordingly, the model would require a slightly different configuration compared to the baseline model in order to achieve the best

#	**Test Acc.**	**Train. Acc.**	**Δ Acc.**	**Train. Loss**	**FL α**	**FL γ**	**Augmentation**	**# Epochs**
1	77.99%	91.83%	13.85%	0.0277	0.3	1	-	310
2	77.87%	91.62%	13.75%	0.0529	0.5	0.5	-	310
3	77.57%	77.06%	-0.51%	0.0373	0.3	1	mix audios	630
4	77.46%	91.40%	13.94%	1.1054	(CE)		-	310
5	77.05%	91.89%	14.84%	0.0902	1	1	-	310
6	76.79%	90.63%	13.85%	0.0904	0.75	0.1	-	310
7	76.75%	80.52%	3.77%	0.0323	0.3	1	mix audios	310
8	76.71%	91.93%	15.22%	0.0179	0.25	2	-	310
9	76.68%	78.46%	1.79%	0.0338	0.3	1	pitch shift	630
10	76.41%	82.05%	5.64%	0.0325	0.3	1	random noise	630

Table 3.10: Summary of experimental results showing the top 10 ranked by test accuracy.

.

possible result. We therefore conclude that a different FL configuration might further improve the model's performance. Even though we did not manage to increase our model's accuracy with the chosen data augmentation techniques, our experimental results could show a significant reduction in overfitting when augmented data were additionally used for training. However, we learned that enlarging the training dataset leads to slowed down learning, longer training times, higher memory requirements and the need for additional calibration. We therefore conclude that data augmentation can be used to improve a model's ability to generalize when the necessary resources are given.

As the classifier for our mobile application we decided to choose the model that achieved the highest test accuracy in our experiments. According to our results presented in Table 3.10, in the ideal case the classifier will make predictions with $\approx 78\%$ accuracy. Considering that the data we used in our experiments for training and testing were of the same origin and similar in terms of recording conditions and quality, we assume that in a real life scenario the error rate will be higher. Nonetheless, we concluded that the classifier is robust enough to be used in real life and continued with the implementation of the prototype which will be explained next.

3.4 Soundscape Overlay

In the previous sections we discussed the design and the implementation of the classifier used in the prototype. Once the acoustic environment of the user was classified, the application plays a sound sample of the recognised scene from a city the user has chosen. Since one of the main goals of this work is to evaluate the psychological effect the overlay of a different soundscape might have on the user, the choice of the sound samples to be played is important.

To understand the significance of the soundscape dataset, we recall some of the research questions mentioned in Section 1.1:

- How does the user experience the overlay of a different soundscape? (How does it feel to walk on a public square in Vienna and experience the soundscape of a public square in Moscow?)
- Will the app manage to provide an acoustic illusion of being in the selected city?

In order to ensure that we achieve the best possible result and can answer the questions above, we defined criteria the soundscape samples have to fulfill as follows:

- The samples are recorded in well-known cities in order to increase the possibility that participants of the case study have visited the chosen city in the past. We recall the requirements stated in Section 3.1, which defined that 4 cities should be offered. To cover different parts of the world, we decided to offer the following cities: New York, Moscow, Delhi and Rio de Janeiro.

- The samples include sounds that contribute to the recognition of the city (e.g. announcements, speech).
- The samples are recorded in well known places (e.g. central areas, sightseeing).
- The samples are of satisfying quality to provide a pleasant acoustic experience.
- The samples are long enough in order to be able to create an acoustic illusion of being in the selected city. We therefore defined the desired length with 5 minutes.

Once we defined the criteria above, we started to search for matching sounds on the internet. As we could not find a publicly available dataset that would meet our requirements, we decided to create our own by crawling the video YouTube[10] platform. During our research, we faced the following challenges:

- While it was easy to find the needed material for New York and Moscow, the available data for Delhi and Rio de Janeiro was comparably sparse.
- It was difficult to find audio of good quality, especially for scenes with loud background noise (metro, metro station, tram).
- Not every city has a tram system as we know it from Europe.
- The available videos for some scenes were shorter than 5 minutes.

In order to overcome those challenges, we decided to use shorter recordings where needed and to loop them to achieve the required length. Furthermore, for cities that did not

Scene	New York	Moscow
Airport	JFK	SVO
Bus	Bus: 42nd Street → Chatham Square	M27 bus: Park Pobedy → Taganskaya
Metro	14 St, 34 St - Penn Station, Times Square - 42 St	Novoslobodskaya, Prospekt Mira
Metro station	Times Square – 42 St	Belorusskaya
Park	Central Park	Vorontsovskiy Park
Public square	Union Square	Red Square
Shopping mall	Hudson Yards Mall	GUM
Street traffic	Canal St, Lafayette St	The Garden Ring
Tram	R68A (B) train: Brighton Beach → Harlem - 145th Str.	Tram 3: Paveletskaya → Danilovsky Market

Table 3.11: The places the sound samples were recorded at in New York and Moscow.

[10] https://www.youtube.com (visited on 26.05.2021)

Scene	Delhi	Rio de Janeiro
Airport	IGI	GIG
Bus	DTC bus: Kapashera Border → Kapashera Thana	Bus: Ipanema → Copacabana
Metro	Vishwavdyalaya, Vidhan Sabha, Civil Lines, Kashmere Gate	Yellow line #4
Metro station	Shastri Park	Botafogo, Copacabana
Park	Hauz Khas Park	Park of Madureira
Public square	Palika Bazar	Cinelândia - Santa Teresa
Shopping mall	Select Citywalk	Leblon Mall
Street traffic	Paharganj New Delhi	Copacabana Str. → Copacabana Beach
Tram	Blue line: Botanical Garden → Noida Sector 52	VLT Linha 1

Table 3.12: The places the samples were recorded at in Delhi and Rio de Janeiro.

have a tram system, we chose recordings of similar public transportation like subways and regional trains operating above the ground. To give an idea which scenes the user could listen to, we summarized the places the sound samples were recorded at in Tables 3.11 and 3.12.

The audio data were extracted from the selected videos in 44.1 kHz 32-bit format. We used the Audacity [11] software to equalize the volume of the soundscape samples and to fade in/out the first/last 10 seconds. The overall soundscape dataset consisted of 36 audio tracks and was of size 267 MB.

3.5 Server Architecture

In Section 3.3.3 we presented the model that we have chosen as the classifier for our prototype. In particular, the classifier was used in the server application. In this section we describe our implementation of the server application and the API it provided.

According to the previously stated requirements, the server application had to provide an API that supports the upload of a 10 seconds audio file and returns the predicted scene or "unknown" as response after getting the predictions from the model for the received audio data. It is important to recall that the used model accepts log-mel spectrograms complemented by log-mel deltas and delta-deltas as input. Therefore, before the received audio file can be classified, its acoustic features have to be calculated using the same procedure as applied for the training dataset. In order to ensure that during the classification process we generate the same input data on the server as during the training of the model, we decided to choose a web application framework that is written in Python called *Flask*[12], which allowed us to continue to work with libraries

[11] `https://www.audacity.de` (visited on 1.06.2021)
[12] `https://flask.palletsprojects.com/en/2.2.x/` (visited on 27.04.2021)

we used before, namely Librosa and Tensorflow. Flask is a popular web framework for building RESTful web services due to its simplicity and extensibility [Gri14]. It is referred to as a micro-framework since it provides a solid core with the basic services without native support for high-level tasks like database access or web form validation. In the Flask framework an URL is mapped to a Python function through *routing*. In our Flask application we defined a route that triggers the classification for the received data and returns the result in the JavaScript Object Notation[13] (JSON) format. In order to reduce the error rate, we defined an accuracy threshold to identify a prediction as valid when it is at least 20% accurate. Therefore, in the success case the returned response included either the label of the best prediction or "unknown" in case of an invalid classification. We deployed our Flask application using *Heroku* [MS13], which is a cloud-based hosting service.

During the design of the web application we took into account that the prototype will be used in a case study. As it will be explained in Chapter 4, the main goal of the case study is to evaluate the classifier's performance in real world scenarios and the overall experience of the users. While the experience of the participants will be evaluated based on a questionnaire, in order to be able to estimate the classifier's performance the performed classification processes have to be documented. We therefore extended the web application with the following functions:

- Saving received recordings: Every audio file for which the classification was executed successfully was saved using *Amazon Simple Storage* Service[14] (Amazon S3).
- Logging classification results: The result of every successful classification was stored in a *PostgreSQL* database [15].

To get a better understanding of how the different components interact with each other Figure 3.2 depicts the process of a classification request. In the following the presented process is described in more details. The mobile application sends a request containing the audio file and the user id to the server application. The user id is required in order to be able to group the logged data to one experience. In the success case, once the received audio file has passed the validation, the web application transforms the audio data and passes it to the classifier. Then, the model's predictions are evaluated. The classification result is either the scene label of the prediction with the highest accuracy or a predefined value in case all predicted accuracies are identified as invalid (too low). Subsequently, the used audio file is uploaded to the cloud storage and the classification data is send to the database. It is important to point out that the user id and the file name are part of both requests to create the required relation between the data. Eventually, the mobile application receives a response containing the classification result. In the process two

[13] `https://www.json.org/json-en.html` (visited on 1.04.2021)
[14] `https://aws.amazon.com/s3/` (visited on 27.04.2021)
[15] `https://www.postgresql.org` (visited on 27.04.2021)

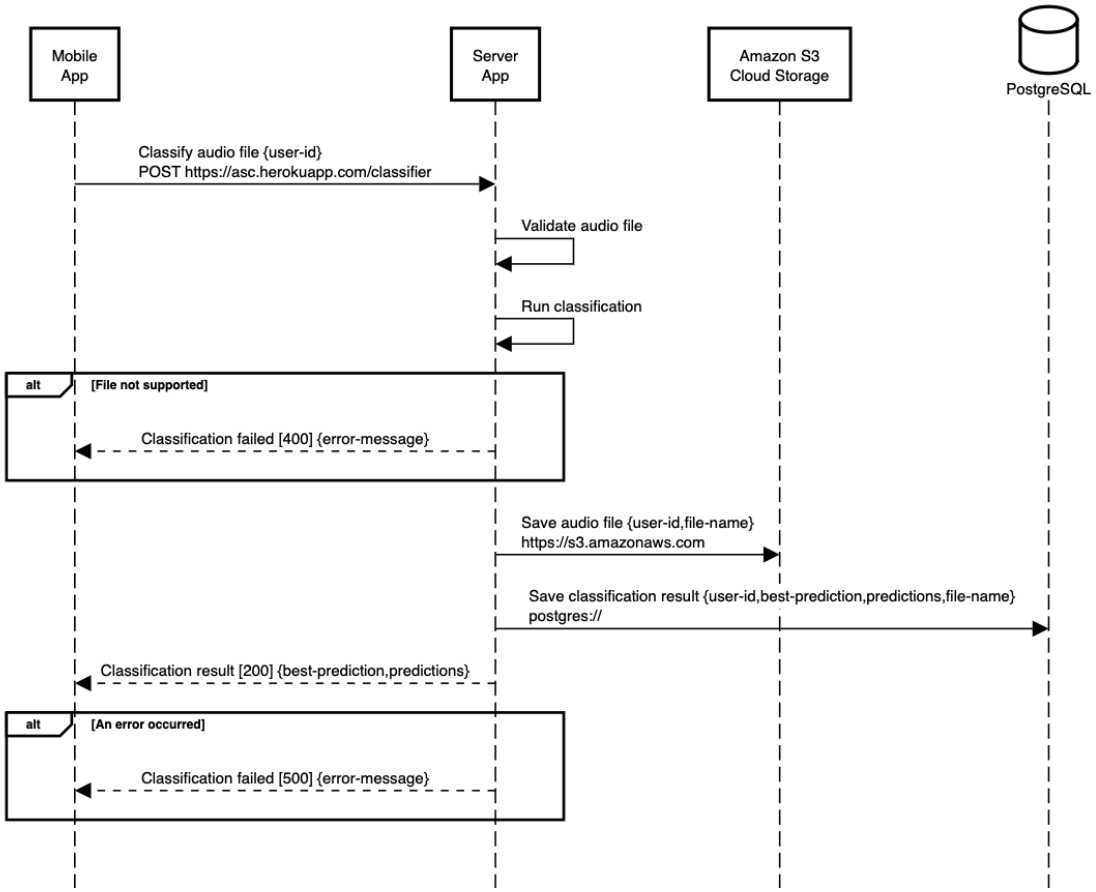


Figure 3.2: A sequence diagram of a classification request.

types of errors are handled by the web application. A request is identified as bad when no audio file of a supported type was attached. Hence, in the case that an error occurs during the classification (e.g. the file is not long enough) the server responses with an internal server error.

An important aspect of the presented architecture is that the client has to provide a unique user id. On Amazon S3 for every user a folder is created and used to save the audio files in. Storing the used audio data allows the analysis of its quality and content, but having the possibility to group the files by user gives an overview over a participant's journey. This type of overview combined with the data about the executed classifications that were stored in the database enables a comparison between the participant's experience based on their answers in the questionnaire and the classifier's performance during the case study. In the evaluation of the case study the technical part analysis this data.

In our prototype the server application has two roles. On the one hand, it provides an ASC API, which enables the mobile application to react to its environment. On the other

hand, it documents the classification processes and allows the technical evaluation of the case study. In the next section we will describe the design and the implementation of our web application as well as its communication with the server.

3.6 Mobile Application

In this section we present the technical details of the mobile application that was tested by the participants of the case study which will be explained in Chapter 4. First, we discuss the framework we used to implement the application. Then, we give a brief overview of its architecture and features. Finally, we describe the design of the user interface and the user experience.

In order to support the most used mobile operating systems, namely Android and iOS, our mobile application was implemented with the Flutter SDK[16]. Flutter is an open source framework which allows the development of mobile, web and desktop applications from a single code base. Flutter applications are written in Dart, which is a class-based and object-oriented programming language[17]. The Flutter framework uses a platform-specific embedder to compile the applications directly to machine code. Hence, Flutter applications are packaged in the same way as native applications and therefore provide an equivalent user experience.

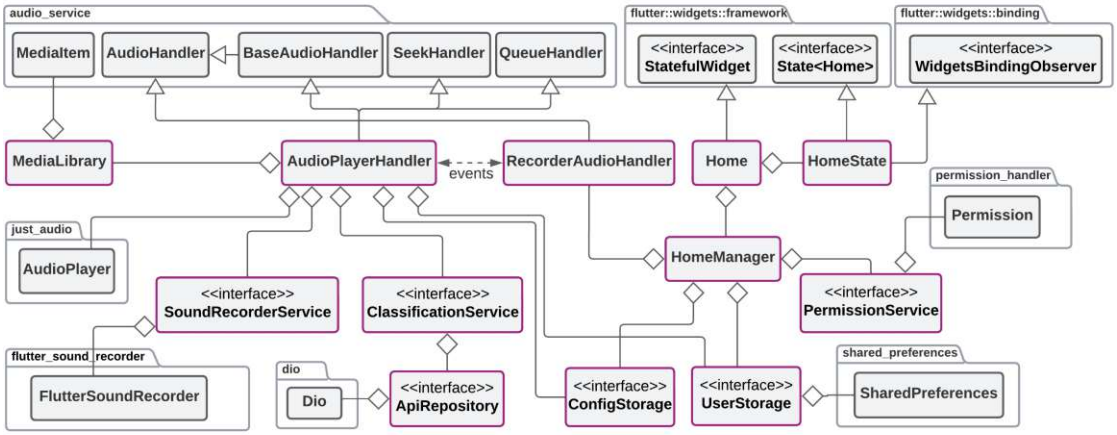


Figure 3.3: An abstracted class diagram of the Flutter application.

[16] https://flutter.dev (visited on 27.04.2021)

[17] https://dart.dev/guides/language/specifications/DartLangSpec-v2.10.pdf (visited on 27.04.2021)

Following the principle of object-oriented architecture, the business logic of the program was divided into several modules. Figure 3.3 illustrates an abstracted class diagram of the Flutter application. The diagram shows the different components and the used packages. In the following the function of each class is explained:

- *Home* represents the main page the user interacts with. In Flutter the UI is build with *widgets* [Win20]. Since the main page can have different states, *Home* is declared as a widget that has a state represented by *HomeState*. A state change triggers an update of the UI.
- *HomeManager* encapsulates the logic for managing the state of the main page. It exposes the data to the UI and reacts to user interactions by notifying the corresponding components. In mobile development the access to restricted data or restricted actions requires user's permission. Therefore, this manager requests the required permissions before starting the classification process.
- *PermissionService* requests the permission to access the microphone of the device using the *PermissionHandler*[18].
- *RecorderAudioHandler* communicates the application's state to the *AudioPlayer-Handler* by sending events using the *AudioHandler*[19].
- *AudioPlayerHandler* reacts to the received events and handles the flow of the classification process including the state of the audio player[20].
- *SoundRecorderService* uses an instance of the *FlutterSoundRecorder*[21] to record the microphone's input. Once the recording is finished, the audio data is saved as a file on the device.
- *ClassificationService* requests the predictions from the server application for the previously saved file and changes the state depending on the response.
- *ApiRepository* sends HTTP requests to the server using *Dio*[22].
- *ConfigStorage* manages the configuration data of the application, which consists of the server URL, and the supported acoustic scenes and cities.
- *UserStorage* manages the data regarding the user, which includes the user id and the selection of the city. *SharedPreferences*[23] is used to store the data permanently such that it can be restored after an application restart.

[18] `https://pub.dev/packages/permission_handler` (visited on 27.04.2021)
[19] `https://pub.dev/packages/audio_service` (visited on 27.04.2021)
[20] `https://pub.dev/packages/just_audio` (visited on 27.04.2021)
[21] `https://pub.dev/packages/flutter_audio_recorder` (visited on 27.04.2021)
[22] `https://pub.dev/packages/dio` (visited on 27.04.2021)
[23] `https://pub.dev/packages/shared_preferences` (visited on 27.04.2021)

- *MediaLibrary* provides the access to the playlist including all audio tracks. Each audio sample for a scene in a city is represented by a *MediaItem*19.

An important aspect of the presented architecture is the encapsulation of the classification process including the play of the audio tracks in the *AudioPlayerHandler*. As it can be seen in the diagram the *AudioPlayerHandler* does not have a reference to an UI component. The *RecorderAudioHandler* broadcasts the UI events to the *AudioPlayerHandler* using a stream provided by the *AudioService*19 package. This was required to avoid that the application is stopped when the user puts it in the background by going to the home screen or opening another app. In order to save resources mobile operating systems like Android and iOS stop applications that run in the background (their UI is not visible) at some point. The *AudioService* package was used because it provides the logic to run an application in a background service that is not stopped by the operating system. The mentioned event stream broadcasts the events between the UI and the background service. This library was designed for audio applications. Hence, it also provides APIs to control the audio player with headset buttons, the Android lock screen and notifications.

To get an overview of the process that is started by the user, Figure 3.4 illustrates the interaction between the user and the application's components for one classification process given that the microphone permission is granted and a city is selected.

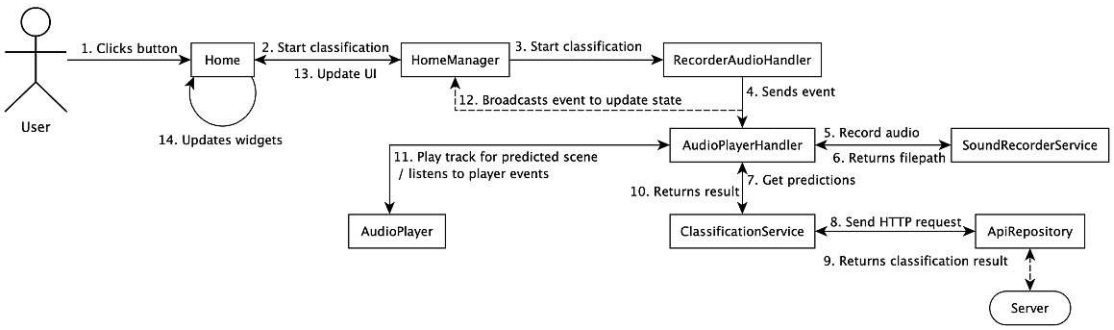
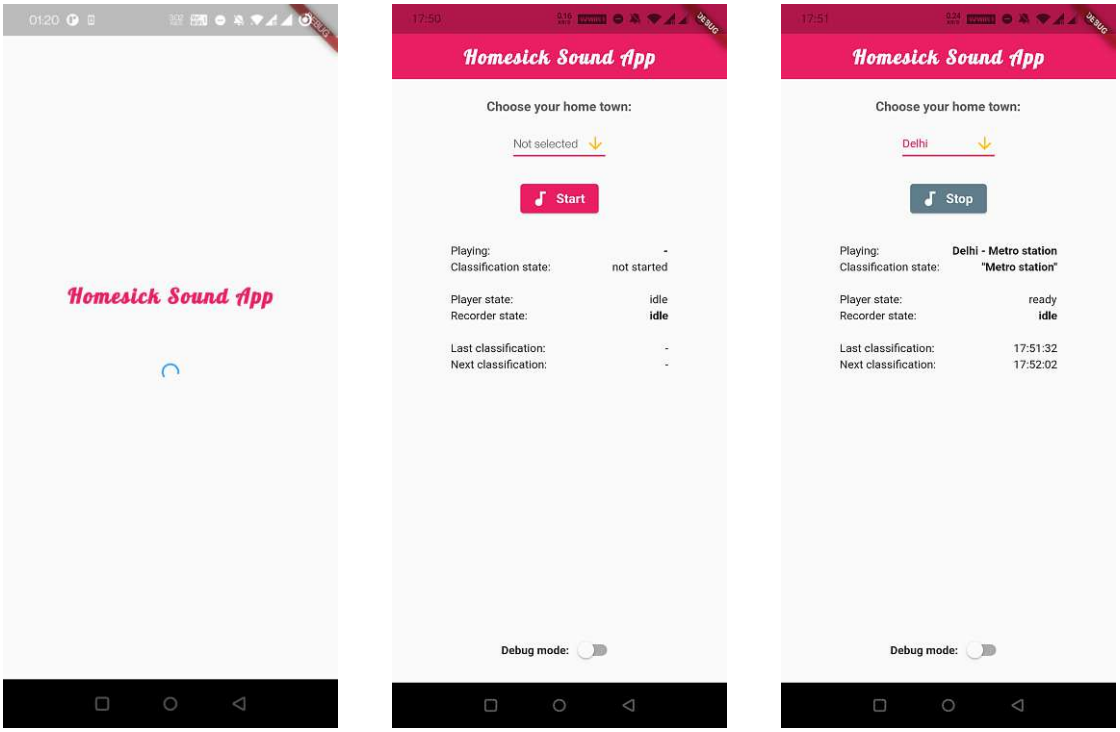


Figure 3.4: A communication diagram showing the interaction between the user and the mobile application's components for one classification process given that the microphone permission is granted and a city is selected.

Once the user has started the classification the depicted process is repeated in a 5 minutes interval. If a new classification result equals the previous one, the audio player continues to play the current track. In case that the classification of the environment has changed, the audio player plays a short audio sample that contains a DJ scratch sound[24] and then starts the new track. This approach was chosen to signalize the user that their scene has changed and to create a smoother transition to the new soundscape. In case an error occurred and the scene could not be classified and the audio player is playing,

[24] `https://www.epidemicsound.com/track/bI1zxiYBxp/` (visited on 27.04.2021)



(a) 1 (b) 2a (c) 2b

Figure 3.5: The screens of the mobile application.

the playback does not change. Otherwise the audio player is started once the next classification succeeds. After the player was started, the application can be put in the background since it is coupled with the background service.

The application's UI has a simple design and consists of two screens, which are shown in Figure 3.5. The first screen the user sees loads a configuration file that specifies which scenes and cities are supported. Furthermore, it includes the URL to the server application. The usage of a configuration file allows to change the settings of the application without the need of a new release. Once the configuration file was loaded, the user is navigated to the main screen. A widget to change the city selection is placed at the top. The button to start and to stop the classification process is placed below, which changes its design depending on the playback state of the audio player as shown in Figures 3.5b and 3.5c. Before the classification process is started, a system dialog is shown to provide the option to grant the permission to access the microphone if needed. The details of the application's state are shown in the middle of the screen for information. For debugging purposes a switch to display additional data about the process was placed at the bottom of the screen. Noteworthy is, that the city can be changed during a running classification process. In this case the audio player switches to the audio track of the same scene for the newly chosen city and plays the DJ scratch sound before switching. The classification

process continues until the user manually stops it. When the application is put in the background, a system notification is shown to inform the user about the application's state and to remind them that the application is still running.

The presented Flutter application was developed to be used by the participants of the case study, which design and results will be described in the next chapter. As it will be shown in the results, all participants used the Android application. Even though it was possible to compile the program for iOS as well, we faced a bug on iOS devices which did not occur during the development on the emulator. We therefore could not use the iOS application in the case study.

CHAPTER 4

Evaluation of the Prototype

In this chapter, the steps and the results of the evaluation of the prototype are presented. The aim of the implementation described in Chapter 3 was to use it in a case study to evaluate the classifier's performance in real world scenarios and the overall experience of the users. Section 4.1 explains by which group of people and under which conditions the prototype was tested. A feedback procedure in the form of a survey was used by the participants to share their experience. The results of the survey are summarised and analyzed in Section 4.2. Lastly, in Section 4.3 the insights of the evaluation are discussed and the following questions are finally answered: Which accuracy does the classifier have when applied to real life recordings? How does the user experience the overlay of a different soundscape? Does the overlay have any effect on the user's perception of their surroundings?

4.1 Case Study

The case study was designed respectively to give insights into several aspects of the prototype's performance. On one hand there are technical questions regarding the classifier's performance when applied to recordings of different devices in different cities. On the other hand, there is the goal to find out if overlaying human's acoustic perception with a similar soundscape has a psychological effect. A detection of a psychological effect raises the question if participants observed the same psychological effect, and for those who did, if they have something in common. Assuming that participants will experience the application differently, the feedback procedure was created that takes into account different outcomes.

4.1.1 Test Group and Scope

The definition of the test group took different aspects into account. The main requirement all participants had to fulfill was to be living abroad or in a different city than they were

originally from. This decision was made based on the premise that people, who have experience with living in different places, might have developed a more sensitive acoustic perception and are thus more likely to experience the effect of an illusion when hearing a different soundscape. Additionally, part of the participants had to meet the criteria for having a *"home town feeling"* towards one of the provided cities in the app. In this context, *"home town feeling"*, stands for a personal relationship with a city, like what a person would develop after living there for an extended period of time. The purpose of choosing a mixed group of people, those who might recognise the sounds played by the app and those who might not, was to cover a wider range of possible experiences as well as to observe the likely differences in people's perceptions. For the same reasons, the test group was not restricted by a specific gender or age.

In order to evaluate the generalizability of the classifier, the participants were supposed to use their own phones and to choose the scenes they want to visit themselves. To ensure that the feedback could be compared, testing conditions were defined and had to be followed by every participant. For this purpose every participant received instructions (see Appendix A[1]for more details). The briefing included a description of the project and the app as well as the testing conditions and a guide about how to install the app. The test scope was defined as following:

- The prototype is tested with a phone with internet connection and with Android version 6.0 or greater
- The participant goes for a one hour walk in a city and visits a few of the classifiable scenes while using the app
- The phone is placed where it can record the audio the best
- The headphones are worn throughout the entire length of the walk
- The participant fills out a survey to give timely feedback after the walk

In addition to the briefing the participants could contact the author any time in case they had questions or needed advice with the installation.

4.1.2 Feedback Procedure

The aim of the feedback procedure was to collect data for the analysis of the participant's experience. For this purpose a survey was designed including quantitative and qualitative questions. The questions referred to some personal background of the participants as well as to their experience with the app and its functionality. The survey was created with the online tool called SurveyPlanet[2]and its composition is presented in Table 4.1.

[1]Appendix A `https://ln5.sync.com/dl/f59e6aca0/9q8h9tzi-g9pv9di4-qqnxa9az-zaj3fw5y` (visited on 03.10.2022)

[2]`https://surveyplanet.com` (visited on 04.09.2021)

The questionnaire did not collect any personal data and included 18 questions with the following answer types:

- Single Choice: Only one of the provided options can be selected
- Multiple Choice: More than one of the provided options can be selected
- Free Text: An answer box to write an essay
- Likert scale: One option can be selected on a rating scale to measure either positive or negative response to a statement

As described in Section 4.1, the case study is about several technical and psychological aspects, which determined the selection of the questions. In the following the structure of the questionnaire is described. Questions 1-3 and 6 refer to the participant's background while questions 4, 5 and 7 refer to the setup the app was tested with. In order to verify that the participant's experience was not falsified by technical problems or by the usability of the app, questions 8 and 9 were asked next. Questions 10 and 11 address the visited scenes and how the classifier's performance was perceived. The following questions, 12-15 use Likert scales to describe the effect the app had on the user. The questions 16-17 were included to get an understanding for the participant's view on using the app again and recommending it to close ones. The survey ends with the option to give additional feedback in form of an essay.

4.1.3 Recordings and Logs

One goal of the case study was to evaluate the classifier's performance in real world scenarios. The participants were not asked to document every scene they visited or the classification results, since that could have distracted them from experiencing the sound effect fully. Section 3.5 explained, that the server saves the received recordings in an Amazon S3 bucket and their classification result and timestamp in a PostgreSQL database. This feature was implemented to have the possibility to match the questionnaire results to their according recordings and predictions. The survey included a question about the visited scenes. The answers to this question and the logged predictions can be compared to estimate the classifier's performance. Additionally, the saved recordings can be analysed and classified by the author to evaluate the quality of the recordings as well as of how difficult they are to be classified by a human. This approach was chosen on the one hand to be able to estimate the classification accuracy and on the other to see behind the scenes of what happened on the technical side. These insights contributed to the technical analysis of the case study.

Question	Type	Answers
1. Please select your age group	Single Choice	16 - 24 years old 25 - 34 years old 35 - 44 years old 45 - 54 years old 55 - 64 years old 65 - 74 years old 75 years old and above
2. Please select your gender identity	Single Choice	Male Female Non-binary I prefer not to answer
3. From which city are you originally from?	Free Text	
4. In which city have you tested the app?	Free Text	
5. Which cities have you tried out?	Multiple Choice	Delhi Moscow New York Rio de Janeiro
6. Do you have a “home town” feeling towards one of the cities?	Multiple Choice Free Text	Yes, a lot Yes, a little No, not at all If Yes, which cities do feel "homish" for you?
7. Have you worn your headphones continuously?	Single Choice	Yes Almost (short breaks) Had longer breaks
8. Did you have any problems with the app?	Single Choice Free Text	No Yes, my problems were:
9. How satisfied are you with the usability of the app?	Likert scale	Very Satisfied Satisfied Neutral Unsatisfied Very Unsatisfied
10. Which environments did you visit while testing the app?	Free Text	
11. How accurate did the classification feel? Were your environments recognised correctly?	Likert scale	Very accurate Mostly accurate Sometimes accurate A little accurate Not accurate at all
12. Did you perceive the overlay of a different soundscape as disturbing?	Likert scale	Extremely disturbing Very disturbing Somewhat disturbing A little disturbing Not disturbing at all
13. Did you experience the illusion of being in the city you chose in the app?	Likert scale	All the time Most of the time Many times A few times Once Not at all
14. How did the app affect your perception of your surrounding?	Likert scale	Very positive Positive Neutral Negative Very negative
15. Has your perception of your surrounding changed over time?	Likert scale	Very much A little Stayed the same
16. How likely is it that you would recommend this app to a friend or family member?	Scale	0 = not at all likely 10 = extremely likely
17. Will you use the app again?	Single Choice	Yes No
18. Feedback: Feel free to share more about your experience	Free Text	

Table 4.1: Case study survey: The questions, their answer types and answer options.

4.2 Results

This section presents the results of the case study and its evaluation. An overview over all answers of the survey and the logged data is available on the web[1]. It is important to point out that the presented results are not statistically representative since the number of participants was far too small. The subsequent described observations and conclusions can be seen as hypotheses rather than a contribution to scientific development.

4.2.1 Test Group

The test group consisted of 7 people from different places of origin. As shown in Table 4.2 the age ranged from 25 to 64 years in which the *25 - 34* group was most highly represented with 71%. The gender was distributed quite equally with 57% identifying themselves as female and 43% as male. Half of the participants had a *"home town feeling"* towards one of the cities included in the app, whereas one of the participants could relate to one of the countries. All participants used different devices and mostly had the newest Android version. The app was tested in Vienna, Stockholm, Jena and Hamburg.

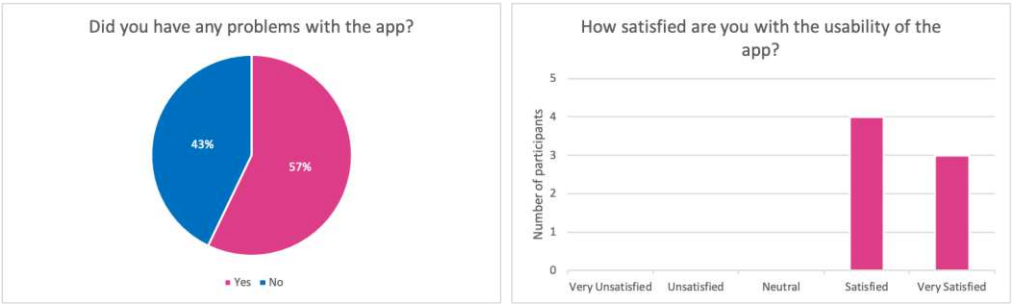
ID	Age	Gender	Origin	Tested	"Home town feeling"	Device	Android
1	25 - 34	Female	Luxembourg City, LU	Vienna, AT	Rockville (MD), US	Huawei Mate 20 lite	10.0
2	25 - 34	Female	Graz, AT	Vienna, AT	New York, US	Google Pixel 3	11.0
3	35 - 44	Male	Ystad, SE	Stockholm, SE	-	OnePlus Nord	11.0
4	25 - 34	Female	Amsterdam, NL	Vienna, AT	Delhi, IN	Samsung Galaxy A51	11.0
5	25 - 34	Male	Litomysl, CZ	Jena, DE	-	Huawei P30 Pro	10.0
6	25 - 34	Female	Schaffhausen, CH	Vienna, AT	-	Samsung Galaxy S6	9.0
7	55 - 64	Male	Moscow, RU	Hamburg, DE	Moscow, RU	OnePlus 7	11.0

Table 4.2: Overview of the case study participants including answers to the questions 1-4 and 6 as well as the logged data about their testing device.

4.2.2 App Usability and Problems

In the survey the participants were asked if they had problems using the app in order to understand if technical issues might have led to an unwanted experience. Even though the majority reported having issues as shown in Figure 4.1a, it seems that the problems could be solved by the participants or did not affect the app's functionality.

The analysis of the answers showed that only one participant reported not having heard the shift sound, which the player plays before switching to a different scene track. This could be an indication that there was a problem with the prototype. This may not be necessarily true however since other aspects like poor internet connection or bad recording quality could have minimised the classification's accuracy instead. Nevertheless it seems the reported issues were not significant enough to ruin the user experience. As shown in Figure 4.1b all participants rated the app's usability as satisfying or better. These are important observations as they let know that the app worked correctly during the walk for most participants and everyone was pleased with its usability. That in turn means it can be assumed that the participants experienced the app as intended and the evaluation



(a) Survey results of question 8. (b) Survey results of question 9.

Figure 4.1: Overview of the case study results regarding having problems with the app and its usability.

of the remaining data could give the wanted insights on the classifier's performance and the psychological effect.

4.2.3 Classification

Section 4.1.3 discussed that no procedure was implemented to verify the classifier's performance. Still, by evaluating the survey responses and the logged data the classification accuracy can be roughly estimated. This knowledge is required to better understand the experiences the participants had when using the app.

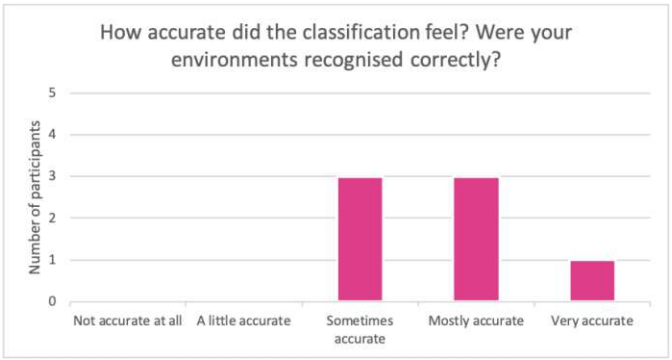


Figure 4.2: Rating of the experienced classification accuracy by the participants.

As illustrated in Figure 4.2 all participants rated the classification as at least sometimes accurate based on their perceptions. More than half of the test group reviewed it as more accurate. Based on these responses it can be concluded that every participant experienced the overlay of their surroundings with an according soundscape at least a few times and therefore might have experienced the effect the case study was designed to investigate. In order to verify the results of the rating, the logged data can be compared with the answers of the participants regarding the locations they visited. An overview over the logged predictions for each participant can be found in Appendix A[1]. The

ID	Visited scenes	Predicted scenes	Recognized	Logged predictions	Wrong predictions	Valid
1	5	4	80%	13	0	100%
2	5	2	40%	4	2	50%
3	4	3	75%	9	1	89%
4	4	3	75%	17	0	100%
5	4	2	50%	14	4	71%
6	5	2	40%	5	2	60%
7	2	2	100%	11	5	55%
			$\mu = 65.7\%$			$\mu = 75\%$

Table 4.3: Case study results: Analysis of the logged data.

Visited scenes: Number of the scenes the participant listed in the survey for question #10.
Predicted scenes: Number of the scenes found in logs which were listed as visited.
Recognized: The ratio of how many of the visited scenes were predicted in the logs.
Logged predictions: Number of the logged prediction results.
Wrong predictions: Logged predictions that can not be true as the scene was not listed as visited.
Valid: The ratio of how many of the predictions might be true and therefore valid.

different number of the logged entries per participant could have several reasons. Due to temporary loss of the internet connection some of the classification requests might have failed causing the player to loop the same track for a longer period of time. Another reason could be that there was an issue between the server and the database so that the data could not be saved but the classification request succeeded. Lastly, the app might have failed to execute the classification for some of the intervals. The logged predictions are sufficient for a simple comparison of the recognised scenes with the places the participants stated to have visited. This comparison is shown in Table 4.3. The left side shows the number of scenes the participants listed and how many of them were part of the logged prediction results. The *recognized* column shows this overlap in percent. These values can be used to estimate how many of the visited scenes were predicted correctly if all logged predictions were accurate. The result of 65.7% can be seen as proof that some scenes could not be recognized. The right side of the table shows the number of prediction logs per participant and how many of those were definitely wrong, since the participant has not listed them as visited. The *valid* column shows this relation in percentages and indicates an estimation of the classifier's accuracy if all logged predictions were correct. The result of 75% is close to the achieved accuracy for the test dataset as we can see from Table 3.10, but since not all visited scenes were recognized correctly, the actual accuracy must have been significantly lower. Due to the lack of data about the time frame of the walks, it can not be assumed that the correctly classified scenes were always recognised at the time of the participant's visit. To get a better understanding for the classifier's performance the author has listened to the logged recordings and made the following observations:

- The sound quality varies between devices.
- Some recordings include additional noises like the foot steps of the user.
- Some recordings were made in a moment when no loud sound events took place such that even for a human ear they are difficult to be classified.

Based on participants' feedback, the estimations and the author's observations it can be assumed that the actual classification accuracy was around 65%. Considering that the test group rated the classification as at least sometimes accurate, it can be concluded that every participant did experience a correct classification of their real world environment, at least at times, whereas the majority of the test group experienced it more often.

4.2.4 Psychological Effect

One of the main goals of the case study was to evaluate the psychological effect the overlay of a different soundscape might have on a human. How does it feel to walk on a public square in Vienna and to hear the soundscape of a public square in Moscow? Will the user's perception of their surroundings be affected by this experience? Finally, will the app manage to provide an acoustic illusion of being in the selected city? As mentioned in Section 4.1.2, the survey was designed taking these questions into account. First, it is important to verify that the participants did not get distracted while using the app, so that their feedback regarding their acoustic perceptions can be considered valid. For this purpose, the participants were asked if they were wearing their headphones continuously during their walk.

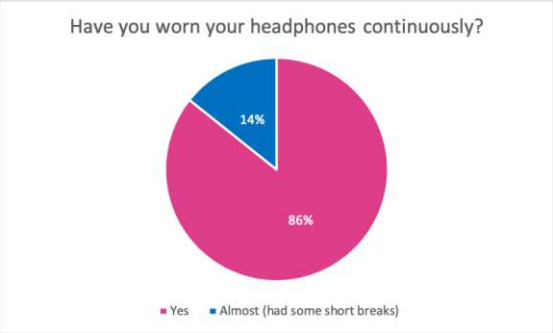
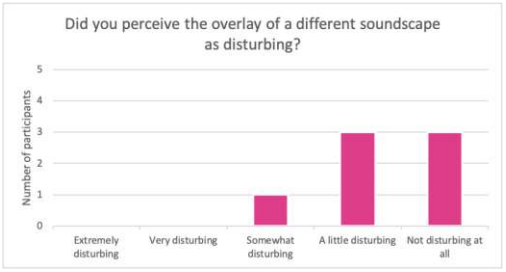
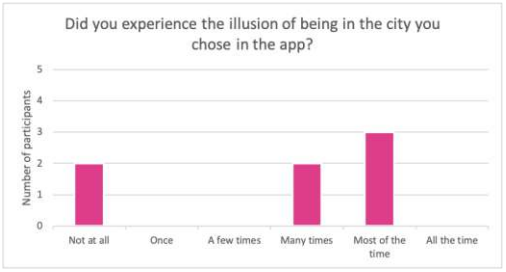


Figure 4.3: Overview of participant's behavior during their walk.

As shown in Figure 4.3 a rather small percentage of the test group experienced short breaks when listening to the app, while the rest was wearing their headphones the whole time. It can be therefore concluded that the received feedback is based on the participant's perception of an overlaid soundscape for one hour with almost no breaks. This means that an important precondition is met and the feedback can be further evaluated. The other survey questions referring to the effect the app had on the user were numbers 12-15.



(a) Survey results of question 12.

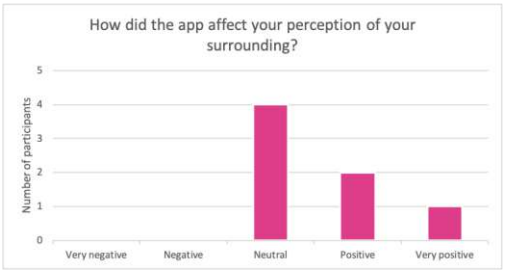


(b) Survey results of question 13.

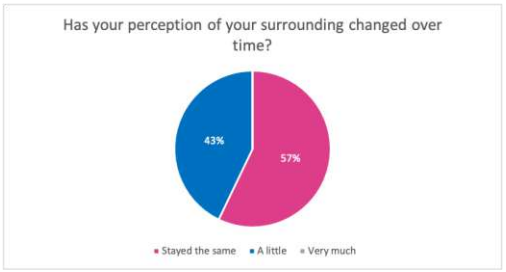
Figure 4.4: Overview of the case study results showing the rating of the perceived disturbing effect and the experienced illusion.

Next, the results of these questions are presented and later a deeper look is taken into the correlations between them.

Figure 4.4a shows that none of the participants perceived the overlay of a different soundscape as extremely or very disturbing. Nonetheless, it can be observed that more than half of the test group experienced listening to a different soundscape as little or somewhat disturbing. Despite the disturbance the app had on some of the participants, Figure 4.4b illustrates that more than half of the participants did experience the illusion of being in the city they selected in the app. It is notable that the participants had similar experiences. Some of them stated to have experienced the feeling of an illusion at least many times, while a few did not experience it at all.



(a) Survey results of question 14.



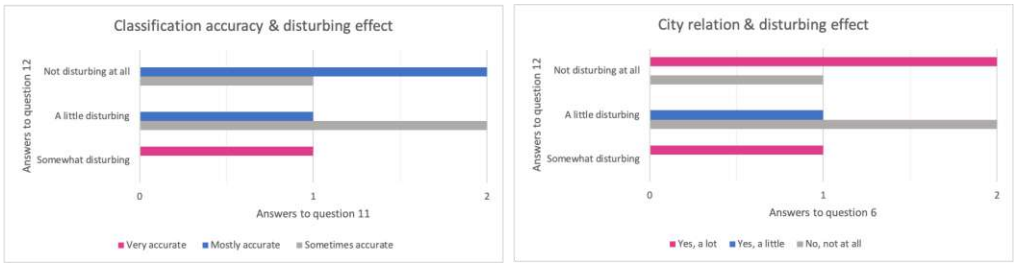
(b) Survey results of question 15.

Figure 4.5: Overview of the case study results showing the app's effect on the participant's perception of their surroundings.

Taking into account that a disturbing effect was experienced and that the effect of an illusion did take place, it is noteworthy that the participants ranked the effect the app had on their perception of their surroundings as neutral or positive as shown in Figure 4.5a. However, as seen in Figure 4.5b, the participants reported that over time there was no significant change in their perception, which rather stayed the same or changed only a little. The presented results give some interesting insights, but there might be correlations between the received feedback worth a further investigation. In the following, the results

of specific questions will be put in perspective in order to achieve a better understanding of the psychological effect the app had on the perception of the participants. To be precise, it will be evaluated if the perceived classification accuracy and participants' relationship with the cities had an impact on participants' perception in regards of the disturbing effect, the app's effect on the perception and the illusion effect. For this analysis, the answers of the corresponding questions will be plotted in grouped bar charts with the answers to one of the questions referring to the perception being on the vertical axis. Even though from the research perspective the number of participants is very small, the expectations are that some similarity aspects can be derived.

First, the impact on the experienced disturbing effect is examined by comparing Figures 4.6a and 4.6b. It can be observed that the answers are mixed and that there is no clear correlation notable. Figure 4.6a shows that even when the participants have experienced the classification as at least mostly accurate, they perceived the soundscape overlay as at least a little disturbing. At the same time Figure 4.6b shows that having a relationship with one of the cities does not necessarily decrease the disturbance. Therefore it can be concluded, that hearing a sound overlay might be experienced as disturbing regardless of the classification's accuracy and the participants' personal background.

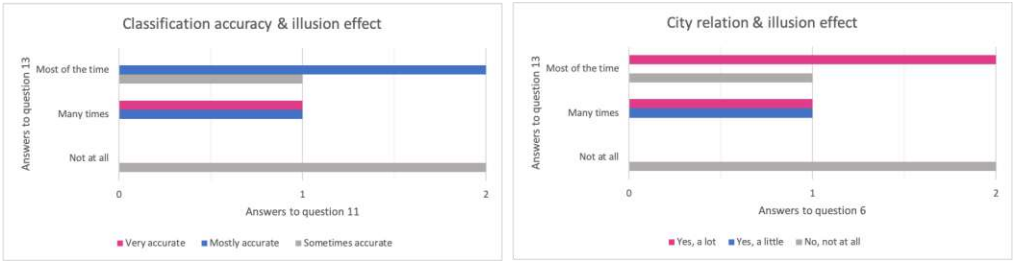


(a) Grouped answers to questions 11 & 12. (b) Grouped answers to questions 6 & 12.

Figure 4.6: Comparison of survey results grouped by questions to illustrate correlations between the perceived classification accuracy, the participants' relation to the cities and the experienced disturbing effect.

Next, we evaluated whether experiencing an illusion of being in a different city was influenced by the classification accuracy and/or the personal relation of the participants to the cities. Following the same approach for this purpose Figures 4.7a and 4.7b are compared. Unlike the disturbance, it seems that the effect of the illusion does depend on the classification accuracy as well as on the personal background. It can be seen that both figures illustrate a similar pattern. Figure 4.7a shows that a mostly accurate classification can lead to experiencing the illusion effect many times. On the other hand, Figure 4.7a shows that the stronger the personal relationship to the cities is, the more often the participant will experience the illusion of being in that city. Additionally it can be concluded that the lower the classification accuracy, the more foreign the selected city is perceived and the less likely an illusion will be experienced.

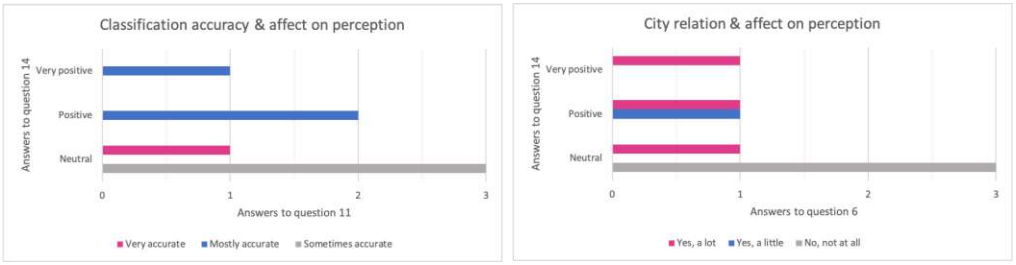
Lastly, the impact of the app's effect on the participants' perception is analysed. The



(a) Grouped answers to questions 11 & 13. (b) Grouped answers to questions 6 & 13.

Figure 4.7: Comparison of survey results grouped by questions to illustrate correlations between the perceived classification accuracy, the participants' relation to the cities and the experienced illusion effect.

comparison of Figures 4.8a and 4.8b shows that the accuracy of the classifier as well as personal background can lead to a more positive effect on participants' perception of their surroundings. Figure 4.8a shows that higher classification accuracy is more likely to implicate a positive effect, while Figure 4.8b shows that the same effect can be achieved by having a stronger personal relationship with the selected city.



(a) Grouped answers to questions 11 & 14. (b) Grouped answers to questions 6 & 14.

Figure 4.8: Comparison of survey results grouped by questions to illustrate correlations between the perceived classification accuracy, the participants' relation to the cities and the app's effect on the perception.

As the final evaluation step of the app's psychological effect, the presented observations are summarised. Based on the presented survey results and the demonstrated correlations, we conclude that:

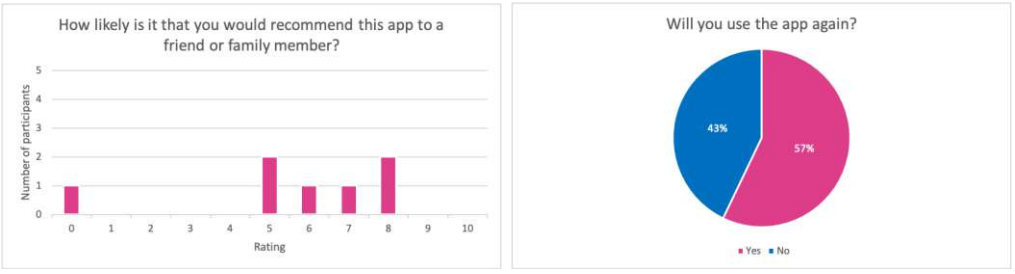
- The overlay of a different soundscape has a psychological effect on the user, whereby those, who perceive a similar level of classification accuracy and share having a personal relation to the cities, experience that effect similarly.
- Listening to a different soundscape can be experienced as disturbing.

- The more accurate the classification is, the stronger the personal relationship with the selected city, the more often the user will experience the illusion of being in that city.
- How positive the app's effect is experienced depends on the classification accuracy and the personal relationship the user has with the selected city.
- The user's perception of their surroundings does not change over time or changes only a little while using the app.

Finally, the questions mentioned in the beginning of this section can be answered. To walk around a public square in Vienna and to hear the soundscape of a public square in Moscow might be experienced a bit as disturbing, but will very likely create a positive feeling. The user's perception of their surroundings might be affected by the app. An acoustic illusion of being in the selected city will more likely take place if the user has a personal relationship to that city.

4.2.5 General Feedback

The survey included a few questions about how the participants value the app. This was done by asking how likely it is that the participants will recommend the app to people close to them and if they would use the app again. The results are shown in Figure 4.9. The previously presented results have shown, that the test group had different experiences depending on several aspects, like the perceived classification accuracy, their personal perception and personal background. It is therefore not surprising that with the exception of one user, the test group valued the app differently even though the majority would rather recommend the app to others, as illustrated in Figure 4.9a. In the previous section it was reported that more than half of the participants experienced the hearing of a different soundscape as disturbing, but it seems that to experience the illusion of being in the selected city was of a greater value such that more than half of the users stated that they would use the app again (as shown in Figure 4.9b).



(a) Survey results of question 16. (b) Survey results of question 17.

Figure 4.9: Overview of the case study results showing the participant's rating of the app.

The last part of the survey was an option where the participants could give additional feedback in form of an essay. Two aspects were reported by several participants. A few shared they wished the classifier's performance would have been better. They had the impression the classifier got stuck and recognised the same scene for too long. Some others suggested the idea of using noise-cancelling headphones in order to get rid of the real sounds around them when the overlay is not loud enough to cover the real noises from the surroundings. For more details, the full answers can be found in Appendix A[1].

4.3 Discussion

In the previous sections the design of the case study and its results were presented. In Section 4.1 the main goal and the scope of the case study were explained. The main focus of the case study referred on the one hand to the classifier's performance in real world scenarios, on the other to the psychological effect the app might have on the user's perception of their surroundings. In the following, the findings will be discussed and some conclusions made focusing first on the classifier's accuracy and later on the psychological effect.

The implementation and the training of the classifier were extensively discussed in Section 3.3. It is important to recall that the development dataset included recordings from only 3 real mobile devices. Even though the development dataset was enriched with simulated data, considering the number of mobile devices that are currently on the market, it can be said that the model has seen only a small fraction of what is possible in reality. The achieved classification accuracy for the test dataset was by 78% as shown in Table 3.10. This is considered as a good performance for the current state of technology when compared to the state-of-the-art systems submitted for the DCASE 2020 challenge. However, it is worth pointing out that this high accuracy could only be achieved by applying the model to data that are of similar quality and origin as the training data. In the case study different devices were used, none of which were the devices used for the creation of the development dataset. The devices were worn differently by the participants when recording the sounds of their surroundings. And lastly, the app was tested in cities (Jena, Hamburg) which were not part of the development dataset. The classifier was challenged to predict the scenes for data that were very different from what it has seen before. Nonetheless, according to the results of the case study and the analysis of the logged data, it seems that the classifier did perform well enough in order to provide the wanted experience. As demonstrated in the previous section, the accuracy in real world scenarios was estimated with 65% which corresponds with the impression of the participants who valued the accuracy on average as mostly accurate. It can be concluded that the classifier is robust enough to be used in real life, but for the mentioned reasons some inaccuracy should be taken into account.

In Section 4.2.1 the composition of the test group was set forth and it was mentioned that a mix of people was chosen based on their personal background in order to cover a wider range of possible experiences. The results presented in Section 4.2 showed that

this expectation was fulfilled. Even though the number of participants was very low, different experiences were reported via the survey whereby some similarities could be found and their probable reasons derived. It was shown that hearing a soundscape of a different city can be perceived as positive and create the acoustic illusion of being in that city. Another observation was that the closer the personal relationship with the selected city is, the more often that illusion will be experienced. However, the quality of the app's experience depends largely on the classifier's performance. As mentioned before, the classifier's accuracy was not very high, therefore it can be assumed that with a better classifier a greater effect on user's perception can be achieved. Based on the general feedback small adjustments like using noise-cancelling headphones might improve the quality of the sound and therefore increase the app's effect. Lastly, the participants reported not having noticed a significant change of their perception over time while using the app. Since the illusion effect did not take place continuously, the change of the user's perception could be researched further using a more accurate classifier.

CHAPTER 5

Conclusion and Future Work

In this chapter we provide a summary of the conclusions and give an outlook on future work. In Section 5.1 we reflect on the steps that have been taken to answer the research questions of this thesis and the achieved results. Eventually, in Section 5.2 we give a brief overview over improvements and further research that could be carried out in the future.

5.1 Conclusion

In this work, we developed a prototype that uses ASC to create an acoustic illusion and to evaluate it's effect on human's perception when used in real world scenarios. In particular, we were interested in the effect the user might experience, when they hear the soundscape of the scene in which they are in, but from a different city.

To achieve this goal, we investigated the current state-of-the-art techniques for ASC by examining the top 10 ranked submissions for Task 1a of the DCASE 2020 challenge, which addressed a very similar problem as this thesis. The research showed that the systems that used a ResNet, either as a single network or as an ensemble, achieved the best performance. Furthermore, we found that all evaluated solutions used log-mel energies with their deltas and delta-deltas as audio features. When comparing the submissions we learned that data augmentation was the main tool to address the generalization problem. Another important observation was that in regards of achieved accuracy ensembles outperformed single network solutions by a very small margin for the price of a significantly higher model complexity. Based on the outcome of the evaluation we chose a baseline model for our classifier and modified it according to our requirements. To train the network we used the TAU Urban Acoustic Scenes 2020 Mobile dataset, whereby in order to fulfill our requirement of supporting 9 scenes and to increase the overall classification performance, we removed the scene that showed the lowest accuracies. Hence, the model was trained to classify the following scenes: *airport*, *bus*, *metro*, *metro station*, *park*, *public square*, *shopping mall*, *street traffic* and *tram*.

During the calibration phase of the network, we conducted several experiments where we trained the model using different hyperparameters, loss functions and data augmentation strategies. The experimental results showed that the applied data augmentation techniques could significantly improve the model's generalization ability. However, we learned that enlarging the training dataset leads to slowed down learning and the need for additional calibration. We achieved the best performance with a test accuracy of 77.99% by using categorical focal loss and the training setup as proposed for the baseline model by Gao et al.[GM20] with Mixup being the only used data augmentation method. The model that had the best performance according to the experiments was chosen to be used as the classifier in our prototype.

The development of the prototype consisted of three parts. First, in order to create a dataset of sound samples to use for the soundscape overlay we searched the internet for sounds that would fulfill our requirements. To cover different parts of the world, we chose the following cities for the prototype: *New York*, *Moscow*, *Delhi* and *Rio de Janeiro*. As we could not find publicly available audio tracks, we extracted the audio data from videos that included the desired scenes. Then, we designed and implemented a server application with Flask that provided an API to get predictions from the model for a given audio file. Moreover, we added the required features to the server program in order to store the needed data for the evaluation of the case study. Lastly, we developed a cross mobile application using the Flutter SDK to support the mobile operating systems Android and iOS. Once started, the mobile application repeats the classification process in a 5 minutes interval until manually stopped by the user, which consisted of the following steps: record the input of the device's microphone for 10 seconds; request predictions for the recorded audio data from the server; play a soundscape sample according to the classification result for the city the user has previously selected. Contrary to our expectations, we found that the application behaved differently on iOS and therefore continued with the case study using the Android application only.

We designed and performed a case study to answer the research questions of this work. In particular, 7 participants tested the mobile application for one hour and reported their experience by filling out a questionnaire. We analyzed the questionnaire results and the logged data on the server and evaluated the classifier's performance in real world scenarios and the psychological effect the acoustic illusion had on the users. Based on the results we estimated the classifier's accuracy with 65%. Moreover, we found that the participants valued the accuracy on average as "mostly accurate". Even though the number of participants was low, we found similarities in the reported experiences. The comparison of the questionnaire answers showed that hearing a soundscape of a different city can be perceived as positive and create the acoustic illusion of being in that city. Another observation was that the closer the personal relationship with the selected city was, the more often that illusion was experienced. Lastly, the overlay of a different soundscape did not appear to affect the user's perception. Since the quality of the experience depends largely on the classifier's performance, we concluded that a greater illusion effect including a change of the user's perception can be achieved with a more accurate classifier.

5.2 Future Work

We propose two main directions in which future work can be carried out: (1) Increase of the classifier's accuracy, and (2) Improvement of the mobile application's user experience.

In order to improve the classifier's performance, a direct follow up of this work could include the usage of additional training data to cover more mobile devices and the execution of more experiments to further calibrate the model. The additional data could be created by using videos as it was done for the soundscape dataset. For the next calibration phase more computation resources could be provided to focus on experiments with data augmentation. In this thesis we evaluated ASC solutions that used ResNets and log-mel spectrograms complemented by log-mel deltas and delta-deltas as audio features. Further scientific research on other approaches can be done as described in Section 2.5.

There are several ways to improve the user experience of the mobile application. First, since a few of the case study participants reported having experienced a potential bug where the classifier gets stuck, the source code should be improved. Second, the quality of the soundscape dataset could be increased by supporting only those cities, for which audio samples in good quality can be found. Third, the users could be advised to use noise-cancelling headphones in order to increase the acoustic illusion by reducing the perception of the real sounds around them. The presented steps would lead to a more pleasant experience which might improve the acoustic illusion and therefore lead to a different effect on the user's perception.

Finally, in order to achieve statistically representative results, the presented case study and its evaluation should be repeated with a larger number of participants.

List of Figures

List of Tables

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`, 2015.

[Abe20] Jakob Abeßer. A Review of Deep Learning Based Methods for Acoustic Scene Classification. *Applied Sciences*, 10, March 2020.

[ADP07] Jean-Julien Aucouturier, Boris Defreville, and Francois Pachet. The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. *The Journal of the Acoustical Society of America*, 122:881–91, September 2007.

[AGO+18] S. Amiriparian, M. Gerczuk, S. Ottl, N. Cummins, S. Pugachevskiy, and B. Schuller. Bag-of-Deep-Features: Noise-Robust Deep Feature Representations for Audio Analysis. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, July 2018.

[AJB+17] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A Closer Look at Memorization in Deep Networks. *arXiv:1706.05394 [cs, stat]*, July 2017. `http://arxiv.org/abs/1706.05394`.

[All77a] J. Allen. Short term spectral analysis, synthesis, and modification by discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3):235–238, June 1977.

[All77b] Jont Allen. Short term spectral analysis, synthesis, and modification by discrete Fourier transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25:235–238, July 1977.

[AMGL17] Jakob Abeßer, Stylianos-Ioannis Mimilakis, Robert Gräfe, and Hanna Lukashevich. Acoustic scene classification by combining autoencoder-based dimensionality reduction and convolutional neural networks, October 2017.

[AMJ+14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, October 2014.

[AMK06] P.K. Atrey, N.C. Maddage, and M.S. Kankanhalli. Audio Based Event Detection for Multimedia Surveillance. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 5, pages V–V, May 2006.

[AR77] J.B. Allen and L.R. Rabiner. A unified approach to short-time Fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11):1558–1564, November 1977.

[ATY+19] Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, 8(3):292, March 2019. https://www.mdpi.com/2079-9292/8/3/292.

[AZ17] Relja Arandjelović and Andrew Zisserman. Look, Listen and Learn. *arXiv:1705.08168 [cs]*, August 2017. http://arxiv.org/abs/1705.08168.

[Bal15] Glen Ballou. *Handbook for Sound Engineers*. Audio Engineering Society Presents. Taylor and Francis, Oxford, 2015.

[BBC+10] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Mach Learn*, 79(1-2):151–175, May 2010. http://link.springer.com/10.1007/s10994-009-5152-4.

[BDSY14] Nicolas Boulanger-Lewandowski, Jasha Droppo, Mike Seltzer, and Dong Yu. Phone sequence modeling with recurrent neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5417–5421, May 2014.

[Ben09] Y. Bengio. Learning Deep Architectures for AI. *Foundations*, 2:1–55, January 2009.

[BGSP15] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D. Plumbley. Acoustic Scene Classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3):16–34, May 2015.

[Bis96] Christopher M Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford [u.a.], 1. publ., reprinted. edition, 1996.

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, 2006.

[BK97] A. Biem and S. Katagiri. Cepstrum-based filter-bank design using discriminative feature extraction training at various levels. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1503–1506 vol.2, 1997.

[BLBV13] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio Chord Recognition with Recurrent Neural Networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pages 335–340, Curitiba, Brazil, November 2013. ISMIR. https://doi.org/10.5281/zenodo.1418319.

[BLD12] Emmanouil Benetos, Mathieu Lagrange, and Simon Dixon. Characterisation of Acoustic Scenes using a Temporally Constrained Shit-Invariant Model. In *DAFx*, York, United Kingdom, September 2012. https://hal.archives-ouvertes.fr/hal-01126770.

[BMP06] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July 2006. Association for Computational Linguistics. https://aclanthology.org/W06-1615.

[Boa16] Boualem Boashash. *Time Frequency Signal Analysis and Processing : A Comprehensive Reference*. EURASIP and Academic Press Series in Signal and Image Processing. Academic Press, Amsterdam Boston, second edition. edition, 2016.

[BSER16a] Victor Bisot, Romain Serizel, Slim Essid, and Gaël Richard. Acoustic scene classification with matrix factorization for unsupervised feature learning. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6445–6449, March 2016.

[BSER16b] Victor Bisot, Romain Serizel, Slim Essid, and Gael Richard. Feature Learning with Matrix Factorization Applied to Acoustic Scene Classification. https://hal.archives-ouvertes.fr/hal-01362864, September 2016.

[BSER16c] Victor Bisot, Romain Serizel, Slim Essid, and Gael Richard. SUPERVISED NONNEGATIVE MATRIX FACTORIZATION FOR ACOUSTIC SCENE CLASSIFICATION. In *IEEE International Evaluation Campaign on Detection and Classification of Acousitc Scenes and Events (DCASE 2016)*, Budapest, Hungary, September 2016. https://hal.archives-ouvertes.fr/hal-02943480.

[BSER17] Victor Bisot, Romain Serizel, Slim Essid, and Gael Richard. Nonnegative Feature Learning Methods for Acoustic Scene Classification. In *DCASE 2017 - Workshop on Detection and Classification of Acoustic Scenes and Events*, Munich, Germany, November 2017. https://hal.inria.fr/hal-01636627.

[BWL20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934 [cs, eess]*, April 2020. http://arxiv.org/abs/2004.10934.

[CB96] H. P. Combrinck and E.C. Botha. On the mel-scaled cepstrum, 1996.

[CB05] Pedro Cano and Eloi Batlle. A Review of Audio Fingerprinting. *Journal of VLSI Signal Processing*, 41:271–284, November 2005.

[CGCB14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. http://arxiv.org/abs/1412.3555, December 2014.

[Cha18a] Pragnan Chakravorty. What Is a Signal? [Lecture Notes]. *IEEE Signal Processing Magazine*, 35(5):175–177, September 2018.

[Cha18b] Eugene Charniak. *Introduction to Deep Learning*. The MIT Press, Cambridge, Massachusetts London, 2018.

[Cho15] Francois Chollet. Keras. https://keras.io, 2015.

[Chr19] M.G. Christensen. *Introduction to Audio Processing*. Springer, 2019.

[CK14] Sachin Chachada and C.-C. Jay Kuo. Environmental sound recognition: A survey. *APSIPA trans. signal inf. process.*, 3:e14, 2014. https://www.cambridge.org/core/product/identifier/S2048770314000122/type/journal_article.

[CLL19] Chuan-Yu Chang, Chien-Chou Lin, and Horng-Horng Lin, editors. *New Trends in Computer Technologies and Applications: 23rd International Computer Symposium, ICS 2018, Yunlin, Taiwan, December 20–22, 2018, Revised Selected Papers*, volume 1013 of *Communications in Computer and Information Science*. Springer Singapore, Singapore, 2019. http://link.springer.com/10.1007/978-981-13-9190-3.

[CNK09] Selina Chu, Shrikanth Narayanan, and C.-C. Jay Kuo. Environmental Sound Recognition With Time–Frequency Audio Features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1142–1158, August 2009.

[CPZ+19] Zhen Cui, Jinshan Pan, Shanshan Zhang, Liang Xiao, and Jian Yang, editors. *Intelligence Science and Big Data Engineering. Visual Data Engineering: 9th International Conference, IScIDE 2019, Nanjing, China, October 17–20, 2019, Proceedings, Part I*, volume 11935 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, 2019. http://link.springer.com/10.1007/978-3-030-36189-1.

[CS03] Michael Cowling and Renate Sitte. Comparison of techniques for environmental sound recognition. *Pattern Recognition Letters*, 24(15):2895–2907, November 2003.

[CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[CvG+14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. http://arxiv.org/abs/1406.1078, September 2014.

[CWSB19] Jason Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3852–3856, Brighton, United Kingdom, May 2019. IEEE. https://ieeexplore.ieee.org/document/8682475/.

[DG17] Monika Doerfler and Thomas Grill. Inside the Spectrogram: Convolutional Neural Networks in Audio Processing. In *SAMPTA*, July 2017.

[DHS01] Richard O Duda, Peter E Hart, and David G Stork. *Pattern Classification*. A Wiley-Interscience Publication. Wiley, New York, NY [u.a.], 2. ed.. edition, 2001.

[Dit17] Tim Dittmar. Audio engineering 101 : A beginner's guide to music production. In *Audio Engineering 101 : A Beginner's Guide to Music Production*. Routledge, London, second edition.. edition, 2017. 10.4324/9781315618173.

[DJV+13] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *arXiv:1310.1531 [cs]*, October 2013. http://arxiv.org/abs/1310.1531.

[dKMR05] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A Tutorial on the Cross-Entropy Method. *Ann Oper Res*, 134(1):19–67, February 2005. https://doi.org/10.1007/s10479-005-5724-z.

[DL18] Guozhu Dong and Huan Liu. *Feature Engineering for Machine Learning and Data Analytics*. Chapman & Hall/CRC Data Mining & Knowledge Discovery Series ; No. 44. CRC Press/Taylor & Francis Group, Boca Raton, FL, first edition.. edition, 2018. 10.1201/9781315181080.

[DLC+20] Bo Dong, Cristian Lumezanu, Yuncong Chen, Dongjin Song, Takehiko Mizoguchi, Haifeng Chen, and Latifur Khan. At the Speed of Sound: Efficient Audio Scene Classification. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 301–305. Association for Computing Machinery, New York, NY, USA, June 2020. https://doi.org/10.1145/3372278.3390730.

[DM80] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, August 1980.

[DRS18] K Deergha Rao and M. N. S Swamy. *Digital Signal Processing : Theory and Practice*. Springer, Singapore, 2018.

[Dub20] Pablo Duboue. The art of feature engineering: Essentials for machine learning. In *The Art of Feature Engineering*. Cambridge University Press, GB, 2020.

[DW06] Guy J. Brown DeLiang Wang. *IEEE Xplore Book Abstract - Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley-IEEE Press, 2006. https://ieeexplore.ieee.org/book/5769523.

[EDW18] Hamid Eghbal-zadeh, Matthias Dorfer, and Gerhard Widmer. Deep Within-Class Covariance Analysis for Robust Audio Representation Learning. *arXiv:1711.04022 [cs, eess]*, November 2018. http://arxiv.org/abs/1711.04022.

[EISA18] Ahmet Elbir, Hamza Ilhan, Gorkem Serbes, and Nizamettin Aydin. Short Time Fourier Transform based music genre classification. In *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*, pages 1–4, April 2018.

[Ell96] Daniel P W Ellis. *Prediction-Driven Computational Auditory Scene Analysis*. PhD thesis, Massachusetts Institute of Technology, June 1996.

[EPT+06] A.J. Eronen, V.T. Peltonen, J.T. Tuomi, A.P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi. Audio-based context recognition.

IEEE Transactions on Audio, Speech, and Language Processing, 14(1):321–329, January 2006.

[Fan68] Gunnar Fant. Analysis and synthesis of speech processes. In Bertil Malmberg, editor, *Manual of Phonetics*, pages 173–277. North-Holland Publishing Company Amsterdam, 1968.

[FMdCH17] G. Z. Felipe, Y. Maldonado, G. d Costa, and L. G. Helal. Acoustic scene classification using spectrograms. In *2017 36th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–7, October 2017.

[Fur86] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(1):52–59, February 1986.

[FVRA20] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A Brief Review of Domain Adaptation. *arXiv:2010.03978 [cs]*, October 2020. http://arxiv.org/abs/2010.03978.

[FXM+18] Dawei Feng, Kele Xu, Haibo Mi, Feifan Liao, and Yan Zhou. Sample Dropout for Audio Scene Classification Using Multi-Scale Dense Connected Convolutional Neural Network. *arXiv:1806.04422 [cs]*, June 2018. http://arxiv.org/abs/1806.04422.

[FZ07] H. Fastl and Eberhard Zwicker. *Psychoacoustics: Facts and Models*. Number 22 in Springer Series in Information Sciences. Springer, Berlin ; New York, 3rd. ed edition, 2007.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. https://www.deeplearningbook.org/.

[GBR+12] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. http://jmlr.org/papers/v13/gretton12a.html.

[GDÇ+18] Shayan Gharib, Konstantinos Drossos, Emre Çakir, Dmitriy Serdyuk, and Tuomas Virtanen. Unsupervised adversarial domain adaptation for acoustic scene classification. *arXiv:1808.05777 [cs, eess]*, August 2018. http://arxiv.org/abs/1808.05777.

[GEF+17] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, New Orleans, LA, March 2017. IEEE. http://ieeexplore.ieee.org/document/7952261/.

[Ger00] David Gerhard. Audio Signal Classification, July 2000.

[GGvv16] Yağmur Güçlütürk, Umut Güçlü, Marcel A. J. van Gerven, and Rob van Lier. Deep Impression: Audiovisual Deep Residual Networks for Multimodal Apparent Personality Trait Recognition. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, volume 9915, pages 349–358. Springer International Publishing, Cham, 2016. http://link.springer.com/10.1007/978-3-319-49409-8_28.

[GM20] Wei Gao and Mark McDonnell. Acoustic Scene Classification Using Deep Residual Networks with Focal Loss and Mild Domain Adaptation. Technical report, DCASE2020 Challenge, June 2020. http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Gao_132.pdf.

[GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330. PMLR, July 2017. https://proceedings.mlr.press/v70/guo17a.html.

[Gra12] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. http://link.springer.com/10.1007/978-3-642-24797-2.

[Gri14] Miguel Grinberg. *Flask Web Development : [Developing Web Applications with Python]*. O'Reilly, Sebastopol, Calif. [u.a.], 1. ed.. edition, 2014.

[GSR13] Jürgen T. Geiger, Björn Schuller, and Gerhard Rigoll. Large-scale audio feature extraction and SVM for acoustic scene classification. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, October 2013.

[Gur97] Kevin Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., USA, 1997.

[GXLA17] Jinxi Guo, Ning Xu, Li-Jia Li, and Abeer Alwan. Attention Based CLDNNs for Short-Duration Acoustic Scene Classification. In *INTERSPEECH*, pages 469–473, August 2017.

[Har96] William Hartmann. Pitch, periodicity, and auditory organization. *The Journal of the Acoustical Society of America*, 100 6:3491–502, 1996.

[Haw04] Douglas M. Hawkins. The Problem of Overfitting. *J. Chem. Inf. Comput. Sci.*, 44(1):1–12, January 2004. https://pubs.acs.org/doi/10.1021/ci0342472.

[HBKK05] K.-P. Hui, N. Bean, M. Kraetzl, and Dirk P. Kroese. The Cross-Entropy Method for Network Reliability Estimation. *Ann Oper Res*, 134(1):101, February 2005. https://doi.org/10.1007/s10479-005-5726-x.

[HDY+12] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012.

[Hin12] Geoffrey E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. http://link.springer.com/10.1007/978-3-642-35289-8_32.

[HKK+10] David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis, editors, *Artificial Neural Networks – ICANN 2010*, volume 6354, pages 92–101. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. http://link.springer.com/10.1007/978-3-642-15825-4_10.

[HL16] Yoonchang Han and Kyogu Lee. Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation, 2016. https://arxiv.org/abs/1607.02383.

[HMV20a] Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. Acoustic scene classification in DCASE 2020 Challenge: Generalization across devices and low complexity solutions. In *arXiv:2005.14623 [Eess]*, November 2020. http://arxiv.org/abs/2005.14623.

[HMV20b] Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. TAU Urban Acoustic Scenes 2020 Mobile, Development dataset. https://zenodo.org/record/3819968, February 2020.

[HMV20c] Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. TAU Urban Acoustic Scenes 2020 Mobile, Evaluation dataset. https://zenodo.org/record/3685828, February 2020.

[HR17] David Miles Huber and Robert E Runstein. *Modern Recording Techniques*, volume 1 of *Audio Engineering Society Presents*. Routledge, Milton, ninth edition, 2017.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–80, December 1997.

[HS14] Kaiming He and Jian Sun. Convolutional Neural Networks at Constrained Time Cost. *arXiv:1412.1710 [cs]*, December 2014. http://arxiv.org/abs/1412.1710.

[HSA+17] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-Excitation Networks. *CVPR 2018*, September 2017. https://arxiv.org/abs/1709.01507v4.

[HYX+20] Hu Hu, Chao-Han Huck Yang, Xianjun Xia, Xue Bai, Xin Tang, Yajian Wang, Shutong Niu, Li Chai, Juanjuan Li, Hongning Zhu, Feng Bao, Yuanjun Zhao, Sabato Marco Siniscalchi, Yannan Wang, Jun Du, and Chin-Hui Lee. Device-Robust Acoustic Scene Classification Based on Two-Stage Categorization and Data Augmentation. Technical report, DCASE2020 Challenge, June 2020. http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Hu_114.pdf.

[HZRS16a] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. *arXiv:1603.05027 [cs]*, July 2016. http://arxiv.org/abs/1603.05027.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, June 2015. http://proceedings.mlr.press/v37/ioffe15.html.

[JH13] Navdeep Jaitly and Geoffrey E Hinton. Vocal Tract Length Perturbation (VTLP) improves speech recognition. *Proceedings of the 30th International Conference on Machine Learning*, page 5, 2013.

[Jia08] Jing Jiang. A Literature Survey on Domain Adaptation of Statistical Classifiers, 2008.

[Jie20] Liu Jie. Acoustic Scene Classification with Residual Networks and Attention Mechanism. Technical report, DCASE2020 Challenge, June 2020. http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Jie_19.pdf.

[JJS93] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, October 1993.

[JKLK17] Youngmoon Jung, Younggwan Kim, Hyungjun Lim, and Hoirin Kim. Linear-scale filterbank for deep neural network-based voice activity detection. In *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, pages 1–5, November 2017.

[JZS15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An Empirical Exploration of Recurrent Network Architectures. *Proceedings of the 32nd International Conference on Machine Learning*, page 9, 2015.

[KC20] Jin-Yeol Kwak and Yong-Joo Chung. Sound Event Detection Using Derivative Features in Deep Neural Networks. *Applied Sciences*, 10(14):4911, January 2020. https://www.mdpi.com/2076-3417/10/14/4911.

[KCH+19] Khaled Koutini, Shreyan Chowdhury, Verena Haunschmid, Hamid Eghbal-zadeh, and Gerhard Widmer. Emotion and Theme Recognition in Music with Frequency-Aware RF-Regularized CNNs. *arXiv:1911.05833 [cs, eess]*, October 2019. http://arxiv.org/abs/1911.05833.

[KEDW19] Khaled Koutini, Hamid Eghbal-zadeh, Matthias Dorfer, and Gerhard Widmer. The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, A Coruna, Spain, September 2019. IEEE. https://ieeexplore.ieee.org/document/8902732/.

[KEW19a] Khaled Koutini, Hamid Eghbal-zadeh, and Gerhard Widmer. CP-JKU Submissions to DCASE'19: Acoustic Scene Classification and Audio Tagging with Receptive-Field-Regularized CNNs. Technical report, DCASE2019 Challenge, July 2019.

[KEW19b] Khaled Koutini, Hamid Eghbal-zadeh, and Gerhard Widmer. Receptive-Field-Regularized CNN Variants for Acoustic Scene Classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, pages 124–128. New York University, 2019. http://hdl.handle.net/2451/60742.

[KHEW20] Khaled Koutini, Florian Henkel, Hamid Eghbal-zadeh, and Gerhard Widmer. CP-JKU Submissions to DCASE'20: Low-Complexity Cross-Device Acoustic Scene Classification with RF-Regularized CNNs. Technical report, DCASE2020 Challenge, June 2020. http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Koutini_142.pdf.

[KL10] Sunil Kumar Kopparapu and M. Laxminarayana. Choice of Mel filter bank in computing MFCC of a resampled speech. In *10th International Conference*

on Information Science, Signal Processing and their Applications (ISSPA 2010), pages 121–124, May 2010.

[KL19] Khaled Koutini and Bernhard Lehner. Acoustic scene classification with reject option based on resnets. Technical report, DCASE2019 Challenge, June 2019.

[Kla06] Anssi Klapuri. *Introduction to Music Transcription*, pages 3–20. Springer-Verlag New York Inc, 01 2006.

[KNS+19] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo K. Rohde. Generalized Sliced Wasserstein Distances. *arXiv:1902.00434 [cs, stat]*, February 2019. http://arxiv.org/abs/1902.00434.

[Kos19] Michał Kosmider. CALIBRATING NEURAL NETWORKS FOR SECONDARY RECORDING DEVICES. *Detection and Classification of Acoustic Scenes and Events 2019*, page 3, 2019.

[Koś20] Michał Kośmider. Spectrum Correction: Acoustic Scene Classification with Mismatched Recording Devices. In *Interspeech 2020*. ISCA, oct 2020. https://doi.org/10.21437/interspeech.2020-3088.

[Kri13] Johannes D Krijnders. A TONE-FIT FEATURE REPRESENTATION FOR SCENE CLASSIFICATION. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, page 3, 2013.

[KS16] Peter Knees and Markus Schedl. Music similarity and retrieval : An introduction to audio- and web-based strategies. In *Music Similarity and Retrieval : An Introduction to Audio- and Web-Based Strategies*, The Information Retrieval Series 36. Springer Berlin Heidelberg Imprint: Springer, Berlin, Heidelberg, 2016. 10.1007/978-3-662-49722-7.

[KSGG19] Chanwoo Kim, Minkyu Shin, Abhinav Garg, and Dhananjaya Gowda. Improved Vocal Tract Length Perturbation for a State-of-the-Art End-to-End Speech Recognition System. In *Proc. Interspeech 2019*, pages 739–743, 09 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

[KY14] Takumi Kobayashi and Jiaxing Ye. Acoustic feature extraction by statistics based local binary pattern for environmental sound classification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3052–3056, May 2014.

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, December 1989.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. https://www.nature.com/articles/nature14539.

[Ler12] Alexander Lerch. An introduction to audio content analysis : Applications in signal processing and music informatics. In *An Introduction to Audio Content Analysis : Applications in Signal Processing and Music Informatics*. Wiley, Hoboken, New Jersey, 2012. 10.1002/9781118393550.

[LGG+17] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, October 2017.

[LH09] Lie Lu and Alan Hanjalic. Audio Representation. In LING LIU and M. TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 160–167. Springer US, Boston, MA, 2009. https://doi.org/10.1007/978-0-387-39940-9_1442.

[LHX+19] Zhitong Li, Yuanbo Hou, Xiang Xie, Shengchen Li, Liqiang Zhang, Shixuan Du, and Wei Liu. Multi-level Attention Model with Deep Scattering Spectrum for Acoustic Scene Classification. In *2019 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 396–401, July 2019.

[LJSL20] Yingzi Liu, Shengwang Jiang, Chuang Shi, and Huiyong Li. Acoustic Scene Classification Using Ensembles of Deep Residual Networks and Spectrogram Decompositions. Technical report, DCASE2020 Challenge, June 2020. http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Liu_64.pdf.

[LLUZ16] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4905–4913, Red Hook, NY, USA, December 2016. Curran Associates Inc.

[LRM21] Valliappa Lakshmanan, Sara Robinson, and Michael Munn. Machine learning design patterns : Solutions to common challenges in data preparation, model building, and MLOps. In *Machine Learning Design Patterns : Solutions to Common Challenges in Data Preparation, Model Building, and MLOps*. O'Reilly, Beijing, 1st edition. edition, 2021.

[LS99] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999. https://www.nature.com/articles/44565.

[LTR17] Henry W. Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *J Stat Phys*, 168(6):1223–1247, September 2017. http://arxiv.org/abs/1608.08225.

[LWC98] Zhu Liu, Yao Wang, and Tsuhan Chen. Audio Feature Extraction And Analysis For Scene Segmentation And Classification. *Journal of VLSI Signal Processing*, 20, April 1998.

[LWL19] Mingle Liu, Wucheng Wang, and Yanxiong Li. THE SYSTEM FOR ACOUSTIC SCENE CLASSIFICATION USING RESNET, 2019.

[Lyo10] Richard F. Lyon. Machine Hearing: An Emerging Field [Exploratory DSP]. *IEEE Signal Processing Magazine*, 27(5):131–139, September 2010.

[Lyo17] Richard F. Lyon. *Human and Machine Hearing: Extracting Meaning from Sound*. Cambridge University Press, Cambridge, 2017. https://www.cambridge.org/core/books/human-and-machine-hearing/3660166B40020EE587D94BB7A309FC12.

[MAGH14] Janvier Maxime, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. Sound representation and classification benchmark for domestic robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6285–6292, May 2014.

[MC01] Danilo Mandic and Jonathon Chambers, 2001. https://onlinelibrary.wiley.com/doi/abs/10.1002/047084535X.ch1.

[McD18] Mark D. McDonnell. Training wide residual networks for deployment using a single bit for each weight. *arXiv:1802.08530 [cs, stat]*, February 2018. http://arxiv.org/abs/1802.08530.

[Med] O'Reilly Media. Pattern Recognition and Machine Learning. https://learning.oreilly.com/library/view/pattern-recognition-and/9780120588305/.

[MG20] M. D. McDonnell and W. Gao. Acoustic Scene Classification Using Deep Residual Networks with Late Fusion of Separated High and Low Frequency Paths. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 141–145, May 2020.

[MHB+18] Annamaria Mesaros, Toni Heittola, Emmanouil Benetos, Peter Foster, Mathieu Lagrange, Tuomas Virtanen, and Mark D. Plumbley. Detection and

Classification of Acoustic Scenes and Events: Outcome of the DCASE 2016 Challenge. *IEEE/ACM Trans. Audio Speech Lang. Process.*, 26(2):379–393, February 2018.

[MHD+17] Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Benjamin Elizalde, Ankit Shah, Bhiksha Raj, and Tuomas Virtanen. DCASE 2017 CHALLENGE SETUP: TASKS, DATASETS AND BASELINE SYSTEM. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, pages 85–92, 11 2017.

[MHV16] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132, Budapest, Hungary, August 2016. IEEE. `http://ieeexplore.ieee.org/document/7760424/`.

[MHV18a] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Acoustic Scene Classification: An Overview of Dcase 2017 Challenge Entries. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 411–415, September 2018.

[MHV18b] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. A multi-device dataset for urban acoustic scene classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pages 9–13, November 2018. `http://arxiv.org/abs/1807.09840`.

[MHV19] A. Mesaros, Toni Heittola, and T. Virtanen. Acoustic Scene Classification in DCASE 2019 Challenge: Closed and Open Set Classification and Data Mismatch Setups. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, pages 164–168, New York University, NY, USA, October 2019.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science : Artificial Intelligence. McGraw-Hill, New York, 1997.

[MKHS14] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional Kernel Networks. *Advances in neural information processing systems*, June 2014.

[MKS+20] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip H. S. Torr, and Puneet K. Dokania. Calibrating Deep Neural Networks using Focal Loss. In *Computing Research Repository*, October 2020. `http://arxiv.org/abs/2002.09437`.

[MLM09] Sheeraz Memon, Margaret Lech, and Namunu Maddage. Speaker Verification Based on Different Vector Quantization Techniques with Gaussian Mixture

Models. In *2009 Third International Conference on Network and System Security*, pages 403–408, October 2009.

[MMM+21] Alessandro Maccagno, Andrea Mastropietro, Umberto Mazziotta, Michele Scarpiniti, Yong-Cheol Lee, and Aurelio Uncini. A CNN Approach for Audio Classification in Construction Sites. In Anna Esposito, Marcos Faundez-Zanuy, Francesco Carlo Morabito, and Eros Pasero, editors, *Progresses in Artificial Intelligence and Neural Systems*, volume 184, pages 371–381. Springer Singapore, Singapore, 2021. http://link.springer.com/10.1007/978-981-15-5093-5_33.

[Moh14] Abdel-rahman Mohamed. *Deep Neural Network Acoustic Models for ASR*. PhD thesis, University of Toronto, 2014.

[MOM12] Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors. *Neural Networks: Tricks of the Trade: Second Edition*, volume 7700 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. http://link.springer.com/10.1007/978-3-642-35289-8.

[MPHK17] Seongkyu Mun, Sangwook Park, David Han, and Hanseok Ko. Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane. Technical report, DCASE2017 Challenge, September 2017.

[MPR05] Shie Mannor, Dori Peleg, and Reuven Rubinstein. The cross entropy method for classification. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 561–568, New York, NY, USA, 2005. Association for Computing Machinery. https://doi.org/10.1145/1102351.1102422.

[MPSN01] S. Molau, M. Pitz, R. Schluter, and H. Ney. Computing Mel-frequency cepstral coefficients on the power spectrum. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 1, pages 73–76 vol.1, May 2001.

[MRL+15] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. Librosa: Audio and Music Signal Analysis in Python. In *Python in Science Conference*, pages 18–24, January 2015.

[MS13] Neil Middleton and Richard D Schneeman. *Heroku*. O'Reilly, Farnham, 1st edition. edition, 2013.

[MSHB21] Puneet Mangla, Vedant Singh, Shreyas Havaldar, and Vineeth Balasubramanian. On the benefits of defining vicinal distributions in latent space. *Pattern Recognition Letters*, 152:382–390, December 2021. https://linkinghub.elsevier.com/retrieve/pii/S0167865521003755.

[MTX+16] Erik Marchi, Dario Tonelli, Xinzhou Xu, Fabien Ringeval, Jun Deng, Stefano Squartini, and Björn Schuller. The up system for the 2016 DCASE challenge using deep recurrent neural network and multiscale kernel subspace learning. Technical report, DCASE2016 Challenge, September 2016. https://dcase.community/documents/challenge2016/technical_reports/DCASE2016_Marchi_1019.pdf.

[MZB10] Dalibor Mitrovic, Matthias Zeppelzauer, and Christian Breiteneder. Features for Content-Based Audio Retrieval. *Advances in Computers*, 78:71–150, January 2010.

[NPK20a] T. Nguyen, F. Pernkopf, and M. Kosmider. Acoustic Scene Classification for Mismatched Recording Devices Using Heated-Up Softmax and Spectrum Correction. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 126–130, May 2020.

[NPK20b] Truc Nguyen, Franz Pernkopf, and Michal Kosmider. Acoustic Scene Classification for Mismatched Recording Devices Using Heated-Up Softmax and Spectrum Correction. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 126–130, May 2020.

[O'S99] Douglas O'Shaughnessy. Speech communications: Human and machine. In *Speech Communications*. John Wiley & Sons, second edition, 1999.

[OSB99] Alan V Oppenheim, Ronald W Schafer, and John R Buck. *Discrete-Time Signal Processing*. Prentice Hall Signal Processing Series. Prentice Hall Internat., London, 2. ed., internat. ed.. edition, 1999.

[PBW19] S. S. R. Phaye, E. Benetos, and Y. Wang. SubSpectralNet – Using Subspectrogram Based Convolutional Neural Networks for Acoustic Scene Classification. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 825–829, May 2019.

[PCK+19] Huy Phan, Oliver Chén, Philipp Koch, Maarten de Vos, Ian Mcloughlin, and Alfred Mertins. Spatio-Temporal Attention Pooling for Audio Scene Classification. In *Interspeech 2019*, pages 3845–3849, September 2019.

[PCZ+19] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *Interspeech 2019*, pages 2613–2617, September 2019. http://arxiv.org/abs/1904.08779.

[PEE+19] Paul Primus, Hamid Eghbal-zadeh, David Eitelsebner, Khaled Koutini, Andreas Arzt, and Gerhard Widmer. Exploiting Parallel Audio Recordings to Enforce Device Invariance in CNN-based Acoustic Scene Classification. *arXiv:1909.02869 [cs, eess, stat]*, September 2019. http://arxiv.org/abs/1909.02869.

[PG18] Chandrasekhar Paseddula and Suryakanth V. Gangashetty. DNN based Acoustic Scene Classification using Score Fusion of MFCC and Inverse MFCC. In *2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS)*, pages 18–21, December 2018.

[PHM+17] Huy Phan, Lars Hertel, Marco Maass, Philipp Koch, Radoslaw Mazur, and Alfred Mertins. Improved Audio Scene Classification Based on Label-Tree Embeddings and Convolutional Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1278–1290, June 2017.

[Pic15] Karol J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, September 2015.

[PKH+17] Huy Phan, Philipp Koch, Lars Hertel, Marco Maass, Radoslaw Mazur, and Alfred Mertins. CNN-LTE: A class of 1-X pooling convolutional neural networks on label tree embeddings for audio scene classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 136–140, March 2017.

[PKK+17] Huy Phan, Philipp Koch, Fabrice Katzberg, Marco Maass, Radoslaw Mazur, and Alfred Mertins. Audio Scene Classification with Deep Recurrent Neural Networks. *arXiv:1703.04770 [cs]*, June 2017. http://arxiv.org/abs/1703.04770.

[PLV+19] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-yiin Chang, and Tara Sainath. Deep Learning for Audio Signal Processing. *IEEE J. Sel. Top. Signal Process.*, 13(2):206–219, May 2019. http://arxiv.org/abs/1905.00078.

[PNJ19a] Sergio R. M. Penedo, Marcio Lobo Netto, and João F. Justo. Designing digital filter banks using wavelets. *EURASIP Journal on Advances in Signal Processing*, 2019(1):33, July 2019. https://doi.org/10.1186/s13634-019-0632-6.

[PNJ19b] Sergio R. M. Penedo, Marcio Lobo Netto, and João F. Justo. Designing digital filter banks using wavelets. *EURASIP Journal on Advances in Signal Processing*, 2019(1):33, July 2019. https://doi.org/10.1186/s13634-019-0632-6.

[Pri90] Roland Priemer. *Introductory Signal Processing*. WORLD SCIENTIFIC, 1990. https://www.worldscientific.com/doi/abs/10.1142/0864.

[PTK+02a] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa. Computational auditory scene recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–1941–II–1944, May 2002.

[PTK+02b] Vesa Peltonen, Juha Tuomi, Anssi Klapuri, Jyri Huopaniemi, and Timo Sorsa. Computational auditory scene recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–1941–II–1944, 2002. https://www.academia.edu/21858078/Computational_auditory_scene_recognition.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis*, 115(3):211–252, December 2015. https://doi.org/10.1007/s11263-015-0816-y.

[RG15] Alain Rakotomamonjy and Gilles Gasso. Histogram of Gradients of Time–Frequency Representations for Audio Scene Classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):142–153, January 2015.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. https://www.nature.com/articles/323533a0.

[Rid18] Imad Rida. Feature Extraction for Temporal Signal Recognition: An Overview. In *arXiv:1812.01780 [Cs, Eess]*, December 2018. http://arxiv.org/abs/1812.01780.

[Rip96] Brian D Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[RS78] Lawrence R Rabiner and Ronald W Schafer. *Digital Processing of Speech Signals*. Prentice-Hall Signal Processing Series. Prentice-Hall, Englewood Cliffs, NJ, 1978.

[RSN13] Gaël Richard, Shiva Sundaram, and Shrikanth Narayanan. An Overview on Perceptually Motivated Audio Indexing and Classification. *Proceedings of the IEEE*, 101(9):1939–1954, September 2013.

[Rub97] Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, May

1997. https://www.sciencedirect.com/science/article/pii/S0377221796003852.

[Rud17] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*, June 2017. http://arxiv.org/abs/1609.04747.

[RV13] Monika Rychtarikova and Gerrit Vermeir. Soundscape categorization on the basis of objective acoustical parameters. *Applied Acoustics*, 74, February 2013.

[SAW95] Bill Schilit, Norman Adams, and Roy Want. Context-Aware Computing Applications. *Proc. of The IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, January 1995.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning : From theory to algorithms. In *Understanding Machine Learning : From Theory to Algorithms*. Cambridge Univ. Press, Cambridge, 1. publ.. edition, 2014.

[SB17] J. Salamon and J. P. Bello. Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters*, 24(3):279–283, March 2017.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015. https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[SGB+15] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D. Plumbley. Detection and Classification of Acoustic Scenes and Events. *IEEE Trans. Multimedia*, 17(10):1733–1746, October 2015.

[SGG20] Jivitesh Sharma, Ole-Christoffer Granmo, and Morten Goodwin. Environment Sound Classification Using Multiple Feature Channels and Attention Based Deep Convolutional Neural Network. In *Interspeech 2020*, pages 1186–1190. ISCA, October 2020.

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks. *arXiv:1505.00387 [cs]*, November 2015. http://arxiv.org/abs/1505.00387.

[Sha49] C.E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, January 1949.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. http://jmlr.org/papers/v15/srivastava14a.html.

[SIVA16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261 [cs]*, August 2016. http://arxiv.org/abs/1602.07261.

[SLM+22] Jianyuan Sun, Xubo Liu, Xinhao Mei, Jinzheng Zhao, Mark D. Plumbley, Volkan Kılıç, and Wenwu Wang. Deep Neural Decision Forest for Acoustic Scene Classification. http://arxiv.org/abs/2203.03436, March 2022.

[SM97] Nitin Sawhney and Pattie Maes. Situational Awareness from Environmental Sounds. *Situational Awareness from Environmental Sounds*, pages 1–7, 1997.

[Smi11] Julius Smith. *Spectral Audio Signal Processing*. http://ccrma.stanford.edu/~jos/sasp/, two thousand, eleventh edition, 2011. https://ccrma.stanford.edu/~jos/sasp/sasp-citation.html.

[Som19] Panu Somervuo. Time–frequency warping of spectrograms applied to bird sound analyses. *Bioacoustics*, 28(3):257–268, May 2019. https://doi.org/10.1080/09524622.2018.1431958.

[SP14] Dan Stowell and Mark Plumbley. Large-scale analysis of frequency modulation in birdsong databases. *Methods in Ecology and Evolution*, 5:901–912, September 2014.

[SPJL20] Sangwon Suh, Sooyoung Park, Youngho Jeong, and Taejin Lee. Designing Acoustic Scene Classification Models with CNN Variants. Technical report, DCASE2020 Challenge, June 2020. http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Suh_101.pdf.

[SPL06] Sigurdur Sigurdsson, Kaare Brandt Petersen, and Tue Lehn-Schiøler. Mel Frequency Cepstral Coefficients: An Evaluation of Robustness of MP3 Encoded Music. *ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October 2006, Proceedings*, page 4, 2006.

[SR00] L.K. Saul and M.G. Rahim. Maximum likelihood and minimum classification error factor analysis for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 8(2):115–125, March 2000.

[SSB+18] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent Advances in Recurrent Neural Networks. *arXiv:1801.01078 [cs]*, February 2018. http://arxiv.org/abs/1801.01078.

[SSP03] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 958–963, August 2003.

[SSS20] Susanta Sarangi, Md Sahidullah, and Goutam Saha. Optimization of data-driven filterbank for automatic speaker verification. *Digital Signal Processing*, 104:102795, sep 2020. https://doi.org/10.1016/j.dsp.2020.102795.

[STIM19] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How Does Batch Normalization Help Optimization? *arXiv:1805.11604 [cs, stat]*, April 2019. http://arxiv.org/abs/1805.11604.

[SV40] S. S. Stevens and J. Volkmann. The Relation of Pitch to Frequency: A Revised Scale. *The American Journal of Psychology*, 53(3):329–353, 1940. https://www.jstor.org/stable/1417526.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*, December 2014. http://arxiv.org/abs/1409.3215.

[SVN37] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937. https://doi.org/10.1121/1.1915893.

[SWY+15] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[SZ15] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, April 2015. http://arxiv.org/abs/1409.1556.

[TA03] J.K. Thompson and L.E. Atlas. A non-uniform modulation transform for audio coding with increased time resolution. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 5, pages V–397, April 2003.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

[TGA+21a] Sabu M. Thampi, Erol Gelenbe, Mohammed Atiquzzaman, Vipin Chaudhary, and Kuan-Ching Li, editors. *Advances in Computing and Network*

Communications: Proceedings of CoCoNet 2020, Volume 2, volume 736 of *Lecture Notes in Electrical Engineering*. Springer Singapore, Singapore, 2021. https://link.springer.com/10.1007/978-981-33-6987-0.

[TGA+21b] Sabu M. Thampi, Erol Gelenbe, Mohammed Atiquzzaman, Vipin Chaudhary, and Kuan-Ching Li, editors. *Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 2*, volume 736 of *Lecture Notes in Electrical Engineering*. Springer Singapore, Singapore, 2021. https://link.springer.com/10.1007/978-981-33-6987-0.

[TLX+19] Guichen Tang, Ruiyu Liang, Yue Xie, Yongqiang Bao, and Shijia Wang. Improved Convolutional Neural Networks for Acoustic Event Classification. *Multimed Tools Appl*, 78(12):15801–15816, June 2019. https://doi.org/10.1007/s11042-018-6991-4.

[TM20] O. K. Toffa and M. Mignotte. Environmental Sound Classification Using Local Binary Pattern and Audio Features Collaboration. *IEEE Transactions on Multimedia*, pages 1–1, 2020.

[TYMO16] Gen Takahashi, Takeshi Yamada, Shoji Makino, and Nobutaka Ono. Acoustic scene classification using deep neural network and frame-concatenated acoustic feature. Technical report, DCASE2016 Challenge, September 2016.

[TYOM17] Gen Takahashi, Takeshi Yamada, Nobutaka Ono, and Shoji Makino. Performance evaluation of acoustic scene classification using DNN-GMM and frame-concatenated acoustic features. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1739–1743, December 2017.

[Udo08] Zölzer Udo. *Digital Audio Signal Processing, Second Edition*. Wiley, Chichester, U.K., 2nd ed. edition, 2008. https://learning.oreilly.com/library/view/digital-audio-signal/9780470997857/.

[VPE18] Tuomas Virtanen, Mark D. Plumbley, and Dan Ellis, editors. *Computational Analysis of Sound Scenes and Events*. Springer International Publishing, Cham, 2018. http://link.springer.com/10.1007/978-3-319-63450-0.

[vRV+10] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther. Process mining: A two-step approach to balance between underfitting and overfitting. *Softw Syst Model*, 9(1):87–111, January 2010. http://link.springer.com/10.1007/s10270-008-0106-z.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You

Need. *arXiv:1706.03762 [cs]*, December 2017. http://arxiv.org/abs/1706.03762.

[VW16] Toan H. Vu and Jia-Ching Wang. Acoustic scene and event recognition using recurrent neural networks. Technical report, DCASE2016 Challenge, September 2016. https://dcase.community/documents/challenge2016/technical_reports/DCASE2016_Vu_1018.pdf.

[WB06] D. Wang and G.J. Brown. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. IEEE press, January 2006.

[WBKW96] E. Wold, T. Blum, D. Keislar, and J. Wheaten. Content-based classification, search, and retrieval of audio. *IEEE MultiMedia*, 3(3):27–36, 1996.

[Wid61] B. Widrow. Statistical analysis of amplitude-quantized sampled-data systems. *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, 79(6):555–568, January 1961.

[Win20] Eric Windmill. *Flutter in Action*. Manning Publications, Shelter Island, NY, 1st edition. edition, 2020.

[WLH00] Yao Wang, Zhu Liu, and Jin-Cheng Huang. Multimedia content analysis-using both audio and visual clues. *IEEE Signal Processing Magazine*, 17(6):12–36, November 2000.

[WPLK18] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional Block Attention Module. *arXiv:1807.06521 [cs]*, July 2018. http://arxiv.org/abs/1807.06521.

[WS20] Shefali Waldekar and Goutam Saha. Two-level fusion-based acoustic scene classification. *Applied Acoustics*, 170:107502, December 2020. https://www.sciencedirect.com/science/article/pii/S0003682X2030606X.

[Wu17] Jianxin Wu. Introduction to Convolutional Neural Networks. *National Key Lab for Novel Software Technology*, page 31, 2017.

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. *arXiv:1611.05431 [cs]*, April 2017. http://arxiv.org/abs/1611.05431.

[XHWP16] Yong Xu, Qiang Huang, Wenwu Wang, and Mark D. Plumbley. Hierarchical learning for dnn-based acoustic scene classification, 2016. https://arxiv.org/abs/1607.03682.

[XLY+18] Jun-Xiang Xu, Tzu-Ching Lin, Tsai-Ching Yu, Tzu-Chiang Tai, and Pao-Chi Chang. Acoustic Scene Classification Using Reduced MobileNet Architecture. In *2018 IEEE International Symposium on Multimedia (ISM)*, pages 267–270, December 2018.

[XZ19] Jie Xie and Mingying Zhu. Investigation of acoustic and visual features for acoustic scene classification. *Expert Systems with Applications*, 126:20–29, July 2019. https://www.sciencedirect.com/science/article/pii/S0957417419300661.

[YCT18] Liping Yang, Xinxing Chen, and Lianjie Tao. ACOUSTIC SCENE CLASSIFICATION USING MULTI-SCALE FEATURES Technical Report. https://www.semanticscholar.org/paper/ACOUSTIC-SCENE-CLASSIFICATION-USING-MULTI-SCALE-Yang-Chen/fcaf5c171c474d161d5749bd2ec9ae4f2588e763, 2018.

[YEG+15] Steve Young, Gunnar Evermann, M.J.F. Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, James Odell, Dave Ollason, Daniel Povey, Anton Ragni, Valtcho Valtchev, Philip Woodland, and Chao Zhang. *The HTK Book (Version 3.5a)*. Cambridge University Engineering Department, December 2015.

[YKM17] Jiaxing Ye, Takumi Kobayashi, and Masahiro Murakawa. Urban sound event classification based on local and global features aggregation. *Applied Acoustics*, 117:246–256, February 2017. https://linkinghub.elsevier.com/retrieve/pii/S0003682X16302274.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv:1611.03530 [cs]*, February 2017. http://arxiv.org/abs/1611.03530.

[ZC88] Zhou and Chellappa. Computation of optical flow using a neural network. In *IEEE 1988 International Conference on Neural Networks*, pages 71–78 vol.2, July 1988.

[ZC18] Alice Zheng and Amanda Casari. Feature engineering for machine learning : Principles and techniques for data scientists. In *Feature Engineering for Machine Learning : Principles and Techniques for Data Scientists*. O'Reilly, Beijing, first edition.. edition, 2018.

[ZCDL18] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond Empirical Risk Minimization. In *Computing Research Repository*, April 2018. http://arxiv.org/abs/1710.09412.

[ZK17] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv:1605.07146 [cs]*, June 2017. http://arxiv.org/abs/1605.07146.

[ZPB+16] Y. Zhang, M. Pezeshki, Philemon Brakel, Saizheng Zhang, César Laurent, Yoshua Bengio, and Aaron C. Courville. Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks. In *INTERSPEECH*, pages 410–414, 2016.

[ZXWY18] Huimin Zhao, Huang Xianglin, Liu Wei, and Lifang Yang. Environmental sound classification based on feature fusion. *MATEC Web of Conferences*, 173:03059, January 2018.

[ZZK+20] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random Erasing Data Augmentation. *AAAI*, 34(07):13001–13008, April 2020. https://ojs.aaai.org/index.php/AAAI/article/view/7000.