Andreas Schmidhofer, BSc

Dataset Generation Guideline for Acoustic Transport Mode Detection

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Roman Kern

Interactive Systems and Data Science
Head: Univ.-Prof. Dr. Stefanie Lindstaedt

Graz, December 2018

This document is set in Palatino, compiled with pdfLATEX2e and `Biber`.

The LATEX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date   Signature

Abstract

Transport mode detection (TMD) is the process of recognizing the means of transportation (such as walking, cycling, driving, taking a bus, riding a metro etc.) by a given sensory input. When this input consists exclusively of audio data then it is called acoustic TMD.
This thesis recherches and presents the methodology for creating datasets, which fulfill all critical requirements for the highly complex task of acoustic TMD. It provides a step-by-step guideline on what needs to be considered when designing, producing and enhancing the dataset.
In order to compile this guideline a recording application was developed, a 9-class dataset with over 245 hours of recordings was created, and experiments were run using this dataset. Those experiments aimed to shed light onto the required number and diversity of recordings, the ideal number of total classes, what is an appropriate sample length, how to remove samples of low quality and which evaluation strategy should be used. Finally, existing external datasets were used to evaluate the classification capabilities.
With the help of our findings it should be easier for future projects to create their own acoustic datasets, especially for TMD.

Contents

List of Figures

1. Introduction

Since their introduction a decade ago, smartphones have rapidly gained in popularity. Nowadays most people own a smartphone. These devices are usually taken everywhere their owner goes. This opens up a market for context sensitive applications.

In order for those applications to work, the phone has to be able to make predictions about its environment. Since smartphones are equipped with several sensors (accelerometers, GPS, compass etc.) there are many possibilities how this can be done. One possibility is to use audio data which can be gained form the microphone.

When using audio data only the task to determine which environment a given recording belongs to is called acoustic scene classification (ASC). ASC can be a form of transport mode detection (TMD) when the environments (the acoustic scenes) are all means of transport (for example bus or train etc.). One use-case of TMD is to provide more relevant information to the user for example by providing traffic information when in a car or inform about delays when in trains. Another use case is to trace back the users taken path by using TMD in combination with real place detection.

Phones as well as computers have been becoming more and more powerful which has allowed advances to be made in many fields. Among them is the field of machine learning, where deep neural networks (DNNs) have become widely adopted even though they have their limitations and are not safe against adversaries [22]. Convolutional Neural Networks (CNNs) are a type of DNNs which have been proven to work for a range of applications such as image classification. They are also applied to ASC and TMD tasks. In general they are very robust and can handle a lot of noise but they also require large amounts of training data.

We developed Android application to ease the creation of a dataset for acoustic TMD. The source code of this application is publicly available online[1]. The created dataset has 9 classes and consists of a total of 245 hours of recordings. The recordings were made in different cities, by different people, using different smartphones. After its creation the dataset has been analyzed and processed to obtain better results in a TMD task.

The EU has recently passed a law called General Data Protection Regulation (GDPR) which took effect in May 2018. This makes it very hard for any EU based research group to publish their acoustic TMD datasets since they usually contain voices (or other identifiable sounds) from other people. A dataset that can be published would have to be post-processed to remove those sounds or to be created in a controlled environment (for instance empty buses). However, both of those strategies would result in a dataset that does not represent a real world environment. Thus strategies on how to create an acoustic TMD dataset are discussed but the dataset will not be published.

The following research question is the basis for this work:

What is an effective way to design and create a dataset that can be used for acoustic transport mode detection?

The main focus of this thesis is to provide a set of guidelines that are useful when it comes to creating an audio dataset for transport mode detection. This is done in two phases. The first phase is the recording phase where the audio data is gathered. It tries to answer the following questions: What can be done to simplify the recording procedure? How many different classes should be chosen? How much data should be acquired? What are the quality requirements of the recordings? How diverse do the recordings need to be? The second phase is the processing phase where the given data is analyzed and filtered to yield better results. It deals with questions such as: How to detect mislabeled recordings? What is an appropriate sample size to split the recordings into? How to detect and remove samples that do not represent the class? How to handle high inter-class variability, low intra-class variability and ambiguous classes?

[1] `https://github.com/aschmidhofer/TMDAudioRecorder`

Chapter 2 provides an outline of previous work done on the subjects of dataset creation as well as transport mode detection. Chapter 3 discusses the principles and implementations used. Chapter 4 contains the results and their evaluation. The findings and lessons learned are compiled into a guide which is presented in chapter 5. Finally, a conclusion can be found in chapter 6.

2. Related Work

This chapter discusses previous work as well as the state of the art of the different aspects of the thesis. Background on the topics of transport mode detection (TMD), acoustic scene classification (ASC) and dataset creation is provided. The two search engines *Google Scholar*[1] and *Scopus*[2] have been used to find publications on those topics. The selection given in this chapter is by no means exhaustive but should include the most relevant publications.

2.1. Acoustic Scene Classification

In 1995 Saint-Arnaud [27] described longterm audio events such as rain or the sound of crowds as *Sound Textures*. Their thesis tried to classify such *Sound Textures* using a clustering approach. Their main focus however was a comparison with the human perception of Sound Textures.

Sawhney from the Speech Interface Group of the MIT Media Lab [28] worked on classifying environmental sounds. In their study from 1997 they used frequency bands from filter-banks as features and nearest neighbor classification, as a result they were able to obtain a 68% accuracy on a 5 class problem.

In 2003 Eronen et al. [7] used mel-frequency cepstral coefficients (MFCCs) as feature sets and a 2-state fully connected hidden markov model (HMM) for classification. They reached recognition rates of 61% on 18 classes and 86% when using 6 higher level categories. They also conducted a study on human recognition ability which outperformed their system by reaching 64% and

[1]https://scholar.google.com/
[2]https://www.scopus.com/

90% respectively. MFCCs are still widely used in ASC tasks nowadays because they provide a good representation of the spectral properties of the audio signal.

Additional to the MFCCs and HMMs Eronen et al. [6] performed linear data-driven feature-transformations such as PCA and ICA to increase classification accuracy. Furthermore they tested different sequence lengths and found that they converge after 30s to 60s. On a data set with 24 classes they achieved a maximum of 72% accuracy. A 50% accuracy can already be reached with a sequence length of 400ms. Humans were able to classify 69% with an average reaction time of 14s. Those findings are relevant to us because finding an appropriate sequence length is one of the sub-tasks of this work.

A convolutional neural network consists of one ore more pairs of convolution and max-pooling layers and one or more fully connected layers that perform the classification in the end. Abdel-Hamid et al [1] have used principles of CNNs for speech recognition which is a similar problem as ASC. They have used a hybrid model where CNN handles the frequency domain and a HMM the time domain. The results showed an improvement in recognition performance which lead to further studies being conducted on CNNs on audio data.

In 2013 Sainath et al. [26] analyzed which architecture and feature vectors produce the best results when performing speech recognition with CNNs. They found that two convolutional and four fully connected layers, as well as a pooling size of three, works best for their tasks. Their CNN showed a relative improvement of 13-30% compared to a baseline GMM. The two CNN layers seem to work well as this architecture of CNNs is still found in state of the art systems.

The key to a good study is its reproducibility. Vandewalle et al. [34] suggest that data as well as algorithms should be made publicly available. Following this initiative, Stowell et al. [33] introduced a challenge in detection and classification of acoustic scenes and events (DCASE). The first DCASE challenge was released in 2013 providing a dataset as well as baseline algorithms. This was important for the field of ASC, since a research community formed around this challenge. Most of the submissions to the challenge used an SVM as the final classifier.

The DCASE challenge has become an annual event with repetitions in the years 2016, 2017 and 2018. It always contains several tasks, one of which is always in the field of ASC. In 2016, most of the submissions which solved the ASC problem used MFCCs as features and CNNs as classifiers [20]. The best performing approach was delivered by Eghbal-Zadeh et al. [4], who fused a deep convolutional network on spectrogram data with binaural i-vectors on MFCC features. It should be mentioned that several approached reached similar accuracy using traditional machine learning techniques like SVMs [15], but a trend towards CNNs can be observed.

The state of the art in ASC is demonstrated by research solving *task 1* of the DCASE 2017 [19]. The best performance on the validation set was achieved by the algorithm of Mun et al. [21], who used SVM and a fully connected neural network as classifiers. However, what had set them apart was the use of generative adversarial networks to generate additional training data. The other two approaches in the top 3 both used CNNs for classification. Han et al. [11] worked on log-mel energies whereas Weiping et al. [37] combined standard spectrograms with CQT spectrograms. Most approaches that perform well in the ASC challenge combine several features and/or classification strategies to achieve the best results.

As by the time of writing, the DCASE challenge 2018 is still ongoing. Therefore, the results have not yet been published. However, the baseline system which was released with the challenge [18] is a state-of-the-art ASC approach. As it is a comparatively light-weight algorithm, and the code is publicly available, it is used as the basis for classification in this work.

2.2. Transport Mode Detection

In the era before smartphones, transport mode detection was performed using GPS data only. Zheng et al. [40] used GPS enabled devices to collect data. From that data, they were able to gather information about the mode of transportation using decision trees. They distinguish between four modes: walking, car, bus and bike. The knowledge they obtained is used to improve geographic applications on the web.

Smartphones allow for access to various sensory data, which lead to most TMD procedures to use a combination of GPS and accelerometer data. Reddy et al. [25] used GPS and accelerometer to classify the modes of stationary, walking, running and motorized travel. Classification was performed using a decision tree followed by a discrete hidden markov model (DHMM), achieving 93.6% accuracy.
Widhalm et al. [38] generated features using GPS and accelerometer data. A combination of randomized ensemble of classifiers and a DHMM were used for classification. They differentiate between eight transport modes (bus, car, bike, tram, train, subway, walk and motorcycle) and report an accuracy of 78%.

A study conducted by Wang et al. [36] use only accelerometer input to predict the mode of transportation. They supported the following 6 classes: bike, bus, car, stationary, subway and walking.
Two years later Hemminki et al. [12] improved the classification, using accelerometer only for the following 7 classes: Stationary, walk, bus, train, metro, tram and car. They introduced features which better characterize acceleration and breaking patters of motorized transport. The mean precision and recall recorded were above 80% which means their approach outperformed those by Wang (around 60%) and Reddy (which reach 66% in a comparable setting).

One of the rare studies that uses audio for TMD was done by Lee et al. [16]. They developed a system called VehicleSense which uses accelerometer input to provide triggers for the classification performed with audio data. Their system claims to have an accuracy of 98.2% on a 5-class problem (stationary, walking, bus, taxi, subway). For sound classification a standard SVM-based machine learning approach is used. To improve the classification results. a pre-processing strategy is applied. The frequency space is transferred from the standard herz scale to a transformed scale. Several different scales have been analysed (including MFCC and LFCC) and the one providing best results (which they called *Steep*) was used as pre-processing. VehicleSense only requires samples of 0.5 seconds to perform a classification.

In their review [23] Prelipcean et al. put TMD papers into three groups: *Location Based Services* (LBS), *Transportation Science* (TSc) and *Human Geography*

(HG). The first group (LBS) tries to determine the mode of transportation in as close to real time as possible to provide the user with useful information. In those scenarios the device predicts the transportation mode based only on the data which is collected at the moment. Possible use cases would be to inform the user of relevant traffic jams, weather reports, or time tables for public transport. The second group (TSc) has its focus on reconstructing the mode of transportation taken given a recording of sensory input. The result are travel diaries, which can be used to study how people are traveling, and assess the impact of infrastructural changes. In the third group (HG), not only the transportation mode is predicted, but also segments with domain-specific semantics. One basic segmentation is whether the subject (be it person or vehicle) moves, or stands still.
As for our work, the classification problem is of type LBS, even though the actual implementation follows TSc strategies, since at the point of classification the recording has already been made. However, after the training phase the classification could be performed on live data, which would make it a LBS use case.
The paper also outlines the difficulty of generating a dataset (i.e. collecting data) and having a benchmark dataset which is agreed upon by the scientific community. The main complications mentioned are the magnitude of the collection, the willingness of users to record and share their data and the legal efforts to publish such a data set.

Fang et al. [8] improved upon previous techniques by introducing improved features for data recorded on accelerometer, magnetometer and gyroscope of a smartphone. They used three machine learning algorithms: decision trees, k-nearest neighbor (KNN) and support vector machines (SVM), and compared their performances. In addition to transport mode detection on ten classes (still, walking, running, biking, motorcycle, car, bus, metro, train, HSR), they also performed vehicle mode detection on the 6 vehicle classes. KNN performed best for vehicle mode detection and for TMD both KNN and SVM show comparable results.

Since there is no public benchmark data set available for TMD, it is difficult to determine which approach yields the best results. However, the segment-based approach proposed by Guvensan et al. [10] is among the overall best performing. Their algorithm takes input from accelerometer, gyroscope and magnetometer sensors of a smartphone. The novelty of their approach is to first distinguish between pedestrian activity (walking, stationary) and vehicle activity (bus, train, etc.). This allows them to get a high accuracy on the walking class, which is a key element for their post processing: the healing algorithm. Without post processing they reach a recall of 84.4% (which already outperforms the 83.6% by Fang et al. [8]), but after applying the health algorithm the recall is as high as 91.6%. The key idea of the healing algorithm is that the mode of transportation does not change rapidly, but there are usually longer periods on the same vehicle and usually during a change the person in walking. With those two assumptions the algorithm replaces misclassified windows. On their own dataset they were able to achieve an overall recall rate of 94.5%.

In 2017 Liang et al. [17] finally published a deep learning approach for TMD. They used a CNN with 6 convolution and max-pooling layers, one fully connected layer and one classification layer. Since they only used accelerometer input their data collection is comparatively lightweight compared to other approaches. An accuracy of 94.5% is achieved using this approach, which outperforms all traditional machine learning models. This demonstrates, once again, the potential of CNNs for TMD.

2.3. Dataset Creation

The book *Designing and Conducting Health System Research Projects* [35] has a whole chapter on data collection techniques. Even though it is written for medical research, the core concepts also apply to other fields. They state that there are different data collection techniques that can be used: using available information, observing, and interviewing. In the context of the creation of an audio dataset, one also has the choice to use available data from other research projects or extracted it from public data. Alternatively the data can be recorded in the field. As with conducting interviews, this is more expensive, but it can be tailored to the very needs of the research project. To keep costs low, but still gather enough data, a combination of both approaches can be used.
Furthermore, the book provides a list of sources of bias that might influence

data quality. First on the list are defective instruments. This also applies to audio data since recording devices have different directionality patterns, frequency responses, and might even add a specific level of noise. Second on the list is the observer bias, which states that the data collector will miss information that they are not interested in. Although this is less common when recording audio data, it is still possible that the person operating the microphone does not deem something worthy of recording, even though it would be crucial for the study. Third item on the list is the effect of the interview on the informant. In the context of an audio recording this would be the presence of the microphone, which might influence the people around it who realize they are being recorded. The final item is the information bias. Sometimes the data was created for a different purpose, or is incomplete, which results in a bias. This is important to note when available data is used, but also when it is newly created. One example in the context of TMD is that buses are more busy at rush hour, so if the recordings are only made during that time they will have a bias. Those four biases should be considered before a dataset is created.

Homburg et al. presented in [13] how to create an audio dataset from data publicly available on the Internet. They crawled about two thousand songs from Garageband[3], a website that allows artists to upload their music and others to download it for free. For each song, a 10 second representational sample was extracted at a random position. The songs were classified into 9 music genres. Additionally to the sample and the label, some metadata (title, artist, length etc.) is stored. The dataset was intended as a benchmark for audio classification tasks. A benchmark algorithm was described and features were extracted. Those temporal, spectral and custom developed features are also part of the dataset.

A free and open dataset with a focus on acoustic scenes, rather than music was created by Stowell et al. [32] in 2013. They downloaded tracks form the audio archive Freesound[4], which contained the tag *field-recording*. Samples of 10 seconds in length were extracted from those tracks and stored alongside some metadata, including further tags (such as city, train, voice, water etc.), which are used for partitioning. The dataset has a total of 7690 entries. In an

[3] GarageBand.com (service was shut down in June 2010)

[4] https://freesound.org/

experiment, they proved that the size is sufficient to perform auto-tagging.

The Million Song Dataset [2] provides features of music tracks on a larger scale. As the name suggests it contains information of a million tracks. Each track is described by metadata and acoustic features, such as pitches, timbre, loudness, tatums, beats, bars and sections. The dataset was created using The Echo Nest API[5], an online resource providing audio analysis and metadata for music tracks. Since the dataset does not contain audio samples for the tracks, it avoids the legal difficulties of re-publishing copyrighted music. However, the use-cases are limited, since further research is restricted to the features available.

Another example for a dataset being created from available data on a larger scale is *ImageNet* [3]. This dataset contains images labeled in a hierarchical fashion. In its first releasem it contained 3.2 million images in 5247 synsets. The current version has more than 14 million images in over 21 thousand synsets. *ImageNet* has been widely used for image classification tasks (such as [14] and [29]), because of its high accuracy and diversity.
To create the dataset, images were retrieved using several image search engines on the web. The cleaning process was performed by humans. To do this on such a large scale, the service of Amazon Mechanical Turk (AMT)[6] was used. Since humans make mistakes or disagree with each other, the labeling setup was designed to cope with mislabeling. The resulting accuracy reported was 99.7%.

In 2017 Gemmeke et al. [9] tried to create a dataset similar to *ImageNet*, but for audio events. Their so called *Audio Set* contains over 1.7 million 10second samples of audio events in 632 classes. The samples were extracted from YouTube[7] videos, which were queried partly using titles and keywords and partly by an internal video-level automatic annotation system. Those samples were then rated by humans. Each sample was checked by three people and a majority vote was applied. More than three quarters of the samples were identified unanimously, leaving less than a fourth of the samples relying on majority vote.

[5] http://the.echonest.com
[6] https://www.mturk.com/
[7] https://www.youtube.com/

To the best of our knowledge there are no studies available that focus purely on design aspects and creation guidelines of an audio dataset.

2.4. Available Datasets

There are several dataset already publicly available that contain recordings that can be used for TMD. Even though, for this work our own dataset was created and used for training and validation, samples of other datasets were used for evaluation.

One of the first datasets that contained modes of transportation was the one provided with the DCASE 2013 challenge [33]. It was recorded in London and split into 30 second samples. The two relevant classes are bus and tube. Also other classes such as busystreet, park or tubestation can be used to represent the *not a transportation mode* class. The dataset contains 10 samples (5min) per class which amounts to a total of 50 minutes. Each sample has been handpicked to avoid microphone handling noises. Therefore, the quality of the samples is high but the quantity is quite low which makes it ideal for evaluation purposes.

A dataset that contains more different modes of transportation is the *LITIS Rouen dataset* [24]. It has 19 classes, 7 of which can be used for TMD: plane, bus, car, metro of Paris (metro-paris), metro of Rouen (metro-rouen), train (train-ter) and high speed train (train-tgv). Their selection is comparatively fine grained since they distinguish between slow and fast trains and metro lines in different cities. Other classes such as train station or tube station can again be used to represent no current transportation mode. In total the dataset contains 3026 samples of 30 seconds in length which amounts to 25 hours of material; 1157 samples (9.6h) belong to the previously mentioned 7 relevant classes.

The DCASE ASC challenges are another valuable source for TMD data. The 2016 challenge contained the classes bus, car, train and tram (among 11 other classes) all of which were recorded in Finland. The authors of the challenge have released a development dataset, which contains 78 samples (38min) per class and (after the challenge was finished) an evaluation dataset with 26

samples (13min) per class. Furthermore, they specify that each vehicle was moving when the recording was made (this is something that is not true for our dataset). In total this dataset holds 208 minutes worth of recordings that we can use for TMD.
For the 2017 challenge they kept the same classes, but changed the sample lengths from 30 seconds to 10 seconds. This is interesting, because 10 seconds of audio seem to contain enough information to perform a classification. The number of samples was increased to 312 per class (52min) in the development dataset and 108 per class (18min) in the evaluation dataset. That yields a total of 280 minutes of audio for TMD.
The dataset for the recent DCASE 2018 challenge also contains modes of transportation, but they are limited to tram, bus and metro. They kept the sample size of 10 seconds, so it can be assumed that this length has proven to provide satisfactory results. In total the dataset holds 8640 samples (24h), so approximately 7 hours are usable for TMD.

There are several more datasets being used for research purposes on ASC, such as the one created by the authors of VehicleSense [16]. We requested access to this dataset, but this was denied, since the data is owned by a company and there is a privacy issue. Those are usually the reasons for why it is hard to get create a public dataset. Either the data is owned by a sponsor who payed for data collection, or the data was collected in an environment where not every person gave their consent to the recording being published. The latter is also the reason why we cannot publish the dataset we create.

3. Methods

3.1. Dataset Creation

There are two main ways to create an audio dataset: using available audio data or recording audio. Examples for the former can be found in section 2.3. Recording new audio data is the more flexible approach which is why we used it for this project. This section discusses the steps required for generating the dataset.

3.1.1. Choice of Classes

One of the primary decisions to be made when planing the creation of a dataset for a classification task is the choice of classes. It can be split into two sub-questions: *How many classes should there be in total?* And *which ones in particular?*

It is easier to merge existing classes than it is to split them. Therefore it is advisable to choose more classes at the creation step ,because they can still be combined at a later point. For instance, the classes *city bus* and *coach* can easily be merged to a parent class *bus*, however, it is not simple to split the *bus* class into the sub-classes, if there was no distinction recorded in the meta data.

The more classes that are required, the higher the effort to record. Not only because more data has to be recorded, but also because it is usually more complicated. For example if you were to record the classes *electric bus* and *combustion engine bus* in one city: The city might have a huge fleet of buses, but only a few of them are electric buses. Thus the chance to randomly enter an electric bus is a lot smaller and therefore you would have to specifically look out for electric buses and ride them. That is more than twice the effort, than if you were to just have one single *bus* class since it could be recorded in your regular commute.

Our Approach

We selected 9 of the most common transport modes for personal transport, namely: *walking*, *bike*, *car*, *bus*, *coach*, *tram*, *metro*, *train* and *plane*. Other datasets have a *taxi* class, which in our case is included in the class of *car*. There are several other popular modes of transportation that we did not include in our dataset. Some were excluded because they are not available in most cities we recorded in (such as ferries, magnetic rails, cable cars etc.). Others were excluded because they are not popular enough among the people performing the recording (such as running, skiing, inline skates, scooters, motor bikes) and some are just too exotic (such as trucks, helicopters, light aircrafts, hot air balloons etc.).
Furthermore there is no *stationary* class in the dataset. That type of class makes sense when using accelerometers, but it does not work for an acoustic dataset. However, we did include a *not a transport mode* class. More details in section 3.2.7.

3.1.2. Choice of Recording Device

When it comes to creating an audio dataset one of the essential decisions is the choice of recording device. There are different types of microphones that give different results. The two main types of studio microphones are condenser microphones which usually record from all directions (having an omni-directional or bi-directional polar pattern) and dynamic microphones which are usually used to record sound from one particular direction (having a cardioid or lobar polar pattern).

The quality of a recording depends on the sensitivity of the microphone used. A high end microphone will be able to pick up more detail than a standard voice tracer, which itself would outperform a mobile phone. The more detail in a recording, the more information is available for performing a classification. Depending on the type of classification performed it might be sufficient to provide recordings with little detail. However, in general the classification performance is equally good or better when more detail is provided.

The same can be said about the sampling rate. The higher the sampling rate, the higher the highest frequency that can be encoded. For some applications, those frequencies might be valuable information. So the higher the sampling rate, the better the classification (in general), but it might have an impact on the performance of the algorithm.

Whatever the choice of recording device may be it, is important to be consistent. If one class is recorded with one type of microphone, but another is recorded using a different microphone, then the classification might pick up the differences of the recording devices, instead of the actual class differences. To avoid this falsification, it is important to use the same audio recording device for all classes. If different recording devices have to be used (for instance because several people in different locations create recordings), then they should record (equal amounts of) several (ideally all) classes.

Our Approach

In this project only smartphone microphones were used to create the dataset. This is due to the intended use-case, which is a classification suite for smartphones. Since most people have a smartphone with them, but they rarely carry around studio microphones. All recordings were sampled with 44.1kHz as that is the highest available sampling rate, using standard Andoid. Comparing different recording devices and sampling rates is out of scope for this work.

3.1.3. Quantity and Quality

There is always a trade off between the quality of the dataset and the minimum time spent to create it. In general the more data (and the more diverse data), the better the results will be but also the more work has to be put into the creation of the dataset. This section will help you answer the following two question: *How many recordings are required?* And *of which quality do they need to be?*

In order to properly answer those questions, we need to first outline the task. What is it that we want to achieve? How many resources are we willing to spend? How reliable does the system need to be? It makes a difference, whether we run a recommender system, where an 80% accurate guess of the transportation mode is already a sufficient hint, compared to if we want to reason with the classification result in a court case, where an accuracy of over 99% is required. So the first step is to define the desired accuracy[1].

The relationship between the amount of data and the accuracy of a given classification task depends on many variables. The main contributors are the number and diversity of classes, the algorithm used and the quality of the data. Given similar quality of the data (for acoustic TMD the data would be the recordings) it usually holds that the more data is available the higher the classification accuracy. However, this relation is hardly ever linear. When there is not enough data adding a bit more can already increase the accuracy significantly. Contrariwise, if there is already a lot of data available then adding more barely affects the accuracy.

> A good strategy would be to continue gathering data, until the resulting improvements are not worth the collection efforts anymore.

Under quality of a recording we understand how well it actually represents the class it is supposed to represent. Reasons for poor quality include noise (that does not originate form the mode of transportation), other sounds (such as music or people talking etc.), or silence.

There is a trade-off between quality and quantity. The higher the quality requirements are, the harder it is to create recordings that meet those requirements. On the other hand, it is easy to create long recording if quality is sacrificed. Similarly, if there is a high quantity of recordings available, it can make up for a lack of quality (given that the algorithm can

[1]whether this is the actual accuracy or an F1-score or similar measure can be defined

handle it), but if there are only a few recordings then the quality must be good in order to get decent results.

Our Approach

Our strategy is to go for high quantity and medium quality. We want to work with *real life* recordings and that means that there will be disturbances. It is not supposed to be a lab environment, since an application that does acoustic TMD will also have to deal with noise. Given enough data, deep neural networks are usually quite robust, so the algorithm should be able to handle it. Furthermore, we perform data processing to increase the quality of the dataset (see secton 3.3).

3.1.4. Diversity

When talking about diversity in the context of acoustic TMD, we mean the extend to which the recordings within a given class are different. There is also the diversity between classes, but that is usually given by the class definitions.

Having a diverse dataset brings advantages and disadvantages. The more different recordings there are in a class (given an appropriate quantity and a good enough algorithm), the more diverse input the trained network will be able to predict correctly. However, the higher the diversity, the higher the chance that one class overlaps with another one, making proper classification impossible. So there is a trade-off between the classification accuracy and the diversity of input that can be handled by the classification.

How diverse your dataset should be, once again, depends on the task at hand. If you only need to detect transportation modes of one particular city, then it is enough to have training data from just this city. However, if it is supposed to work in any city around the world then you will have to provide more diverse recordings for training. Of course, you could also provide a more diverse dataset when you do not need it, but you might loose accuracy and you risk having more confounding variables. Besides, creating a more diverse dataset requires more effort. Therefore, it makes economic sense to provide just enough diversity to fulfill the task.

Our Approach

We aim to be as diverse as possible, while still maintaining a decent accuracy. Therefore, we tried to get recordings from transportation modes in different cities across Europe. The goal is to be able to correctly classify samples from external datasets. Those external recordings from different cities are not used for training, but merely for evaluation purposes. The degree of success is described in section 4.9.

3.1.5. An Application for Recording

An Android application was developed to record audio using smartphones. The advantage of having a custom-developed application is that it can be set up to suit the user needs. The application was specifically designed for creating our acoustic dataset for transport mode detection.

Metadata

The most obvious feature of the application is the storing of metadata. For each recording the user can pick the mode of transportation, the phone's position, and the sampling rate using drop-down menus (so called spinners in Android). Furthermore, the location (usually the name of the city) as well as additional notes can be entered using text input. All metadata is stored in the filename to avoid having to parse additional files when grouping recordings.
An example filename might look as follows:

$$\underbrace{\texttt{1536661337142}}_{\text{timestamp}}\_\underbrace{\texttt{Rv6}}_{\text{vers.}}\_\underbrace{\texttt{Train}}_{\text{TM}}\_\underbrace{\texttt{lyingOpen}}_{\text{phone pos.}}\_\underbrace{\texttt{Munich}}_{\text{city}}\_\underbrace{\texttt{ICE\_emptycoach}}_{\text{notes}}\texttt{.wav}$$

The filename consists of the time of recording (in ms since 1970), the version of the app, the transport mode, the phones location, the recording location and notes, each separated with underscore. If the notes contain more than one term, then they are also separated with underscores. All files are stored to the external storage (usually an SD-card) of the phone as wavelets (`.wav`).

Lessons learned

Several people have tested the application over several months and suggested some improvements.

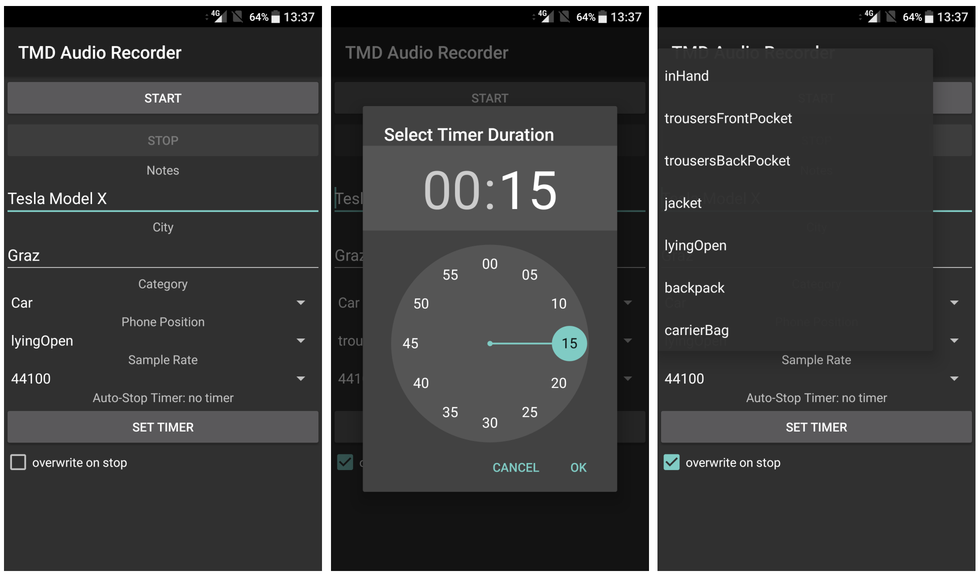
The first suggestion we received was to add trimming functionality. The request was to be able to listen back to the recording and set new start and stop points. This is helpful, because people sometimes forget to stop recording, especially if they are not used to using the application. Or when they board a train and it doesn't start for another 10 minutes. Trimming can be useful the increase the quality of the recording.

Another feature that was requested is a timer. The user should be able to set a time after which the recording stops automatically. This is useful because people often know how far they are going with their chosen mode of transportation (be it tram/metro/bus but also car and bike) and they usually also know how long it will take them approximately. So they might want to set the timer to not have to think about stopping the recording later.

The last feature that was added is the ability to overwrite metadata when stopping the recording. This has many usecases, for instance, when the user set the wrong mode of transportation, but only realize it after taking out the phone to stop the recording. Often, the settings seem to fit, but while recording the phone was mainly in the hand, instead of the pocket where it started out. Or when the environment changes and it suddenly begins to rain while on the bike, or there suddenly is a screaming baby on the train (which should be added as a note). In all those cases, it is very helpful, if changes are applied when the recording is stopped. The application now features a checkbox that allows the user to select whether or not the file should be renamed when stopping the recording.

User Interface

Figure 3.1 shows the user interface of the application. It was not made to look aesthetically pleasing, since the application is only used internally and to us functionality is more important than style. Hence a default Android look and feel is sufficient.



Notes: Tesla Model X
City: Graz
Category: Car
Phone Position: lyingOpen
Sample Rate: 44100
Auto-Stop Timer: no timer
overwrite on stop

(a) The main activity (b) The timer (c) A spinner

Figure 3.1.: Examples of the user-interface of the Android application.

There is one main activity as shown in figure 3.1a, which handles all user input. It contains a button to start and one to stop the recording. There are text fields for input and spinners for choosing metadata for the recording. One example is the spinner to select the phone position, which is shown in figure 3.1c.

When the `set timer` button is pressed, a time picker dialogue is displayed, as shown in figure 3.1b. Once a duration is selected (in this case 15 minutes), a new recording will stop after the specified amount of time. This feature cannot be combined with the overwrite-on-stop feature. However, the recording can be stopped before the maximum duration in which case the changes are applied.

Availability

The source code of the application, as well as a compiled version can be found online[2]. It is provided *as is*, and licensed under the MIT License, which allows anyone to use, modify, and redistribute the software.

3.2. Classification

The crucial step of any transport mode detection is the classification. This is where a given data sample is labeled with a class prediction. Most classification algorithms require a training phase, where audio samples with the correct label are provided (supervised learning) to create a model. After the training phase, the model can be used to perform the actual classification of unlabeled data. In our case the audio sample is labeled with a predicted mode of transportation.

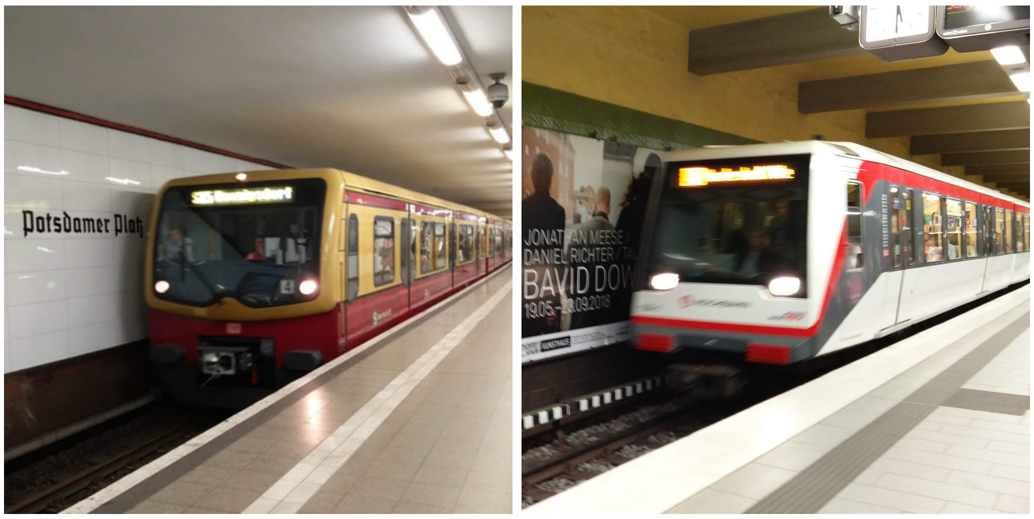
3.2.1. Difficulties

The classification of audio samples is not a trivial task. This is due to the complexity of the problem. There are two main complications: high intra-class variability and low inter-class variability.

[2] `https://github.com/aschmidhofer/TMDAudioRecorder`

Low Inter-Class Variability

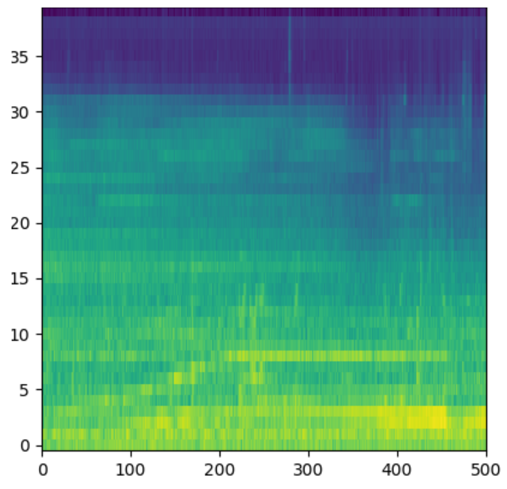
Samples from different classes sound similar. On the one hand this is the case because some classes are inherently similar, such as *bus* and *coach*. Also, all transport modes operating on rails are similar in the sounds they produce. Especially in a bigger city, where the *train* performs like a *metro* in the inner city, and the *metro* sounds like a *train* further out. One example is the S-Bahn (suburban train) in Berlin, which looks (see figure 3.2a) and behaves like a metro line.
That it does not just appear like a metro, but also sound like one is demonstrated in figure 3.3 which compares the log mel-band energies. It shows that the Berlin S-Bahn (3.3a), which travels underground, produces sounds closer to the Hamburg U-bahn (3.3b) than the Dresden S-Bahn (3.3b) or the Intercity-Express, which (3.3d) both travel above ground. Further details on log mel-band energies can be found in section 3.2.4 and more plots from different classes are available in section 4.1.4.

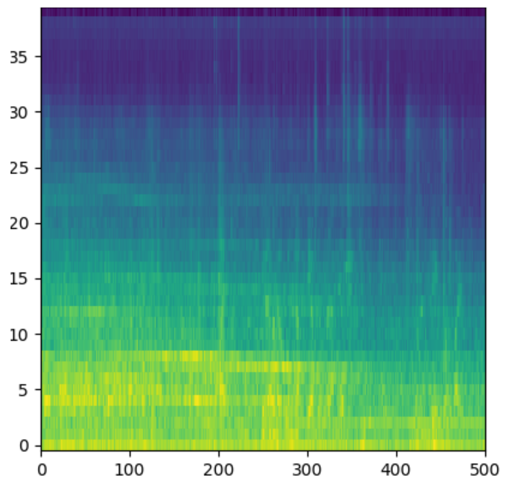


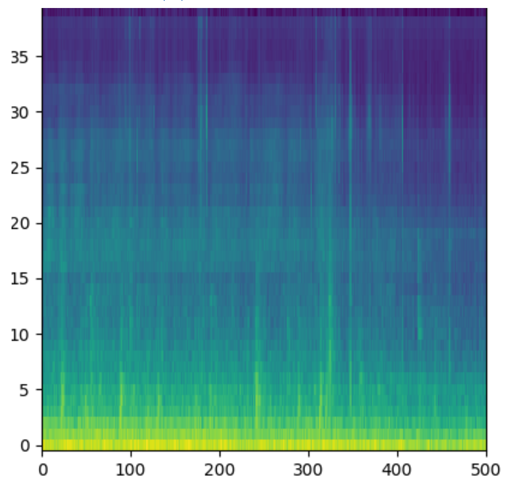
(a) S-Bahn Berlin (b) U-Bahn Hamburg

Figure 3.2.: A suburban train of Berlin compared to a metro in Hamburg.

The second reason for low inter-class variability is identical surrounding sounds. *Walking* on the sidewalk picks up the same street noises as riding the *bike* on the same street. Another example are passengers whose talking


(a) S-Bahn Berlin


(b) U-Bahn Hamburg


(c) S-Bahn Dresden

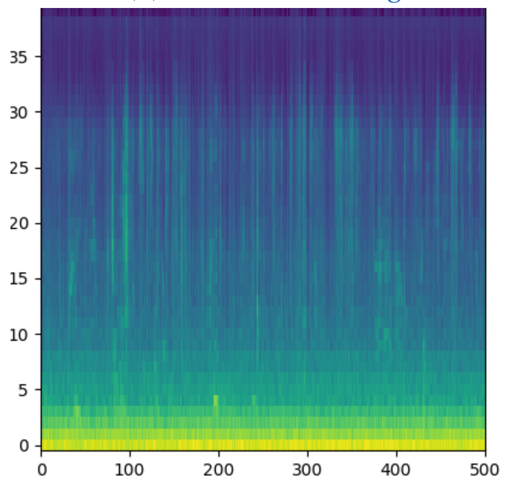

(d) DB ICE

Figure 3.3.: Feature comparison between the Hamburg metro (top right) and different German trains. It shows the low inter-class variability between the train and metro class and the high intra-class variability inside the train class.

sounds the same whether they ride a *tram*, or a *train* (or any other transport mode).

High Intra-Class Variability

Samples within the same class sound significantly different. There are several possible reasons for this. Firstly, it depends on where the recording is taken. The front of the cabin of a *plane*, for instance, does sounds different to the rear of the cabin. The same is true for a *bus*, which usually has the engine in the back.
Broad class definitions are another reason for high-intra class variability. An electric car sounds vastly different from one with a combustion engine, yet they are both part of the class *car*. The same is true for regional trains and fast trains which are both part of the class *train*.
Another contributing factor is the mode of operation. A car on the motorway sounds different to a car in city traffic. Also the driving style is crucial, because a high-revving engine from a car that is raced produces a noise distinctive from one that is driven by a calm driver.
Then, of course, there are external influences such as people. An empty *bus* sounds different from a full one. It also makes a difference whether you ride your *bike* in the city in the middle of the night, or during rush hour where there is the most traffic and noise on the street.

3.2.2. Training, Validation and Evaluation Set

The whole dataset, consisting of recordings of different lengths, is split into three subsets: The training set, the validation set and the evaluation set. The *training set* is used in the learning process (also called training) of the deep neural network. Features of the samples are provided with a ground truth to allow for the network to adjust its weights, using backpropagation of error.
The *validation set* is used to check the results to avoid over-fitting (which is when the network adjusts too much to the training set). Some algorithms implement a stopping criterion (eg. when the validation accuracy starts to drop), after which the training is ended. Others (like the one we use) train from a fixed number of epochs and select the model of the epoch where the validation set performed best.
The purpose of the *evaluation set* is to rate the network once the training

is complete. All the samples in the set are classified and the results are compared to the real label. This is how the confusion matrix, as well as statistics, such as recall, accuracy, and F1-score are created.

The evaluation set is taken from roughly 10% of the data. It is selected by hand, to assure that the evaluation set is as diverse as possible, but at the same time not too overloaded. This is done by selecting shorter recordings, but from different locations, operation modes, facilitators etc. Doing this split manually requires some effort by a human, but it is essential to get a representative evaluation result especially when the number of recordings is not very high.
This split is based on raw recordings, which means that either all or none of the samples from one recording are in the evaluation set. The reason being that samples from one recording might sound almost identical (especially consecutive samples). Identical training and evaluation data cause unrealistically high evaluation results so it should be avoided.

The remaining recordings which are not in the evaluation set are again split into 70% training and 30% validation data. This split is done automatically at runtime, which means that the recordings are already split into samples (see section 3.2.3). Samples are associated with validation or training set based on the identifier of the sample. The identifier (in our case this is the timestamp) is identical for all samples in a recording which means that there can not be samples from the same recording in both sets. The selection is random but consistent throughout tests (so the same samples will be in the validation set for each execution).

3.2.3. Splitting Recordings

The raw recordings have to be split into samples. This has to be done because the algorithm expects samples of equal length. However, the length of the recordings vary. So a python script was created that breaks down the raw recordings into 10 second samples. The samples are created in such a fashion that they do not overlap and there is no unused space between them. In other words, if they were to be put together sequentially one could obtain the original recording. Only as many samples are created, as fully fit in the length of the recording. There is no padding in the end. A few seconds of the recording (anything less than the sample length) might not be used.

Splits for Sample Length Analysis

Special rules apply when creating the samples for analyzing different sample lengths. The script works as described above with two extensions: Firstly, there is an initial offset of one minute. So the first sample that is created starts at second 60. This is to counteract the fact that some recordings require trimming. Secondly, not the full length of the recording is used. To ensure that each class has similar amounts of training data only a percentage of the recording is split into samples. This percentage is calculated using the minimum total length of training data per class and dividing it by the total length of training data of the current class. The recordings of the evaluation set are split in their entire length (excluding offset).

3.2.4. Algorithm

The choice of algorithm is an important one. There are several algorithms that can perform acoustic transport mode detection. Simpler algorithms are usually quicker and can work with smaller training sets. More complex algorithms are more robust to noisy input data and in general, given enough training samples, they will produce better results. As pointed out by Wolpert [39], there is no single best algorithm, just one that fits the given task and dataset.

The current trend for acoustic sound classification is to use convolutional neural networks (CNNs). CNNs were originally developed for image classification tasks, such as those performed by Krizhevsky et al. [14] and Sermanet et al. [29], and they are still the dominant choice in that field. In recent years they have also gained dominance in the field of ASC.

The algorithm used for this project does not have to be the algorithm which performs best, since our goal is to create a high quality dataset, not a high quality classification system. Our first candidate was the open-source baseline algorithm of the 2017 DCASE challenge. It gave decent results with our dataset; however, it is based on a multilayer perceptron architecture. The baseline algorithm of the DCASE challenge 2018 [18] uses CNNs and it is a state-of-the-art ASC algorithm. There are other algorithms that might yield better classification results but for our purposes this algorithm is sufficient. Since it does not have a name it shall be referred to as *the algorithm*. The facts that *the algorithm* is state-of-the-art and that it is also open-source were the reasons to choose it as our classification algorithm.

The algorithm operates by performing the following steps: feature extraction, feature normalization, learning, testing and evaluating. *Learning* uses the training set to adjust the parameters of the network, *testing* uses the trained network to give a label to each item in the evaluation set and *evaluating* compares those labels to the provided true labels and compiles a summary. The upcoming two subsections describe the feature extraction process as well as the architecture that is used for learning and testing.

Features

The first step *the algorithm* carries out is the feature extraction. Log mel-band energies are extracted for each 10 second audio-signal. This is done by performing a fast Fourier transformation, then bringing the values on a mel-scale (as defined by Stevens et. al in [31]) and then applying the log function (base e) on the values themselves.
The analysis frame is 40ms wide and has a hop size of 50% (walk-forward sliding window). Each frame is represented by 40 bands, so the input shape for the CNN is 40 x 500 (40 bands and $\frac{10s}{0.04s}\frac{1}{50\%} = 500$).

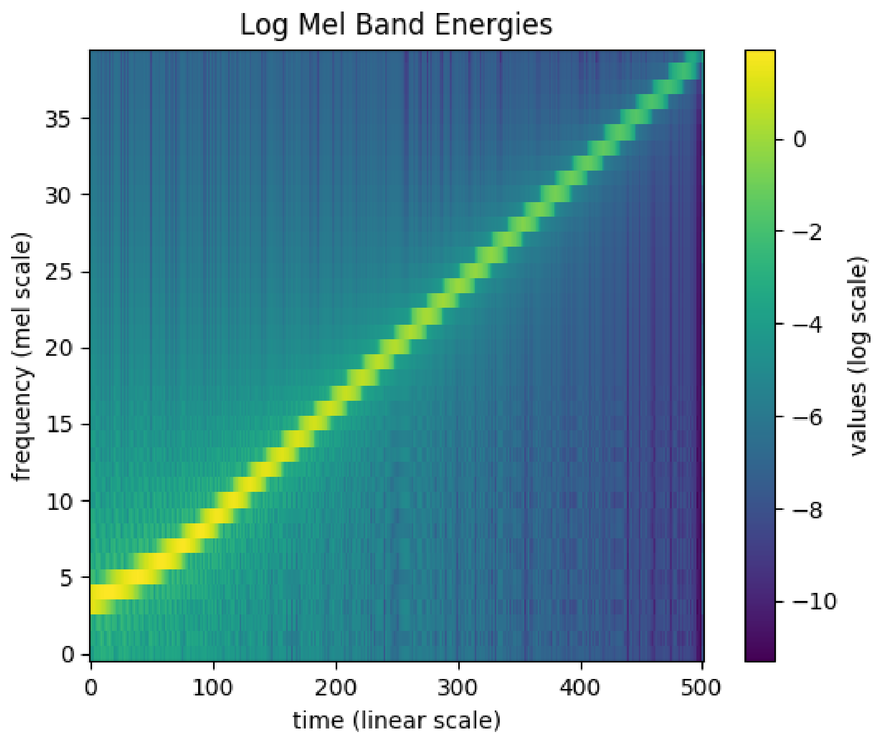


Figure 3.4.: The log mel-band energies of a demonstration sample.

Those features can be visualized by plotting them. Figure 3.4 contains the features of a demonstration sample. It contains only one chirp in sine waveform which starts at a frequency of 440Hz and an amplitude of 80% and over the 10 seconds it is logarithmically interpolated to end at 22kHz and an amplitude of 10%.
In figure the increase seems to contain steps because of the discretization. There are only 40 values in the y-axis compared to 500 on the x-axis and the figure was scaled (each pixel has a $\frac{1}{12.5}$ ratio).
In the plot, the amplitude is represented by color: the brighter the color the louder the given frequency (due to the log function the intensity is on a logarithmic scale). The x-axis represents the time (on a linear scale, 500 corresponds to 10s) and the y-axis the frequency (on a mel-scale, which is itself logarithmic). The highest frequency that can be encoded is half the

sampling rate. With a sampling rate of 44.1kHz that would be 22.05kHz, which is also the maximum value on the y-axis (index 40 on scale). The axis are labeled with the index, as provided to the network. In subsequent plots the labels are removed to increase the figures scalability.

Architecture

An illustration of the architecture of *the algorithm* is shown in figure 3.5.

There is a two layer CNN, each with a 2D convolutional layer (with batch normalization and ReLu activation) and a 2D max pooling layer. The convolutional layers have a kernel size of 7 and 32 or 64 filters respectively. The max pooling layers have a dropout rate of 30% and a pool size of (5,5) or (4,100) respectively. What follows is a layer to flatten and then a dense layer (again with ReLu activation and a dropout rate of 30%). The output layer is a softmax layer. For the training, a batch size of 16 is used for 200 epochs. The Adam optimizer with a learning rate of 0.001 is applied to change the parameters of the model in the leaning process. The parameters of the model which performed best on the validation set during the 200 epochs are then used.

It is not our intention to improve upon *the algorithm*, since the focus of this work is on creating a dataset not an algorithm. The given algorithm is just used as a tool for performing classification.

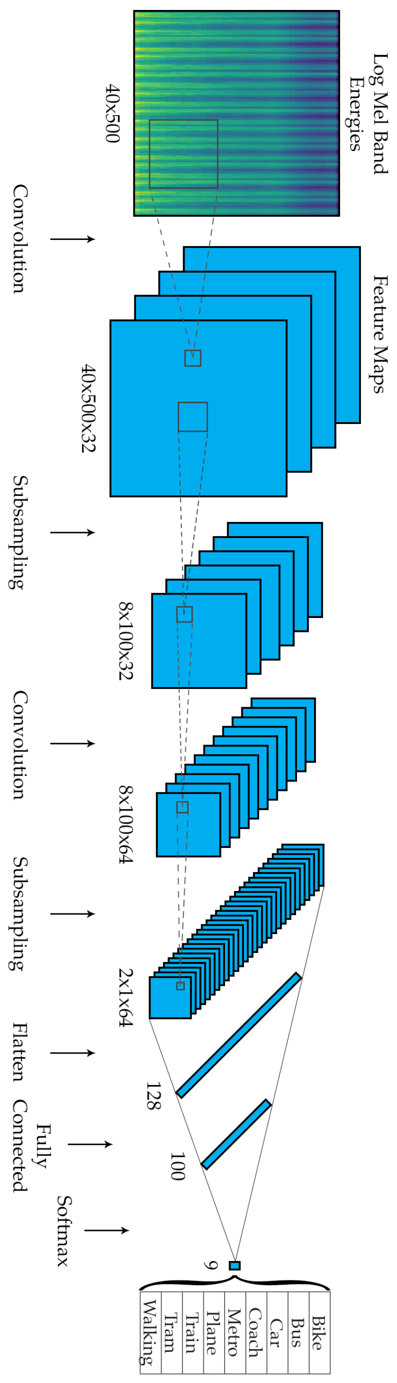


Figure 3.5.: The architecture of the CNN used.

Implementation

Several changes had to be made on the software in order for it to fit our purposes. The source code used as a starting point is licensed under MIT license and is available online[3].

The first change that had to be made was necessary to allow for our dataset to be used as an input. The baseline algorithm does not intend custom datasets, since the dataset for the challenge is provided by the organizers. So a bit of code had to be changed to allow the correct integration of our samples and classes.

Furthermore, *the algorithm* had to be adapted slightly in order for it to allow different sample lengths. The default sample length is 10 seconds, but for our sample length analysis we require arbitrary sample lengths. Fortunately, due to the design of the software those changes could be made by changing the parameters in the configuration file.

The last minor change that had to be made was to allow GPU usage. The frameworks used by the algorithm already support GPUs, however they need to be set up correctly. The GPU cluster of the Know Center has eight Tesla P100 GPUs. In order to not block all of them, only one GPU should be selected by its ID. This is done by adding a few lines of code in the main file.

3.2.5. Sample Length

Usually, the raw recordings have a duration of several minutes. In our case the shortest ones will be approximately one minute in length (for instance recordings of walking in a plane) and the longest ones can have a duration of several hours (for instance longer train or car rides).

It would not be advisable to use the original recordings for the classification process directly. There are two reasons for this: First of all, the classification system is designed to compare files with equal length. Secondly, there would not be enough training data if every file is only used once.

[3] https://github.com/DCASE-REPO/dcase2018_baseline

Thus, the recordings have to be split into sub-clips, so called samples. The length of the samples can be chosen. Ideally, the samples are just long enough to contain the relevant information required to perform the classification. If the chosen sample length is too short, the classification accuracy will decrease because the samples don't include enough information. However, a higher sample length increases the training time and might require more training data, since there will be less samples in total. Moreover, in a real time classification task the sample length also represents the minimum time required for labeling. This is because a recording with at least the sample length has to be made before a first estimation can be made. A quick response is essential for a good user experience, which is why the sample length should not be set for too long.

The ideal sample length depends on the classification problem as well as the classification algorithm used. To figure out which is an appropriate value, a test has to be conducted. If there is no reference value available, then it might pay off to start with a broad test on a logarithmic scale. For instance, by picking sample rates as a multiple of two. This first result will show which area has to be examined closer to figure out a suitable sample rate.

In [10] Guvensan et al. analyzed different sample lenghts (they refer to it as window sizes since they do not work with audio signals) for their mobile transport mode detection system. They found that their system performs best with a window size of 60 seconds. However, this cannot be generalized since they use accelerometer, magnetometer and gyroscope as data input, and they also have different classes. When plotting the recall of each class over different window sizes it can be observed that the classes tram and ferry benefit the most from a larger window size. We also use this strategy to gain more insight on how the performance of each class depends on the sample length. The results can be found in section 4.2.

3.2.6. Ambiguous Classes

Sometimes a sample does not belong to strictly one class. There might be two or more classes that it can be categorized in. In our case, this applies mainly to combinations of the *walking* class and another class. For instance walking in a plane, or walking in a train etc.

There are different ways to handle those special cases. One possibility would be to put them into one of the two classes and train it accordingly. In other words, they are not treated differently than any other sample. For our walking while in a vehicle that would mean to either give priority to walking (and count all samples that are labeled differently as misclassified) or to the other transport mode (and count samples labeled as walking as misclassifications). As a first approach, the latter is what we are doing.

Another possibility is to create a separate class for each special case. This would be advised if you want to be able to distinguish the special case from the regular transport mode. In our case, this would be to deliberately distinguish between walking in a plane and walking in a train. However, in order to accomplish this, enough recordings of the special case need to be available for the algorithm to be able to train the network correctly.

The third way of handling ambiguous classes is by considering both classes as correct. This means whenever the sample is labeled in either of the two (or more) classes that it could belong to, it is counted as correctly classified. So, for example whenever a *walking in a plane* sample is classified as *walking* or *plane* it is a true positive. This strategy allows you to put the special case samples in either of the classes for the training process. It also allows you to work with a dataset where the special case sometimes is in one class and at other times in the other class. As long as it is always in one of the two where it should belong (and there are not too many special cases relative to total samples) it should still work.

A last strategy would be not to allow ambiguous classes. If the use-case does not required those special cases to be matched correctly, then it might make sense not to include samples of that kind in the training process, as those samples can cause confusion and result in more misclassification.

Our Approach

We will start out with samples being associated with only one class. Then we try to increase the accuracy by also counting samples labeled as the other appropriate class as true positives. We will analyze in which of the classes the training samples need to be, in order to cause the least false negatives. If a significant amount of ambiguous samples are still classified incorrectly, then the removal of all ambiguous samples will be considered.

3.2.7. The Unknown Class

It can happen that an input sample does not represent any of the transportation modes. This can be either be because there is so much noise overlay that the sounds from the target TM cannot be heard, or it can simply be a recording that was made outside of any transportation mode.

For those cases, it might make sense to add an *unknown* class. This is an additional class that is added alongside the transportation modes and it represents everything that is not one of those TMs. So, for instance, a different transportation mode like ferry (that is not in our 9 classes), the noise of a street, people talking, even random noise, and silence would all belong to this class.

Of course, it depends on the use case. If you ensure that only samples of transportation modes are entered in the classification process, or if you define that anything else causes undefined behaviour, then you might not need to add the *unknown* class.
However, even then it might make sense to add a *trash* class (which is essentially the same as the *unknown* class). This is because of how CNNs (and deep neural networks in general) operate. They always have to give a label so there is usually a class that is represented as *if it is none of the others then it is this one*. In other words whenever nothing else fits, this label is applied (hence one might call it a *trash* class). When the network is presented with something unseen, they usually put it in this class. The reverse conclusion to this is that the class is not well represented in the network. So adding an extra class might sometimes even increase the accuracy because the former *trash* class is properly trained to recognize its features.

Another reason to add the *unknown* class is for quality assurance. It can be used to filter samples which do not properly represent their associated class.

In fact, the *unknown* class is essential for our cleaning process as described in 3.3.3 and it is also helpful for trimming (see 3.3.2).

In order to add the *unknown* class, data has to be gathered that represents it. Basically, every audio file that is not one of the 9 selected transportation modes can be used. For this project we use recordings that were made in the dataset creation phase (however there are not as many for this class than there are for the other 9), as well as samples from other datasets. It is advisable to select soundscapes that might be encountered in the recording processes of the main classes such as: a busy street, a metro station etc., but also some that vaguely resemble the sound of a transportation mode such as: for a washing machine or a lawn-mower.

The results of adding an *unknown* class can be found in the 4 Results chapter.

3.3. Data Processing

Even when taking care in the recording process, mistakes are made. Sometimes the wrong label might have been picked or a recording was not stopped even when the mode of transportation changed. To counteract this, a simple preprocessing is applied before the classification model training.

3.3.1. Relabeling

It can happen that a recording was not labeled correctly in the recording process (ie. the metadata contains an incorrect class). This is mostly due to human error by forgetting to change the class in the recording application. Those mislabeled recordings need to be relabeled correctly to increase the quality of the dataset.

Given a working classification system, automatic label changing can easily be implemented. Assuming that there are not too many mislabeled recordings (which is a justifiable assumption since we expect no more than one percent of labels to be incorrect), the algorithm (as described in section 3.2.4) will still give a decent model, even when those recordings are in the training set. The model can then be used to classify all samples (including the ones in the training set) and relabeling can be performed based on the results.

There are several ways to decide whether a recording is mislabeled or not. One possible strategy is to analyze all samples from a given recording and check whether the class that was predicted by the most samples corresponds to the given label. If it does not (ie. another class is predicted more often that the class the recording is supposed to have), then the recording should be relabeled accordingly.
Another possibility is to define a threshold λ, above which the percentage of correctly predicted samples (that is, stored label and prediction are identical) of a recording must be. If the percentage is below the threshold, the recording is flagged for relabeling.
We chose the latter strategy with $\lambda = 50\%$. This assures that if the threshold is reached, it is also certain that the majority of samples are predicted to be that class (because there can not be another class with $> 50\%$), so the criterion of the first strategy is also fulfilled. Furthermore, it allows us to filter recordings with low confidence because, even though a majority might be reached, if less than half the samples are predicted correctly there might be something wrong with the recording. It could also be a sign that a complex class should be split into sub-classes, or that the overall class accuracy is too low.

Each recording that is marked for relabeling is checked by a human. This is feasible since not a lot of recordings should require renaming. By having a human listen to the recording they determine whether the recording actually has a wrong label or whether there was a mistake in the relabeling algorithm.

To demonstrate the robustness of this relabeling approach some recordings are deliberately given the wrong label. The training is done including the samples from the mislabeled recordings in the wrong classes. The percentage of wrong labels is increased until the approach fails. Results of this experiments can be found in section 4.8.1.

3.3.2. Trimming

One of the simplest ways to remove samples that do not represent their class is by trimming. Trimming is the process of removing parts at the beginning and/or the end of the recording. Some recordings require trimming because the recording was started too early or stopped too late. This might be due to several reasons, for example the bike had to be unlocked after starting the recording, or the train did not leave the station for another minute, or the person simply forgot to turn off the recording when leaving the mode of transportation etc.

Trimming is performed by analyzing the samples at the beginning and at the end of a recording. The same principles as for relabeling apply: as long as there are relatively few samples that need to be trimmed, the algorithm can be trained using all samples. A minimum number of samples n is defined. The samples at the beginning are then analyzed and all samples are thrown out until n consecutive samples have the correct class prediction. The same is done for the samples at the end, but starting with the last sample and going backwards.

The most likely other class predicted at the start and end of a recording is *walking*. This is because usually people walk into and out of another mode of transportation.
Having a separate *not a transportation mode* class would help in this scenario. Otherwise there is likely to be one class (*bike* in our case) that cannot be trimmed this way, because even samples that do not represent the class are predicted to be of this class.

The optimal value for n depends on a lot of factors, such as the sample length, the lengths of recordings, the total size of the dataset, the classes used, the quality of the recordings etc. The ideal n can even differ between classes. There is a trade-off between not removing too many *good* samples versus making sure that all the *bad* (those which do not represent the class) samples are removed. So, in that sense there is no best value for n, just one that is appropriate.
Therefore, our strategy is to test different values for n and select one that provides good results. This can be assessed by including recordings where

the desired trimming points are known and comparing them against the number of removed samples.

This simple trimming strategy does not always work. In some cases, when there are too many false positives, or too few true positives, the algorithm would trim nothing or trim the entire recording, respectively. To counteract that, the trimming strategy can be extended based on another assumption: There is more to be trimmed at the back of a recording, than an the front. This should be true for our dataset since the recording is usually started when one enters the mode of transportation, but often times it is forgotten to be turned off (it might have equally often been forgotten to turn on, but this results in a lack of samples, not a surplus).
The improved trimming procedure analyses the amount of samples that are to be trimmed in the front and compares them to the number of samples to be trimmed at the back. If there are significantly more at the back (for instance 2 minutes worth of samples, this threshold can be defined as well), only then the recording should actually be trimmed. This also allows for an automatic adjustment of n: it can be increased if none of the samples would be trimmed and it can be reduced if all of the samples would be trimmed.

3.3.3. Cleaning

Cleaning the dataset is the process of removing samples that do not represent the given class. This could be because there is additional noise (eg. music, people talking or other external events) that predominates the sound of the transportation mode, or because the transportation mode is not active in the given sample (eg. waiting at a pedestrian crossing or intersection, tram/train is in station etc.). It serves the same purpose as trimming, that is, increasing the datasets' quality. The main difference is that it takes into account all samples of a recording and not just beginning and end. A good cleaning strategy would make trimming obsolete.

Removing misclassified samples is not a constructive approach. It works for trimming because it is not important if a few samples that should have been kept are thrown out. However, it is inappropriate for cleaning, since it would remove way too many samples that do represent the class (but the algorithm was not able to classify them correctly). This would just simplify the classification task but not increase the quality of the dataset (in fact the quality would actually decrease).

Using an additional *trash* class can help the cleaning process. In the training process, this class should be provided with samples of likely scenarios that the other recordings might contain but are not part of the transportation modes (such as silence, white noise, music, sounds of a busy road, a construction site, people talking etc.). It should be considered that this class should not dominate any other class. It should be ensured that the number of samples for this class is less than the minimum number of samples of any other class. When analyzing all samples (from the 9 original TMD classes) those which are predicted to be the *trash* class are likely to not properly represent their original class and therefore can be removed.
This strategy works for two reasons: Firstly, if the algorithm predicted a sample to be in this class (,even though it was in the training set of a different class,) then the dominating aspect of the sample might be one of the scenarios that was put into the *trash* class. Secondly, a sample that does not fit in any class (including the *trash* class) is likely to be classified as *trash* because this class has the most diverse input.

There are different degrees to which the cleaning process can be performed. The minimum should be to remove everything that is outside of the class definition (for instance, for the *walking* class, samples where the person is stationary should be removed). The other extreme is to remove everything that is not exclusively the given class (for example when a car ride contains music). Any degree in between (in other words: how dominant the actual class features need to be in the sample) can be chosen. The question which is the best strategy, does not have a trivial answer. It depends on the specific circumstances and needs to be assessed individually. If the given class is already lacking diversity, then cleaning out too much is contra-productive. On the other hand, if the use case defines a clear pattern that is required, but the recordings are cluttered with other input then filtering more samples is the strategy to go with.

A superior, but extensive way, for filtering samples is done on a class by class basis. Each possible interference is handled separately. For instance, in the *tram* class you might want to filter samples in which the tram is halted

at the station. To do so, you require two sets of samples: one for tram in the station and one for tram in regular transit. Those can be acquired manually from the raw recordings of the dataset. Since there are only two classes, not a lot of training data is required (although the more the better). The algorithm can then be trained on those two classes to separate all remaining *tram* samples into *in station* and *in transit*. Then the *in station* samples can be removed, since they do not represent the *tram* class.
The same strategy can be applied to other classes (such as *metro* and *train*) or to filter waiting times at the *bike* and *walking* classes.
This method can be improved further by taking sample proximity into account. Usually there are patterns between the separated classes. In the *tram* example, it is highly unlikely that in one sample the tram is moving, in the next it is stationary, and in the following it is moving again. The prediction of samples that fall out of line can be changed to fit the neighboring samples similar to the healing algorithm introduced in [10]. One simple way to implement this is by defining a minimum sample group size m (should depend on the sample length) and ignore (relabel) patches of samples that are smaller than m.
More sophisticated methods for the detection of collective outliers (such as [5], or others, suggested by [30]) can be performed. However, that is outside the scope of this project.

When evaluating the cleaning process it is important to have a consistent evaluation set. If the evaluation set is also cleaned, then the resulting accuracy scores do not represent the original problem anymore. Keeping the evaluation set the same also allows to identify whether too many samples of the training set have been removed (and the lack in diversity results in a decrease in accuracy).
However, the evaluation set might also contain samples that do not represent the class and should be cleaned. It has to be made sure that the samples removed from the evaluation set are actually outside the TMD class description. Therefore, those samples should be checked by a human.

3.4. Scope

The purpose of this work is to answer questions about dataset design and creation. As such, it is not meant to improve upon the algorithms used for classification. Several aspects, such as the positioning of the device when recording, as well as different sampling rates, are not analyzed. We are trying to provide a comprehensive guide on how to create a dataset effectively, but we are aware that there are limitations given by the volume of work for this thesis.

4. Evaluation

This chapter discusses the findings of this project. This includes the data that was gathered and information which can be derived from that data. Some sections also contain a discussion that interprets the information.

First, the created dataset is described and its properties are listed. Then, an experiment is conducted to find an appropriate sample length to split the recordings into (section 4.2). To be able to make statements about the quantity and diversity of the samples, tests are run on smaller subsets of the data in section 4.4. How many classes should be contained in a dataset is discussed in section 4.5, and better ways to deal with misclassification are presented in section 4.6. Ambiguous classes are dealt with in section 4.7, followed by sample processing strategies (section 4.8). Finally, our dataset is compared to existing datasets in section 4.9.

4.1. The Dataset

This section describes the dataset which was created for this project. It consists of audio recordings of different transportation modes with a total length of 245 hours (82.3GB). The recordings were made between February and October 2018 by different people in multiple cities across Europe.

4.1.1. The Classes

The dataset contains recordings from the following 9 modes of transportation: Bike, Bus, Car, Coach, Metro, Plane, Train, Tram and Walking. It is worth noting that the number and total length of recordings is not uniformly distributed. The exact distribution as well as average recording length can be obtained form table 4.1.

Class	**number of files**	**average recording time [min]**	**longest recording [min]**	**total recording time**
bike	258	17	45	74h 48′
train	142	20	118	48h 19′
car	102	21	129	37h 20′
walking	145	10	55	26h 21′
metro	93	11	42	18h 30′
plane	55	17	52	16h 17′
bus	53	10	28	9h 12′
coach	15	29	58	7h 27′
tram	54	08	28	7h 23′
Total	**917**	**16**	**129**	**245h 40′**

Table 4.1.: Average, maximal and overall recording lengths per transport mode.

If not specified otherwise, the location of the recording device can be anywhere inside the mode of transportation. The following locations were tagged as meta-data: lying open, in hand (can be in use), inside a backpack, a jacket, or a front/side/rear pocket of trousers.

We aimed at creating a broad dataset that features versatile recordings. Therefore, recordings had to be made in different environments. One example for this variety are different weather conditions. Especially for *bike* and *car* journeys the sound changes significantly if it suddenly starts to rain. This is why the dataset contains recordings in different weather conditions from dry to heavy rain, and in some cases, even snowfall. The given weather is noted in the meta data (if no weather tag exists then it was sunny or cloudy and dry).

Bike

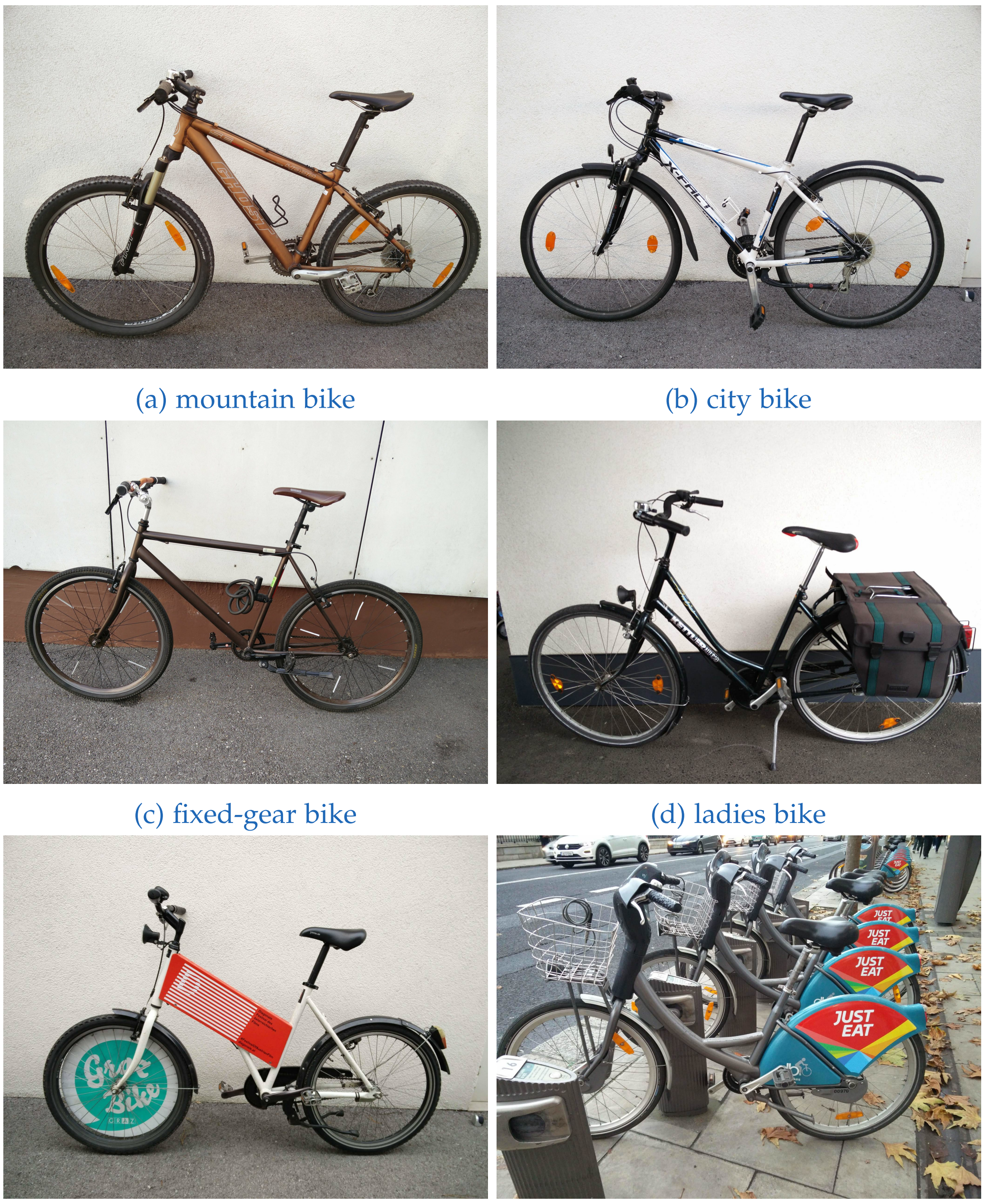
The bike class represents rides using a bicycle. In other words: the activity of cycling. Per our definition of the class, the whole of the journey is to be considered cycling even when waiting at a traffic light. This is due to practical reasons, as it would have been too much effort to stop and restart the recording at every short stop.

The location of the recording device is always a pocket in the cyclists pants. This decision was made to allow for the best pick-up of the typical leg movement that occurs when riding a bike. So the resulting sound-scape will depend heavily on the trousers worn, and which of the pockets was used. Therefore, the type of trousers and a pocket label (front-pocket, back-pocket or lower-side-pocket) are part of the meta data. There are a few recordings in the dataset that were made with the phone in a different location, such as the backpack, which can be used for testing.

The recordings were made on different bikes, in different locations, with different weather, at different times and by different people. The dataset includes rides on cycling paths, on busy city streets, in the countryside, and even up a mountain, and a bit of down-hill. Some recordings were made at night, when there is little to no traffic, and others were done in the middle of the day. Some of the bikes that were used to create the dataset are pictured in figure 4.1. The bikes that were used the most are the mountain bike (4.1a) and the city bike (4.1b). The majority of the data was generated in, and around, the city of Graz, but there are also recordings from riding rental-bikes in Brussels and Dublin (4.1f).

Bus

This class represents the transportation mode of city buses. This includes electric and hybrid buses (as shown in figure 4.2a), but most recordings were made in buses with a combustion engine. The class definition does not require the bus to be moving in order for it to be a valid recording. Thus, a recording can include waiting at a bus-station or at a traffic light.



(a) mountain bike (b) city bike

(c) fixed-gear bike (d) ladies bike

(e) single speed (f) rental bikes

Figure 4.1.: Some of the bikes that were used to record on.

Recordings in this class differ for several reasons. One reason is of course the bus itself because different models produce different sounds. Then of course, it depends on the location inside the bus, especially in double-decker buses such as those found in Dublin, and pictured in figure 4.2b, where the upper deck sounds significantly different from the lower deck.
Another aspect that should not be neglected are other passengers. An empty bus sounds different to a bus full of people who might, in some cases, completely overpower the engine sounds. That is why the dataset contains recordings with different amounts of passengers.

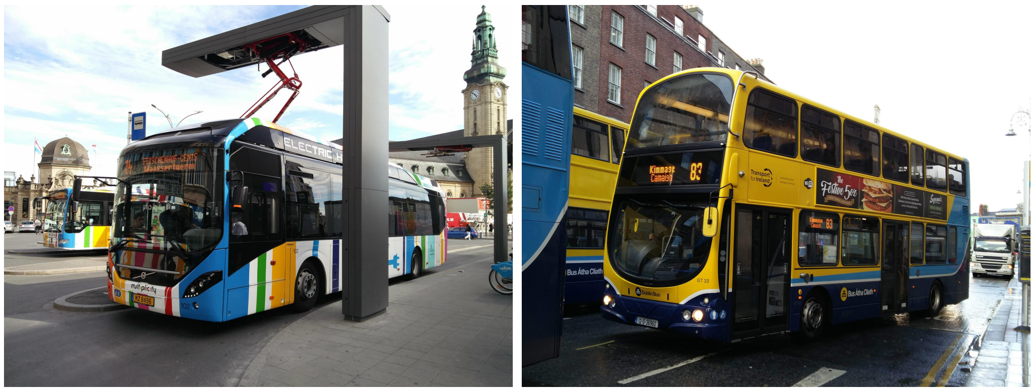

(a) Luxembourg electric hybrid bus (b) Dublin double-decker

Figure 4.2.: Two examples of exotic buses.

Car

Nowadays, one of the most common modes of transportations is the car. The *car* class contains any type of automobile that is used for personal transportation. This also includes taxis (even though our dataset does not contain recordings made in taxis). Racing cars are outside the scope of this class.

There are many different types of cars. The two main factors that contribute to an individual sound-scape of the car are: the engine, and the tires. Our dataset contains recordings of more than ten different models. Some of them had petrol engines (Opel Zafira, Renault Clio, Volvo XC60), some had diesel engines (VW Passat, Skoda Octavia and another Opel Zafira), two were electric cars (Tesla Model S and Tesla Model X) and one was a hybrid (Toyota C-HR).

The one factor that influences the sound inside the car the most is its velocity. The higher the speed, the louder the noise generated by tires on the road (and the engine if driven in highest gear). This is why cars on motorways sound very different from cars in city traffic.
As with other classes, in this dataset the whole of the journey is considered driving the car (even when it is not in motion) which includes standing at traffic lights or pedestrian crossings etc.
Manufacturers try to design cars so there is as little noise in the cabin as possible. This is why it is very hard to classify cars in city traffic (especially electric cars).

Some of the recordings of this class contain music (and have a *music* tag in the meta data). In fact, if a recording has music then it most likely belongs to the *car* class since most people listen to the radio while driving. However, it is not a signal that should be used for classification, because some coaches have music playing, or there might be music in the background of a person walking. Thus, samples which mainly contain music should be filtered out.

Coach

A coach is a comfortably equipped bus used for longer journeys. They usually travel between cities on country roads, or motorways. Judging by the vehicle, the *bus* class is relatively close, however, the resulting sounds inside a coach are very distinct from a bus.

The dataset contains coach journeys recorded in different countries in Europe. Those countries are: Austria, France, Germany, Ireland, Italy and even Luxembourg. Although, it should be mentioned that the country in which the coach is traveling does not influence the sound it produces (at least not to a notable extend). What does make a difference is the model of the bus that is used and different countries have different transportation providers, which have different coaches.

Metro

Several bigger cities in Europe operate metro systems. They consist of underground trains which, in some cases, submerge further outside the city center. Most carriages are on two rails but there are exceptions, such as the rubber-tired metro in Toulouse and some metro lines in Paris.
The distance between stops and the associated travel speeds are higher than for trams and lower than those of trains.

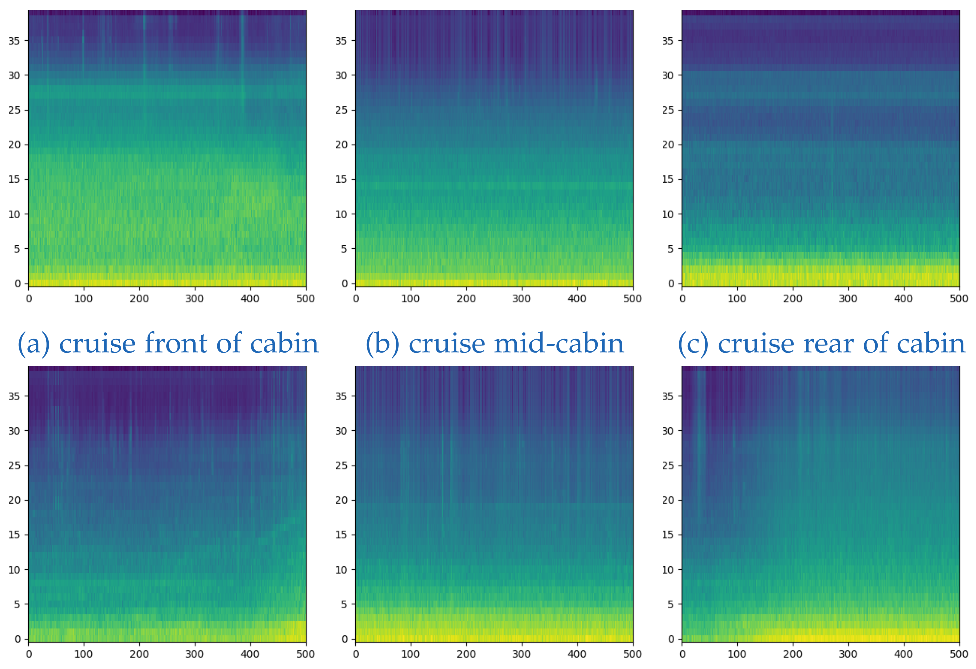
Our dataset contains recordings of metro lines in Hamburg, Paris, Toulouse and Vienna. Often times the sound-scape also varies by metro lines within one city. For instance, the U1 in Vienna sounds significantly different than the U6, because other vehicles are used. This is why the meta data will usually also specify the underground line name.

There are edge cases where it is not entirely obvious whether they should be specified as *metro* or not. One such example would be the Stadtbahn in Stuttgard (as pictured in figure 4.4). In parts of the city, it behaves a lot like a tram in that its rails are on the streets and the stops are fairly close together. However, in the very center of the city it behaves more like a metro, having its own underground tracks. The dataset contains one recording of the Stadtbahn U6 for testing purposes.

Plane

This class represents airborne transport using commercial passenger airliners. The dataset contains recordings from intra-continental flights inside Europe in economy class. The airplanes used on those flights were short- to medium-range airliners (Boeing 737-800 or Airbus A319/A320/A321 in various versions).
Light aircrafts are not part of this class. If they should be added, it is recommended to put them in a separate class.

Different stages of each flight are recorded. The main part is the cruise, but take-off, and landing, as well as rolling on the taxiway are available as separate files.



(a) cruise front of cabin (b) cruise mid-cabin (c) cruise rear of cabin

(d) engine start (e) taxiway (f) turn engine up

Figure 4.3.: A set of typical features of the plane class during different stages of the flight and at different positions inside the cabin.

Since the location inside the cabin comes with a significant noise difference, the recordings were made from different positions. The meta data either contains the row number (usually 1-37), or a tag such as *front of cabin* or *rear of cabin*. A selection of extracted features is shown in figure 4.3 to demonstrate the difference in sound between the individual positions and stages through the flight.

Train

Any train that is operated on two rails and does not belong to any other class (such as *metro* or *tram*) is part of the *train* class. This includes inner-city trains, regional trains, express trains, and fast trains. The class also does not limit the type of traction system. Most trains (especially fast trains) are powered electrically, but others (on routes with no overhead cables) rely on fuel. Technically, locomotives driven by a steam engine also fit in this class, however, the dataset does not contain such recordings.

It is possible to split this class into slow trains and fast trains. The meta data contains the type of train (such as ICE, Railjet, EC, S-bahn) to allow for this. The reason why it is not done by default is that none of the facilitators commute with slow trains on a regular basis. So it was not easily possible to create a large enough sample set to support a separate class.

All recordings were made in trains of central Europe. The following countries are contained in the dataset: Austria (ÖBB), Belgium (SNCB), France (SNCF, Thalys), Germany (DB), the Netherlands (NS), and Switzerland (SBB).

Tram

Most medium to large cities in Europe operate tramways. They are electrically powered by overhead cables and run on rails in city roads. The dataset contains recordings from tram lines in several cities in Europe (Amsterdam, Brussels, Dresden, Graz, Lyon, Paris and Vienna).

As is the case with buses, other passengers can be a significant noise factor. It seems to be the social norm that in buses and trams people speak (often loudly) with each other, take phone calls or even play music on their phones. This is not the case (at least not as much) in public transport with a longer journey time, such as coaches and trains where people are usually quiet as to not disturb others.

There are several tram lines that are somewhat unusual. One example can be found in some trams in Paris. Most tram lines have two rails (as trains do) but these have tires and have only one central rail.
Usually, trams operate on street level. However, line 3 and 4 in the city of Brussels are underground and have underground stations that resemble those of a metro line (as pictured in figure 4.5).
Another example can be found in Lyon. The airport of Lyon is connected to the city with an express tramway. As the name suggests this tram line has fewer stops and travels at higher speeds than regular trams making it sound more like a train than a tram.

The high diversity of the *tram* class, combined with its relatively low share of total time in the dataset, make it hard to classify. Plus, there is another problem concerning the class definition: the vehicle does not need to be in motion. This is fine for buses because there will still be an engine running, but trams do not make any sound if they are halted. To achieve a higher classification accuracy, the recordings would have to be revised.

Figure 4.4.: Stuttgard Stadtbahn

Figure 4.5.: Brussels underground tram

Walking

The *walking* class represents when a person is traveling on foot. Same as for the *bike* class the phone location has to be in a pocket of the trousers of that person. The walking speed can vary, from strolling, to fast marching. However, jogging and running is out of the scope of this class. Climbing and descending stairs also counts as walking.

Recordings for the *walking* class have been made by different people, in different speeds, in different places with different terrain and different environments. Some were walks in cities, others in the woods or the country side. A few recordings were even made when walking on festival grounds with loud background music.

There are some shorter recordings in our dataset that feature walking inside another mode of transportation, such as planes, trains, metros, trams, and coaches. Those are special cases and it has been decided that they should belong to the primary mode of transportation they occur in (not *walking*), since it can happen that one has to walk inside another vehicle. In other words, it is more likely that you are walking in a plane (or respective other transportation mode) than it is that you are in a plane (or respective) when intending to go for a walk. In our dataset each of those classes contain a separate folder `walking` with the recordings of this kind.

4.1.2. The Cities

City	bike	bus	car	coach	metro	plane	train	tram	walk	Total
Graz	4087	302	914	209	0	0	332	107	424	6377
Vienna	0	9	60	0	771	231	448	62	185	1769
Klagenfurt	165	28	716	113	0	0	70	0	18	1111
Brussels	235	77	0	28	43	0	158	22	87	652
Paris	0	5	76	0	120	112	0	52	219	586
Hamburg	0	17	0	0	67	0	272	0	97	455
Amsterdam	0	0	0	0	0	179	18	38	105	341
Lyon	0	0	0	96	0	127	0	28	30	281
Edinburgh	0	50	0	0	0	198	0	0	16	266
Toulouse	0	0	100	0	62	21	0	0	57	240

Table 4.2.: Recording times of each mode of transportation for ten cities sorted by total recording time. All entries in minutes.

The recordings for our dataset were made in different cities across Europe. Table 4.2 lists the top 10 of the cities where most recordings were created. It also breaks down how much of each transport mode was recorded in each city. The top 3 are Austrian cities, because most of the facilitators are Austrians and live in one of those cities. The other cities had to be traveled to.

A recording belongs to a city if the journey started in the given city. The classes *coach* and *plane* only have the city as a place of departure. Some *car* and *train* rides stay within city bounds, but for most of the recordings the vehicle also leaves the city. All other classes usually fully stay within the city.

It can be observed that all of those cities feature recordings of different modes of transportation. Graz has most of the classes covered except *plane* and *metro*, because there is no metro system in Graz. Therefore, there are a lot of *metro* recordings coming from Vienna. There is no active *tram* line in Klagenfurt, otherwise it would have also been covered.
The cities not on the list include Dijon, Dresden, Dublin, Firence, Montpellier, Munich, Rottadam, Stuttgart, and Venice. Those, however, only feature one to three transportation modes each.

4.1.3. Evaluation Set Split

The dataset was split by hand into training and evaluation set. Table 4.3 shows the distribution of training and evaluation files. The aim was to have roughly 10% of the overall data in the evaluation set. However, since some classes (such as *bike*) have very long and repetitive recordings, only a proportionally small amount is required in the evaluation set. Other classes are already short on recordings in the training set (such as *bus*) so we did not take away too many.

Since the split is done on the basis of the recordings, we are bound to recording lengths. However, this also complicates the selection procedure. To ensure that there is a high diversity in the evaluation set, it tends to contain the shorter recordings. Furthermore, we tried to keep the same proportions of characteristics (city, vehicle, phone position etc.) to ensure that the evaluation result is not predominated by any one trait. This is more important than reaching the 10% target. For example, if recordings were made in three different cities and we added recordings with equal

Class	training	evaluation	evaluation ratio [%]	Total
bike	71h 04′	3h 43′	5.0	74h 48′
bus	8h 33′	0h 39′	7.1	9h 12′
car	35h 07′	2h 13′	5.9	37h 20′
coach	6h 59′	0h 27′	6.3	7h 27′
metro	17h 22′	1h 07′	6.1	18h 30′
plane	14h 52′	1h 24′	8.7	16h 17′
train	44h 47′	3h 32′	7.3	48h 19′
tram	6h 29′	0h 53′	12.1	7h 23′
walking	23h 10′	3h 11′	12.1	26h 21′
Total	**228h 27′**	**17h 13′**	**7.0**	**245h 40′**

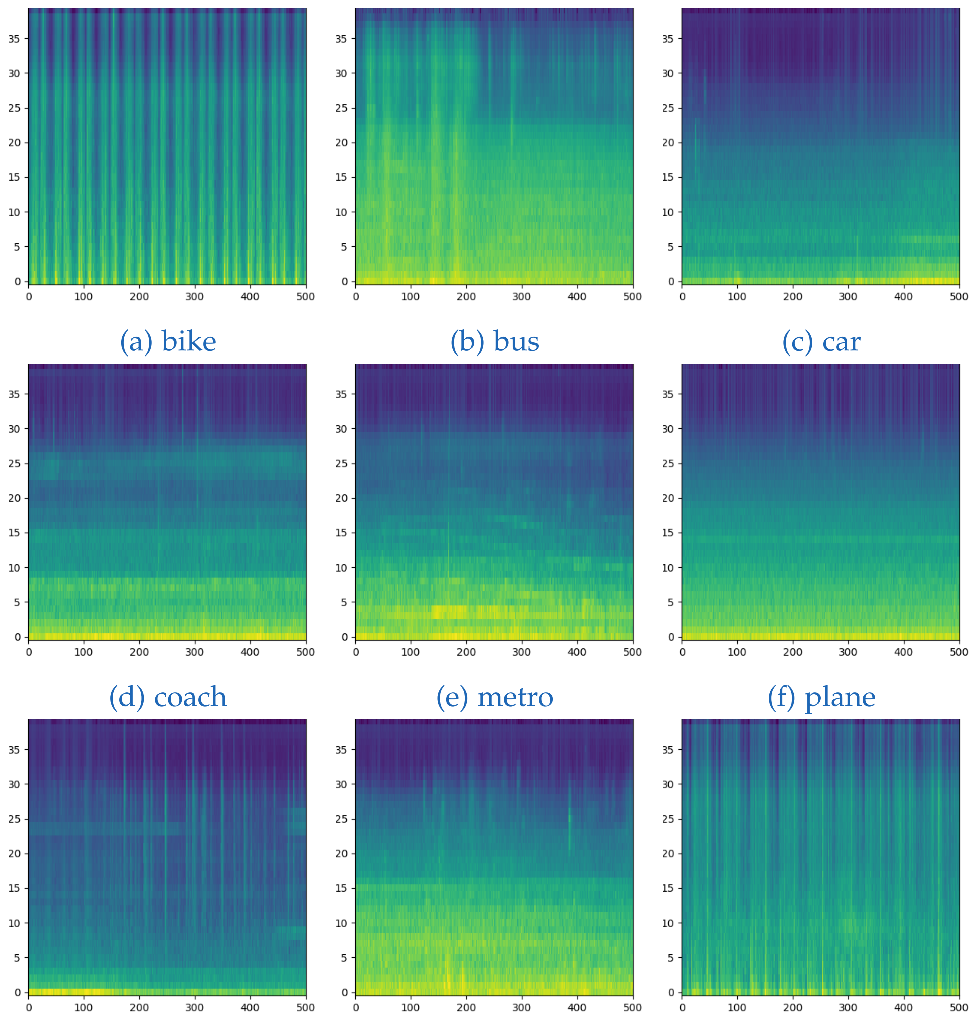
Table 4.3.: The split of the dataset into training set and evaluation set. The recording times and the resulting percentage are listed for each class as well as for the overall dataset.

length from each city but they amount to only 5% and there are only long recordings left, then we do not add any more to the evaluation set, even though we could add another long recording, but then the ratio of cities in the evaluation set would be biased.

Finding an ideal split is not a trivial task. It is always a matter of balancing the training and evaluation set. The presented approach results in an appropriate split for our proposes.

4.1.4. Log Mel-band Energies

The features of one representational sample of each class are displayed in figure 4.6 for a side-by-side comparison. It can be observed that *car* and *train*



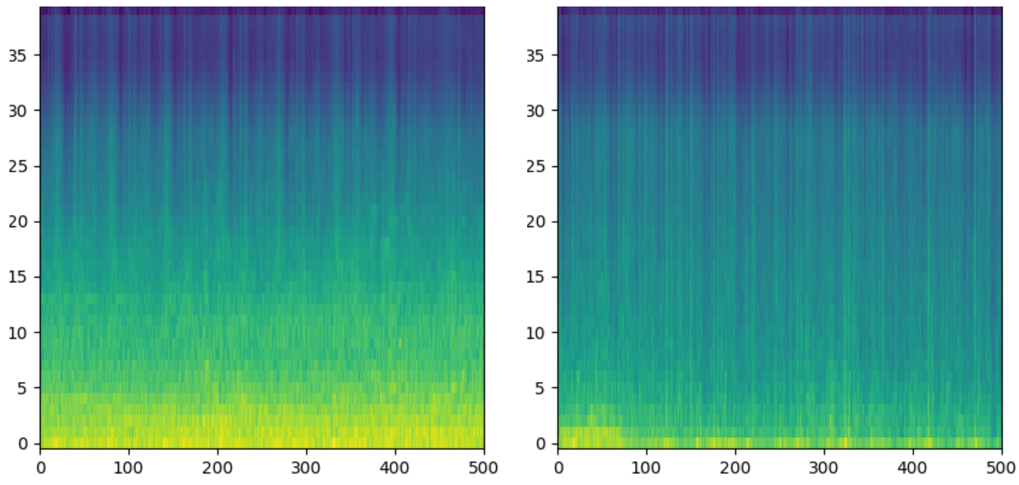
(a) bike (b) bus (c) car

(d) coach (e) metro (f) plane

(g) train (h) tram (i) walking

Figure 4.6.: A set of typical examples for features of the 9 classes for direct comparison.

are the most quiet of the classes. Samples in *plane* have a uniform spectrum, because the sound is always the same during the 10 seconds. Inside a *bus* or a *metro* there are more changes in the noise, which can also be observed in the features. *Walking* and *bike* have periodical signals. For these examples, *walking* has the longer period, suggesting that two steps take more time than one rotation of the pedals on the *bike*. Since we know one sample is 10 seconds long the step speed can be derived (this person made 18 steps in 10 seconds, and the cyclist did 12 full rotations).



(a) walking in a plane

(b) walking in a train

Figure 4.7.: The features of examples for walking inside another mode of transportation. The left image shows walking in a plane and the right one walking in a train.

The features of the special cases of walking inside a plane or a train are displayed in figure 4.7. As one would expect, they are a combination of what has been shown above. There are the periodical walking steps but also a higher background noise level. The steps are not as clearly visible anymore, but it can still be deducted that the person in the plane walked 16 steps in 10 seconds and the person in the plane did approximately 19.

More plots of log-mel band energies of interesting cases can be found in the appendix.

4.2. Sample Length

To figure out which sample length is appropriate for our given TMD dataset and *the algorithm*, several sample lengths have been tested. For this experiment the dataset was slightly reduced to contain only 69 hours, that is 28%

of the total recordings (the exact procedure is described in section 3.2.3). This was done to ensure an equal distribution of samples between classes, and to remove ambiguous cases (like walking in a plane). To ensure that the evaluation results are comparable, the evaluation samples are always extracted from the same recordings. Those recordings were handpicked to ensure that the evaluation set consists of a variety of different scenes.

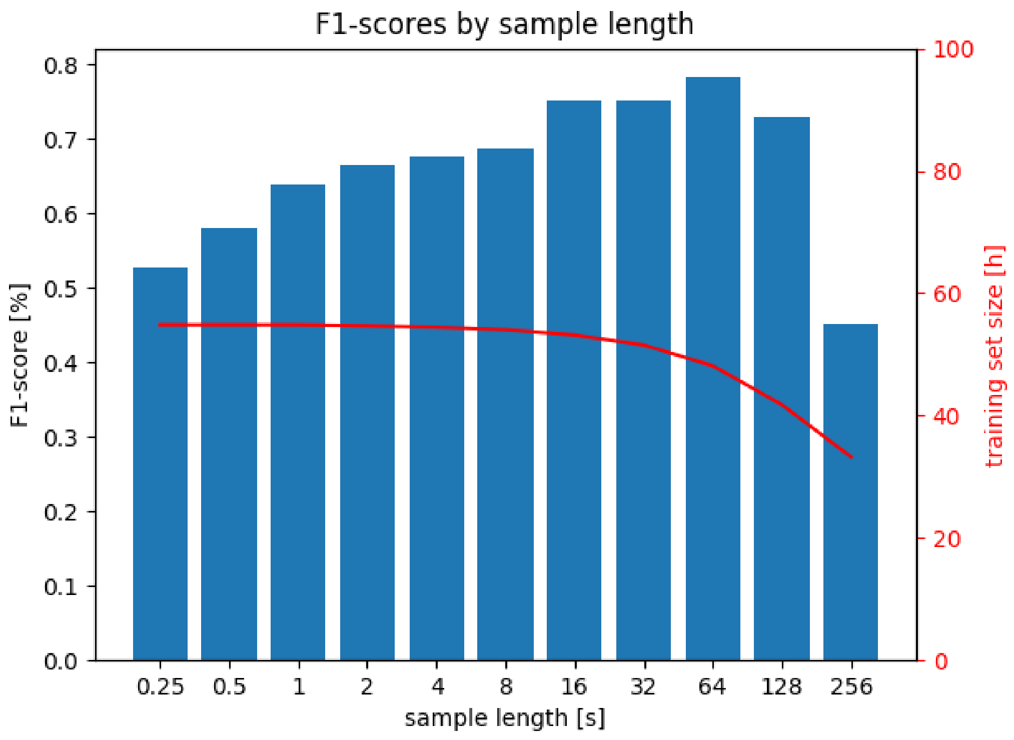


Figure 4.8.: The F1-scores of evaluation runs after training with different sample lengths. The corresponding training set size in hours is plotted in red.

In a first rough analysis the samples lengths are selected from a logarithmic scale. Specifically, starting with a base of one second, multiples of two are analyzed in either direction. The resulting F1-scores, which can be seen in figure 4.8, show that the algorithm performs better the longer the sample lengths are. At one point (128 seconds) the F1-score decreases, but that is simply due to the reduced size of the dataset. With the given limitations (see section 3.2.3) fewer samples are extracted with larger sample lengths. The combined length of all samples used in each training can also be obtained from figure 4.8. At larger sample lengths some classes will not have enough samples to be properly represented, which is why the F1-score drops.

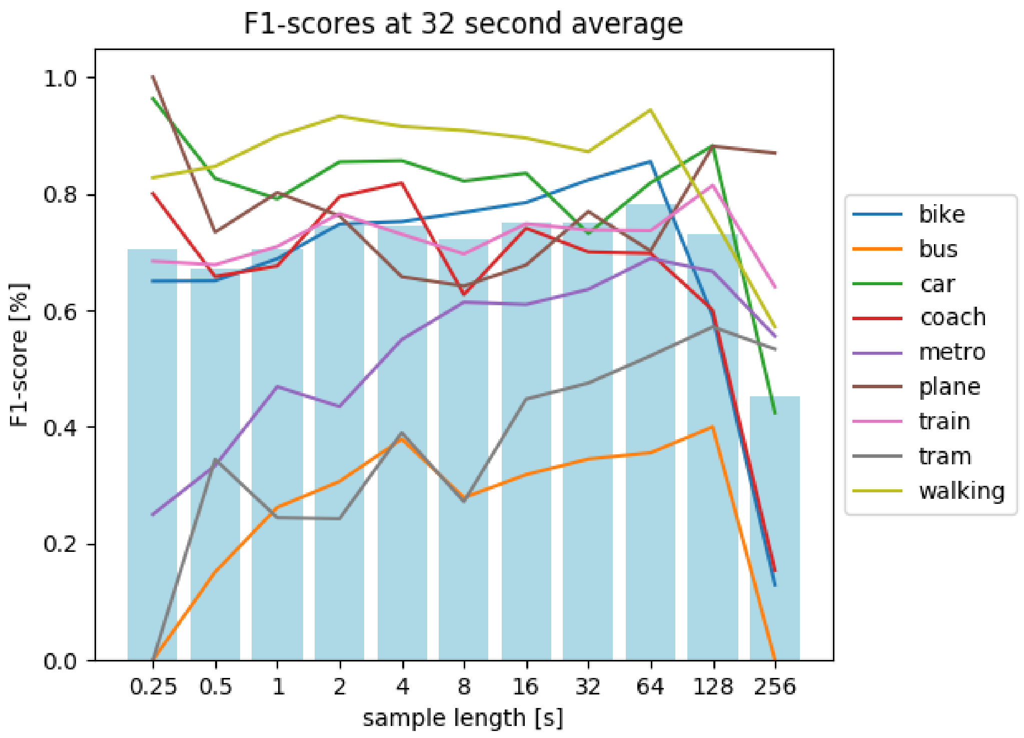


Figure 4.9.: The individual F1-scores of different sample lengths with a majority vote on 32 second batches. Each line represents the F1-scores of one class and a weighted average F1-score is shown by the bar plot.

To conclude that the longer the samples the better the result (even given long enough recordings), would be a misinterpretation. One also needs to consider that in the time it takes to perform the classification with a longer sample, several short samples can be classified. In order to account for this, a majority vote for samples within 32 seconds is applied (so for 1s samples a best of 32 samples is performed, for 2s a best of 16 and so on). Figure 4.9 shows the results.

It can be observed that starting at two seconds and up to a minute the weighted F1-score (shown by the light-blue bars in the background) does not change very much. There is one outlier at 8s, but this might just be due to the validation set selection, or other random factors during the training. Figure 4.9 also contains plots of the F1-scores of the individual classes. It can be noted that transport modes such as *plane*, *coach*, and *car* which have a distinct background noise, are correctly classified at very low sample lengths and their F1-scores tends to decrease at longer samples, due to the influence of the other classes before it recovers again.
Other classes, such as *bike* and *walking*, which produce periodic sounds with a period of 1-2 seconds, tend to increase in accuracy at least up until the point where the period can be detected inside a sample (around 2 seconds). That the *bike* class then still increases (whereas *walking* decreases) is probably due to the fact that less of the other classes are misclassified as *bike* (see confusion matrix in chapter 4.3).
The classes *bus*, *metro* and *tram* gain the most with an increased sample length. Those are also the classes with the most discrepancies in the recordings, due to their stopping in stations. However, *bus* and *tram* are also the classes with the fewest samples so that might be another reason for the comparatively low F1-scores.

Additional trainings were performed to have a closer look at possible sample lengths. This time a linear scale was chosen with an interval of two seconds up to a maximum of 20 seconds. The resulting F1-scores can be obtained from figure 4.10. It can be observed that there is a slight increase for all the classes with longer samples. The *plane* class performs worse at first but then recovers (as also seen on the logarithmic scale). The biggest imprecision is due to the *coach* class which oscillates quite a bit. The overall best result is produced with a sample length of 16 seconds. It has been ensured that this is not just a random outlier by performing the training several times (and observing similar results).To see whether this is because of the validation set (which stays if the samples do not change), or because of other parameters in the training (maybe it works well with the given sequence length or pooling size which both had to be adjusted to the sample length) would require more experiments to be run. However, the gain is not significant enough to justify further analysis.

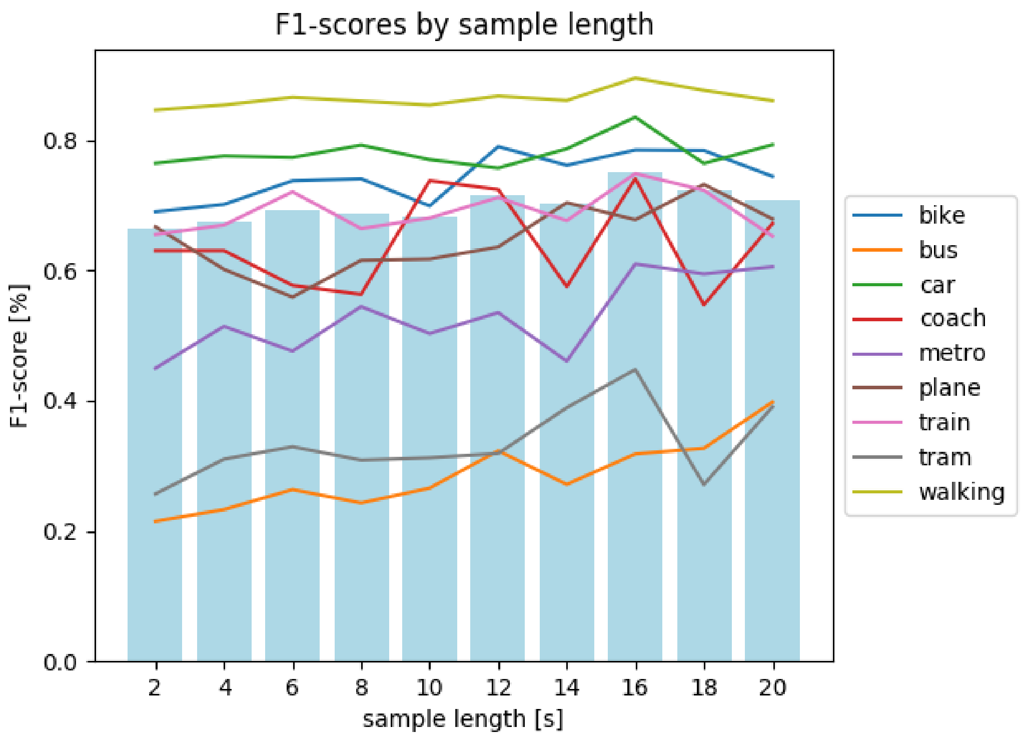


Figure 4.10.: The individual F1-scores of different sample lengths on a linear scale.

Decision

We had to decide which sample length to proceed with. All the following experiments are run with that sample length. The overall best score was provided with a sample length of one minute (64 seconds to be precise). However, having such a long sample length is impractical for our usecase, since we do not want a user to have to wait a minute before the application displays the first result. The little gain in accuracy is not worth the loss in usability.
When it comes to smaller sample lengths the best performance was at 16 seconds. This is also the sample length we would have selected if we were using only our dataset. Other researchers have used sample lengths of 10 seconds in their experiments (such as Mesaros et al. [18] for the DCASE

challenges). So, if we wanted to include their samples in our experiments we could have a maximal sample length of 10 seconds. Since there is no dramatic difference in F1-scores we decided to apply their sample length and go with **10 seconds**.

4.3. Starting Point

		Prediction								
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	98.9	0.0	0.0	0.0	0.2	0.2	0.2	0.0	0.7
	bus	24.5	23.1	3.9	0.0	24.0	0.4	20.1	0.4	3.5
	car	8.6	0.3	83.1	0.5	0.6	1.5	5.2	0.1	0.1
	coach	19.8	6.6	12.0	42.5	0.6	0.0	18.6	0.0	0.0
	metro	34.3	0.2	0.7	0.0	51.7	1.5	9.7	1.2	0.5
	plane	19.5	1.7	2.5	0.0	4.7	67.6	3.9	0.0	0.0
	train	18.2	0.4	1.9	0.2	0.8	1.2	72.2	1.5	3.6
	tram	47.3	3.2	8.6	0.3	5.4	5.1	12.5	15.7	1.9
	walk	19.7	0.0	0.3	0.0	1.9	0.0	0.4	0.1	77.7

Table 4.4.: The confusion matrix of the classes of our dataset when all the samples are used. All values in percent. The same confusion matrix with absolute values can be fond in the appendix.

Now that we had decided upon a sample length, all samples could be extracted. This is done following the procedure described in section 3.2.3. The restrictions do not apply anymore, so for each file as many samples as would fit into the recording are extracted. Henceforth, those 10 second samples are the samples we are talking about if we mention *our samples*, *the samples* or simply *samples* (unless specified otherwise).

A normal training is performed using all our (training) samples on *the algorithm*. When testing all samples from the evaluation set, the resulting F1-score is 72%. This is the baseline we will be comparing against, when we reduce the sample set, change classes, or perform preprocessing in the following sections.

A confusion matrix can be found in table 4.4. Each given percentage indicates the amount of samples of a class that were predicted as the charted class. In other words: which classes it was confused as (thus confusion matrix). Furthermore it is color coded: the diagonal elements (which represent the correct predictions) change their shade of green to be more saturated the closer the value is to 100%, and all other entries (which represent mispredictions) become more red if the value increases.

The most obvious discovery to be made from this confusion matrix is that most mislabeled samples are predicted to be of the *bike* class. The *bike* class itself has a very high amount of true positives (almost 99%), but all the false positives reduce the F1-score of the class significantly. It has become a so-called *trash* class, because it is what the classifier picks if it does not know where to put it. We will deal with this issue in section 4.6.

Runtime

On a Tesla P100 (which is a GPU designed for data centers) with 16GB of memory and 9.3 teraFLOPS single-precision performance, the training process with the full dataset takes 3 hours and 40 minutes. That is a bit more than one minute per epoch.
Creating the features also takes several hours but that only needs to be done once.

4.4. Dataset Size

This section attempts to find the appropriate size for the dataset. Different sizes are tested, both in terms of the lengths and the number of recordings. Since we only have one given dataset, we can only create smaller subsets from it. In the following experiments the training set is reduced but the evaluation set is kept the same to assure comparable results.

4.4.1. Number of Samples

The first experiment which was conducted reduced the dataset by samples. This means that random samples were selected to be removed from the dataset. The impact this has on the performance of *the algorithm* is shown in figure 4.11. It is worth noting that the the x-axis has a logarithmic scale: from right to left the dataset is halved in each step.

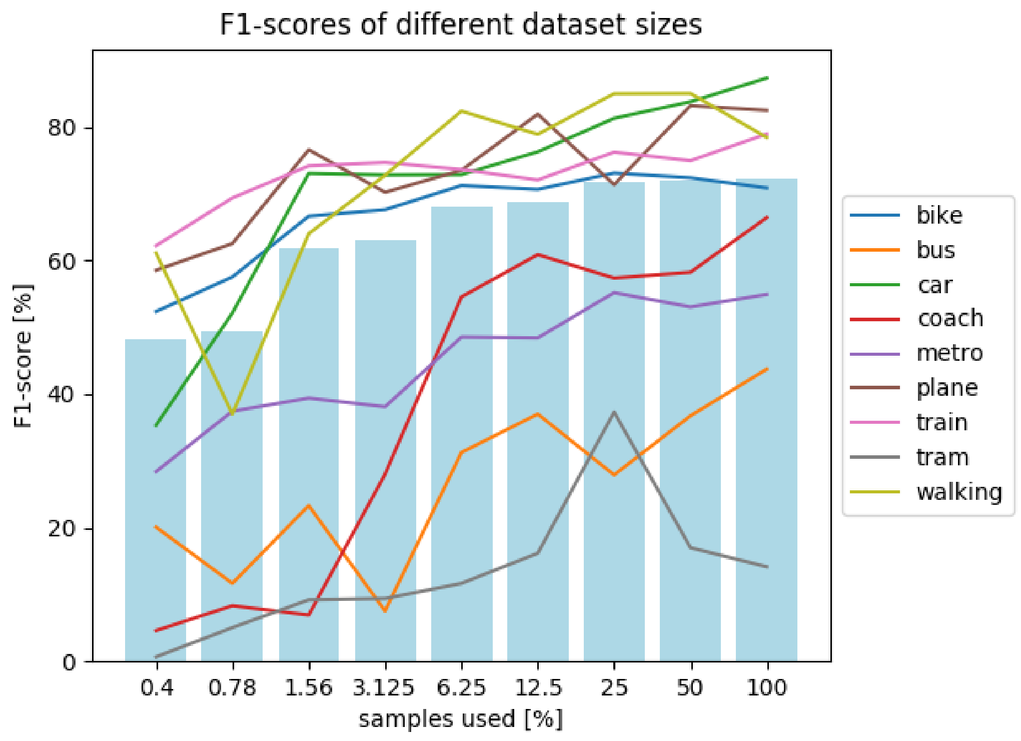


Figure 4.11.: The F1-scores of the individual classes when the samples of the dataset are reduced.

It can be observed that up to 75% of the samples can be removed without causing a noteworthy drop in performance. The F1-scores of the individual classes change slightly (in particular *car* and *plane* are worse off), but the overall score stays level.
The dataset can be further reduced down to around 5% until the performance really starts to drop. This is best seen by the trend of the *coach* class (red line in figure 4.11), which increases significantly between a dataset size of 1.56% and 6.25%.
Even when 99.6% of the samples have been removed the F1-score is still at 45%. Three of the classes, namely *bus*, *coach* and *tram*, cannot be predicted correctly anymore but the others (especially *bike*, *plane* and *train*) can still be told apart to a large extend.

This means that for applications with strict memory requirements a reduced subset of our dataset would suffice. Also, if we were to publish the dataset it would be a possible to use only a quarter of the samples to reduce the required storage space (and the connected transmission times) from over 80GB down to around 20GB.

The reason as to why so many samples can be removed without losing performance is that our recordings are quite repetitive in nature. So a 10 minute recording will have a lot of very similar samples. To see if this lack of diversity exists for the full length of a recording, (such that the recording could be trimmed) or just within parts of the recordings, requires further analysis.

4.4.2. Length of Recordings

It was established that a lot of samples (around three quarters) can be removed from the dataset without notably reducing the diversity and the thereby associated classification accuracy. The next step is to figure out whether those samples can be removed in a certain pattern, namely, at the end of each recording. If this change provides similar results (compared to when samples are removed at random), then this would mean that the recordings in the dataset are unnecessarily long.

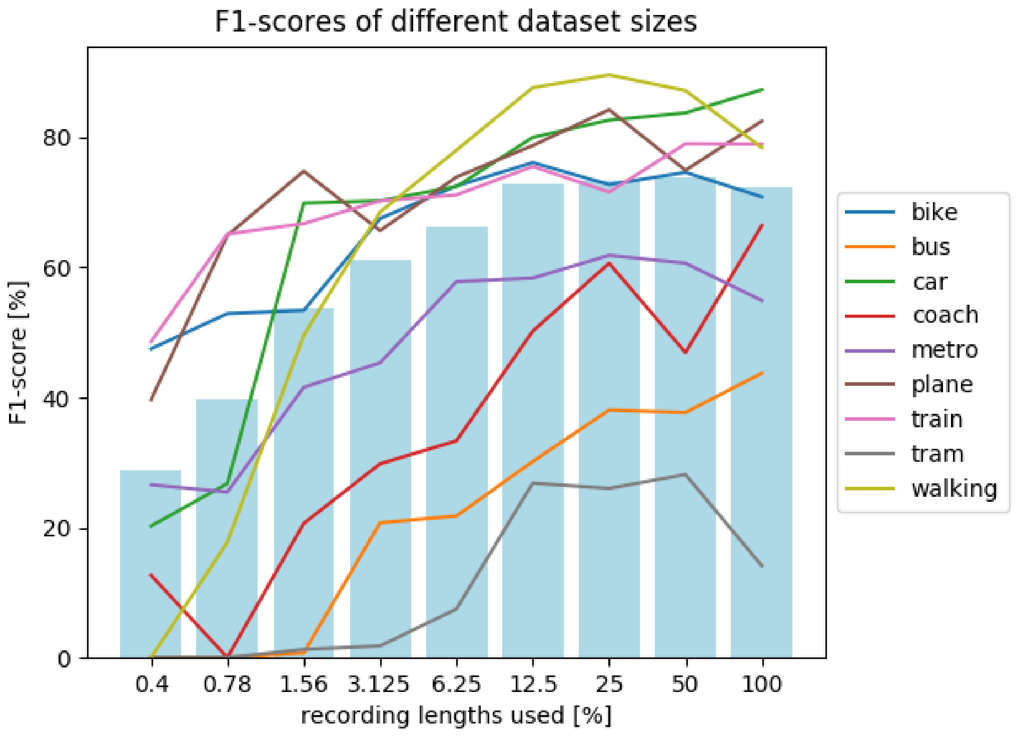


Figure 4.12.: The F1-scores of the individual classes when the recording lengths of the dataset are reduced.

The datasets for this experiment were created by removing a certain percentage of samples at the end of each recording of the full dataset. The resulting scores are displayed in figure 4.12. This plot is comparatively smoother than the previous one because the recording length was shortened incrementally (ie. the same samples have been removed).

Similar to the previous experiment, the F1-scores are not significantly affected when up to 75% of the recording length is removed. Beyond this point the scores drop more drastically than if samples are reduced at random. Some classes (eg. *tram*) decline earlier than others (eg. *car* or *plane*).

The average length of recordings in the dataset is 16 minutes (see table 4.1), and 17 minutes for the training set only. This means that even with average recording lengths of $\frac{17}{4} = 4.25$ minutes the results would not differ significantly.

Some classes (such as *walking* and *metro*) seem to suggest that recordings that are too long hinder performance. However, it is more likely that some recordings of those classes just require trimming at the end of the file (see section 4.8.2).

4.4.3. Number of Recordings

This experiment tries to answer the question of how many recordings are required to achieve a certain performance. Therefore, the number of recordings in the dataset was reduced. The selection of recordings happened at random and with the full dataset at each step. So there is no incremental removal to avoid falsifying the results, if essential recordings are removed early on.

The F1-scores of each simulation (as well as those of the individual classes) can be obtained from figure 4.13. The x-axis in this plot has a linear scale (as supposed to the logarithmic scale in the two previous figures).

It is worth noting that in this experiment the classes *bus*, *coach* and *tram* were exempt from the removal. This is because they do not contain many recordings. If they were included then their accuracy would drop too quickly, which in turn would negatively affect the F1-scores of the other classes.
This is why in figure 4.13, the graphs for those classes do not seem to be majorly affected. The *tram* score even decreases with a bigger dataset (possibly because the ratio of the already too few *tram* recordings decreases).

The *metro* class, which would have been the next on the list to expect, does show a significant decline when only one sixteenth (6.23%) of the recordings are used. So do the other classes that have been reduced, except *bike*, because it is our *trash* class.

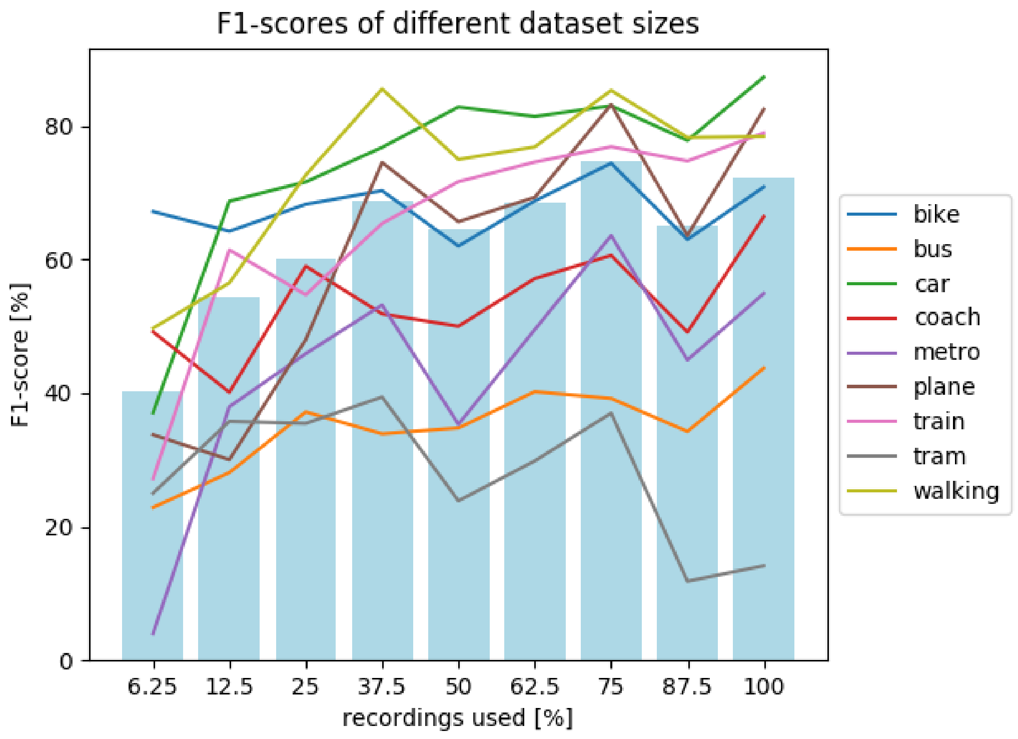


Figure 4.13.: The F1-scores of the individual classes when the recordings of the dataset are reduced.

With 75% of recordings removed, the prediction capabilities are already notably restricted. However, with only 25% of the recordings removed, the precision actually increased, which suggests that there are recordings in the dataset which are either incorrectly labeled, or otherwise negatively influence the training procedure.

Given the graph at hand it is hard to tell if more data would further improve the results. The fluctuation between tests is too high to predict a trend.

However, it is unlikely that more recordings (of similar quality) will worsen the classification results.

Conclusion

Based on the data provided we would not suggest removing recordings from the dataset. In some cases, it might be appropriate to thin out the dataset by removing random samples throughout recordings. It has been shown that the recording length can be reduced to around five minutes. What is more important than the length, is the diversity between recordings. This should be considered for future data creation.

4.5. Number of Classes

The decision of how many and which classes to choose can influence the classification capabilities significantly. This section examines three different approaches that change the number of classes used. At first, existing classes are merely removed, then classes are merged to reduce the number of classes, and finally classes are split to increase the number of classes.
The goal of this analysis is to illustrate the ratio between the number of classes and the classification scores, as well as provide an overview on how different classes influence each other.

4.5.1. Excluding Classes

The simplest way of changing the number of classes is by simply leaving out some existing classes. This experiment reduces the number of classes from all nine to two. If a class is not removed then all of its samples are being used.

Only a subset of possible combinations has been tested. Since for each number of classes c there would be $\frac{9!}{c!(9-c)!}$ possibilities. A selection has been made to cover a range of different strategies: combining only small/big classes (in terms of number of samples), combining classes that are similar (eg. *bus* and *coach*) or very different (eg. *bike* and *plane*). The strategy for 4-7 classes was to have two different sets and always add one class to each set. With 8 classes all combinations (removing each class once) were tested.

Table 4.5 documents the resulting F1-scores of each of the tests. When analyzing the total scores (in the right-most column), it can be said that *in general* the score worsens, the more classes are tested.

The table helps identify which classes have clear decision boundaries and which ones might be overlapping. For example *bike* and *plane* can be clearly distinguished which results in high F1-scores. Even *bike* and *tram* are fairly different. When testing the two classes *bike* and *walking*, however, more confusions are recorded. The same is true for the combinations *metro* and *train*, and *bus* and *coach*. *Bus* and *tram* yield the worst results of the tested 2-class combinations.
By analyzing 3-classes in one test in can be shown that *car*, *plane*, and *walking* are far apart on the vector space (and can therefore be easily distinguished), whereas the three inner-city public transport modes (*bus*, *metro*, and *tram*) are more easily confused by the classifier.

The middle part of table 4.5 shows two sets of classes (for 4-7 classes always upper row and lower row) and how the F1-scores develop when additional classes are added. The sets start out with 4 classes each and are increased to the total of 9 classes (last row). It is shown that in most cases the score decreases with each added class. However, if a class is added that has a lot of samples and does not influence other classes significantly (such as *car* or *walking*), then the F1-scores could also increase with more classes. This does not mean that the accuracy of the individual classes is improved but in total proportionally more samples are predicted correctly.

Another noteworthy observation can be made when analyzing the relationship between *tram* and *train*. The two columns can be found next to each other in table 4.5. They show that the the score for *tram* is significantly higher if the *train* class is not present. A similar behaviour (but not as extreme) can be observed for the *coach* class. Those findings coincide with the data from the confusion matrix (table 4.4) which shows that *coach* and *tram* are often labeled as *train*.

	bike	bus	car	coach	metro	plane	train	tram	walk	Total
2 classes	99.7					99.4				**99.6**
	98.3							92.1		**97.1**
	94.6								93.0	**93.9**
					82.7		95.2			**92.3**
		87.8		79.2						**84.2**
		63.9						73.7		**69.5**
3 cl.			97.1			97.9			98.8	**98.0**
	94.1		90.1				90.4			**91.8**
		63.4			78.2			66.9		**70.9**
4 cl.	97.0			71.1		92.4		76.5		**91.4**
		58.2	92.8		80.9				96.8	**89.6**
5 cl.	95.7			79.9	75.8	91.6		62.5		**87.4**
		37.5	88.0		72.0		83.4		93.2	**83.3**
6 cl.	94.7	47.2		73.7	66.0	85.5		55.2		**80.3**
		33.4	86.9		65.3		83.6	30.3	91.3	**77.9**
7 cl.	90.9	47.1	87.7	68.4	66.8	88.4		53.5		**80.7**
		22.1	86.2	52.1	61.8		80.1	23.2	90.1	**74.0**
8 classes		19.9	82.1	51.0	51.4	82.9	79.8	21.5	90.9	**73.5**
	72.3		78.1	50.0	62.7	73.8	74.6	20.5	77.6	**70.8**
	77.5	44.9		52.2	63.1	86.9	81.4	32.1	84.0	**75.1**
	69.6	28.7	85.1		53.6	84.4	76.7	6.6	82.7	**71.3**
	78.6	41.0	86.2	49.4		82.8	77.6	24.9	87.1	**76.3**
	74.5	35.9	85.2	60.8	56.4		79.2	20.1	85.2	**73.0**
	84.2	55.3	88.8	68.0	64.2	91.0		50.9	89.2	**81.4**
	74.7	33.3	87.1	55.4	61.0	82.4	80.2		79.4	**76.3**
	75.3	31.0	81.9	65.6	61.7	82.0	78.7	19.3		**71.3**
9cl.	70.9	43.7	87.3	66.4	54.9	82.5	79.0	14.1	78.4	**72.3**

Table 4.5.: This table lists the individual and total F1-scores (in percent) of evaluations with a reduced number of classes used. Each row represents one evaluation. If a cell is empty that means that the corresponding class was removed.

The large influence of the *train* class is also indicated by the block of 8-classes, where the best result is achieved when the *train* class is removed.

4.5.2. Merging Classes

Merging classes can be a helpful strategy to improve the classification results. It comes with the disadvantage that the results are not as detailed (because there will be fewer classes), but it has the potential to increase accuracy and depending on the use-case that might be of higher importance.
However, the advantages only take effect when the merging is done right. This section discusses merging strategies by the means of performing tests on our dataset.

Not all merging improves the performance. This can be demonstrated by merging two classes which were proven to be fairly independent, such as *car* and *plane*. A confusion matrix can be found in table 4.6. The total F1-score of the evaluation after the algorithm was trained with the merged classes is 74.7%. This is only a minor improvement to the 72.4% that would be achieved if all classes are kept in the training and the merge happened on the evaluation results.

That merging two classes can also have negative effects is shown by merging *bus* and *plane*. Table 4.7 compares the two confusion matrices, once with the merge before, and once after the training. It can be observed that, even though the number of true positives is higher if the classes are merged before training, the number of false positives also increases. In total the F1-score decreases from 72.4% (training with all classes) to 71.9% (training with *bus* and *plane* merged).

		Prediction							
Actual		bike	bus	car & plane	coach	metro	train	tram	walk
	bike	99.3	0.0	0.2	0.0	0.1	0.0	0.0	0.5
	bus	24.9	27.9	2.6	0.0	23.1	16.6	1.3	3.5
	car & plane	6.4	0.3	87.5	0.1	0.8	4.5	0.3	0.1
	coach	11.4	19.2	18.6	47.3	0.0	3.6	0.0	0.0
	metro	31.3	0.5	1.2	0.0	50.7	13.9	2.2	0.0
	train	14.7	0.6	5.8	0.0	1.1	74.5	0.8	2.5
	tram	42.5	1.3	12.1	0.3	7.0	22.0	13.4	1.3
	walk	19.8	0.0	0.3	0.0	3.4	0.4	0.0	76.3

Table 4.6.: The confusion matrix when the classes car and plane are merged.

	Prediction when trained with merged classes							
Actual	bike	bus & plane	car	coach	metro	train	tram	walk
bike	99.2	0.5	0.0	0.0	0.2	0.1	0.0	0.2
bus & plane	15.2	69.7	1.9	0.1	6.9	5.1	0.8	0.3
car	12.1	2.5	77.9	0.4	0.0	6.3	0.8	0.0
coach	20.4	13.2	6.6	53.9	1.2	4.8	0.0	0.0
metro	30.6	9.2	1.0	0.0	44.3	14.4	0.5	0.0
train	15.7	2.9	3.0	0.4	0.1	75.8	0.4	1.7
tram	43.5	8.0	7.3	0.3	5.8	27.2	6.7	1.3
walk	28.8	0.0	0.1	0.0	1.1	0.4	0.0	69.7

	Prediction when trained with all classes							
Actual	bike	bus & plane	car	coach	metro	train	tram	walk
bike	98.9	0.2	0.0	0.0	0.2	0.2	0.0	0.7
bus & plane	20.8	57.2	2.9	0.0	9.8	8.2	0.1	0.9
car	8.6	1.8	83.1	0.5	0.6	5.2	0.1	0.1
coach	19.8	6.6	12.0	42.5	0.6	18.6	0.0	0.0
metro	34.3	1.7	0.7	0.0	51.7	9.7	1.2	0.5
train	18.2	1.6	1.9	0.2	0.8	72.2	1.5	3.6
tram	47.3	8.3	8.6	0.3	5.4	12.5	15.7	1.9
walk	19.7	0.0	0.3	0.0	1.9	0.4	0.1	77.7

Table 4.7.: The confusion matrices when the classes bus and plane are merged. The upper table shows results after a training with the merged classes whereas the lower table shows the result of a training with all classes but the entries in the matrix of bus and plane have been combined.

The correct way to merge classes is by combining classes which are inherently similar. An example would be all classes that operate on rails (in our dataset those are *metro*, *tram* and *train*). Doing so yields a weighted F1-score of 81.8% (as compared to 74.9% when merged after training with all classes). When, additionally, the two bus classes (*bus* and *coach*) are merged the F1-score increases to 83.2% (75.1% without retraining). This step is only a small gain, but it does not cause additional misclassification. As shown in the confusion matrix (table 4.8), most of the false positives (especially those from the merged *busses* class) belong to the *rails* class (those in *bike* can be ignored for reasons that will be discussed in section 4.6).

		Prediction					
		bike	busses	car	plane	rails	walk
Actual	bike	98.6	0.0	0.0	0.1	0.3	1.0
	busses	9.3	41.7	6.1	0.0	39.6	3.3
	car	6.6	1.3	82.9	0.3	9.0	0.0
	plane	10.1	0.0	1.4	71.5	16.5	0.5
	rails	8.7	0.9	0.9	0.6	86.1	2.9
	walk	15.1	0.1	0.1	0.0	1.4	83.3

Table 4.8.: The confusion matrix when the classes bus and plane are merged. The upper table shows results after a training with the merged classes whereas the lower table shows the result of a training with all classes, but the entries in the matrix of bus and plane have been combined.

4.5.3. Splitting Classes

To increase the number of classes in our dataset (without using external data), the existing classes have to be split into subclasses. The precondition for splitting is that there is enough meta data available to associate each recording of a class to either one of those subclasses.
Splitting a class into two subclasses can have three outcomes (in different

manifestations thereof). The best case is that the two classes can be cleanly separated by the classifier. This is when it makes sense to split a class, so it is what we are aiming for. The second case is when there is one dominant class that the second class is confused with, but which itself is not mistaken for the second class. The third possibility is that both classes influence each other with false positives on both sides. This is usually a sign that the classes cannot be separated and the split should not be applied.
This section demonstrates all three cases by means of splitting some of the 9 classes in our dataset into subclasses.

		Prediction								
		bike	bus	car	coach	ecar	metro	plane	train	...
Actual	bike	99.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	...
	bus	22.3	22.3	0.4	0.0	0.9	23.6	0.0	21.4	...
	car	9.0	0.7	80.7	0.5	0.0	0.4	1.8	6.8	...
	coach	29.3	26.3	2.4	37.1	0.6	0.6	0.6	3.0	...
	ecar	39.3	4.9	26.2	0.0	26.2	0.0	1.6	1.6	...
	metro	39.8	1.2	0.0	0.0	0.0	44.3	0.5	12.7	...
	plane	24.1	0.6	2.8	0.2	0.0	1.1	67.2	3.8	...
	train	15.7	0.9	1.7	0.2	0.3	0.4	0.7	76.3	...
	tram	53.0	2.9	1.6	0.0	1.0	3.5	7.7	21.1	...
	walk	21.0	0.0	0.0	0.0	0.0	1.0	0.1	0.4	...

Table 4.9.: The confusion matrix when samples from electric cars are split from the car class into a separate ecar class. The columns tram and walk have been omitted because neither of the car subclasses were confused with them.

A dominant subclass frequently emerges when the subclasses are difficult to distinguish and the other subclass has significantly fewer samples. An example in our dataset would be splitting the *car* class into electric cars (*ecar*) and regular (non-electric) cars (*car*). The confusion matrix (table 4.9) shows that *ecar* (which has fewer samples) is confused with *car* but not vice

versa. The accuracy of electric cars being correctly identified as cars is only around 50% and then there is another 50%/50% chance that a sample is put in the wrong subclass. Therefore, it is not advisable to keep the subclasses separate.
The same is true when suburban trains are split from the *train* class. The prediction accuracy of the subclasses are even worse in that case (details can be found in the appendix).

		Prediction							
		bike	...	easyjet	metro	plane	train	tram	walk
Actual	bike	98.6	...	0.0	0.2	0.2	0.0	0.0	0.9
	bus	18.8	...	0.0	23.1	0.0	17.9	0.0	5.2
	car	10.7	...	0.0	0.0	1.6	9.6	0.0	0.0
	coach	26.3	...	0.0	0.0	1.8	4.8	0.6	0.0
	easyjet	15.4	...	52.8	0.0	30.9	0.0	0.0	0.8
	metro	27.4	...	0.2	58.0	2.0	10.9	0.7	0.2
	plane	25.0	...	21.7	3.7	44.9	3.1	0.2	0.2
	train	15.8	...	0.3	0.5	0.3	75.2	0.6	4.7
	tram	48.2	...	0.3	10.5	1.0	16.0	12.8	2.2
	walk	17.1	...	0.0	1.1	0.0	0.4	0.0	81.5

Table 4.10.: The confusion matrix when samples from easyjet flights are split from the plane class into a separate class. The columns bus, car and coach have been omitted because neither of the plane subclasses were confused with them.

When two equally sized subclasses are hard to distinguish, then in most cases both of their accuracy scores suffer from it. This is the case, when in our dataset the *plane* class is split into plane rides with the operator EasyJet and all others. In the confusion matrix (table 4.10) it results in a square, where one diagonal are the correct predictions and the corners of the other diagonal are the confusions with the respective other subclass. Even though both subclasses are confused with each other, the majority of samples is still

predicted correctly (53% for *easyjet* and 45% for others (*plane*)).
Whether or not to keep a class split in such a case depends on the desired use-cases. If the application should be able to distinguish between subclasses, then it makes sense to keep them. Additional labeling strategies, such as those for ambiguous classes (see section 4.7), for example if *easyjet* is classified as *plane* it is also correct, can be applied. If the application does not gain anything from the split (which is the case for this example), then the split should be reversed.

		Prediction					
		bike (other)	bike in black pants	bike in blue jeans	bike in shorts	other	walk
Actual	bike (other)	69.7	26.8	0.7	1.4	0.7	0.7
	bike in black pants	19.9	73.5	0.0	3.1	2.1	1.4
	bike in blue jeans	5.4	2.0	92.5	0.0	0.0	0.0
	bike in shorts	9.8	0.8	0.0	88.6	0.3	0.5
	other	18.1	0.5	0.0	0.4	79.5	1.5
	walk	16.5	1.2	0.0	2.2	1.4	78.7

Table 4.11.: The confusion matrix when samples from the bike class are split into separate classes depending on the trousers that were worn when creating the recording. Note that the classes bus, car, coach, metro, plane, train and tram have been combined to other for better readability (they were trained separately though).

The best case is demonstrated by splitting the *bike* class based on which trousers were worn during the recording process. The four subclasses were defined by wearing *jeans* (trousers made of denim, phone in the right front pocket), wearing *shorts* (different knee-long trousers with the phone in a lower side-pocket), wearing *black pants* (similar to jeans, just not as tight,

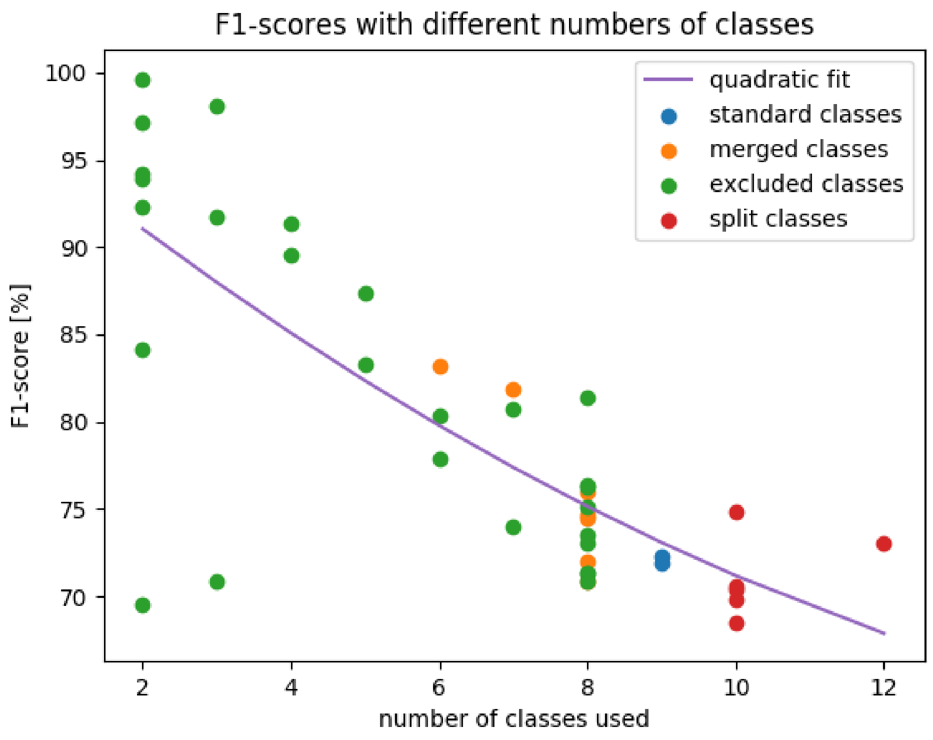


Figure 4.14.: The F1-scores of evaluations with different numbers of classes. The individual data points are color-coded by the strategy used to change the amount of classes. Additionally, the purple line shows a quadratic fit through the data.

phone also in the right front pocket), and everything else (recordings from other people and/or other trousers). For *jeans* and *shorts* approximately 9 out of 10 samples are predicted correctly. This is evidence of the high diversity that exists within the bike class. The subclass for *black pants* only has a success rate of 3 in 4. The exact percentages are listed in table 4.11. There exists more confusion between this class and the *bike (other)* class than at the other subclasses. The reason for this might be that *black pants* were worn most often and sometimes it might not have been logged in the meta-data (with the result that some recordings that should be in *black pants* are still in *bike (other)*).

Conclusion

The more classes are being used the lower the classification score will be. This is illustrated by figure 4.14 which contains a scatter of F1-scores from all the evaluations made in this section. It also contains a fit through the data points, which highlights the decline when more classes are used. The plot is by no means to be taken as a ground truth, but merely as guidance on what to expect when choosing the number of classes.

With more classes, more complex problems can be handled. Thus, there is a trade-off between accuracy and manageable complexity. The number of classes should be chosen in a way, that allows for the required complexity, while still maximizing accuracy. In other words: one should choose as many classes as required for their use cases, but not more.

4.6. Handling Low Confidence

With the given setup of the 9 classes and *the algorithm* there is one class which contains most of the samples with low confidence in the classification. In our case this is the *bike* class. This section provides a solution on how to circumvent that and produce a more meaningful result.

4.6.1. The Not-A-Transport-Mode Class

The first strategy (as described in section 3.2.7) was to create an additional class with a diverse set of samples that should take over the role of the *trash* class (ie. the class where *the algorithm* puts samples if they do not match anywhere).

To train this class, we added all classes, which were not a transport mode, of the *LITIS Rouen dataset*, and the *DCASE* 2017 and 2018 challenge datasets. The resulting class, which was labeled *other*, contained around 15 thousand samples. That is slightly more than half of the samples in the training set of the *bike* class (those amount to 28 thousand). This indicates the superior size of our dataset (more on that in section 4.9.1).

A training was performed with the *other* class in the training set. Contrary to our expectations, the results (as shown in the confusion matrix in table 4.12) show that almost none of the samples were predicted to belong to that class.

		Prediction									
		bike	bus	car	coach	metro	other	plane	train	tram	walk
Actual	bike	98.7	0.0	0.0	0.0	0.1	0.2	0.2	0.0	0.0	0.8
	bus	29.3	27.1	0.9	0.0	23.1	1.3	0.0	15.3	0.4	2.6
	car	11.1	0.4	77.7	0.9	0.0	0.0	0.3	9.6	0.0	0.0
	coach	34.7	7.2	12.6	38.3	0.0	0.0	2.4	4.8	0.0	0.0
	metro	36.6	1.0	0.5	0.0	46.8	0.0	1.7	12.9	0.5	0.0
	other	71.5	0.8	3.8	1.3	0.4	7.9	1.3	6.3	1.3	5.4
	plane	21.9	0.0	1.6	0.0	1.3	0.0	69.3	5.7	0.0	0.3
	train	15.6	0.4	1.2	0.1	0.5	0.1	0.5	77.4	0.1	4.1
	tram	52.4	1.0	11.2	0.0	2.2	0.0	0.6	25.6	5.8	1.3
	walk	16.0	0.0	0.0	0.0	2.7	0.1	0.0	0.3	0.0	80.9

Table 4.12.: The confusion matrix with an additional class added. The training set of the class consists of all non-TM classes of the 3 external datasets and the evaluation set contains our own recordings form a street, a ferry, a washing machine and moving around chairs.

To rule out the possibility that the samples in the *other* class are just to different to those from our dataset, another test was performed with the classes *station* and *street* instead of *other*. *Station* contained all samples from external classes that were recorded in a metro station, and *street* contained samples from any sort of city street (quiet and noisy, pedestrian and traffic). Those samples are more likely to be contained in our classes. However, the evaluation result still did not indicate a change in *trash* class.

4.6.2. Binarization Strategies

The reason why adding a *trash* class did not work is connected to how *the algorithm* performs the classification, specifically how it assigns the label. The final layer of the network provides confidence scores for each of the 9 classes. Those scores then have to be translated to a binary output: For each class either 1 (this sample belongs to the class) or 0 (this sample does not belong to the class).
One strategy to do so is by putting a 1 to the class with the highest confidence (it is referred to as `frame_max`). This is what we expected to happen.

That *the algorithm* uses a different approach per default becomes obvious when having a look at the confusion matrix of a training where the *bike* class was removed (additional confusion matrices can be found in the appendix). In those cases the *bus* class is suddenly the new *trash* class. It is unlikely that a class which had almost no false positives before, suddenly has a lot. This indicates that there might be a different cause than the removal of a class.

		Prediction								
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	96.5	0.0	0.1	0.0	0.2	0.3	1.6	0.0	1.4
	bus	8.7	30.1	6.6	0.4	23.6	0.4	28.4	0.9	0.9
	car	3.5	1.0	75.1	3.0	0.3	2.1	14.3	0.6	0.0
	coach	0.6	20.4	13.8	39.5	1.2	1.2	23.4	0.0	0.0
	metro	4.5	2.0	0.2	0.0	50.7	4.2	35.3	1.5	1.5
	plane	2.2	2.2	6.8	0.3	1.9	68.7	17.8	0.2	0.0
	train	4.0	0.1	3.0	0.3	0.5	0.9	87.0	0.9	3.3
	tram	6.1	3.8	11.2	0.0	2.9	7.3	52.4	13.1	3.2
	walk	13.3	0.0	0.4	0.0	1.2	0.1	1.0	0.0	84.1

Table 4.13.: The confusion matrix for the 9 classes of the dataset when the `frame_max` strategy is applied at classification.

In fact, the binarization strategy that is used (called `global_threshold`) defines a confidence threshold above which the value is 1 and otherwise it is 0. The default threshold is 50% to assure that there is at most one class that is assigned (because the sum of the confidence values is 100%). The issue is that when none of the classes reach this threshold, then all of the values stay 0 and the subsequent code that applies the label returns the first one (because the index which is initialized with 0 is never changed). So, the reason why the *bike* class had so many false positives is that it was the first in the list.

One possible course of action is to use `frame_max` instead of the default strategy. When doing so, the *bike* class still has false positives but significantly fewer (see confusion matrix in table 4.13). It can also be observed that the percentage of correctly predicted samples increases slightly (because some of those with a low confidence still had the highest confidence in the true class). The real trash class is *train*, which has numerous mis-predictions from all classes except *bike* and *walking*.
However, `global_threshold` can be used to our advantage as described in the next section.

4.6.3. The Unknown Class

The alternative course of action is to keep the default strategy but add another class. That class should represent all samples with a low confidence and it should be called the *any* class. We could have called it the *unknown* class, but since *any* starts with an A it is the first element in the list of classes and we do not have to change *the algorithm*. The *any* class does not have to have any training or evaluation data, it just needs to be added to the list of classes. All samples with a low confidence will automatically be assigned to the *any* class in the classification process. The resulting confusion matrix can be found in table 4.14.

This setup is superior to adding a *trash* class in that it does not require additional data. It affects all classes equally and, most importantly, it works. Furthermore, it brings major advantages for the sample processing steps which are described in section 4.8.

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	1.0	98.0	0.0	0.0	0.0	0.2	0.2	0.3	0.0	0.4
	bus	15.3	5.2	30.6	1.3	0.0	23.6	0.0	20.5	0.0	3.5
	car	6.6	2.9	1.0	84.1	0.5	0.3	1.1	3.5	0.0	0.0
	coach	12.6	0.0	32.9	9.0	37.7	1.2	0.0	6.6	0.0	0.0
	metro	16.9	3.7	0.0	0.2	0.0	58.0	3.7	15.9	1.2	0.2
	plane	11.0	0.9	0.5	2.0	0.6	5.5	74.2	4.6	0.0	0.6
	train	8.9	5.9	0.7	2.8	0.1	0.4	0.7	76.1	0.2	4.2
	tram	40.6	8.3	1.9	5.1	0.3	13.7	0.6	18.2	9.9	1.3
	walk	4.0	13.8	0.0	0.0	0.0	1.1	0.1	0.3	0.0	80.8

Table 4.14.: The confusion matrix with an additional class added that contains all predictions with a confidence below 50%.

4.7. Ambiguous Classes

The dataset contains samples where more than one class would be a correct prediction. Those are from the *walking inside another mode of transportation* recordings (see section 4.1.1). As described in section 3.2.6 there are several approaches on how to handle those special cases. The strategy that has been used thus far was to keep them in one class and refrain from giving them special treatment. This section discusses how they can be handled differently, and what is the best strategy to proceed with.

By default, the ambiguous samples are not in the *walking* class but in the class of the other transport mode. To estimate the impact of those samples, the evaluation set is temporarily reduced to contain only ambiguous samples from the *plane* and *train* class. A confusion matrix can be found in table 4.15 (on top). It shows that most *walking in a plane* samples are classified as *plane* and a few are confused with *bike*. Interestingly, none of them were predicted to contain walking. The *walking in a train* samples are distributed more evenly between those three classes, with less than a quarter of samples being incorrect (when both *train* and *walking* are counted as correct).

An additional training has been performed where the ambiguous samples were put in the *walking* class of the training set. The resulting confusion matrix can also be found in table 4.15 (below). It illustrates that more of the evaluation samples are classified as *walking*, but there is also a notable increase in the confusion with the *bike* class. Therefore it has been concluded that the training samples should not be moved to the *walking* class.

	Prediction when trained in plane/train									
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	plane	7.6	0.8	0.0	0.0	0.0	90.8	0.8	0.0	0.0
	train	23.6	0.0	0.0	0.0	0.0	0.9	43.6	0.0	31.8

	Prediction when trained in walking									
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	plane	26.0	0.0	0.0	0.0	0.8	64.9	0.0	0.0	8.4
	train	31.8	3.6	0.9	0.0	0.9	3.6	17.3	0.0	41.8

Table 4.15.: The confusion matrices of the ambiguous samples in the train and plane class. The upper table shows the results when ambiguous samples of the training set belonged to plane/train and the lower table shows results for when those samples belonged to the walking class. Both evaluations used frame maximization as binarization.

There is one more possibility in handling those samples, which would be to remove them. Since all the samples in those special cases are known, they can easily be eliminated from the dataset. However, it does require the evaluation set to be changed, which would in turn simplify the task at hand and counteract the comparability of results. Besides, the evaluation accuracy of the ambiguous samples is high enough that only little could be gained by eliminating those special cases. When trained and evaluated with the samples removed the accuracy increased by 1% (to 76%) and the total F1-score by 2% (to 78.7%). The corresponding confusion matrix can be found in the appendix (table B.4).

4.8. Sample Filtering

This section deals with improving the dataset. This is done in three steps. At first the samples of recordings in the wrong class are relabeled, then an appropriate number of samples at the beginning and end of each recording are removed, and finally individual samples within a recording are removed if they do not comply with the class definition.

4.8.1. Relabeling

The simplest way to find incorrectly labeled recordings is by checking their label against the meta-data. Since the label is derived from the meta-data this should match, otherwise something went wrong on the way. By doing so, we actually found 4 recordings with the wrong label. All of them were put in a wrong folder because they had label names in the notes. One recording was *walking* in a `metro_station`, which contains *metro*. Another file noted the weather condition `slightrain`, which contains *train*; and two more recordings were made on the `Intercitybus`, which contains *bus* but is actually a *coach*. So, in conclusion the recordings should be sorted by the right field in the meta data and not by searching for class names. The four files have been assigned the right label and we continued with the relabeling process as described in section 3.3.1.

To figure out which recording has an incorrect label in the meta-data, a training including the *any* class was performed. Afterwards the trained network was evaluated using the training set. The resulting evaluation scores are not meaningful and can be ignored (they are notably higher compared to when the evaluation set is used). However, the evaluation provides a list of predictions for each of the samples. This list is the basis of the relabeling process.

The samples are grouped by recording, and the ratio of incorrect predictions is analyzed. With a threshold of 50% (incorrectly classified samples in the recording), 91 recordings are listed as candidates for relabeling. Those are too many false positives, so the threshold is increased. At 90% mismatches, 24 recordings are listed, most of them are of the *tram* class (this was to be expected because the class has the lowest accuracy).

Additionally to the number of mismatches, the suggested class can be analyzed. The suggested class of a recording is the class, that most samples are predicted as. If this class is the *any* class that just means that the recording has a low confidence, but if it is any other class, it might have been labeled incorrectly. So, when only counting the recordings that are suggested to be another transport class, then at 50% there are 48 recordings, and at 90% there are 14 recordings, that are candidates for relabeling.
Among those 14 recordings are mainly false positives, such as the *walking in a train* recordings and more of the *tram* class. Those recordings also contain some interesting exceptional recordings, such as cycling with the phone in the backpack and walking with the phone in hand. Those were created for testing purposes and should not have been in the training set.
Two more recordings were found which need attention. Both of them contain a phone call during the recording. After the call the recording goes silent. There seems to be a bug in the application when the microphone is used by another app. These recordings can be added to the list of recordings that require trimming.
When going through the list that was created with a threshold of 50%, we were able to find one true positive. It was a recording of the LyonExpress tram that had the incorrect label of *plane* (apparently we forgot to change the label in the application).

When running *the algorithm* using the relabeled training set, it changes the results as shown in table 4.16. The *coach* class is less often confused with *bus*, resulting in higher accuracy for both classes. The *tram* class also improved. In this case it is to be interpreted as a good sign when more samples were classified as *any*, because that means that less samples were predicted to be other classes. This is the case for the *metro* class, even though the accuracy dropped slightly there are less confusions with the plane and train class.
Interestingly, the *plane* class performed slightly worse. The misplaced tram recording might have given the class more diversity (so overall more samples

were predicted as *plane*) which was now corrected.
Overall the weighted F1-score improved from 76.8% to 77.9%. Using the `frame_max` evaluation strategy the score can be further increased to 78.3% (confusion matrix in table B.6 in the appendix).

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	0.0	0.1	0.0	0.1	0.0	-0.1	-0.1	-0.1	0.1	0.0
	bus	-2.2	-3.5	7.4	0.0	0.9	0.4	0.0	-2.2	0.9	-1.7
	car	-1.8	-2.0	-1.0	1.3	1.5	-0.3	-1.0	3.3	0.0	0.0
	coach	0.6	0.6	-19.8	6.0	12.0	-1.2	0.0	1.8	0.0	0.0
	metro	9.7	-0.5	0.7	0.5	0.0	-1.5	-3.5	-4.7	-0.5	-0.2
	plane	3.6	0.2	0.9	3.5	2.7	-4.7	-7.9	1.1	0.3	0.3
	train	2.0	-3.2	0.5	-0.4	0.1	0.4	0.2	-0.8	2.3	-1.0
	tram	9.3	-7.3	2.9	-1.0	-0.3	-8.3	1.9	-8.6	11.2	0.3
	walk	-1.1	2.3	0.1	0.0	0.0	0.2	-0.1	0.1	0.0	-1.5

Table 4.16.: The changes in the confusion matrix after the relabeling process. This matrix contains the differences of the entries of the confusion matrix after relabeling (table B.5 which can be found in the appendix) and those of the confusion matrix of an evaluation with all samples (table 4.14).

It is hard to tell if we only found one true positive with this strategy because there are not more samples that require relabeling in the dataset, or because the relabeling strategy is not good. To gain more insight another experiment was conducted. One random recording was selected from each class and incorrectly labeled as another (the alphabetically next) class. This way each of the 9 classes has one recording that requires relabeling. *The algorithm* was trained and evaluated using this set of samples. Then the relabeling process was applied. With 50% mismatches in a recording, a total of 95 recordings were listed, containing 6 of the 9 recordings that were relabeled. When increasing the required mismatches to 90%, then 3 of the 9 recordings were found among 35 in total. To find all 9 recordings the required mismatches would have to be reduced to 10%, which would also return 300 false positives (and is therefore impractical). Figure 4.15 shows the relationship between false positives and recordings found, which actually require relabeling, plotted over the different input values for the relabeling algorithm. The classes that were easiest to find were *bike* (in *bus*), *bus* (in *car*) and *plane* (in *train*) and the ones hardest to find were *walking* (in *bike*) and *train* in (*tram*).

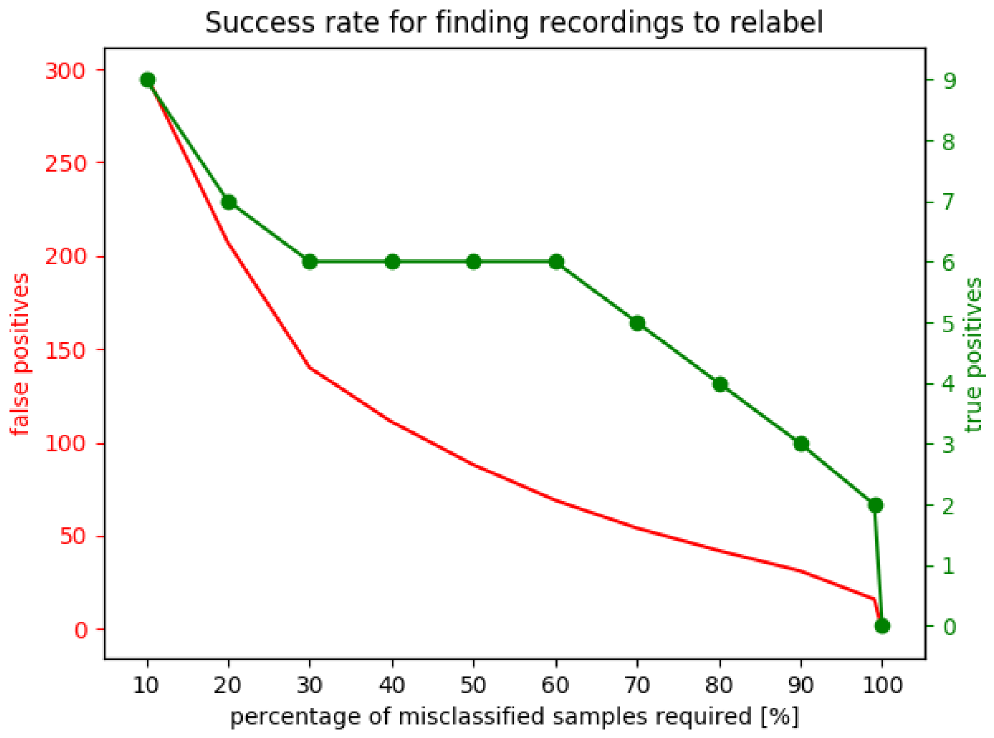


Figure 4.15.: The comparison of the number of recordings that are found (true positives) to the number of recordings that need to additionally be analyzed (false positives). The x-axis shows the required percentage of incorrectly labeled samples in a recording for it be flagged for relabeling.

Furthermore it is worth noting that the F1-score drops form 78.3% to 77.5%, when the network, trained with 9 incorrectly labeled recordings, is evaluated with the evaluation set. A confusion matrix can be found in the appendix (table B.7).

4.8.2. Trimming

Some recordings contain parts at the beginning and/or the end, that do not belong to the mode of transportation, that the recording represents. The process of trimming attempts to filter the corresponding samples of those parts of the recordings.

The trimming process is similar to the relabeling, in that is using a trained network (including the *any* class) to evaluate the training samples. Those predictions of the individual samples can then be grouped by recording and brought in the right order (sorted by sample ID). Afterwards, the beginning and end of each recording can be analyzed as described in section 3.3.2.

The basic approach does not work for each class. The accuracy of the *tram* class is too low, so the entire recordings would be trimmed. The counterexample is the *bike* class, where the accuracy is so high, that none of the samples would be trimmed (even samples that would required removal are classified as *bike*). The only class where we found that the trimming worked was the train class (with $n = 6$), but only when the part to be trimmed contained walking. With the other classes we only found false positives, which might also be contributed to the lack of testing data, since no recordings that require trimming were intentionally created.

Since most recordings that require trimming only require trimming at the back, the second strategy can be applied. A difference of 2 minutes (12 samples) has been defined to be the threshold. The trimming is only performed if at least 2 minutes more would be trimmed from the back than at the front.
Using this addition (with $n = 6$), there were still a lot of false positives, but it also found several recordings that actually required trimming. Most of them in the *walking* class. This is a class that is prone to require trimming, because we are more likely to forget turning off the recording when we stop walking (as this just happens), than when we exit a train or bus (because

this is a conscious action).

To further reduce the number of false positives, the number of samples to be trimmed at the front was required to be below a certain threshold that we defined to be 6 (1 minute). This is to ensure that the prediction accuracy of the recording has a certain reliability. Among the recordings flagged for trimming there were still quite a few false positives. For most of them there was an apparent change in sound somewhere in the recording, such as a car that left the motorway, or a plane that landed, and the second part happened to be harder to predict correctly.

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	0.5	-0.8	0.0	-0.1	0.0	0.0	0.2	0.1	-0.1	0.2
	bus	1.3	0.9	-5.7	0.4	0.0	-0.4	0.0	5.7	-0.9	-1.3
	car	2.0	1.5	0.1	-11.0	-0.3	0.3	1.5	5.8	0.0	0.0
	coach	7.2	0.6	3.6	-6.0	-1.8	0.0	0.6	-4.2	0.0	0.0
	metro	-8.7	0.0	1.5	0.2	0.0	-1.5	5.0	4.2	-0.7	0.0
	plane	2.8	-0.2	0.3	-3.6	-3.3	-0.3	9.1	-3.9	0.0	-0.9
	train	-1.2	0.4	-0.5	-1.3	0.1	-0.1	0.5	5.5	-2.2	-1.2
	tram	-2.6	2.2	0.6	1.3	0.0	2.9	0.3	6.7	-10.5	-1.0
	walk	6.1	8.6	-0.1	0.1	0.0	-0.4	0.4	0.3	0.0	-14.9

Table 4.17.: The changes in the confusion matrix after the trimming process. This matrix contains the differences of the entries of the confusion matrix after trimming (table B.8) and those of the confusion matrix of an evaluation after relabeling (table B.5).

To be able to also trim the bike class, the value of n had to be increased to 30. This means that 30 consecutive samples have to be classified as *bike* before the trimming stops (those 30 samples are then not included in the trim, of course). With this value, and the aforementioned extensions, all of the recordings that we knew required trimming were correctly flagged to be trimmed. Even the point where the recording should be cut was predicted with an acceptable level of accuracy (17 samples for 2 minutes (should be 12), 53 for 10 minutes (should be 60) etc.).

Even though there were still a lot of false positives, it was decided to perform the trimming anyway because, as we saw in section 4.4.1, a lot of samples can be removed without losing classification accuracy. In total 5866 samples were removed in the trimming process (that is 6.8% of all samples). Most of them (4083) from the *bike* class.

The resulting changes are displayed in table 4.17. The *bike* class is largely unaffected. Probably because it still holds a lot of samples. The classes profiting the most from this trim are *train* and *plane*, simply because they are predicted more often. This behavior was observed before. The *train* class becomes, what we referred to as, a *trash* class. The biggest loss is registered for *walking*. Over-all the F1-score decreased to 74.7% (-3.2%).

When performing a manual trim of all the files that were known to require trimming (90 minutes of *bike*, 27 minutes of *walking*, 10 minutes of *metro* and 6 minutes of *train*) no significant changes in the evaluation result were observed. The amount of samples removed is not enough to account for an observable difference in accuracy.
An additional evaluation has been made using the `frame_max` strategy (without the *any* class) and the confusion matrix is presented in table 4.18. The overall F1-score of this evaluation is 77.2%.

	Prediction								
Actual	bike	bus	car	coach	metro	plane	train	tram	walk
bike	97.4	0.1	0.2	0.0	0.2	0.2	0.4	0.1	1.5
bus	3.9	33.6	0.9	1.3	24.5	0.0	20.5	9.6	5.7
car	2.8	0.9	83.4	2.3	0.8	0.8	8.7	0.4	0.0
coach	1.8	22.8	18.6	33.5	0.6	3.6	19.2	0.0	0.0
metro	6.7	2.5	1.7	0.0	61.4	3.5	17.4	5.2	1.5
plane	0.5	2.8	12.8	0.5	6.1	68.5	3.5	0.5	4.9
train	3.6	0.7	1.5	0.0	1.6	0.6	80.3	6.0	5.7
tram	5.4	3.5	11.2	0.0	15.0	1.0	14.7	45.0	4.2
walk	14.2	0.0	0.2	0.0	1.1	0.0	0.5	0.4	83.7

Table 4.18.: The confusion matrix of an evaluation after manual trimming.

4.8.3. Cleaning

Cleaning in this context is the process of removing samples from the dataset, which do not properly represent their true class. This section attempts to apply the strategies presented in section 3.3.3 to our dataset.

As a first step the *tram* class was to be trimmed. A separate dataset was created from recordings of the *tram* class where we knew whether the vehicle was in the station or not. *The algorithm* was trained with only two classes: *in motion* and *in station*. Unfortunately this did not work because there were not enough samples. We only had recordings from three tram rides that were split accordingly and each halt in the station only yields 1-3 samples. So, *the algorithm* could not make the difference and 100% of samples were classified as *in motion*.

A second attempt was trying to clean the *bike* class. The idea was to filter samples, in which the cyclist is not actually cycling. The most likely stop is at a traffic light where the background noise are cars passing by. Since we did not have our own recordings of streets we used the *busystreet* class from the *LITIS Rouen dataset* and the *street_trafic* class from the *DCASE 2018 development dataset*. We made sure to put equal amounts of samples from the *bike* class and the newly created *streets* class into the training set. The idea was that even though samples from our training set are used (which might include samples that contain only street noise and therefore should be cleaned) the other class will still have more street noise. However, when evaluating all our *bike* samples still 100% of them were classified as *bike*, possibly because of confounding variables (such as the microphones used or the fact that for one class the sound is dampened by trousers). So no cleaning could be performed.

The third attempt was using external data only to clean the *bus* class. The idea was to filter samples where a lot of people are speaking (which occurs regularly in buses). A class had to be found, which contains people talking. The *market* class from the *Rouen dataset* fit this description quite well. So a training was performed using only the *market* and *bus* samples from that external dataset. It was made sure that the binarization method was set to `frame_max` but still all of our bus samples in the training set were classified as *bus*.

Interestingly when evaluating the *bus* samples from the evaluation set only 90% of samples were classified as *bus*. The remaining samples (which were classified as *market*) were all from the same recording: a bus ride in Vienna. This is the only bus ride of that particular city in the whole dataset. So even though the cleaning did not work, it raises another question of whether this recording should be in the evaluation set (or even the dataset) at all.

		Prediction					
		bus	car	metro	plane	train	tram
Actual	bus	46.8	1.5	18.6	0.0	27.4	5.6
	car	32.0	9.8	2.8	0.2	36.1	19.2
	metro	8.3	1.9	10.2	0.1	67.9	11.7
	plane	17.5	31.4	10.5	0.0	29.1	11.5
	train	28.2	4.4	1.4	0.0	43.1	23.0
	tram	13.9	1.8	3.7	0.0	45.8	34.8

Table 4.19.: The confusion matrix when trained using external datasets and evaluated using our training set. All values are in percent. The absolute values for this matrix can be found in table B.10 in the appendix.

In a last, highly experimental, attempt to clean our dataset, *the algorithm* was trained using all transportation modes from the external datasets. In this case the idea was that if a sample really represents the class then it should still be correctly classified in this network. That this is not the case becomes evident when having a look at the confusion matrix of when our training samples are evaluated on the model trained with external data (table 4.19). All of the classes have an accuracy below 50%, which is too low even when accounting for samples that should be cleared out. The reason that so few samples are classified as *plane* is that only the *Rouen dataset* contains a *plane* class which only hods 69 samples (see also section 4.9.1).
When the cleaning of the five classes (*bus*, *car*, *metro*, *train* and *tram* (*plane* was excluded)) is performed, 29 thousand samples are removed (only 11 thousand remain in those 5 classes, for an exact distribution refer to table B.10). To evaluate, *the algorithm* has been trained with the cleaned training set and

evaluated using our evaluation set. The results (see table 4.20) show that all the classes, which have been cleaned, decreased in prediction accuracy. The accuracy increases for two classes may partly be contributed to the fact, that overall more samples are classified as the given class (which is the case for *walking*), and partly to pure chance (which is probably the case for *coach*, since it has shown to have a higher variance in previous experiments, see for instance figure 4.10).

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	-0.3	-1.6	0.0	0.0	0.0	-0.1	-0.3	-0.2	0.0	2.5
	bus	-1.7	3.5	-3.5	-1.7	0.4	-3.1	0.9	-5.7	0.0	10.9
	car	8.3	3.0	2.9	-22.0	2.5	-0.3	9.1	-5.2	1.3	0.3
	coach	-10.2	1.2	-8.4	-6.6	15.6	0.0	3.6	4.8	0.0	0.0
	metro	3.0	6.5	2.2	-1.0	0.0	-37.1	1.2	16.2	2.2	6.7
	plane	-6.1	0.2	-1.7	-0.9	0.2	-0.5	4.6	2.5	1.1	0.8
	train	1.0	1.3	3.6	0.6	3.4	-0.4	0.1	-16.0	2.7	3.8
	tram	-19.5	5.4	-1.3	-4.8	1.9	-8.3	4.2	17.9	1.0	3.5
	walk	-8.3	-15.9	0.1	-0.1	0.0	-0.8	-0.4	-0.3	0.0	25.6

Table 4.20.: The changes in the confusion matrix after the cleaning process. This matrix contains the differences of the entries of the confusion matrix after cleaning (table B.9) and those of the confusion matrix of an evaluation after relabeling (table B.5).

To conclude the topic, it can be noted, that the discussed cleaning mechanisms do not work for this particular dataset. Implementing more sophisticated cleaning strategies requires human effort and the question has to be raised whether this is worth it or not. We were not able to show that removing individual samples results in better performance. Further tests with a dedicated test set would have to be issued. The only performance gains that were achieved, were caused by removing entire recordings that were either incorrectly labeled or contained non-representative data (such as silence). Therefore we suggest that the priority should be to create high-quality recordings, rather than trying to improve on existing ones.

4.9. Comparison

This section compares our dataset to other datasets that contain transport modes, in particular the *LITIS Rouen dataset* and the *DCASE 2017* and *DCASE 2018* development sets. Since the former contains samples with a length of 30 seconds, each of those samples was split into three 10 second samples. The other two datasets already contained samples with a length of 10 seconds.

4.9.1. Dataset Size

A comparison of the number of classes and samples between the four datasets can be found in table 4.21. It also contains information on how many of the classes can be used for transport mode detection. A detailed description of each dataset, as well as information on which of its classes can be used for TMD, was provided in section 2.4.

Dataset	**number of classes [TMD / total]**	**average samples per class**	**number of samples [TMD / total]**
Rouen [24]	7 / 19	495 / 478	3471 / 9078
DCASE 2017 [19]	4 / 15	312	1248 / 4680
DCASE 2018 [20]	3 / 10	864	2592 / 8640
Our dataset	9 / 9	10188	91691 / 91691

Table 4.21.: Comparison of different external datasets with our dataset. The number of classes as well as the number of samples are compared, the first value applies for TMD classes only and the second value takes all classes into account. All samples are 10 seconds in length.

Our dataset contains a significantly higher number of samples compared to the external datasets. We have created four times as many samples as are contained in all of the three other datasets combined. In terms of data for TMD the external datasets amount to only roughly 8% of our number of samples.

What this table does not show is the difference in size of the individual classes. The DCASE datasets have equally many samples in each class, whereas the samples from the *Rouen dataset* and our dataset are unevenly split (see table 4.1 for the distribution of classes in our dataset).

4.9.2. Transferability

The level to which our classes generalize can be analyzed by using the external datasets. If a class truly represents a transport mode then *the algorithm* should be able to classify samples no matter where they originate. Table 4.22 shows a confusion matrix of the evaluation with all external TMD samples in the evaluation set. The classes *bike*, *coach* and *walking* are not represented because they do not have corresponding classes in the external sets.
The resulting accuracies are lower than we had hoped. That the *tram* class does not perform well was to be expected. The other classes do hit the mark of random chance (which for 9 classes is at 11%), but some of them only barely.

There is a non-neglectable amount of confusions with the *bike* and *walking* classes. To avoid these, *the algorithm* can be trained with only the required 6 classes. The evaluation result is shown in table 4.23. The classes *metro* and *plane* increased the number of true positives, *tram*, *car* and *bus* still perform badly and *train* even lost evaluation accuracy.

So in conclusion our dataset does not generalize enough to support classification of external data. However, this is also true the other way around (see table 4.19). The samples in the individual datasets are just too far apart in the feature space.

		Prediction								
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bus	3.7	39.2	14.4	0.0	11.6	0.8	22.1	0.1	7.9
	car	10.6	3.8	16.2	0.1	20.1	22.9	18.3	2.0	6.0
	metro	19.6	19.4	6.1	0.0	20.4	0.9	21.1	1.2	11.3
	plane	56.5	0.0	0.0	0.0	5.8	34.8	0.0	0.0	2.9
	train	4.7	13.3	3.2	0.0	9.6	9.1	52.1	3.3	4.7
	tram	0.3	43.1	15.3	0.0	20.2	0.0	2.7	0.0	18.4

Table 4.22.: The confusion matrix of an evaluation with TMD data from external datasets. All values in percent (same table with absolute values available in the appendix).

		Prediction					
		bus	car	metro	plane	train	tram
Actual	bus	26.0	18.0	25.5	13.6	16.6	0.4
	car	1.7	16.9	11.0	57.3	12.7	0.5
	metro	13.9	6.5	42.8	22.8	12.1	1.9
	plane	0.0	0.0	8.7	91.3	0.0	0.0
	train	6.2	6.3	21.6	29.4	33.8	2.7
	tram	12.6	21.6	52.1	0.9	12.8	0.1

Table 4.23.: The confusion matrix of an evaluation with TMD data from external datasets when only the relevant classes have been trained. All values in percent.

4.10. Limitations

The results presented in this chapter depend to a large extend on the dataset we created. We gave our best to assure quantity, diversity and quality. However, we only touched on a small subset on what is acoustic TMD. The same experiments might yield different results when a different dataset is used. Generalization should be made with discretion.

Even with the same dataset, the results can vary between trainings. This is due to the selection of the validation set and the randomization process that is part of *the algorithm* (the order in which samples are selected for the patches is not always the same). Especially the *coach*, *bus* and *tram* classes (the classes with the fewest samples) have shown a variance in accuracy of up to 20%. We did our best not to have a bias when selecting between different evaluation runs.

5. The Guideline

This chapter summarizes the lessons learnt from this project into a useful guide for people who also want to create a dataset for acoustic transport mode detection (or a similar field).

5.1. Know what you want

The objective as to why a dataset is created should be evident. Simply creating a dataset for the sake of creating a dataset is not constructive. There should be an obvious goal, or task that you want to achieve. With this goal in mind, different decisions, such as the number and types of classes, can be made. The classification accuracy depends on the number of classes (as shown in 4.5), as well as how well they fit your class definitions. In some cases, it might even be enough to use existing datasets and the extensive process of creating a new dataset can be avoided.

5.2. Have sound Class Definitions

Once you have decided which classes will be in the dataset, a class definition should be written down for each of the classes. This definition should state what exactly is contained within a class, and what is not, to avoid ambiguities (such as *is the Lyon ExpressTram a tram?*).

The procedure is connected to section 5.1, in that you need to know the desired outcome, to appropriately confine classes. It is absurd to believe that a few examples in the dataset will generalize to all possible situations (as we have seen in 4.9.2). This applies to different modes of transportations, different cities, different recording device positions etc. So, a sound class definition must also cover those aspects.

A decision that served us well was to specify the location of the recording device to be inside trousers for cycling and walking. If we were to start over, we would add for all classes that the vehicle or person must be in motion. This would increase the effort of recording, but reduce the confusion between classes (see also section 5.8).

5.3. Be Consistent

Consistency is important in many aspects of dataset creation. Here it refers to the recording process. It is important to have similar (ideally identical) recording devices to avoid that the classification can be based on the device used (as discussed in section 3.1.2). This also applies to the settings of the device, such as the encoding and the sampling rate.

Any inconsistencies can become confounding variables. If it is not possible to avoid them, they should at least be tracked as meta-data.

5.4. The User is King

When it comes to creating the dataset you often rely on other people. In our case, we needed help in recording different transport modes in different locations. It is essential to make the task for other people as simple as possible. To do so, we created an application such that the people who help us (the users of the app) do not have to deal with matters that can be automated (such as logging timestamps and locations). The less a user has to do, the fewer mistakes can be made, and therefore the higher the quality of the dataset will be.

Listening to the user's feedback and improving the tools available to them is also highly suggested. We added several features to our application (such as the timer and the overwrite functionality, details in section 3.1.5) to better serve the user's needs. Especially, being able to change the meta-data once the recording is stopped, saved a lot of time that would have otherwise been required for manual editing.

5.5. Meta Data is Gold

Meta-data can be used to split classes, help interpreting the evaluation results, or even allow for the dataset to be used in completely unrelated experiments. Therefore, it makes sense to collect a lot of it. As discussed in the previous section, the more that can be logged automatically the better. Still, it is also recommended to manually add tags.

Without our efforts of including meta-data, some of the splits would not have been possible. Splitting the *bike* class based on the trousers worn is one example. We did not create the meta-data because we planned to do the split, but we could do the split because the meta-data existed. Refer to section 4.5.3 for more details on splitting classes.

We would even go as far as to argue that there is no such thing as too much meta-data (however, there can be not enough storage space). All the meta-data that is not required for a certain experiment can be ignored anyway.

5.6. Balance the Classes

As we have seen in section 4.3 and again in section 4.9.2, classes which had considerably fewer samples than the other classes performed significantly worse. Even though the bad performance of *bus* and *tram* cannot be entirely contributed to their underrepresentation, this skew did certainly have a negative effect.

This is not to say that all classes need to have an equal amount of samples, but it should be of the same order of magnitude. In this case we can refer to our dataset for a negative example: The *bike* class contains 10 times as many samples as the *tram* class (see table 4.1).

5.7. Mind your Resources

The scope of the project should be adequate to the resources available. This includes budget, access to hardware, time constraints and other limiting factors.
We would have continued gathering data, but at some point a recording stop had to be made in order to progress with the project.

As for computing hardware, it is worth knowing how much your system can handle. For us a standard computer (with 12GB memory and 2.3GHz quad core CPU) could only handle a fourth of our dataset before *the algorithm* ran out of memory and started swapping (which made it impractical to use). Fortunately we were granted access to a server with 8 dedicated GPUs.

So, you should not assume that your hardware can handle everything that is presented to it. If there is no possibility to upgrade, or get access to better hardware, then this also limits the maximal size of your dataset.

5.8. Invest Early

The early stages of the dataset creation have the highest return on effort invested. So, if you want to efficiently create an effective dataset you should not neglect the planning (as discussed in the first two section of this guide), and the subsequent recording process should be handled with care. It is a lot harder to remove recordings of low quality once they are in the dataset, than it is to avoid recording them.

This is also a well known principle in software engineering. The more time is spent in software design, the fewer revisions have to be implemented; and the more energy is spent to produce high quality code, the less maintenance has to be done. Since the costs for fixing an error increase with each step of the process (cheapest in design, highly expensive once deployed), it makes sense to invest early.

The strategies we used for removing samples that do not represent the class, did not perform well (see sections 4.8.2 and 4.8.3). More sophisticated strategies would have to be used in order to get the desired effect. The alternative, however, is to not have to use those methods at all because care was taken throughout the recording process.

Especially trimming can easily be performed when a recording is stopped. If you realize that you recorded for too long then the recording should be trimmed immediately and not postponed or forgotten about with the argument that the trimming algorithm will deal with it anyway. The more the algorithm has to trim, the worse it performs, and a manual trim by a human will in all likelihood be more exact.
This is also related to 5.4: if the user has the ability to easily trim a recording then he is more likely to trim the recording than if that would result in additional perceived effort for him.

The cleaning process becomes redundant if a recording is paused once the quality standards are not met anymore. This can happen for example if the vehicle stops, the radio is turned on, one has to wait at a traffic light, a group of noisy school children enter etc.

5.9. Make your Backup

It is generally a good idea to create backups. Whether it be from your code or the data you created, from everything that is stored digitally there should exist at least a second copy (ideally on a different storage device). Data storage is cheap enough that this should be afforded, even for a huge dataset. After all it would be vastly more expensive to recreate the dataset once it is lost (that is true for any size).

A part of a good backup strategy is to *keep the raw data*. Whenever a manipulation is performed on the dataset (to create a new dataset), the original data should still be kept. In other words, your dataset should be *immutable*. This applies for instance when splitting the dataset into samples (see section 3.2.3). The raw data should not be deleted once the samples are extracted. Not only for the case that the samples are lost (accidentally deleted, storage device failure etc.), but also in case you want to change the sample length at a later point.

5.10. Take care when handling Files

File manipulations should always be executed with caution. Whether that is importing (*moving*) files from a recording device, *copying* files onto a server, *deleting* files that are not required anymore, or simply *renaming* a file; it requires attention to be done right.

If not done right, removing files is the most likely to cause unexpected data loss; (`rm` is a powerful command, you don't want to hit enter until you typed the full path). However, also moving or copying data can cause unintentional overwrites, especially when scripts are being used to repeatedly perform those actions.

When data is being imported or exported (or copied in general) it should be checked that all the data was transfered correctly. The best method is comparing checksums, but the comparison of the total number of files and their size can already indicate if everything was copied. When copying the raw recordings into the GPU server some files were lost during transmission. We only found out about this after the first experiments had already been run (and they had to be re-run).

Not only missing files can cause undesired outcomes, but also incorrectly renamed or moved ones. This was discussed in section 4.8.1, where we found out that some of the files had the wrong label because we did not handle the files with enough care.

5.11. Keep it Realistic

The algorithm will only be able to classify what it is trained on. Therefore, it does not make sense to put recordings in the evaluation set when there are no corresponding recordings in the training set. The classification might be able to handle some degree of generalization, but you should not expect too much. This has been mentioned before in 5.2, but it should also be kept in mind when performing the training/evaluation set split.

When analyzing the evaluation set with the relabel algorithm, we have found that there are recordings, such as an airport bus ride on Venice airport, where there is only one of a kind in the dataset. Those recordings should either be removed from the dataset, or put into the training set. Having them in the evaluation set will falsify the results, because 100% of the samples of those recordings are misclassified.

So it should be made sure that the recordings are truly *iid* (independently identically distributed).

5.12. Know your Algorithm

The mode of operation of the algorithm used should be known well. In particular, it is important to understand what it requires as an input and what the output represents. Even if the algorithm is just used as a tool for classification, without the intention to improve on it, it should not be treated as a black box. The more you know about the function, the better you can prepare your data and interpret the results.

This might be the most impactful mistake we have made during this project, even though it seems so obvious. We did analyze all the steps of *the algorithm* (as described in section 3.2.4), from the feature generation up until the softmax classification layer. However, what we forgot to look into was how the output of the network is used to derive one class prediction (discussed at length in section 4.6.2). If we would have known about that earlier, the first half of the classification results of this project would look different (in particular the *bike* class would have a higher score).

5.13. More is More

Our attempts in trying to clean out the dataset (section 4.8.3) did not come with the desired increase in accuracy. The opposite was the case, whenever we removed significant amounts of samples, the performance decreased as well. This suggests that creating more data might be more effective than trying to increase the quality of the existing data.

CNNs are proven to work well will a lot of data. So this is particularly true if you use a similar algorithm to the one we used. If your resources allow it, go for creating more data instead of post-processing them. This suggested increase in quantity goes hand in hand with the suggested quality mentioned in section 5.8.

5.14. Analyze the Evaluation Data

Once you have created the dataset and implemented all necessary steps to perform the classification, it is time to analyze the classification results. Projects where the classification is just one part in a pipeline might tend to just work with what the classifier presents. However, having a closer look at mis-predictions can give you valuable insight on how to improve the classification accuracy. Visualizations, such as confusion matrices (which have been used extensively thought this project), can simplify this task.

We have evaluated the trained networks, not only with the evaluation set, but also with the training set. Doing so helped us figure out which recordings where hard to classify, or often confused with another class. Further decisions on whether to remove individual recordings, or to change the class definitions can be based on such findings.

5.15. Be Smart

Some of the recommendations in this guide might seem trivial and not important for you. However, you should take the time to reflect if that is actually the case. After all, this list of tips is based on the mistakes we made because we did not consider those things.

That being said, it might indeed be the case that some of those guidelines do not apply for your project, in which case they can be disregarded. There might even be situations where the opposite of what has been said is true. So, not everything presented here should be interpreted as the ultimate truth. You do not have to rely exclusively on this thesis, but you can also refer to additional sources on the topic (refer to chapter 2 for a start and continue from there) and/or use critical thinking to figure out what works best for your situation.

Furthermore, this is not a comprehensive guide on everything that needs to be considered. It is just meant to be a solid starting point. As with each knowledge project, you will not succeed unless you put your own thoughts and ideas into it.

6. Conclusions

This thesis discussed best practices when it comes to the design and creation of a dataset for transport mode detection. At first a background in acoustic scene classification, dataset creation, and general transport mode detection was given. Then the strategies and tools were presented. An Android application has been developed, which aided the recording process. Recordings have been made in different cities across Europe to create a dataset with **9 classes**, namely: bike, bus, car, coach, metro, plane, train, tram, and walking, at a total size of **245 hours**.

Experiments were performed on the dataset to gain insight into different aspects of classification. It was found that the sample length which achieves the best classification result is somewhere between 16 and 64 seconds. However, to stay compatible with other datasets and to ensure a low response time for a prospective application, a sample length of **10 seconds** was selected.

By analyzing different sizes of datasets it was found that up to three quarters of individual samples can be removed from the full dataset without causing notable changes, but when removing full recordings the performance drops more rapidly. This suggests that (for our dataset) diversity is more important than quantity.

Strategies on how to find the an appropriate size, as well as the best number of classes were presented. The latter depends on the task at hand. One should select as many classes, as are required to be distinguishable and none more. The more classes that are in use, the lower the classification scores will be.

Additionally, we demonstrated how to handle ambiguous classes. The solution that worked best for the given dataset was to keep special cases in their main class, such as *walking in a plane* in the *plane* class and *walking in a* *train* in the *train class*. When samples of those recordings are classified as *walking*, they are also counted as true positives.

Different strategies on how to assign a label based on the confidence scores for all classes have been discussed and explored. Selecting the class with the highest confidence should be used whenever the highest accuracy is desired, for instance: for the final evaluation. However, a different strategy, which implements a global threshold, in combination with an additional label, that is given if the threshold is not met, has proven to be very beneficial for sample filtering.

Sample filtering techniques, such as relabeling and trimming recordings, were described and applied. Relabeling required some human input, but the process turned out to be reliable and effective. Trimming only worsened results if done automatically, and only caused insignificant changes when done manually. Cleaning could not be performed in a way to improve results to the extend that the effort would be justifiable.
Using all these schemes to modify the training set, while leaving the evaluation set the same, the total performance was improved from an **F1-score** of 72% to a score of **78%**.

The findings throughout the project were compiled into a guideline. It can be referred to when expertise on creating a dataset on acoustic transport mode detection is required.

7. Future Work

Several interesting aspects of acoustic transport mode detection are still open to be analyzed. One such aspect would be classifying a large number of classes. Whether it is better to group them, perform a rough classification and then a separate sub-classification, or if all classes should be trained in only one network. It could be explored what the advantages and disadvantages of either of those strategies are.

We have analyzed aspects such as sample length and number of recordings but there are so many more variables that could be analyzed. Examples would be to analyze if different microphones produce different results, or how the sampling rate effects the classification accuracy.

Furthermore, more sophisticated methods for increasing the quality of an existing acoustic dataset could be explored. We have only touched on simple sample filtering techniques which do not produce the desired effect. Other approaches, such as various clustering techniques, might yield better results.

Throughout the project we only worked with one classification algorithm. It would be interesting to know if the dataset gains or looses accuracy when other algorithms are used. The experiments that were conducted might also produce different results when another algorithm is used.

The evaluation set was selected by a human and kept the same for all experiments. Since the evaluation set has a high impact on the classification scores, it would make sense to evaluate the choice of evaluation set. This could be done for instance be automatically creating an evaluation set (for instance randomly selecting a percentage of recordings) and comparing evaluation results.

If we were to continue working on this project then we would gather more data for the classes which are under-represented. Those were the classes of bus, coach and tram. This is required to figure out if the bad performance of those classes is caused by the lack of data or if it is inherently given by the class definition (ie. those classes are just harder to classify). Hopefully, more data in these classes would increase their classification accuracies and result in an overall better performance.

Smartwatches are another interesting topic that could be relevant for acoustic TMD. There seems to be a trend towards wearing watches with computing capabilities. Those watches could be used in the recording process of the dataset generation. This could bring advantages since they are usually better exposed to the environment. It would also fundamentally change the sounds for cycling and walking.

8. Acknowledgements

I would like to thank Roman Kern, my supervisor, for coming up with many great ideas, a few of which we were able to implement, and for keeping the motivation high throughout this project. It was a pleasure working with you. Your serenity made it easy to empathize. Thanks for supporting my wanderlust!

Additionally I would like to express my gratitude towards the Know Center Graz, for granting me access to their GPU cluster. Without it I could not have run experiments on that scale.

Furthermore I would like to acknowledge that I am grateful that Karl Voit created and published this latex template. It has simplified the formating a lot and the modular design has helped me keep everything organized. So thanks for the great work in the open-source community.

Special thanks also to Andreas Kraschitzer, Camille Sigoillot, Katharina Kovacs and Markus Meyer for creating recordings and giving feedback on the application.

Last but not least I want to thank my parents, Annemarie and Wolfgang, for providing me with the required mental and financial support during my many years in education.

Appendix

Appendix A.

Additional Figures

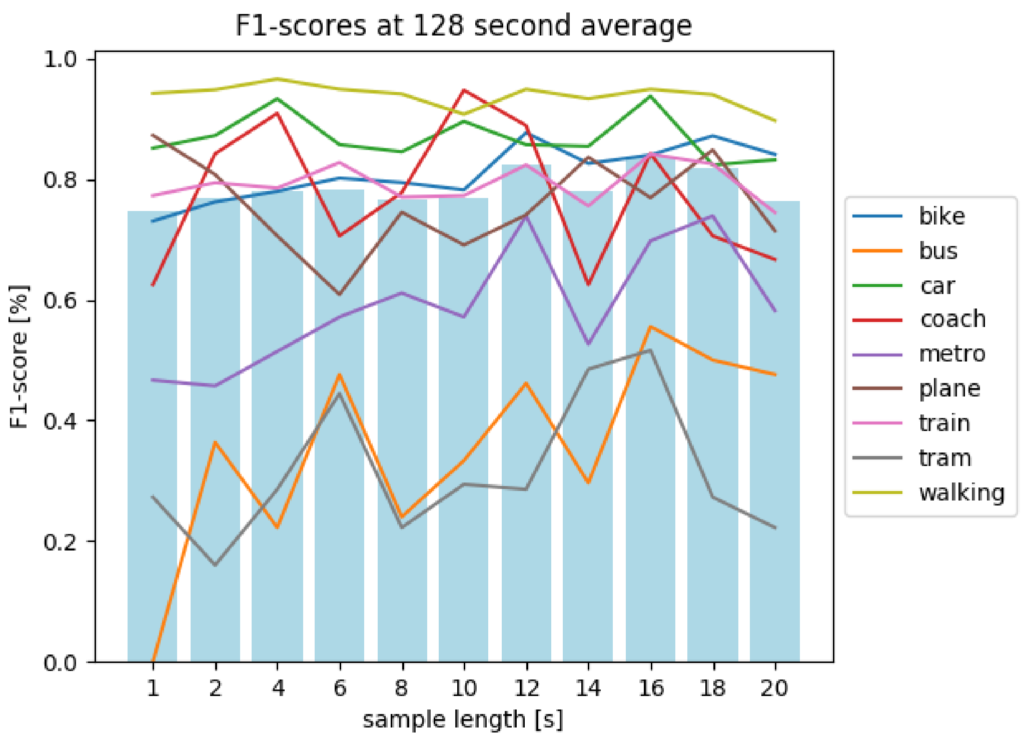
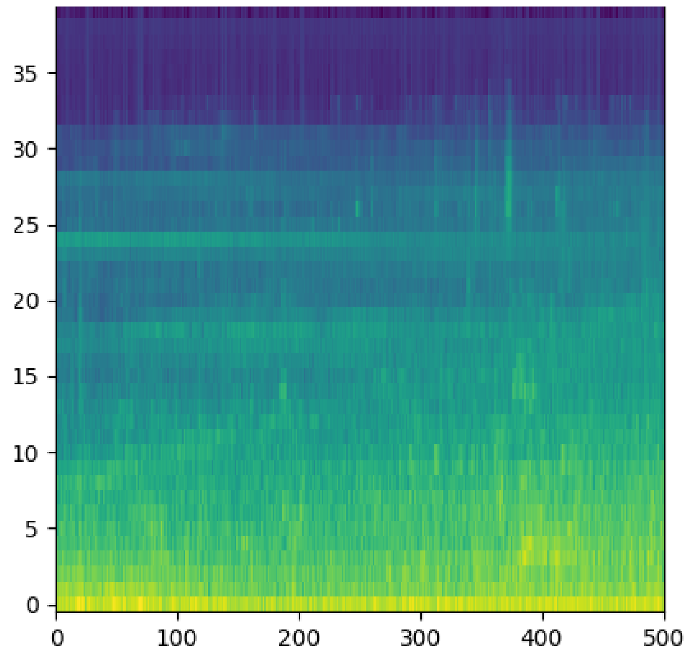
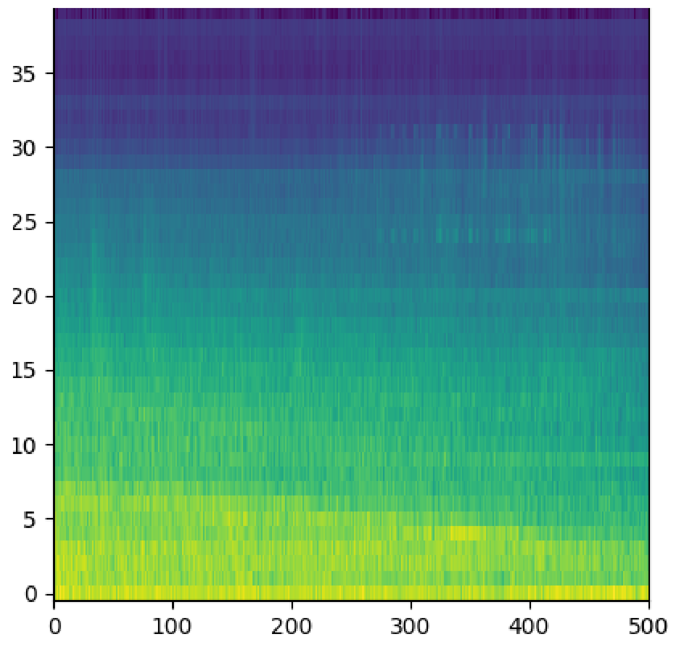


Figure A.1.: The individual F1-scores of different sample lengths on a linear scale with best of 128 second averages. This means for 1s a best of 128 was applied, for 2s best of 64, and so on up until best of 6 for 20s.

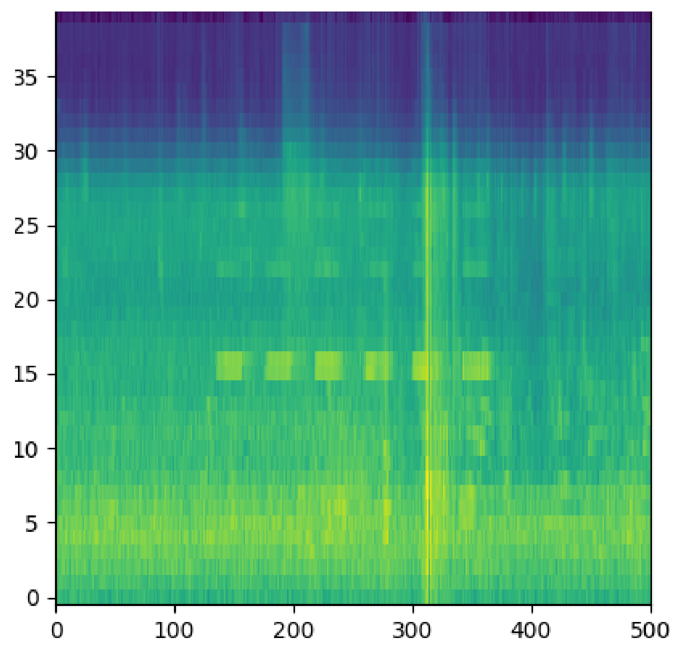


(a) tram: accelerating

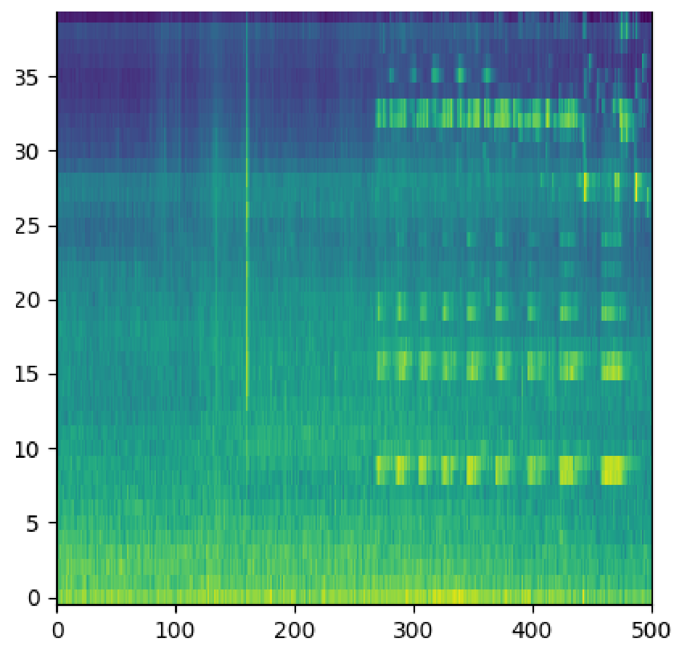


(b) metro: slowing down

Figure A.2.: Examples for log-mel band energies of vehicles changing speed. The features on the left were created from a sample of a Viennese tram accelerating. The plot on the right shows the Paris metro slowing down.



(a) metro: doors closing



(b) bike: squeaking breaks

Figure A.3.: Examples for log-mel band energies of sound events. The left plot contains the features of a sample recorded in the metro when the doors were closing. The peeping as well as the slam of the door are visible. The features on the right represent cycling and the audible squeaking of the breaks.

Appendix B.

Confusion Matrices

		Prediction								
Actual		bike	bus	car	coach	metro	plane	train	tram	walking
	bike	1315	0	0	0	2	2	2	0	9
	bus	56	53	9	0	55	1	46	1	8
	car	68	2	657	4	5	12	41	1	1
	coach	33	11	20	71	1	0	31	0	0
	metro	138	1	3	0	208	6	39	5	2
	plane	124	11	16	0	30	429	25	0	0
	train	250	6	26	3	11	16	991	20	50
	tram	148	10	27	1	17	16	39	49	6
	walking	222	0	3	0	21	0	5	1	877

Table B.1.: The confusion matrix of the classes of our dataset when all the samples are used. This is the same as table 4.4 but with absolute values to get an idea of the sample distribution.

B.1. Merging

		Prediction						
		bike	bus	car	coach	plane	rails	walk
Actual	bike	98.4	0.0	0.0	0.0	0.2	0.8	0.6
	bus	11.8	21.8	2.6	0.0	0.4	62.0	1.3
	car	4.0	0.1	79.9	0.8	2.1	13.0	0.0
	coach	12.0	15.6	9.0	48.5	0.0	15.0	0.0
	plane	7.6	0.2	1.3	0.0	77.0	13.9	0.2
	rails	6.5	0.6	1.5	0.1	1.6	86.7	3.0
	walk	20.1	0.0	0.0	0.0	0.0	1.6	78.3

		Prediction						
		bike	bus	car	coach	plane	rails	walk
Actual	bike	98.9	0.0	0.0	0.0	0.2	0.3	0.7
	bus	24.5	23.1	3.9	0.0	0.4	44.5	3.5
	car	8.6	0.3	83.1	0.5	1.5	5.9	0.1
	coach	19.8	6.6	12.0	42.5	0.0	19.2	0.0
	plane	19.5	1.7	2.5	0.0	67.6	8.7	0.0
	rails	25.7	0.8	2.7	0.2	1.8	66.0	2.8
	walk	19.7	0.0	0.3	0.0	0.0	2.4	77.7

Table B.2.: The confusion matrices when the classes metro, tram and train are merged. The upper matrix shows when the merge happened before the training, and the lower one only combined the classes in the confusion matrix of a regular 9-class training.

B.2. Splitting

		Prediction									
		bike	bus	car	coach	metro	plane	sbahn	train	tram	walk
Actual	bike	98.9	0.0	0.0	0.0	0.2	0.3	0.0	0.1	0.0	0.5
	bus	26.6	27.1	1.3	0.0	23.1	0.0	0.0	18.3	0.4	3.1
	car	12.5	0.5	80.5	0.3	0.3	2.4	0.0	3.5	0.0	0.0
	coach	38.9	12.6	10.2	31.7	0.0	0.0	0.0	6.6	0.0	0.0
	metro	29.4	0.5	0.0	0.0	59.5	1.7	0.2	8.5	0.2	0.0
	plane	14.6	0.5	1.1	0.2	1.3	81.9	0.0	0.0	0.0	0.5
	sbahn	44.1	0.0	2.3	0.0	1.9	7.5	10.3	32.9	0.5	0.5
	train	24.0	0.5	3.3	0.0	0.0	0.9	0.1	67.7	0.5	3.1
	tram	57.5	1.9	5.1	0.3	10.2	5.4	1.6	6.7	9.3	1.9
	walk	27.4	0.0	0.0	0.0	1.0	0.0	0.0	0.2	0.0	71.5

Table B.3.: The confusion matrix when samples from city trains are split from the train class into a separate sbahn class.

B.3. Ambiguous Classes

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	any	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
	bike	0.9	97.8	0.0	0.0	0.0	0.0	0.4	0.1	0.0	0.8
	bus	17.9	1.7	35.8	0.4	0.0	23.6	0.0	18.8	0.9	0.9
	car	6.4	2.4	2.0	81.7	1.0	0.1	0.9	5.4	0.0	0.0
	coach	18.0	0.6	29.9	7.2	38.3	0.0	0.0	6.0	0.0	0.0
	metro	20.9	4.0	3.0	0.5	0.0	60.0	2.5	8.7	0.2	0.2
	plane	9.1	0.0	0.2	2.8	0.0	0.2	85.3	2.2	0.2	0.0
	train	14.3	1.9	1.3	1.3	0.1	1.0	1.8	76.3	1.6	0.4
	tram	39.6	2.9	2.2	4.5	0.0	9.9	5.1	14.4	19.8	1.6
	walk	4.9	16.3	0.1	0.0	0.0	1.7	0.0	0.2	0.0	76.9

Table B.4.: The confusion matrix when trained without the ambiguous walking samples.

B.4. Relabeling

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	1.0	98.0	0.0	0.1	0.0	0.1	0.2	0.2	0.1	0.4
	bus	13.1	1.7	38.0	1.3	0.9	24.0	0.0	18.3	0.9	1.7
	car	4.8	0.9	0.0	85.3	2.0	0.0	0.1	6.8	0.0	0.0
	coach	13.2	0.6	13.2	15.0	49.7	0.0	0.0	8.4	0.0	0.0
	metro	26.6	3.2	0.7	0.7	0.0	56.5	0.2	11.2	0.7	0.0
	plane	14.6	1.1	1.4	5.5	3.3	0.8	66.3	5.7	0.3	0.9
	train	10.9	2.7	1.2	2.4	0.2	0.7	0.9	75.3	2.5	3.2
	tram	49.8	1.0	4.8	4.2	0.0	5.4	2.6	9.6	21.1	1.6
	walk	2.9	16.1	0.1	0.0	0.0	1.2	0.0	0.4	0.0	79.3

Table B.5.: The confusion matrix of an evaluation after the relabeling process.

		Prediction								
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	98.3	0.0	0.1	0.0	0.4	0.2	0.2	0.2	0.6
	bus	2.6	40.6	1.3	1.3	24.5	0.0	23.6	3.9	2.2
	car	1.3	0.5	87.0	2.5	0.1	0.3	8.3	0.0	0.0
	coach	0.6	17.4	16.2	52.7	0.6	0.6	12.0	0.0	0.0
	metro	5.5	2.0	1.2	0.0	69.7	0.2	18.4	2.0	1.0
	plane	1.6	3.3	7.6	5.5	1.3	69.9	8.0	1.3	1.6
	train	4.1	1.6	2.9	0.3	1.6	1.1	80.1	3.5	4.8
	tram	4.2	6.7	8.9	0.0	14.4	3.8	24.0	35.5	2.6
	walk	17.0	0.2	0.0	0.0	1.4	0.0	0.4	0.2	80.8

Table B.6.: The confusion matrix after the relabeling process when frame_max was used in evaluation. All values in percent.

		Prediction								
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	98.5	0.0	0.0	0.0	0.2	0.4	0.4	0.0	0.6
	bus	5.2	34.1	1.7	0.4	24.0	1.3	31.0	1.3	0.9
	car	6.1	0.9	80.8	1.5	0.0	2.1	8.2	0.4	0.0
	coach	1.8	28.7	13.2	50.3	0.0	0.6	5.4	0.0	0.0
	metro	6.7	2.5	1.0	0.5	66.9	2.5	18.2	1.0	0.7
	plane	4.3	8.3	1.3	0.6	1.4	78.0	4.3	0.6	1.3
	train	6.8	0.4	1.7	0.3	1.0	1.8	81.9	1.3	4.7
	tram	5.4	4.8	7.7	3.2	12.5	8.6	31.0	24.0	2.9
	walk	17.2	0.1	0.1	0.0	0.7	0.0	0.4	0.0	81.5

Table B.7.: The confusion matrix when trained with 9 incorrectly labeled recordings. The evaluation was performed using frame_max on the evaluation set.

B.5. Trimming

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	1.5	97.2	0.0	0.0	0.0	0.1	0.4	0.3	0.0	0.5
	bus	14.4	2.6	32.3	1.7	0.9	23.6	0.0	24.0	0.0	0.4
	car	6.8	2.4	0.1	74.3	1.8	0.3	1.6	12.6	0.0	0.0
	coach	20.4	1.2	16.8	9.0	47.9	0.0	0.6	4.2	0.0	0.0
	metro	17.9	3.2	2.2	1.0	0.0	55.0	5.2	15.4	0.0	0.0
	plane	17.5	0.9	1.7	1.9	0.0	0.5	75.4	1.7	0.3	0.0
	train	9.6	3.1	0.7	1.1	0.4	0.7	1.4	80.8	0.4	2.0
	tram	47.3	3.2	5.4	5.4	0.0	8.3	2.9	16.3	10.5	0.6
	walk	9.0	24.7	0.0	0.1	0.0	0.8	0.4	0.6	0.0	64.4

Table B.8.: The confusion matrix of an evaluation after the trimming process.

B.6. Cleaning

		Prediction									
		any	bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	1.2	95.6	0.0	0.0	0.0	0.0	0.1	0.1	0.0	3.0
	bus	12.7	6.1	28.8	0.0	1.3	20.5	0.9	18.3	0.0	11.4
	car	15.2	5.4	3.0	52.3	4.3	0.0	10.7	7.5	1.3	0.3
	coach	10.2	2.4	8.4	2.4	63.5	0.0	4.2	9.0	0.0	0.0
	metro	20.9	9.7	4.5	0.0	0.0	17.9	6.5	31.6	2.2	6.7
	plane	11.3	1.1	0.0	0.9	0.2	0.0	80.0	4.3	1.4	0.8
	train	10.6	4.4	4.2	1.7	3.7	0.2	1.5	64.9	3.1	5.8
	tram	27.8	8.6	4.2	0.6	1.9	0.0	7.0	34.2	11.5	4.2
	walk	0.7	8.9	0.1	0.0	0.0	0.0	0.0	0.4	0.0	90.0

Table B.9.: The confusion matrix of an evaluation after the cleaning process.

B.7. Using External Datasets

	Prediction						
		bus	car	metro	plane	train	tram
Actual	bus	1433	47	570	0	838	171
	car	4038	1236	349	20	4547	2416
	metro	518	116	632	4	4217	727
	plane	933	1673	562	0	1554	612
	train	4525	713	219	0	6919	3689
	tram	328	43	86	0	1077	819

Table B.10.: The confusion matrix when trained using external datasets and evaluated using our training set in absolute values. The same confusion matrix using percentages can be found in table 4.19

	Prediction									
		bike	bus	car	coach	metro	plane	train	tram	walk
Actual	bike	0	0	0	0	0	0	0	0	0
	bus	65	687	253	0	204	14	388	2	139
	car	110	40	169	1	209	238	191	21	62
	coach	0	0	0	0	0	0	0	0	0
	metro	397	393	123	0	414	18	428	25	230
	plane	39	0	0	0	4	24	0	0	2
	train	59	165	40	0	119	113	649	41	59
	tram	3	507	180	0	238	0	32	0	216
	walk	0	0	0	0	0	0	0	0	0

Table B.11.: The confusion matrix of an evaluation with TMD data from external datasets in absolute values. The percentage values are shown in table 4.22

Bibliography

[1] O Abdel-Hamid et al. 'Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition'. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2012, pp. 4277–4280. DOI: 10.1109/ICASSP.2012.6288864.

[2] Thierry Bertin-Mahieux et al. 'The Million Song Dataset'. In: *Proceedings of the 12th International Conference on Music Information Retrieval ({ISMIR} 2011)*. 2011.

[3] J Deng et al. 'ImageNet: A large-scale hierarchical image database'. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[4] Hamid Eghbal-Zadeh et al. 'CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks'. In: *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)* (2016).

[5] V Emamian, M Kaveh and A H Tewfik. 'Robust clustering of acoustic emission signals using the Kohonen network'. In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*. Vol. 6. June 2000, 3891–3894 vol.6. DOI: 10.1109/ICASSP.2000.860253.

[6] A J Eronen et al. 'Audio-based context recognition'. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (Jan. 2006), pp. 321–329. ISSN: 1558-7916. DOI: 10.1109/TSA.2005.854103.

[7] A Eronen et al. 'Audio-based context awareness - acoustic modeling and perceptual evaluation'. In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)*. Vol. 5. Apr. 2003, pp. V–529. DOI: 10.1109/ICASSP.2003.1200023.

[8] Shih-Hau Fang et al. 'Transportation Modes Classification Using Sensors on Smartphones'. In: *Sensors* 16.8 (2016).

[9] J F Gemmeke et al. 'Audio Set: An ontology and human-labeled dataset for audio events'. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, pp. 776–780. DOI: 10.1109/ICASSP.2017.7952261.

[10] M Amac Guvensan et al. 'A Novel Segment-Based Approach for Improving Classification Performance of Transport Mode Detection'. In: *Sensors* 18.1 (2018).

[11] Yoonchang Han, Jeongsoo Park and Kyogu Lee. 'Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification'. In: *the Detection and Classification of Acoustic Scenes and Events (DCASE)* (2017), pp. 1–5.

[12] Samuli Hemminki, Petteri Nurmi and Sasu Tarkoma. 'Accelerometer-based Transportation Mode Detection on Smartphones'. In: *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. SenSys '13. New York, NY, USA: ACM, 2013, 13:1–13:14. ISBN: 978-1-4503-2027-6. DOI: 10.1145/2517351.2517367. URL: http://doi.acm.org/10.1145/2517351.2517367.

[13] Helge Homburg et al. 'A Benchmark Dataset for Audio Classification and Clustering.' In: *ISMIR*. Vol. 2005. 2005, pp. 528–531.

[14] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Advances in Neural Information Processing Systems 25*. Ed. by F Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[15] Anurag Kumar and Bhiksha Raj. 'Audio Event Detection Using Weakly Labeled Data'. In: *Proceedings of the 2016 ACM on Multimedia Conference*. MM '16. New York, NY, USA: ACM, 2016, pp. 1038–1047. ISBN: 978-1-4503-3603-1. DOI: 10.1145/2964284.2964310. URL: http://doi.acm.org/10.1145/2964284.2964310.

[16] S Lee, J Lee and K Lee. 'VehicleSense: A reliable sound-based transportation mode recognition system for smartphones'. In: *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. June 2017, pp. 1–9. DOI: 10.1109/WoWMoM.2017.7974318.

[17] X Liang and G Wang. 'A Convolutional Neural Network for Transportation Mode Detection Based on Smartphone Platform'. In: *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. Oct. 2017, pp. 338–342. DOI: 10.1109/MASS.2017.81.

[18] Annamaria Mesaros, Toni Heittola and Tuomas Virtanen. 'A multi-device dataset for urban acoustic scene classification'. 2018. URL: https://arxiv.org/abs/1807.09840.

[19] Annamaria Mesaros et al. 'DCASE 2017 Challenge setup: Tasks, datasets and baseline system'. In: *DCASE 2017 - Workshop on Detection and Classification of Acoustic Scenes and Events*. Munich, Germany, Nov. 2017. URL: https://hal.inria.fr/hal-01627981.

[20] A Mesaros et al. 'Detection and Classification of Acoustic Scenes and Events: Outcome of the DCASE 2016 Challenge'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.2 (Feb. 2018), pp. 379–393. ISSN: 2329-9290. DOI: 10.1109/TASLP.2017.2778423.

[21] Seongkyu Mun et al. 'Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane'. In: *Proc. DCASE* (2017), pp. 93–97.

[22] N Papernot et al. 'The Limitations of Deep Learning in Adversarial Settings'. In: *2016 IEEE European Symposium on Security and Privacy (EuroS P)*. Mar. 2016, pp. 372–387. DOI: 10.1109/EuroSP.2016.36.

[23] Adrian C Prelipcean, Gyözö Gidófalvi and Yusak O Susilo. 'Transportation mode detection – an in-depth review of applicability and reliability'. In: *Transport Reviews* 37.4 (2017), pp. 442–464. DOI: 10.1080/01441647.2016.1246489. URL: https://doi.org/10.1080/01441647.2016.1246489.

[24] Alain Rakotomamonjy and Gilles Gasso. 'Histogram of Gradients of Time-frequency Representations for Audio Scene Classification'. In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 23.1 (Jan. 2015), pp. 142–153. ISSN: 2329-9290. DOI: 10.1109/TASLP.2014.2375575. URL: http://dx.doi.org/10.1109/TASLP.2014.2375575.

[25] Sasank Reddy et al. 'Using Mobile Phones to Determine Transportation Modes'. In: *ACM Trans. Sen. Netw.* 6.2 (Mar. 2010), 13:1–13:27. ISSN: 1550-4859. DOI: 10.1145/1689239.1689243. URL: http://doi.acm.org/10.1145/1689239.1689243.

[26] T N Sainath et al. 'Deep convolutional neural networks for LVCSR'. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. May 2013, pp. 8614–8618. DOI: 10.1109/ICASSP.2013.6639347.

[27] Nicolas Saint-Arnaud. 'Classification of sound textures'. PhD thesis. Massachusetts Institute of Technology, 1995.

[28] Nitin Sawhney and Pattie Maes. 'Situational awareness from environmental sounds'. In: *Tech-nical Report, Massachusetts Institute of Technology* (1997).

[29] Pierre Sermanet et al. 'OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks'. In: *CoRR* abs/1312.6 (2013). URL: http://arxiv.org/abs/1312.6229.

[30] Karanjit Singh and Shuchita Upadhyaya. 'Outlier detection: applications and techniques'. In: *International Journal of Computer Science Issues (IJCSI)* 9.1 (2012), p. 307.

[31] Stanley Smith Stevens, John Volkmann and Edwin B Newman. 'A scale for the measurement of the psychological magnitude pitch'. In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190.

[32] Dan Stowell and Mark D Plumbley. 'An open dataset for research on audio field recording archives: freefield1010'. In: *CoRR* abs/1309.5 (2013). arXiv: 1309.5275. URL: http://arxiv.org/abs/1309.5275.

[33] D Stowell et al. 'Detection and Classification of Acoustic Scenes and Events'. In: *Multimedia, IEEE Transactions on* 17.10 (Oct. 2015), pp. 1733–1746. ISSN: 1520-9210. DOI: 10.1109/TMM.2015.2428998.

[34] P Vandewalle, J Kovacevic and M Vetterli. 'Reproducible research in signal processing'. In: *IEEE Signal Processing Magazine* 26.3 (May 2009), pp. 37–47. ISSN: 1053-5888. DOI: 10.1109/MSP.2009.932122.

[35] Corlien M Varkevisser, Indra Pathmanathan and Ann Templeton Brownlee. *Designing and conducting health systems research projects*. Vol. 1. IDRC, 2003.

[36] S Wang, C Chen and J Ma. 'Accelerometer Based Transportation Mode Recognition on Mobile Phones'. In: *2010 Asia-Pacific Conference on Wearable Computing Systems*. Apr. 2010, pp. 44–46. DOI: 10.1109/APWCS.2010.18.

[37] Zheng Weiping et al. 'Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion'. In: *Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*. 2017.

[38] P Widhalm, P Nitsche and N Brändie. 'Transport mode detection with realistic Smartphone sensor data'. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. Nov. 2012, pp. 573–576.

[39] David H Wolpert. 'The Lack of A Priori Distinctions Between Learning Algorithms'. In: *Neural Computation* 8.7 (1996), pp. 1341–1390. DOI: 10.1162/neco.1996.8.7.1341. URL: https://doi.org/10.1162/neco.1996.8.7.1341.

[40] Yu Zheng et al. 'Learning Transportation Mode from Raw Gps Data for Geographic Applications on the Web'. In: *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. New York, NY, USA: ACM, 2008, pp. 247–256. ISBN: 978-1-60558-085-2. DOI: 10.1145/1367497.1367532. URL: http://doi.acm.org/10.1145/1367497.1367532.