TU WIEN Informatics

Classification of vehicles based on audio data

Classifying vehicle types with audio preprocessing and machine learning

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

066 646 Master programme Computational Science and Engineering

by

Bernd Schönfelder, BSc
Registration Number 01326497

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. DI Dr. Thomas Grechenig

Vienna, February 6, 2025

Bernd Schönfelder | Thomas Grechenig

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

TU WIEN Informatics

Fahrzeug Klassifizierung basierend auf Audiodaten

Fahrzeugklassifizierung basierend auf Audio Datenvorverarbeitung und Machine Learning

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

066 646 Masterstudium Computational Science and Engineering

eingereicht von

Bernd Schönfelder, BSc
Matrikelnummer 01326497

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. DI Dr. Thomas Grechenig

Wien, 6. Februar 2025

Bernd Schönfelder	Thomas Grechenig

Erklärung zur Verfassung der Arbeit

Bernd Schönfelder, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 6. Februar 2025

Bernd Schönfelder

Danksagung

Ich möchte meinen Betreuern meinen aufrichtigen Dank für ihre unschätzbare Unterstützung, Anleitung und Ermutigung während dieser Arbeit aussprechen. Ihr Fachwissen und ihr konstruktives Feedback waren maßgeblich für die Ausrichtung und das Ergebnis dieser Arbeit.

Mein herzlicher Dank gilt außerdem der Technischen Universität Wien, dem INSO-Institut und der RISE GmbH für die Bereitstellung der notwendigen Ressourcen und Infrastruktur, die diese Forschung ermöglicht haben.

Besonderer Dank gilt meiner Familie und meinen Freunden für ihre unerschütterliche Unterstützung und ihr Verständnis während dieser Reise. Ihr Glaube an mich war eine stetige Quelle der Motivation.

Abschließend danke ich allen, die direkt oder indirekt zum erfolgreichen Abschluss dieser Arbeit beigetragen haben. Ihre Hilfe und Ermutigung weiß ich sehr zu schätzen.

Acknowledgements

I would like to express my sincere gratitude to my supervisors for their invaluable guidance, support, and encouragement throughout this thesis. Their expertise and constructive feedback were instrumental in shaping the direction and outcome of this work.

I also extend my heartfelt thanks to Vienna University of Technology, INSO Institute and RISE GmbH for providing the necessary resources and infrastructure that made this research possible.

Special thanks go to my family and friends for their unwavering support and understanding during this journey. Their belief in me has been a constant source of motivation.

Finally, I am grateful to everyone who contributed, directly or indirectly, to the successful completion of this thesis. Your help and encouragement are deeply appreciated.

Kurzfassung

Diese Arbeit untersucht das Potenzial der Fahrzeugklassifizierung auf Basis von Audiodaten und bietet eine Alternative oder Ergänzung zu traditionellen video-basierten Verkehrserkennungssystemen. Vier zentrale Forschungsfragen leiten diese Arbeit: die Genauigkeit der Fahrzeugzählung (RQ1), die anhand von Referenzdaten bewertet wird, die Machbarkeit der Fahrzeugtypklassifikation mittels maschinellem Lernen (RQ2), die ebenfalls gegen dieselben Referenzdaten evaluiert wird, der Einfluss der Hardware auf Echtzeitberechnungen (RQ3), der sich auf die bereits entwickelte und eingesetzte Infrastruktur bezieht, sowie die Gesamtzuverlässigkeit der audiobasierten Verkehrserfassung (RQ4) im Hinblick auf die verfügbaren Referenzdaten und die Messkonfiguration.

Zur Beantwortung dieser Fragen wurden Daten aus einer Sensorbox mit Audioaufzeichnungskapazität verwendet, die neben einem bestehenden Videoerkennungssystem eingesetzt wurde. Die von der Videoerkennung gewonnenen Daten dienten als Referenz für die Bewertung der Leistung des Audioerkennungssystems. Ein Softwareprototyp, der maschinelles Lernen einsetzt, wurde entwickelt und erreichte eine Fahrzeugzählgenauigkeit von bis zu 98 was zeigt, dass eine Ereigniszählung die Zuverlässigkeit video-basierter Systeme erreichen kann. Es konnte zudem beobachtet werden, dass die Hypothese, wonach Videoerkennungssysteme bei schlechter Sicht Schwierigkeiten haben könnten, während Audio-basierte Systeme weiterhin zuverlässig bleiben, nicht verworfen werden kann. Für die Klassifizierung von Fahrzeugtypen erreichten die im Rahmen dieser Arbeit trainierten maschinellen Lernmodelle eine Spitzengenauigkeit von 95–97% und eine durchschnittliche Genauigkeit von 88,7%. Dies demonstriert die Machbarkeit der Audio-basierten Klassifizierung, obwohl Klassenungleichgewichte zugunsten von Autos dazu führten, dass der Klassifikator letztlich als binär einzustufen ist.

Hardwarebegrenzungen stellten eine Herausforderung dar; Der Khadas VIM3 - Mikrocomputer, der das Herzstück der verfügbaren Sensorbox bildet und auch die Erkennungspipeline hosten sollte, erwies sich aufgrund von Speicherbeschränkungen als ungeeignet für die Verarbeitung an Bord, wodurch eine Echtzeitbereitstellung unter den aktuellen Bedingungen nicht möglich war. Nichtsdestotrotz repräsentierte

der entwickelte Workflow den vorbeifahrenden Verkehr mit hoher Genauigkeit, insbesondere unter der Bedingung, dass Autos den Großteil des Verkehrs ausmachten, was auch die am Teststandort beobachtete Zusammensetzung war. Während die Methoden am Teststandort validiert wurden, sind weitere Studien erforderlich, um ihre Anwendbarkeit in unterschiedlichen Verkehrsumgebungen zu bewerten.

Diese Ergebnisse unterstreichen das Potenzial der Audio-basierten Fahrzeugerkennung als komplementären Ansatz zu Videosystemen und bieten bedeutende Implikationen für skalierbare und kosteneffiziente Lösungen zur Verkehrserfassung.

Abstract

This thesis explores the potential of classifying vehicles using audio data, offering an alternative or addition to traditional video-based traffic detection systems. Four primary research questions guide this work: the accuracy of vehicle counting (RQ1) which is evaluated against reference data, the feasibility of vehicle type classification via machine learning (RQ2) which is also evaluated against the same reference data, the influence of hardware on real-time computations (RQ3) which adresses the already developed and deployed infrastructure, and the overall reliability of audio-based traffic representation (RQ4) in respect to available reference data and the measurement setup.

To address these questions, data from a sensor box with audio recording capabilities which is deployed alongside an existing video recognition system was used. The data obtained by the video recognition system served as reference data for evaluating the performance of the audio recognition system. A software prototype leveraging machine learning was developed, achieving a vehicle counting accuracy of up to 98%, proving that a counting of events can approach the reliability of video based systems. It could also be observed that the hypothesis that video recognition systems would struggle with bad visibility while audio based systems will still be reliable can't be abandoned. For vehicle type classification, the machine learning models trained in during thesis reached a peak detection accuracy of 95–97% and an average of 88.7%, demonstrating the viability of audio-based classification despite class imbalances favoring cars which lead to the classifier being of binary type.

Hardware limitations posed challenges; the Khadas VIM3 - microcomputer which is the heart of the available sensorbox and should also have been able to host the detection pipeline was insufficient for on-board processing due to memory constraints, making real-time deployment infeasible under current conditions. Nevertheless, the developed workflow successfully represented passing traffic with high accuracy, provided the prevalence of cars dominated the traffic composition which was also the observed composition at the test site. While the methods were validated at the test location, further studies are required to assess their applicability across varying traffic environments.

These findings underscore the promise of audio-based vehicle detection as a complementary approach to video systems, with significant implications for scalable and cost-effective traffic monitoring solutions.

Contents

CHAPTER 1

Introduction

In contemporary traffic control systems, video recognition technology has become ubiquitous and plays a key role in monitoring and managing vehicular activities. However, this prevalent method is not without limitations, which pose challenges such as high costs, substantial computational demands, and inherent restrictions under adverse weather conditions. As traffic surveillance technology continues to evolve, it is imperative to explore additional approaches that address these drawbacks while maintaining efficiency and cost effectiveness.

One of the main methods for detecting and classifying vehicles at the current time is video recognition, as it provides an easy approach for almost every situation where a camera can be mounted. Video-based vehicle detection and classification systems have been developed and have been proven to be viable solutions for real-time traffic data collection. However, they are susceptible to drawbacks such as dependency on good weather conditions and clear visibility, which can hamper detection in adverse circumstances.

This research investigates the potential advantages of audio classification of vehicles as a viable addition, or even alternative, to video-based traffic control systems. The focus on audio classification stems from the recognition of its unique strengths, particularly in overcoming challenges posed by inclement weather conditions such as snow and fog, under which conventional video recognition may falter.

The methodology used in this research endeavors to address research questions regarding the number of passing vehicles, deduction of vehicle types, and representation of traffic, exploring the potential of audio recognition technology in traffic monitoring. This includes an in-depth description of the equipment, data collection processes, preprocessing techniques, feature extraction methods, machine learning

model, experiment design, validation methods, ethical considerations, limitations, data analysis approaches, and an overall summary of the research methodology.

To ensure correct results and verifiability of predictions made by the machine learning model, input data has to be correctly labeled. The basis for this is the reference data acquired from video recognition, which is automatically synchronized with results from audio preprocessing. Resulting labeled data is then analyzed for plausibility, ensuring that data used for verifying the model is indeed correct in respect to the reference data. Data labeling has to be automated to handle large input datasets to quickly accommodate new input data, where manual labeling of such large data sets would be very time consuming and prone to error, which is not feasible in the scope of this thesis.

In the results section, there will be a detailed description of the results achieved during this thesis, as well as a comparison to existing video recognition results utilizing confusion matrices to show how well the model could detect a car or predict the correct vehicle type from the available classes. The results will also be discussed in detail with respect to the relevant research questions.

The primary motivation behind this research lies in the pursuit of a robust, real-time, and cost-efficient traffic monitoring system. Unlike video-based approaches that often require complex setups and significant computational resources, audio recognition presents the prospect of a streamlined and economically viable solution. Through the usage of a sensor box equipped with microphones and a microcomputer, this research seeks to showcase the feasibility of implementing audio recognition technology in traffic monitoring.

The objectives of this thesis are multifold. The first aim is to assess the accuracy of vehicle detection derived from the audio recordings and ascertain its comparability with the already deployed measurement setups. Subsequently, the potential to accurately calculate the number of passing vehicles, to discern the vehicle type, and to evaluate the influence of the preliminary setup on the computation speed is explored. By addressing these research questions, audio recognition of passing vehicles is established as a reliable and efficient alternative to traditional video-based traffic monitoring.

This introduction sets the stage for a comprehensive exploration of audio recognition in the realm of traffic monitoring. First, the current state of the art will be explored to provide an overview of available methods. The subsequent sections will delve into the methodology employed, present the results of the experiments conducted, and provide a thorough discussion of the findings in relation to the research questions outlined.

CHAPTER 2

Contribution of this Work

This thesis provides a reliable alternative to an existing video monitoring setup for classifying vehicles passing by a recording station, consisting of a video camera of which data is analyzed by a subsequent machine learning system to generate labels for passing vehicles, as well as a sensorbox containing microphones and other equipment, as further described in Section 4.3. It is shown that it is possible to have a machine learning setup based on the available audio data that produces as reliable results as the already proven video recognition system, and which is able to predict the correct vehicle type for new data, which can be verified with results video data recognition from the same time period.

Furthermore, by getting accurate results, it is shown that a very simple setup consisting of Micro-Electro-Mechanical Systems (MEMS) microphones in a sensor box and a machine learning classifier for the resulting audio stream can compete with a camera recognition system used by Austrian Motorway and Expressway Financing Joint-Stock Company (ASFINAG). This provides a viable and cheaper alternative that is also easier to set up, as well as posing less concerns regarding privacy infringements via camera recording, as the recorded audio snippets would only be available as a spectrogram, yielding no sensible data about possibly recorded conversations. By showing that similar detection accuracy as vehicle classification via video recognition can be achieved with this simple audio recognition system, it is shown that this audio-based setup provides a viable alternative for the existing video-based system, and provides a proof of concept which can be tested in other locations in the future.

2.1 Research questions

A short overview of the main research questions tackled in this thesis, further evaluated in chapter 7

2.1.1 Calculation of Passing Vehicles (RQ1)

Can the number of passing vehicles be calculated accurately with respect to the number of lanes, or for a single lane only? Here, a software prototype will be built and tested against reference data obtained from video data. Research through development will help to refine the computations so that they can be verified by comparison with results from video analysis.

2.1.2 Vehicle Type Deduction (RQ2)

Can the type of a passing vehicle be deduced accurately with machine learning? The results will be tested against video data obtained at the same location and time as the audio data, and photoelectric barriers can also be used to gather vehicle count and timestamps for verification.

2.1.3 Influence of Hardware Setup (RQ3)

How does the hardware setup, which is based on a Khadas VIM3 as the microcomputer in use with a Neural Processing Unit (NPU) Machine Learning (ML) module for ML workloads, influence the computation, and is it possible to finish computations close to real-time, where results can be acquired within a maximum constant time window of a few seconds after each event? A window of five seconds is the desired maximum here, as the evaluation period on the sensor box has a sliding window of five seconds in length.

2.1.4 Overall Representation of Passing Traffic (RQ4)

Can the workflow developed in this thesis lead to an accurate representation of passing traffic in regard to the existing video based detection system? Could this detection method be used in different locations as well, and how would that change in surroundings affect detection accuracy? This depends on the first three research questions and will be examined by research through development and comparing results from audio detection with reference measurements for the same period of time at the same location obtained from video analysis. All of these results will then be interpreted and described and will yield the final verdict.

CHAPTER 3

State of the Art, Literature Review and Impact

3.1 Introduction

In the era of smart cities and advanced traffic management systems, the accurate detection and classification of vehicles play a pivotal role in ensuring efficient transportation, safety, and security [1]. One of the main methods for detecting and classifying vehicles at the current time is video recognition, as it provides an easy approach for almost every situation where a camera can be mounted. Video-based vehicle detection and classification systems have been developed and proven to be viable solutions for real-time traffic data collection [2]. Using such systems has some drawbacks, however, like the dependency on good weather conditions as well as clear visibility as the main impacting factors, which can hamper detection in adverse circumstances.

Other popular detection methods with a lesser susceptibility to these influences can be based on inductive loops, radars, infrared sonar, microwave detectors, or combinations of those technologies, but those are not relevant to this work.

3.2 Video Recognition in Traffic Monitoring and Analysis

In traffic monitoring and analysis systems, video recognition plays a pivotal role, as it can offer real-time insights into traffic flow, safety, and congestion, as well as tracking vehicles and vehicle types at certain locations [3]. Numerous methods implementing video recognition can be employed for video-based detection and classification of vehicles, each posing its own advantages and disadvantages [4]. As the importance

of efficient vehicle detection is ever increasing, especially with more vehicles being on the roads, and with the expansion of road networks, video monitoring has become an essential method for automatic traffic analysis [5].

Video recognition plays a crucial role in traffic monitoring systems, offering real-time insights into traffic flow, congestion, and safety. Several methods and technologies are employed for video-based vehicle detection and classification, each with its own strengths and limitations [4].

3.2.1 A short introduction of traditional computer vision techniques employed in traffic monitoring

There are many different traditional algorithms that can be employed for implementing computer vision, such as background subtraction [6], edge detection [7], [8], and object tracking [8]. These basic methods are widely used in traffic surveillance systems to detect and track vehicles [8]. Although being computationally efficient and easy to implement, these presented computer vision methods can struggle with complex scenarios, including obstructions in the field of view, varying lighting conditions, and noisy or cluttered backgrounds, as they all rely on handcrafted features and heuristic rules to detect vehicles in input video streams [9]–[11].

3.2.2 Classification approaches depending on deep learning algorithms

Deep Learning-Based Approaches

Deep learning models, particularly convolutional neural networks (CNNs), have revolutionized video recognition tasks by automatically learning hierarchical representations from raw pixel data [11]. CNN-based architectures, such as Faster R-CNN [12], YOLO [13], and SSD [14], have demonstrated remarkable performance in real-time object detection and tracking in traffic videos. Deep learning models excel at capturing complex spatial and temporal patterns in video data, leading to highly accurate and robust vehicle detection results [15]. However, training deep neural networks requires large annotated datasets and significant computational resources [16].

Multi-View and 3D Reconstruction

Multi-view and 3D reconstruction techniques leverage multiple camera perspectives and depth information to enhance the accuracy and reliability of vehicle detection [17]. By fusing information from different viewpoints, multi-view systems can mitigate occlusion and improve the accuracy of object localization. Furthermore, 3D reconstruction enables the estimation of vehicle trajectories and motion dynamics,

providing valuable insights into traffic behavior [18]. However, deploying multi-view and 3D reconstruction systems may require extensive infrastructure and calibration efforts, limiting their scalability and deployment in real-world environments [19].

Foreground Segmentation and Motion Analysis

Foreground segmentation methods segment moving objects from static backgrounds in video sequences, allowing for the detection and tracking of vehicles based on motion cues [20]. Motion-based approaches, such as optical flow analysis and motion history images, capture temporal dynamics and motion patterns of vehicles in traffic scenes. These methods are effective at detecting moving objects and distinguishing them from static background elements [21]. However, motion-based techniques may struggle with handling complex motion patterns, occlusions, and variations in object speed [22].

Hybrid Systems and Ensemble Methods

Hybrid systems combine multiple video processing techniques, such as deep learning models, motion analysis, and background subtraction, to improve overall detection performance. Ensemble methods integrate predictions from diverse detection algorithms to enhance robustness and reliability in challenging scenarios [23]. By leveraging the strengths of different approaches, hybrid and ensemble systems can achieve superior detection accuracy and resilience to environmental factors. However, designing and optimizing hybrid systems may require careful integration and parameter tuning to achieve optimal performance [24].

In summary, video recognition methods in traffic monitoring encompass a diverse range of techniques, from traditional computer vision algorithms to deep learning-based approaches. Each method offers unique advantages and challenges, highlighting the importance of selecting the most appropriate approach based on specific application requirements and environmental conditions [25].

3.3 Audio Recognition in Traffic Monitoring and Analysis

To support existing video analysis and monitoring systems, audio recognition can play a vital role in augmenting those systems by providing additional sensory information about surrounding traffic conditions [26]. Several different methods and techniques can be employed for audio-based vehicle detection, classification and monitoring, which come with their unique advantages and challenges [27]–[29].

3.3.1 Spectral Analysis and Feature Extraction

Commonly utilized techniques for extracting discriminative features from audio signals are spectral analysis techniques, such as Fourier Transform, most often Fast Fourier Transform (FFT), and Mel-Frequency Cepstral Coefficients (MFCC), which can also be used for vehicle classification based on audio data [30], [31].

The audio signal is decomposed into constituent frequency components with Fourier Transform, while the Mel-Frequency Cepstral Coefficients method captures both spectral and temporal characteristics of input sounds. Both methods combined enable the identification of vehicle-specific acoustic signatures [32]. These methods can also be combined with machine learning algorithms for further interpretation of results. To achieve that combination of spectral analysis and machine learning, the output of feature extraction methods can be passed to a machine learning model, for example as spectrogram data which the model can be trained on, as well as identify samples later. A known caveat is the susceptibility of spectral analysis methods to variations in environmental noise and/or background clutter, as well as data from different locations, which can affect classification accuracy and has to be handled separately [33].

3.3.2 Machine Learning-Based Approaches

There are a couple of machine learning models which are employed in vehicle classification based on audio-data, such as random forest classifiers, support vector machines (SVM) and neural networks, where such models distinguish different vehicle types. Different kinds of machine learning models are employed for audio-based vehicle classification tasks, such as support vector machines (SVM), random forests, and neural networks. While support vector machines utilize hyperplane classifiers for separation of vehicle classes in feature space, neural networks capture audio patterns via using hierarchical representations [34], [35].

Machine learning approaches offer good scalability and high flexibility and can be used to model diverse acoustic features, a caveat is that most of them require extensive and well labeled datasets for training to produce reliable results, and are susceptible to overfitting in noisy environments [34].

As support vector machines have shown promise in varying audio classification tasks such as environmental audio classification and music genre classification, an SVM model has also been chosen as a first approach to testing a proof of concept for this thesis [36]–[38]. They also sport a high ability for generalization and can efficiently deal with high-dimensional input features [34], which is another reason that qualifies them for testing. Furthermore, they can also be combined with ensemble methods, such as artificial neural networks, which can further improve their classification accuracy [39]. As such, machine learning approaches can offer robust methods to

accurately differentiate between various audio inputs, thus leading to the possibility of classifying vehicle types based on their passing-by sound.

3.3.3 Time-Frequency Analysis and Wavelet Transform

Localized spectral decomposition of audio signals is possible by utilizing Wavelet Transform, which is especially useful for examining and characterizing transient sounds and extracting acoustic features [40], which can also be useful for vehicle classification. During the Wavelet Transform, the audio signal is decomposed into different frequency components at varying resolutions [41], which enables enhanced time-frequency localization and feature extraction for both stationary and non-stationary signals as it offers arbitrary time-frequency resolution [42], which underscores the resiliency of Wavelet Transform methods to background noise variations and allow capture of dynamic acoustic patterns, which are crucial for vehicle classification. A challenge arising with these methods is the appropriate selection of wavelet basis functions and decomposition scales, which affects their practical implementation [43].

The MEL scale

The MEL scale is a scale of pitches judged by listeners to be equal in distance one from another. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 MELs [44].

The Morlet Wavelet Transformation

The challenges of choosing appropriate wavelet basis functions and decomposition scales are addressed by the Morlet Wavelet Transformation, which offers an empirically validated solution, where the effectiveness of the synchrosqueezed wavelet transform is discussed by [45], a method providing a solution for the selection of a precise wavelet basis function. This provides an indication that a complex-valued wavelet like the Morlet Wavelet can offer improved outcomes compared to real-valued wavelets regarding the selection of wavelet basis functions and decomposition scales, where the analytical expression for the complex Morlet Wavelet is

$$\Psi(t) = \pi^{-1/4} \mathrm{e}^{\mathrm{i}\omega_0 t} \mathrm{e}^{-t^2/2}, \tag{3.1}$$

where ω_0 is the non-dimensional frequency [46].

The Rectified Linear Unit (ReLU) activation function

ReLU is an activation function which is commonly used in neural networks [47]. It allows the model to learn from and make predictions on complex data by introducing a non-linearity to the network, with this simple mathematical expression describing it:

$$f(x) = \max(0, x) \tag{3.2}$$

Advantages of using this function include computational efficiency, sparse activation and mitigating the vanishing gradient problem, which can occur in deep networks during backpropagation. Disadvantages include the possiblity of dead neurons, happening if a neuron receives a large amount of negative input data, and that it is not very suitable for data containing a lot of negative values, which in our case poses no problems, since the input data passed to the model from preprocessing contains only positive values, as they describe MEL spectrograms in RGB.

3.3.4 Train-test split and separate dataset for testing

In machine learning, a train-test split is a fundamental technique used to assess a models performance and generalizability. By dividing the dataset into two subsets—one for training the model (training set) and one for evaluating it (test set) one can ensure that the models ability to predict new data is accurately measured. This process helps to prevent overfitting, where the model learns the training data too well, capturing noise instead of the underlying patterns. Typically, a common split ratio is 70-80% for training and 20-30% for testing, though this can vary based on dataset size and specific needs. Additionally, cross-validation methods, such as k-fold cross-validation, can be employed for a more robust evaluation, where the dataset is split multiple times to ensure consistent performance across different subsets. This practice is essential for developing models that are not only accurate but also reliable when applied to real-world data.

Advantages of Using a Separate Dataset for Testing

Utilizing a separate dataset for testing offers several key advantages. Primarily, it allows for the detection and prevention of overfitting, ensuring that the model generalizes well to new data rather than merely memorizing the training set. This separation provides a clear and unbiased evaluation of the model's performance, highlighting its true predictive power on unseen data. Furthermore, it facilitates hyperparameter tuning and model comparison by providing a consistent benchmark against which different models and configurations can be measured. Overall, this approach leads to the development of more robust and reliable machine learning models.

3.3.5 Environmental Noise Reduction Techniques

As analytical as well as machine learning methods benefit greatly from clear data, it is important to be able to reduce environmental and background noise in audio signals to ensure that only the desired part of the signal can be extracted and trained on, if necessary. Thus, noise reduction algorithms can improve the accuracy of audio-based classifiers by preprocessing audio signals to enhance signal-to-noise ratio (SNR) and reducing or suppressing background noise. Commonly utilized methods of noise reduction are spectral subtraction, Wiener filtering, and adaptive noise cancellation and can enhance audio quality of available signals [48], which can also be helpful to improve the accuracy of vehicle detection in acoustic signals.

3.3.6 Integration with Video-Based Systems

By using audio-visual cues, moving vehicles can be detected utilizing self-supervised approaches, where cross-modal model distillation using auditory information enhances the detection capabilities of a video-based model, thus highlighting the importance and possibilities of adding audio recognition of vehicles to existing video recognition [49].

In video-based systems, auditory cues can be used to enhance the existing model, but this dual approach can also be used to train models relying solely on auditory data [49], which means by leveraging spectral analysis, machine learning, and noise reduction techniques, systems based on audio recognition can augment existing traffic surveillance systems and partly even replace them [50].

3.4 Significance of Audio Recognition in Traffic Monitoring

Audio recognition plays a significant role in enhancing traffic control and management capabilities by providing additional sensory information about surrounding traffic conditions. Unlike video-based approaches, which rely primarily on visual cues, audio recognition offers complementary insights into vehicle behavior, road conditions, and driver interactions. The significance of audio recognition in traffic control can be understood from several perspectives.

3.4.1 Enhanced Situational Awareness

Audio recognition enhances traffic controllers' situational awareness by providing real-time auditory cues about traffic flow, congestion, and emergency situations. The sounds of vehicle engines, horns, sirens, and tire screeches can convey critical information about traffic dynamics and potential hazards. By integrating audio cues with visual data from surveillance cameras, traffic controllers can gain a more

comprehensive understanding of traffic conditions and make informed decisions to optimize traffic flow and ensure safety [51].

3.4.2 Early Detection of Anomalies

Audio recognition enables early detection of anomalies and irregularities in traffic patterns, such as accidents, breakdowns, and road closures [@anomaly_detection]. Unusual sounds, such as collisions, vehicle malfunctions, or tire blowouts, can serve as early warning signals for traffic incidents, allowing authorities to respond promptly and mitigate potential disruptions. By continuously monitoring audio signals from roadside sensors or microphones, traffic control centers can identify abnormal events and initiate appropriate interventions to minimize traffic congestion and ensure efficient traffic management [51].

3.4.3 Non-Visual Monitoring

Audio recognition provides a non-visual monitoring capability that complements traditional video-based surveillance systems. Unlike video cameras, which may have blind spots or limited coverage areas, audio sensors can capture sounds from all directions and distances. Keeping in mind that most microphones still are directional, this potential omnidirectional sensing capability enables audio recognition systems to detect and localize traffic events beyond the field of view of cameras, such as incidents in adjacent lanes or obscured by obstacles, which can be a huge advantage. Non-visual monitoring using audio sensors enhances overall situational awareness and reduces reliance on line-of-sight visibility.

3.4.4 Advantages of Audio Recognition in Traffic Monitoring

In summary, audio recognition technologies offer valuable capabilities for enhancing traffic monitoring by providing non-visual sensory information about traffic conditions, detecting anomalies and irregularities, enabling non-visual monitoring, and facilitating analysis of traffic flow. By integrating audio recognition with existing traffic surveillance systems, traffic monitoring can be enhanced by no longer being reliant only on line-of-sight data.

3.5 Machine Learning in Audio Classification

As some data cannot be covered by cameras and video recognition alone, audio signals have to be used to detect special events, i.e. tire skidding, or even just a vehicle passing by in adverse weather conditions like heavy snowfall or rain, or otherwise low visibility where camera systems will face problems [52]. Machine learning plays a pivotal task in the accurate classification of any detected events

in the audio stream, as vehicles have to be distinguished by their acoustic features alone while still resulting in a high detection accuracy [53].

3.5.1 Popular Machine Learning Approaches for Audio Classification

There are many different methods to utilize machine learning for audio classification, where some of the most common approaches will be briefly outlined.

Support Vector Machines (SVM)

SVM is a popular and commonly used machine learning approach, and is used in audio analysis for both binary and multiclass classification tasks [54]. This makes SVMs effective in classifying different audio inputs, making them suitable for vehicle type classification by determining the optimal hyperplane that separates different classes in feature space, while being robust against overfitting as well as being able to generalize to unseen data, making them suitable for classification with limited labeled data [55]. Limitations arise with large-scale datasets and higher dimensional feature space, where computational inefficiencies and memory constraints become the main problemns. Interesting insights come from the use of well-known audio features like MFCCs (Mel-frequency cepstral coefficients) for classification by SVMs [56], which is an approach also employed in this thesis.

Random Forests

Random forests are ensemble learning algorithms that combine multiple decision trees to improve classification performance and robustness [57]. In (audio) classification tasks, random forests learn to differentiate between classes based on extracted features and labeled training data [58]. Random forests offer scalability, interpretability, and resistance to overfitting, making them suitable for handling complex audio patterns and environmental variations [59]. However, random forests may require careful hyperparameter tuning to achieve optimal performance and may be susceptible to biases in training data distributions, as well as depending on larger data sets which require more resources [60]. As the dataset which is available for testing in this thesis is rather small as far as datasets for machine learning go, this method was not tested in this thesis.

Neural Networks

Neural networks, particularly deep learning architectures, have emerged as powerful models for audio classification tasks. Convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variants learn hierarchical representations from raw audio signals, enabling end-to-end learning of discriminative features [61], [62]. CNNs operate directly on time-frequency representations of audio signals, while

RNNs capture temporal dependencies and long-range contexts in audio sequences [63], [64]. Neural networks offer scalability, flexibility, and superior performance in audio classification tasks but require large annotated datasets and significant computational resources for training [65]. As neural networks also require larger datasets, this approach was only employed once a certain amount of available data was reached, but should pose beneficial for future developments, as neural networks scale well with added data to improve their accuracy, whereas random forest classifiers plateau after a certain amount of data is reached [66].

Gaussian Mixture Models (GMM)

Gaussian Mixture Models (GMMs) are commonly used probabilistic models in audio classification for clustering and density estimation tasks [67]. GMMs excel in modeling the distribution of audio features within different vehicle classes by utilizing a mixture of Gaussian distributions, facilitating probabilistic inference and classification [68]. They are appreciated for their simplicity, interpretability, and robustness to noise and variability in audio data [69]. However, GMMs may struggle with capturing complex nonlinear relationships and often require careful initialization and parameter tuning to achieve optimal performance [70].

Ensemble Learning and Model Fusion

Ensemble learning techniques, such as bagging, boosting, and stacking, combine predictions from multiple base classifiers to improve overall classification accuracy and robustness [71]. By aggregating diverse classifiers' predictions, ensemble methods mitigate individual models' weaknesses and enhance overall performance [72]. Model fusion approaches integrate outputs from different machine learning models, such as SVMs, random forests, and neural networks, to leverage their complementary strengths and improve classification results [73]. However, designing and optimizing ensemble systems may require careful selection of base classifiers, feature representations, and fusion strategies to achieve optimal performance [74].

3.5.2 KERAS CNN model

The Keras Convolutional Neural Network (CNN) model is described in a little more detail with its advantages highlighted, as it is a major part in this thesis.

Advantages of Keras [75]

1. **User-Friendly Abstraction**: Keras, as an open-source neural network library, provides a high-level abstraction for building and training neural networks. This abstraction simplifies the process of constructing complex neural network

architectures, allowing researchers and practitioners to focus on the design and experimentation rather than dealing with low-level implementation details. Thus, a working model can be built with ease and without having to dive too deep into a new framework.

2. **Modularity and Extensibility**: Keras follows a modular design philosophy, enabling users to easily build, modify, and experiment with various neural network components. This modularity facilitates the incorporation of different layers, activation functions, and optimizers, making it well-suited for adapting to the specific requirements of audio classification tasks.

3. **Integration with TensorFlow**: Keras is tightly integrated with TensorFlow, one of the most popular deep learning frameworks. This integration offers the advantage of combining the simplicity of Keras with the computational efficiency of TensorFlow, resulting in a powerful and flexible platform for building and training neural networks.

4. **Community Support and Documentation**: Keras benefits from a large and active community, which translates into extensive documentation, tutorials, and a wealth of pre-trained models. This community support can significantly accelerate the development and deployment of audio classification models, providing valuable resources for troubleshooting and optimization.

5. **Implementation Considerations**: The practical implementation of audio classification using spectrogram images with Keras involves pre-processing the audio data, constructing a suitable neural network architecture, and training the model on labeled datasets. The subsequent evaluation and fine-tuning of the model are crucial steps in achieving optimal performance.

6. **Conclusion**: In conclusion, Keras emerges as a preferred choice for developing audio classification models based on spectrogram images. Its user-friendly abstraction, modularity, integration with TensorFlow, and robust community support make it a compelling option for researchers and practitioners seeking a balance between simplicity and flexibility in their machine learning endeavors. The subsequent chapters will delve into the details of the implementation, experimentation, and evaluation of the proposed audio classification model.

3.5.3 Machine Learning Metrics [76]

Three of the most important metrics for machine learning are precision, recall, F1 score and accuracy.These metrics provide valuable data for comparing different Machine Learning models.

1. **Precision:** Precision is the ratio of true positive predictions to the total number of positive predictions made by the classifier. It measures the accuracy of the positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.3}$$

2. **Recall:** Recall is the ratio of true positive predictions to the total number of actual positive instances in the dataset. It measures the ability of the classifier to find all the positive instances.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.4}$$

3. **F1 Score:** The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It's a useful metric when there is an uneven class distribution.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

4. **Accuracy:** Accuracy measures the proportion of correctly classified instances (both positive and negative) among all instances in the dataset.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}} \tag{3.6}$$

As Accuracy (Acc) is the defining metric chosen in this thesis, as it is a binary metric that fits very well, it is described in some m Mathematically, accuracy (Acc) is defined as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.7}$$

For a binary classifier, this calculation simplifies to:

$$Acc = \frac{TP_{car} + TP_{noise}}{Totalsamples} \tag{3.8}$$

Where:

- TP = True Positives (correctly predicted positive samples)
- TN = True Negatives (correctly predicted negative samples)
- FP = False Positives (incorrectly predicted positive samples)
- FN = False Negatives (incorrectly predicted negative samples)

Confusion Matrices

Another important metric for evaluating a models performance that is especially suited for visualising results is the confusion matrix, which is built from true and false positives as well as negatives. It is a square matrix that compares the actual labels of a dataset (true classes) with the labels predicted by the model. The matrix has the following components:

- **True Positives (TP):** The number of instances correctly classified as a particular class.
- **True Negatives (TN):** The number of instances correctly identified as not belonging to a particular class.
- **False Positives (FP):** The number of instances incorrectly classified as a particular class (also known as Type I error).
- **False Negatives (FN):** The number of instances incorrectly classified as not belonging to a particular class (also known as Type II error).

The confusion matrix provides a detailed breakdown of how the model is performing across all classes, allowing for the identification of specific strengths and weaknesses in its predictions. It is particularly useful for understanding not only overall accuracy but also where the model may be making systematic errors.

3.5.4 The loss function

In the context of machine learning (ML), loss is a measure of how well or poorly a model's predictions match the actual data. It quantifies the difference between the predicted outputs and the true values.

The loss function calculates this difference and outputs a single number, known as the loss, which represents the model's error on a given data point or batch of data. The goal of training an ML model is to minimize this loss, meaning you want to adjust the model's parameters (weights and biases) to make the predictions as accurate as possible. Minimizing the loss during training helps the model improve its predictions over time.

3.5.5 The role of machine learning in audio classification

In summary, machine learning plays a crucial role in audio classification for traffic monitoring applications, offering diverse algorithms and frameworks for vehicle detection and identification. By leveraging SVMs, random forests, neural networks, and ensemble methods, audio-based systems can achieve robust and accurate vehicle classification results in various environmental conditions.

3.6 Integration of Audio and Video Technologies

Integration of audio and video technologies in traffic monitoring systems enables multimodal data fusion and enhances overall situational awareness [77]. By combining audio and visual cues, multimodal systems can improve vehicle detection and classification accuracy, especially in challenging scenarios with poor visibility or occlusions [78]. Several approaches and techniques are employed for integrating audio and video technologies, each offering unique advantages and challenges [79]. Subsequently, some of the most common methodologies and techniques for multimodal fusion will be discussed to provide a detailed overview.

3.6.1 Synchronous Data Acquisition

Synchronous data acquisition involves capturing audio and video signals simultaneously using synchronized sensors and recording devices [80]. By aligning audio and video streams temporally, synchronous systems enable direct correlation between audio events and visual observations, facilitating multimodal fusion and analysis [81]. Synchronous data acquisition ensures temporal consistency and accuracy in audio-video integration but may require specialized hardware and synchronization mechanisms to maintain temporal coherence [79].

Feature-Level Fusion

Feature-level fusion combines audio and visual features extracted from raw data streams to generate comprehensive representations of scenes [82], which can also be helpful in traffic monitoring. Audio features, such as spectrograms [83] and MFCCs [84], are extracted from audio signals, while visual features, such as color histograms and motion vectors, are extracted from video frames [85]. Feature-level fusion techniques concatenate or combine audio and visual features into a single feature vector, which is then fed into a classifier for vehicle detection and classification [86], [87]. Feature-level fusion offers flexibility and modularity in integrating heterogeneous data sources but may require careful feature selection and normalization to ensure compatibility between modalities [88].

Decision-Level Fusion

Decision-level fusion combines decisions or predictions from independent audio and video classifiers to make final classification decisions [89]. Audio-based classifiers and video-based classifiers operate independently to detect and classify vehicles based on extracted features [90]. Decision-level fusion algorithms combine the output probabilities or labels from audio and video classifiers using fusion rules, such as majority voting, weighted averaging, or Dempster-Shafer theory [91], [92]. Decision-level fusion offers robustness to individual classifier uncertainties and noise but may

require calibration and optimization of fusion rules to achieve optimal performance [91].

Deep Multimodal Learning

Deep multimodal learning architectures integrate audio and video information within a unified neural network framework, enabling joint feature learning and representation [93]. Multimodal neural networks incorporate audio and visual streams as input channels and learn hierarchical representations from raw data. These architectures leverage shared representations and cross-modal interactions to capture complementary information from audio and video modalities [94]. Deep multimodal learning offers end-to-end optimization of fusion parameters and enhances model interpretability and robustness [95]. However, training deep multimodal networks requires large annotated datasets and may suffer from domain mismatches between audio and video modalities [96].

3.6.2 Possibilities of combining audio and video technologies

In summary, integration of audio and video technologies in traffic monitoring systems enables synergistic exploitation of complementary information from multiple sensory modalities. By leveraging synchronous data acquisition, feature-level fusion, decision-level fusion, late fusion, and deep multimodal learning techniques, multimodal systems can achieve enhanced vehicle detection and classification performance in diverse traffic scenarios.

3.7 Challenges and Limitations in Existing Approaches for Traffic Monitoring and Analysis

Despite significant advancements in video and audio-based traffic monitoring technologies, several challenges and limitations persist in existing approaches. These challenges stem from environmental factors, technological constraints, and algorithmic limitations, affecting the overall effectiveness and scalability of traffic monitoring systems.

3.7.1 Camera Systems

Camera based traffic monitoring system are some of the most widely used solutions, but they are not without issues. Subsequently, we will discuss the most important concerns concerning the viability of such systems.

Environmental Variability

Environmental factors, such as lighting conditions, weather variations, and background clutter, pose significant challenges for video and audio-based traffic monitoring systems. Changes in lighting conditions, such as shadows, glare, and reflections, can affect object visibility and pose challenges for object detection algorithms. Similarly, adverse weather conditions, such as rain, fog, and snow, can degrade sensor performance and reduce data quality. Background clutter, such as foliage, signage, and other vehicles, can introduce distractions and occlusions, complicating object segmentation and tracking [97]–[99].

Sensor Limitations

Sensor limitations, including resolution, field of view, and sensitivity, impact the quality and reliability of data collected by traffic monitoring systems. Low-resolution cameras may struggle to capture fine-grained details of vehicles, leading to inaccurate object localization and classification. Narrow field-of-view sensors may miss critical events or objects outside their coverage area, limiting situational awareness and detection capabilities. Moreover, sensor sensitivity to environmental noise and interference can affect signal quality and introduce false positives or negatives in detection results.

Data Annotation and Labeling

When using supervised models, data annotation and labeling require substantial human effort and expertise, particularly for large-scale video and audio datasets. Manual annotation of vehicle instances, attributes, and behaviors in video sequences and audio recordings is time-consuming, labor-intensive, and prone to errors. Moreover, subjective interpretations and inconsistencies in labeling criteria can affect dataset quality and compromise algorithm performance. Automated or semi-automated labeling tools and crowdsourcing approaches may alleviate annotation burdens but may introduce annotation biases and variability, or even errors. Unsupervised learning does exist, where models can cluster data on characteristics they decide on, but the results still have to be verified afterwards.

Algorithmic Complexity

Algorithmic complexity and computational resource requirements pose challenges for real-time processing and deployment of traffic monitoring systems. Deep learning models, such as CNNs and RNNs, require significant computational resources for training and inference, limiting their scalability and applicability in resource-constrained environments. Moreover, complex algorithms may suffer from long

processing times, latency issues, and high energy consumption, hindering their real-world deployment in edge computing scenarios [100].

Privacy and Ethical Considerations

Privacy and ethical considerations arise from the collection, storage, and analysis of video and audio data in traffic monitoring systems. Continuous surveillance and monitoring of public spaces raise concerns about individual privacy and data protection. Unauthorized access to sensitive data, such as license plate numbers, vehicle trajectories, and audio conversations, can lead to privacy breaches and misuse of personal information. This can be somewhat alleviated by not storing data persistently, or only recording where privacy is not infringed on, for example by placing the recording equipment near a highway, where personal conversations will not take place and can't be recorded in the audio stream, and the only personal data that has to be removed from video data should be license plates.

3.7.2 Thoughts on possible improvements

In summary, addressing the challenges and limitations in existing video and audio-based traffic monitoring approaches requires interdisciplinary efforts and holistic solutions. By mitigating environmental variability, overcoming sensor limitations, improving data annotation practices, optimizing algorithmic complexity, and addressing privacy and ethical considerations, traffic monitoring systems can achieve greater accuracy and reliability.

3.8 State of the Art Synopsis

In this thesis section, we explored the state of the art in video and audio-based traffic monitoring technologies, machine learning approaches for audio classification, integration of audio and video technologies, challenges and limitations in existing approaches, the significance of audio recognition in traffic control, and the future directions for research and development. Literature suggests that by leveraging diverse methods and techniques from computer vision, signal processing, machine learning, and sensor technologies, traffic monitoring systems can achieve greater accuracy, reliability, and efficiency in monitoring and managing traffic flow, ensuring road safety, and improving transportation infrastructure.

CHAPTER 4

Methodology

This chapter introduces the methodical approach employed in this project, where audio pre-processing provides input data for a machine learning model counting vehicles and categorizing them purely on the basis of their passing-by sound.

The experiment design as well as both hardware and software setups will be described. As data collection and processing is an integral part of this thesis, both the collection of raw audio data and the acquisition of reference data, together with their joining and comparison with following verification are explained in detail. The most important step between data acquisition and model training is the pre-processing of audio data, which is done in a preprocessing pipeline that handles audio processing as well as automated labeling.

The pre-processing pipeline produces input data for machine learning by analysing audio data and filtering out passing-by events, producing audio snippets for each timestamp along with corresponding spectrogram data representing that audio. Timestamps for these events are then compared with reference timestamps, and matching pairs are saved with the data for timestamp, audio data, reference label and audio spectrogram and are passed to the machine learning model in with these parameters as input.

As preliminary information, all code in this project was realized in python, specifically python 3.11.0 [101]. Input data format for audio files is MP3, while reference data is available as JSON. Output data formats are MP3 for audio snippets, and PNG for spectrogram data generated by pre-processing for further evaluation by the model.

4.1 Introduction

First, the existing setup is shortly introduced, which is described in detail in section 5.1.3, as well as how data is collected from that setup. From this data follows the data preprocessing, which is discussed in the following sections.

The microcomputer in use is a Khadas VIM3, which sports a High-performance Amlogic A311D SoC as well as a built-in NPU with 5 TOPS performance [102]. In currently deployed sensor boxes microphones of the type SPK0641HT4H-1 from the manufacturer Knowles are in use, which are well-suited for use near load sites like roadsides, as they have a very good THD to SPL relation, as shown in figure 4.1. Data from these microphones can be verified against reference data from video recognition in the labeling step, which generates labeled data as input for machine learning, and is an important step in the pipeline, as it ensures that available training data can be validated by checking if the resulting data passed to the machine learning model corresponds to its label. After the labeling step, audio snippet data is saved together with corresponding labels and spectrograms. In this step, noise is also extracted from input data to let the model adapt to all kinds of noise encountered between events.

The resulting dataset is then passed to the machine learning model, which can be either trained on a full dataset, or on a standard train-test split. As a final step, results are then validated against reference data, which gives a measurement of the models quality. The methodology employed in this research endeavors to address the research questions described in 7.1.1, 7.1.2, 7.1.3 and 7.1.4, exploring the potential of audio recognition technology in traffic monitoring. This chapter presents an in-depth description of the equipment, data collection processes, preprocessing techniques, machine learning model, experiment design, validation methods, ethical considerations, limitations, data analysis approaches, and an overall summary of the research methodology.

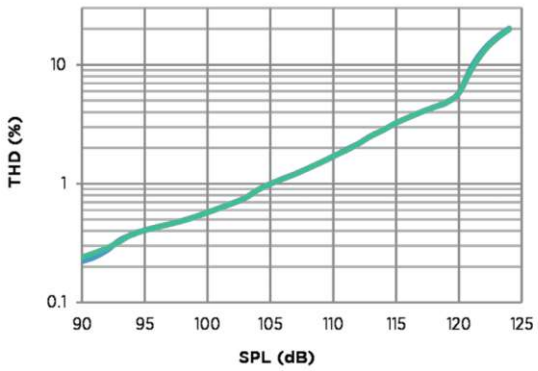

Figure 4.1: Typical THD vs. SPL of Knowles SPK0641HT4H-1

4.2 Experiment Design

The implemented experiment setup is dependent on the existing hardware measurement setup with a sensor box as shown in figures 4.2, 4.3, 4.4, 4.5, described in more detail in section 5.1.3 and handles data of a two-way country road with in- and outbound traffic, which is treated the same way for audio recording, therefore the model neither receives information about the direction of a vehicle, nor does it predict direction. The model is setup in a way that it can be deployed in different locations, and this is a testcase for future development when there are more sensor-boxes available with cameras for labeling, currently this is the only existing setup with video data for reference.

Figure 4.2: Sensorbox topview with microphone shields marked

Figure 4.3: Sensorbox sideview

Figure 4.4: Sensorbox bottomview

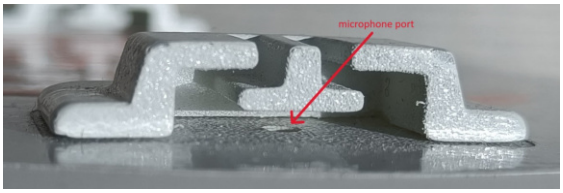

Figure 4.5: Sensorbox microphone input detail with microphone port marked

4.3 Equipment and Setup

Currently deployed sensor boxes are small self-contained setups with approximately 20cm, 15cm and 30cm as width, height and length respectively, as shown in figures 4.2, 4.3, 4.4, 4.5. They contain a power supply, thermal regulation, a microcomputer, antenna, the microphones relevant to this thesis, and several other sensors which are not discussed in this thesis. Shielded ports with a mesh provide protection from environmental factors for the microphones. This ensures fidelity of audio-recordings even in high-noise environments, as the microphones also posses a good Sound Pressure Level (SPL) as well as Total harmonic Distortion (THD), which are essential to making good recordings.

The camera is setup on the side of the road, facing aslant in the roads direction. For more details on the exact camera setup and viewing angle, refer to section , refere to figures 4.6 and 4.7.

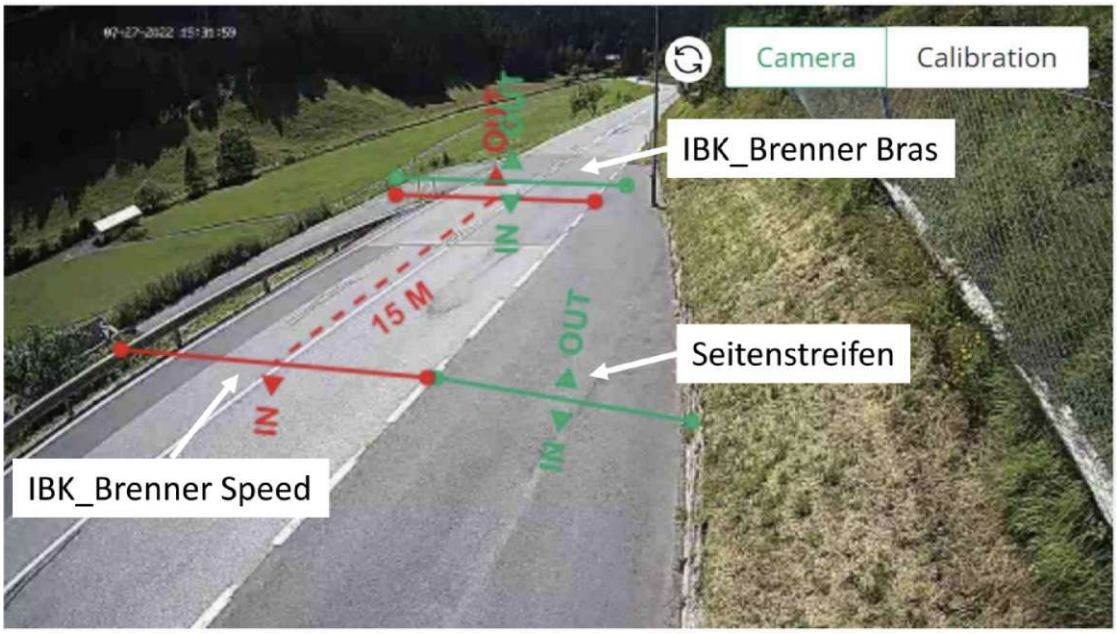

Figure 4.6: Camera setup at Brenner with Sensorbox (SRB)1030, which denotes the deployed sensorbox designation
(provided by Bernard Group - Bernard Technologies GmbH)

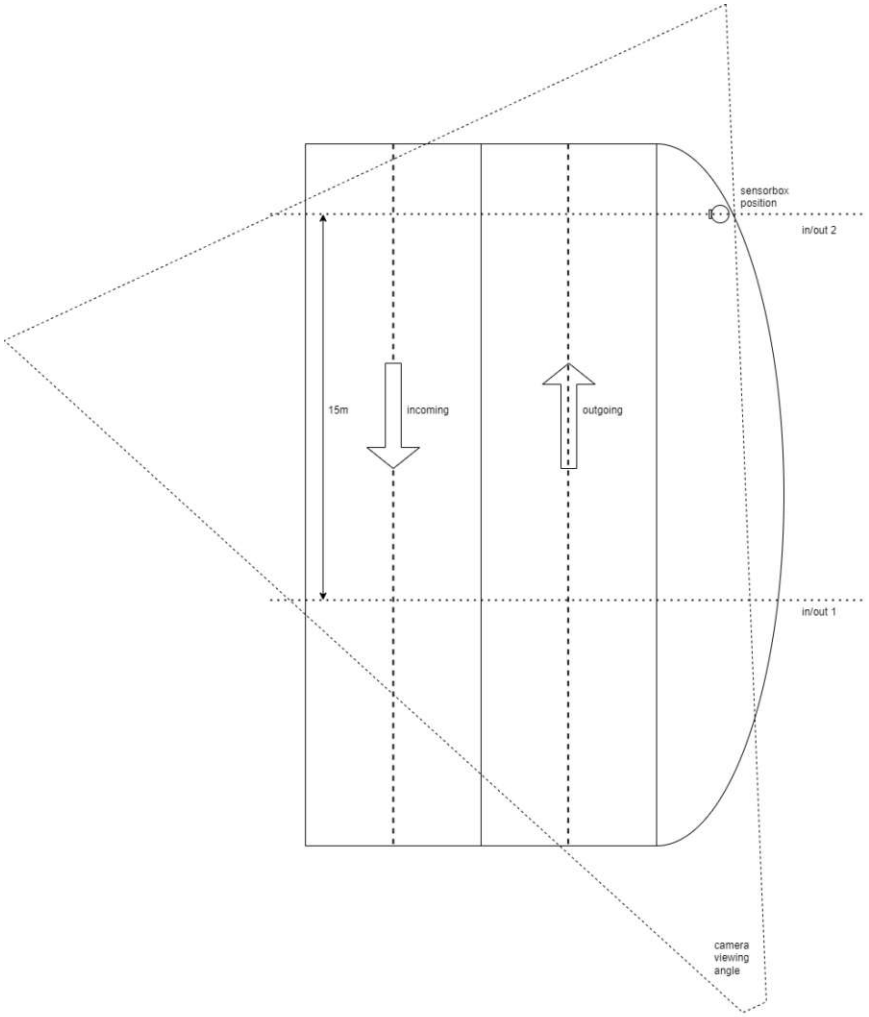

Figure 4.7: Diagram of camera setup at brenner with srb1030

4.4 Data Collection

Collecting data consists of programming time windows via a Representational State Transfer (REST) Application Programming Interface (API) from a starting timestamp to an end timestamp. Raw audio data for this timeframe is then recorded and saved as MP3 files,. Corresponding reference data and passing-by events from recognition are also gathered through a REST API and saved as JSON files for future reference. Both sets of data together allow for a comprehensive analysis of traffic events and

the comparison between results from audio recognition with the results from video recognition.

4.4.1 Raw Data Collection

Data is permanently collected by the microphones and used for loudness calculations, but can also be saved as MP3 files on an SD card mounted on the microcomputer, which is then retrievable by either downloading the files or manually removing the card. As data is per default not saved persistently, time windows have to be programmed for which recordings are made.

4.4.2 Reference Data Collection

The camera deployed at this site is permanently monitoring traffic at this location, and recorded data is passed to a machine learning model which generates output data in the form of JSON files containing type of vehicle, timestamp of the passing-by event and estimated speed as shown as example in listing 4.1. This setup acts as a black box from which reference data is extracted, which is regarded as golden dataset for this thesis, as it provides reference data which can be trusted to be accurate. This data can be downloaded via Client for URL (CURL) commands from an API provided by our partner Bernard Group - Bernard Technologies GmbH [103], which is then used in preprocessing to match extracted audio snippets and their respective timestamps to timestamps with associated labels from video recognition reference data.

Listing 4.1: Exemplary JSON data

```
1       {
2               "crossingLineEvent": {
3                       "class": "car",
4                       "direction": "out",
5                       "lineId": "cdd0ddda-d6d9-40b5-8190-538507719966",
6                       "lineName": "Brenner Speed",
7                       "speedestimate": "89.700996",
8                       "subClass": "van",
9                       "timestamp": "2022-11-10T15:36:42.369552Z",
10                      "trackId": 125829136
11              },
12              "eventSchema": "https://swarm-analytics.com/schema/event/",
13              "node": {
14                      "id": "1ce96134-cf3f-481b-b9ec-f25ed411502c",
15                      "name": "202205B0023_Land Tirol - Brenner"
16              },
17              "stream": {
18                      "id": "9d1b97a0-271d-43b1-be88-e09af6ba6c57",
19                      "name": "202205B0023_Land Tirol - Brenner"
20              },
21              "version": "4.0"
22      },
23      {
24              "crossingLineEvent": {
25                      "class": "car",
26                      "direction": "in",
27                      "lineId": "cdd0ddda-d6d9-40b5-8190-538507719966",
28                      "lineName": "Brenner Speed",
```

```
                        "speedestimate": "79.528717",
                        "timestamp": "2022-11-10T15:36:44.226722Z",
                        "trackId": 1158676485
                },
                "eventSchema": "https://swarm-analytics.com/schema/event/",
                "node": {
                        "id": "1ce96134-cf3f-481b-b9ec-f25ed411502c",
                        "name": "202205B0023_Land Tirol — Brenner"
                },
                "stream": {
                        "id": "9d1b97a0-271d-43b1-be88-e09af6ba6c57",
                        "name": "202205B0023_Land Tirol — Brenner"
                },
                "version": "4.0"
        },
        {
                "crossingLineEvent": {
                        "class": "car",
                        "direction": "out",
                        "lineId": "cdd0ddda-d6d9-40b5-8190-538507719966",
                        "lineName": "Brenner Speed",
                        "speedestimate": "80.000000",
                        "timestamp": "2022-11-10T15:36:48.647166Z",
                        "trackId": 127926288
                },
                "eventSchema": "https://swarm-analytics.com/schema/event/",
                "node": {
                        "id": "1ce96134-cf3f-481b-b9ec-f25ed411502c",
                        "name": "202205B0023_Land Tirol — Brenner"
                },
                "stream": {
                        "id": "9d1b97a0-271d-43b1-be88-e09af6ba6c57",
                        "name": "202205B0023_Land Tirol — Brenner"
                },
                "version": "4.0"
        },
        {
                "crossingLineEvent": {
                        "class": "truck",
                        "direction": "out",
                        "lineId": "cdd0ddda-d6d9-40b5-8190-538507719966",
                        "lineName": "Brenner Speed",
                        "speedestimate": "82.066872",
                        "subClass": "single-unit-truck",
                        "timestamp": "2022-11-10T15:37:20.977848Z",
                        "trackId": 1169162245
                },
                "eventSchema": "https://swarm-analytics.com/schema/event/",
                "node": {
                        "id": "1ce96134-cf3f-481b-b9ec-f25ed411502c",
                        "name": "202205B0023_Land Tirol — Brenner"
                },
                "stream": {
                        "id": "9d1b97a0-271d-43b1-be88-e09af6ba6c57",
                        "name": "202205B0023_Land Tirol — Brenner"
                },
                "version": "4.0"
        },
        {
                "crossingLineEvent": {
                        "class": "car",
                        "direction": "in",
                        "lineId": "cdd0ddda-d6d9-40b5-8190-538507719966",
                        "lineName": "Brenner Speed",
                        "speedestimate": "63.305981",
                        "timestamp": "2022-11-10T15:37:31.651439Z",
                        "trackId": 1170210821
                },
                "eventSchema": "https://swarm-analytics.com/schema/event/",
                "node": {
```

```
                        "id": "1ce96134-cf3f-481b-b9ec-f25ed411502c",
                        "name": "202205B0023_Land Tirol - Brenner"
                },
                "stream": {
                        "id": "9d1b97a0-271d-43b1-be88-e09af6ba6c57",
                        "name": "202205B0023_Land Tirol - Brenner"
                },
                "version": "4.0"
        }
```

4.5 Project Workflow Pipeline

In figure 4.8, the general data flow within the project is illustrated, abstractly showing the unification of audio and reference data, as well as the choice between different transformations for the pre-processing output, and the choice which model to train, which can then be compared.

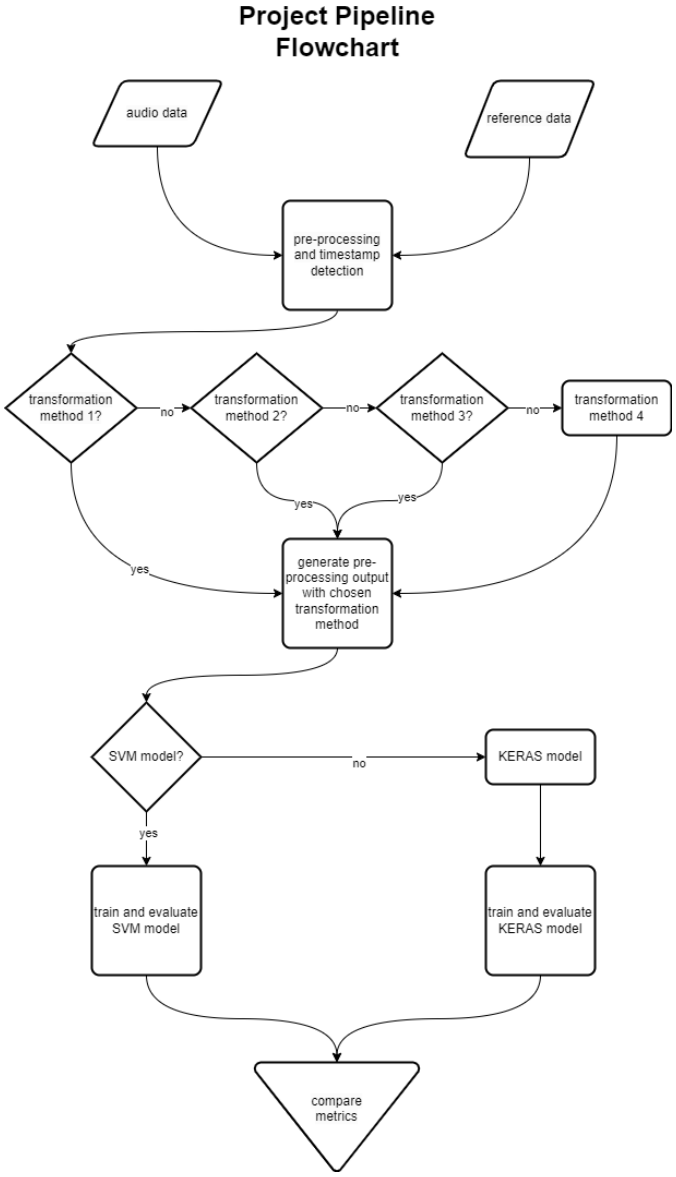

Figure 4.8: Flowchart of project workflow

4.6 Data Setup and Preprocessing

Project data is setup according to the guidelines presented by DrivenData in their article "Cookiecutter Data Science", as per [104], which specifies how to setup the folder structure of a data science project in a way that leaves raw data immutable, intermittent and result data mutable, and still pertaining a clean and easy to understand project structure, as shown in figure 4.9. This leads to a clean project setup, with clear indications where raw data can be found, both for audio data and reference JSON files, as well as intermediate data, results, and where the actual project code resides. Project data is managed by Data Version Control (DVC) [105], which is used to keep the data synchronized across machines used for development and ensures that the same experiments always use the same data and versions of contained files across machines, even if the data should have changed. DVC also allows the setup of pipelines, which can execute different parts of the code in order, or only certain stages of it. It also provides the possibility of setting up experiments by passing variable parameters to the program and tracking the output as well, thus making it very easy to generate metrics for different setups, and comparing them to find the optimal interplay of possible parameters.

Paragraph 4.6.1

Parameters to set are the maximum frequency for Melody or melody scale, a perceptual scale of pitches judged by listeners to be equal in distance from one another (MEL) to generate spectrograms, as well as the number of epochs to be used, and which spectrogram transformation to use. Epochs specifies how often the entire data set has to be worked through during training.

Spectrogram transformations are different ways to represent data as spectrogram images, leading to different impact on the ML model. Metrics contain information about model accuracy, F1 score, precision and recall, and also of the support the dataset provides for each class.

The rest of the project is set up via GitLab [106]. GitLab contains the project code, as well as the configuration files for DVC for specifying which data is held by DVC, and which by GitLab. Typically, all code is held by GitLab, and all data by DVC.

All input data is checked into dvc and synchronized across machines using the repository, as that ensures that an experiment will always have the correct data to use regardless of the machine it is run on, and has the added benefit of being able to handle large datasets which GitLab alone cannot upload.

```
├── LICENSE            <- Open-source license if one is chosen
├── Makefile           <- Makefile with convenience commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default mkdocs project; see www.mkdocs.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── pyproject.toml     <- Project configuration file with package metadata for
│                         {{ cookiecutter.module_name }} and configuration for tools like black
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.cfg          <- Configuration file for flake8
│
└── {{ cookiecutter.module_name }}   <- Source code for use in this project.
    │
    ├── __init__.py             <- Makes {{ cookiecutter.module_name }} a Python module
    │
    ├── config.py               <- Store useful variables and configuration
    │
    ├── dataset.py              <- Scripts to download or generate data
    │
    ├── features.py             <- Code to create features for modeling
    │
    ├── modeling
    │   ├── __init__.py
    │   ├── predict.py          <- Code to run model inference with trained models
    │   └── train.py            <- Code to train models
    │
    └── plots.py                <- Code to create visualizations
```

Figure 4.9: Cookiecutter Data Science project structure example [104]

4.7 Preprocessing of Audio Data

To ensure the quality and accuracy of the audio data, a meticulous preprocessing pipeline is implemented to ensure correct detection of passing-by events and extraction of audio snippets for detected timestamps, as well as noise data for intermittent periods, with data structured as presented in section 4.6. Initial steps involve the ex-

traction of timestamps for passing events by identifying local maxima in a smoothed Root Mean Square (RMS) curve, generated through denoising steps followed by Savitzky-Golay interpolation [107], [108]. Subsequently, two-second audio snippets are extracted for each timestamp, with one second before and one second after the timestamp. A two-second noise sample is collected between timestamps if those are far enough apart in time to ensure no part of an event is included in noise data, meaning that this sample only contains noise, and no part of an audio signal that should be detected, as shown in figure 4.10. If the time between two events is long enough that the extraction time of the noise snippet does not collide with the extraction time of an event snippet, it is extracted and labeled as noise. As is evident from figure 4.10, event extraction times can overlap, where both will be extracted and labeled if reference timestamps exist, but an audio snippet can contain the leading or edge of the following snippet, or the trailing edge of the snippet extracted before. This however happens rarely as most events are far enough apart in time, and the model should also learn to detect these overlaps.

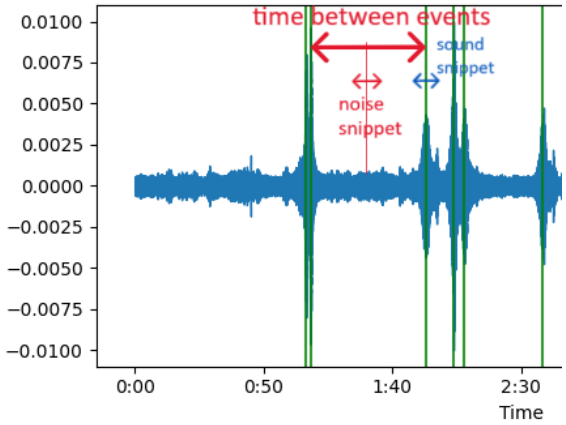

Figure 4.10: Snippets for extraction around timestamps (exaggerated)

The audio data undergoes transformation into spectrograms using various methods to test which of these transformations leads to the best results once the generated spectrograms are given to the ML model for training. Implemented transformations are the Morlet wavelet transformation, realized through the librosa package and its Constant Q Transform (CQT) functionality, a MEL spectrogram also processed with librosa [109], and two different spectrograms generated with scipy [110].

The scipy spectrogram is once generated directly from the input data snippet, and once with machine learning preprocessing of the image utilizing `sklearn.preprocessing import minmax_scale` [111], which drastically improves results for this simple transformation by transforming the input to an appropriate uniform scale. These preprocessing steps aim to enhance the discriminative features of the audio data for

effective machine learning classification.

After ascertaining that both librosa implementations produced better results combined with machine learning than their scipy counterparts over all examined frequency ranges from 1kHz to 10kHz, different frequencies were tested again for setting the MEL frequency scales, where a value of fmax=8000Hz provided the best results.

Results and the exact procedure of all those preprocessing steps are explained in more detail in chapter 5.2. The final results of preprocessing are audio and noise snippets with their corresponding spectrograms, where each filename contains both timestamp and label, where labels are *car, bus, truck, motorbike, noise*.

4.8 Machine Learning Model

In this chapter, the reasoning behind the final ML model choice, as well as the implementation of used models, will be explained. The workflow within the project along with questions towards validation, ethical considerations and limitations is also adressed.

4.8.1 Model Choice

As a machine learning framework for this project, the final choice fell to Keras [75] due to its versatility, simplicity of use and some other advantages, which are further discussed in section 3.5.2. One of the main tasks for the model was to be able to recognize spectrograms capturing complex temporal patterns and frequency variations, which are generated from audio data snippets. These, together with an efficient labeling system, allow the model to learn the available classes, and later perform predictions on either labeled data for verification, or predict types for unknown data, which is the main objective of this thesis. Other machine learning frameworks are also available for building such classification models, including TensorFlow [112], PyTorch [113], and scikit-learn (former sklearn) [114], where an SVC model was implemented for testing. While each of these frameworks has its merits, Keras stands out for several reasons, particularly in the context of simplicity, abstraction, and ease of use.

The Keras model was implemented in python by using the tensorflow library, the exact model setup can be reviewed in section 4.8.4. A Support Vector Machine (SVM) model as it's predecessor is described in more detail in section 4.8.3.

4.8.2 Model Input Data

For model training, a dataset consisting of spectrograms generated from audio data snippets, correlated with their respective labels, is passed to the model as input

data where this training data represents 80% of the available dataset. This input data is the same regardless of which model is trained, as that allows a comparison between the implemented Keras and SVM models. Both model types are trained as supervised models without clustering unlabeled input data themselves. Inference is then done on the remaining 20% of the dataset to verify that the models can also predict unknown data correctly. There are for now five different types which are labeled, as only supertypes of vehicles are considered and not subclasses.

Subclasses will be evaluated when there is more data available, but since they only form a fraction of a percent of available data, there is not enough audiodata available for them to train the model, hence it is trained on following five classes:

- Car . . . provided by reference data
- Truck . . . provided by reference data
- Bus . . . provided by reference data
- Motorbike . . . provided by reference data
- Noise . . . generated from audio data

where car, truck, bus and motorbike are classes provided by the camera machine learning model, passed as events in JSON files, and noise is generated by preprocessing by extracting data between events to have a noise baseline to evaluate against. The extraction of noise data is necessary to ensure that this class can also be trained and more clearly differentiated from vehicle data.

4.8.3 Implementation of Support Vector Machine (SVM)

The first model developed is based on the sklearn library and is built by training a Support Vector Machine (SVM) machine learning model for image classification. Input data is resized from the input shape generated by pre-processing to a standardized dimension of 64x64 pixels, ensuring consistency in input dimensions, where an array is then generated from the resulting resized image data to which labels are appended in a final input step.

Available data is then split into a training and a test dataset, where 80% of the data are used for training, and the remaining 20% for testing the model.

The model is then trained over a set amount of epochs, where for each epoch accuracy is used as the determining factor of the models performance. From all epochs, the model with the highest accuracy over all generations is then selected as the best model and returned as resulting machine learning model.

A Confusion matrix of the results of the trained and tested model is then generated to show how well the model performs for each class, as well as calculating additional metrics, such as precision, recall and F1 score and presenting them in a classification report.

The best results achieved during training and inference on the available data with this approach yielded an accuracy of 84% for discerning events with cars passing by from noise, which was good enough for a proof of concept, but not the desired quality of results, as an accuracy of at least 90% was a self-imposed goal for this thesis, with the further goal of getting past 95% accuracy if possible.

4.8.4 Implementation of Keras Model

The chosen Keras implementation is widely used state of the art, and this best practice was implemented as the Keras Sequential Model. To build this type of model, data in the form of image arrays is passed directly from the preprocessing algorithm, along with corresponding timestamps and labels, as this simplified the handover of data between pre-processing and model training and inference from having to save and load image files to simply passing arrays containing the same data, resulting in a faster pipeline.

Based on TensorFlows Keras Sequential API, a Convolutional Neural Network (CNN) model is constructed, which expects data in the form of MEL spectrograms as input, which it parses in its input layer based on the shape of the input, which in turn is determined by the spectrogram transformation used in preprocessing, as the results of all four transformations have similar, but not the same shape. The reason to implement and test four different transformation types as described in section 4.7 is to find the best method for this usecase out of these common transformation types, and to prove that the wavelet transformation was the best suited candidate.

This first layer is followed by a 2D convolutional layer [115] as second layer, which contains 32 filters and a (3,3) kernel, followed by a ReLU activation function as described in section 3.3.3.

As a third layer, MaxPooling2D is applied to reduce spatial dimensions, the result of which is flattened in a fourth layer. Layers five and six are dense layers, with the former using 128 units and ReLU activation, whereas the second uses the available number of classes, which is calculated from the available labels in input data, as well as softmax activation, which converts a vector of real numbers into a probability distribution, which can be described as such:

$$\text{For each element } i \text{ in the output vector:} \tag{4.1}$$

- z is the input vector of real numbers.

- i represents the index of the element in the output vector.
- K is the total number of classes.
- $\text{softmax}(z)_i$ denotes the i-th element of the softmax output vector.

$$\text{The softmax function is given by:} \quad \text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4.2}$$

The model is then compiled using the Adam optimizer and categorical crossentropy loss, with accuracy as the evaluation metric. Using the Adam optimizer has the advantages that it converges faster than traditional gradient descent-based optimizers, having an adaptive learning rate as well as being a robust to hyperparameter choices while also having low memory requirements [116].

The compiled model is then saved for training and testing, where it is first fit to a training dataset over a predetermined number of epochs, and once the model is fully trained, it is tested on a test dataset which is either the remaining 20% of a train-test split, or a dataset which is new to the model.The train-test-split approach is usually used for training the model and testing inference, but with the possibility of also testing the model on another dataset it has not been trained on, which is useful to test if the model overfit on the data.

A summary of the resulting model, as well as the trained epochs with their corresponding accuracy, is then printed to the console for inspection along with an overview of the dataset, where in this example listing 4.2 a test dataset of 27 samples is used. Conv2D describes the convolutional layer, MaxPooling2D the reduction of spatial dimensions, Flatten the flattening process and both Dense layers providing activation function in the former and a probability distribution in the latter, as described in the beginning of section 4.8.4. The total number of parameters as well as trainable parameters along with their data size is then printed, followed by the training Epochs with their respective runtimes per step and both loss and accuracy values.

The final accuracy and F1 score are then reported, followed by a confusion matrix denoting which type was detected how well, along with the available number of supporting data points.

Listing 4.2: Model Summary Output

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 82, 85, 32)        320
```

```
max_pooling2d_1 (MaxPoolin   (None, 41, 42, 32)        0
g2D)

flatten_1 (Flatten)          (None, 55104)             0

dense_2 (Dense)              (None, 128)               7053440

dense_3 (Dense)              (None, 3)                 387

=================================================================
Total params: 7054147 (26.91 MB)
Trainable params: 7054147 (26.91 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
4/4 [==============================] - 2s 62ms/step - loss: 119.8548 - accuracy: 0.4615
Epoch 2/10
4/4 [==============================] - 0s 63ms/step - loss: 72.5732 - accuracy: 0.5481
Epoch 3/10
4/4 [==============================] - 0s 57ms/step - loss: 21.0999 - accuracy: 0.7981
Epoch 4/10
4/4 [==============================] - 0s 54ms/step - loss: 25.5614 - accuracy: 0.6154
Epoch 5/10
4/4 [==============================] - 0s 62ms/step - loss: 20.4485 - accuracy: 0.7308
Epoch 6/10
4/4 [==============================] - 0s 58ms/step - loss: 9.5498 - accuracy: 0.8654
Epoch 7/10
4/4 [==============================] - 0s 57ms/step - loss: 6.2013 - accuracy: 0.9712
Epoch 8/10
4/4 [==============================] - 0s 61ms/step - loss: 5.5833 - accuracy: 0.9135
Epoch 9/10
4/4 [==============================] - 0s 60ms/step - loss: 2.8641 - accuracy: 0.9712
Epoch 10/10
4/4 [==============================] - 0s 55ms/step - loss: 1.6281 - accuracy: 0.9712
1/1 [==============================] - 0s 189ms/step

Accuracy: 0.8889
F1 Score: 0.8556
Classification Report:
              precision    recall  f1-score   support
         car       0.82      1.00      0.90         9
       noise       0.94      0.94      0.94        16
       truck       0.00      0.00      0.00         2
    accuracy                           0.89        27
   macro avg       0.59      0.65      0.61        27
weighted avg       0.83      0.89      0.86        27
Confusion Matrix:
 [[ 9  0  0]
 [ 1 15  0]
 [ 1  1  0]]
```

These results are from a relatively small dataset of only 27 unique spectrograms, including noise data, and show that the model can already achieve 89% accuracy with limited data. Expanding the dataset to at least a support of 300 labeled events leads to significantly improved results with resulting accuracy of 96%.

4.9 Threats to Validity

As the developed approach has to rely on some limitations which concern the available training and testing data as well as ethical considerations, it is important to acknowledge that and take possible concerns into account, which will be discussed

in this section.

4.9.1 Validation

Model predictions are currently validated against the output of the video recognition, and this available reference data is treated as golden dataset against which the results of this thesis are evaluated, as there is currently no information about the inner workings of that system, which is a known but accepted caveat, as the data is used by Asfinag, the Austrian Motorway and Expressway Financing Joint-Stock Company and therefore deemed accurate. A setup containing a sensorbox and a camera for manual labeling of raw video data and corresponding audio data is currently in planning, but an exact setup date is not yet known.

4.9.2 Ethical Considerations

Ethical considerations play a crucial role in research design. To safeguard privacy and anonymity, the sensorbox is deployed in a location with minimal foot traffic, like in this case a country road and next to a highway; should this change in the future, this point will have to be re-evaluated. The audio recognition algorithm also does not persistently store any data, mitigating potential privacy concerns.

4.9.3 Limitations

While the methodology employed in this thesis is robust, it is essential to acknowledge certain limitations as mentioned before. The reliance on labeled reference data from a video recognition model introduces an element of trust in the accuracy of this data, where it has to be accepted that the available reference data is regarded as true, and build the model from there, as the only data available from the reference system are the timestamps with associated type and speed, without access to the raw video data. Additionally, the current setups restriction to a simple two-way road presents limitations in assessing the models performance in complex traffic scenarios, which will be part of future developments, as the outcome of this thesis is for now to be treated as a proof of concept.

4.9.4 ML Result Data Analysis

The analysis of data involves extracting valuable insights from the ML training and inference experiments with different transformations and frequency parameters. The transformed audio data is systematically analyzed, and metrics such as accuracy rates, confusion matrices, and other relevant measures are computed. This step is crucial for drawing meaningful conclusions from the results and to garner information as to what parameters and transformation steps are most benefitial to model performance, as well as finding out which tested model performs best. All

transformation modes were tested in the frequency range from 1kHz to 10kHz, in 1kHz steps, with the resulting pre-processing output presented to the SVM model as well as the Keras model, resulting in 80 different parameter pairings, from which the wavelet transformation at 8kHz and the Keras model for subsequent evaluation performed the best. Experiments can be replicated with the original data, as only the best results have been saved as models for further evaluation. A known caveat is the scarcity of data available for this thesis, as there are a few thousand events to be classified, but most of those events are cars, producing a bias towards that class, which can be remedied in future work by recording a lot more data and extracting as many events of other classes as possible, so those also are represented with at least a few hundred examples each.

4.10 Data workflow for this thesis

How data is treated within the project structure can easily be visualized with a simple flowchart shown in 4.11, starting from data aquisition and ending with the output metrics which lead to decisions about which model or parameters to use. Timestamps are matched to ensure that a labeled audio snippet has the correct label from its corresponding reference timestamp, which can be verified by sifting the available data. Data transformation is implemented as described in section 4.7, and model training and testing are repeated until accuracy converges to a stable value. According to the output metrics, the best model is then chosen for further evaluation, where the defining metric is accuracy, as it is essentially a binary model. All experiments can be repeated either by starting a DVC pipeline, or running a python script by hand.

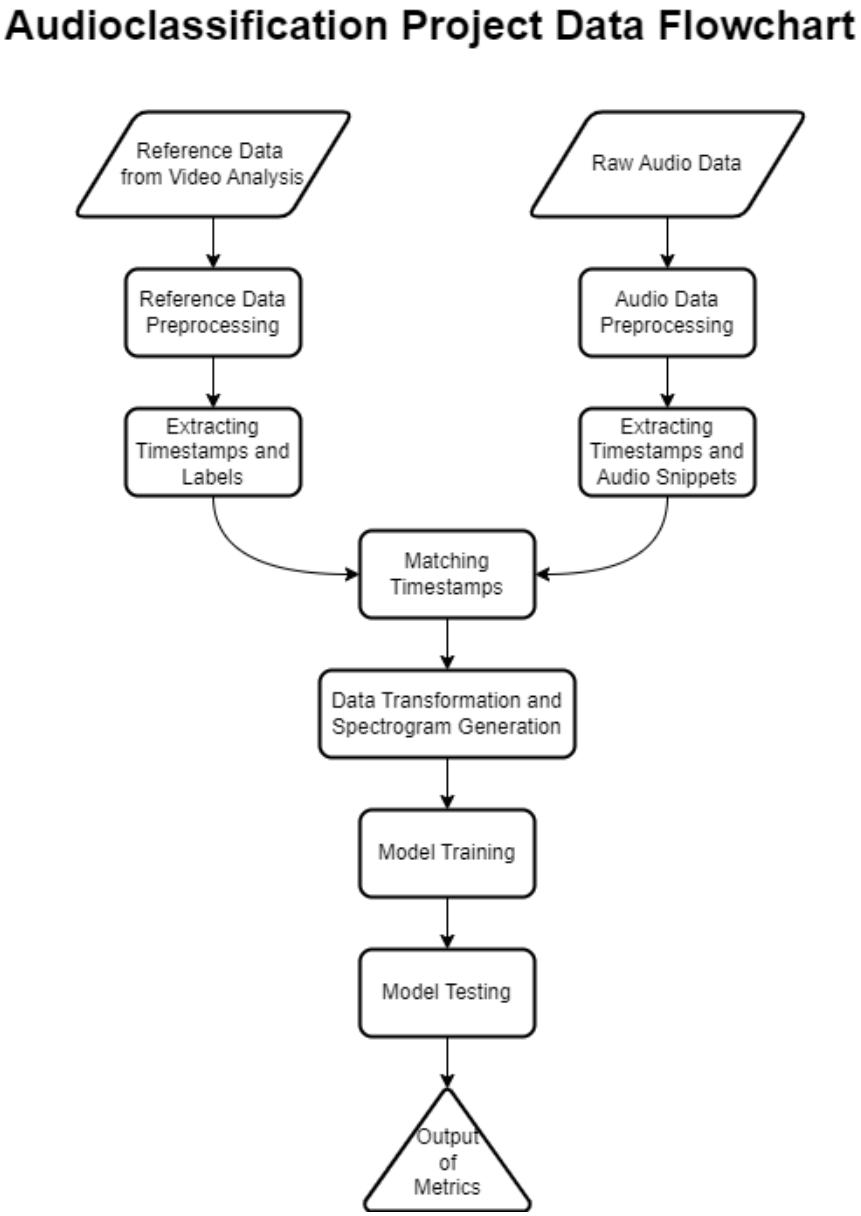

Figure 4.11: Flowchart of the Project Data

4.11 Synopsis of Methodology

In summary, the methodology encompasses a holistic approach to audio recognition in traffic monitoring, starting with utilizing provided equipment and software, data collection and preprocessing, machine learning model implementation as well as experiment design as described in secti 4.9.4, validation and data analysis, as well as

considering ethics and known or possible limitations of the project. This comprehensive methodology is designed to address the research questions as stated in section 7.1 and contribute insights to the field of traffic control how audio classification could support or even replace video classification, and how the usage of audio classification could be beneficial for traffic monitoring by adding more data to classification and making it possible to reliably detect events even under weather conditions where a camera cannot operate efficiently anymore.

CHAPTER 5

Implementation of Methodology

In this chapter, the implementation of methodology explained up until this point will be elaborated, as well as providing exemplary intermediate result plots which are used by the algorithm to extract the correct audio segments,and how these segments are transformed for use by the ML model, leading to classification results.

5.1 Data Collection and Preprocessing

How reference and raw data are collected and treated are an essential part of this thesis and the workflow for obtaining good results from machine learning, therefore the gathering of said data is described in detail. The most important part is the feature extraction of available audio data, which ensures that well prepared and verified data can be presented to the model for training, and is explained in detail in section 5.2, which covers all the steps between loading raw audio files and reference data into the pipeline and producing spectrograms of audio snippets at event timestamps to be passed to the model.

5.1.1 Data Collection

For training the model, we use audio data gathered by a sensorbox mounted on "B182 Brennerstraße" at coordinates `47°01’37.4"N 11°30’03.4"E` where the camera and sensor box setup is shown in figures 4.6 and 4.7, described further in the following sections to give a detailed overview of how both the camera and the sensor box are setup in section 5.1.2, and how audio data is collected in section 5.1.3, as at this location there is also a traffic camera with an underlying video object tracking machine learning model present, as elaborated in section 5.1.4. Data from both the camera setup and the sensor box is gathered, where the data from video recognition

of the camera images is refered to and used as reference data which is available in JSON format, and the raw audio data gathered by the sensor box is available in MP3 format.

5.1.2 Camera and sensorbox location and setup

In this section, images of the setup of the recording camera as well as of the mounted sensor box are presented to give a better understanding of the overall setup of the system, and where the data is gathered from.

Figure 5.1: Sensor box setup at B182 Brennerstraße

Figure 5.2: Camera and sensorbox setup at B182 Brennerstraße

It has to be noted that this is not the same camera that records reference data, the camera where reference data is from can be seen in 5.3 below the cameras on top

Figure 5.3: Camera setup at B182 Brennerstraße

Figure 5.4: Sensor box setup at B182 Brennerstraße

Figure 5.5: Sensor box closeup of setup at B182 Brennerstraße

Figure 5.6: Mounting of camera and sensorbox

that point in both directions, the camera relevant to this thesis is the one pointing left in the image.

Figure 5.7: Camera view at B182 Brennerstraße provided by Land Tirol [117]

5.1.3 Audio Data Collection

Acoustic training data is collected by a sensorbox with denotion srb01030 at a location at coordinates `47°01’37.4"N 11°30’03.4"E` on "B182 Brennerstraße", where there are two lanes present with a single sidelane on the right hand side in relation to driving in the direction of the camera setups alignment direction, as shown in figure 4.6 as well as in figures 5.1 to 5.7. The audio data is recorded by the sensorbox utilising two MEMS microphones mounted approximately 20cm from each other, as shown in figures 4.2 to 4.5 as dual channel audio. For detecting passing by events, only one channel is needed, so a mono signal is desired and can be extracted by either layering both channels over each other, or just choosing one of them for further evaluation. After all pre-processing steps to extract and verify the event timestamps, the audio snippet from which the spectrogram for ML model training or inference is extracted from the original raw audio data at that specific timestamp, so that all available data is used for generating the model input. For the sake of using all avaliable data points, both channels are overlayed on each other and passed on as mono audio by utilising `librosa’s librosa.to_mono(y)` functionality, which convert an n-channel audio input to mono by averaging samples across channels. This is even benefitial to the fidelity of the resulting mono signal because more data points are present for each timestep. Due to the setup of the sensorbox, it always provides two channels, without the option of just recording one, so the data can either be converted to single channel, or only one channel can be selected from the resulting data array, but as an input, the full dataset is always present. The sensorbox

can be remotely programmed to start persistently recording data to an SD card for a specified period, which can then be retrieved by downloading the generated MP3 files or manually removing the SD card from the sensorbox by visiting the deployment site and opening the box after taking it offline.

5.1.4 Video Data Collection

The results of the ML evaluation of video data corresponding to the audio data described in section 5.1.3, containing passing-by timestamps and vehicle type, are collected via a REST API and downloaded as JSON files, where the starting time and length of the desired timeframe have to be matched to the starting time and length of the available audio data.

The JSON files contains parameters like vehicle type, estimated speed, direction of movement, timestamp of the event, and if available, also the subtype of vehicle, and also the recording stations designation, which in this case is `Brenner Speed`. Of this data, for this thesis only the timestamp as well as the vehicle type are relevant, while the direction of movement, subtype and estimated speed are not evaluated. Data in JSON has the form of the following example in listing 5.1, and only timestamp and vehicle type are extracted from these datasets.

Listing 5.1: Exemplary JSON data

```
{
    "crossingLineEvent": {
        "class": "car",
        "direction": "in",
        "lineId": "cdd0ddda-d6d9-40b5-8190-538507719966",
        "lineName": "Brenner Speed",
        "speedestimate": "84.639496",
        "timestamp": "2022-12-09T13:00:51.204668Z",
        "trackId": 3278897164
    },
    "eventSchema": "https://swarm-analytics.com/schema/event/",
    "node": {
        "id": "1ce96134-cf3f-481b-b9ec-f25ed411502c",
        "name": "202205B0023_Land Tirol - Brenner"
    },
    "stream": {
        "id": "9d1b97a0-271d-43b1-be88-e09af6ba6c57",
        "name": "202205B0023_Land Tirol - Brenner"
    },
    "version": "4.0"
}
```

A known caveat is that the process gathering reference data from video recognition is to be regarded as a black box where no insight into how events are classified exactly is possible, and the reference data received with the assurance of it being correct leads to the reference data being regarded as correct and as baseline for evaluationg the audio recognition models with the knowledge that the baseline might not be 100% correct, but still being the only reference available. This is due to the

fact that reference data is only available as said JSON files, and no access to the raw video material for manual evaluation is possible.

Reference data from provided JSON files functions as the baseline for data verification, and events recorded there are used to collate timestamps extracted from audio files with labels for vehicle types. This reference data is saved as a tuple of timestamp and vehicle type, while the direction of travel and the estimated speed from video analysis are not considered.Timestamps in reference data and audio data drift by a few milliseconds, as the camera points at the street at an angle, while the microphones record data facing the road at a 90° angle, as seen in figure 4.6 and figures 5.1 to 5.7. This results in a slightly different timestamp from the camera by a few milliseconds, depending on the direction of travel of the observed vehicle, leading to a necessary adjustment between audio and video timestamps by matching timestamps by finding the closest timestamp in reference data and then comparing if the resulting number of timestamps matches the reference data. A very relevant downside of video detection has to be noted, as the video camera can be obscured by fog, snow, heavy rain, otherwise poor visibility or similar phenomena, whereas the microphones do not suffer from those problems. This leads to testing showing that sometimes audio preprocessing picks up signals that video recognition misses, which show up as differences in data analysis and upon further investigation sound like cars passing, therefore it can be assumed that they are indeed valid registered events, just not present in reference data. As there is no access to raw video data for reference, further verification than that is not possible, so signals picked up by audio processing but not present in video data are discarded from the dataset, as their label is unknown and cannot be found out. This data is still present and can be used to let the model predict which type it could be, but as there is no reference data for it and therefore no label to go by, the results cannot be verified.

5.2 Feature Extraction and audio pre-processing

This section describe the most important workflow in this thesis before Machine Learning in detail, which is the processing of input audio data to produce spectrograms which can be used to train the ML model

Audio files are imported for preprocessing via Librosa [109], which is a python library providing a large spectrum of audio processing algorithms. Since data is recorded by two microphones, providing stereo data, the first step in preprocessing is getting the absolute Root Mean Square (RMS) throughout the signal to be able to build an envelope for the existing audio data and subsequent peak detection, as only one channel is required for further processing, but this way, no data is lost. The resulting data as well as the log power spectrogram are shown in figure 5.8

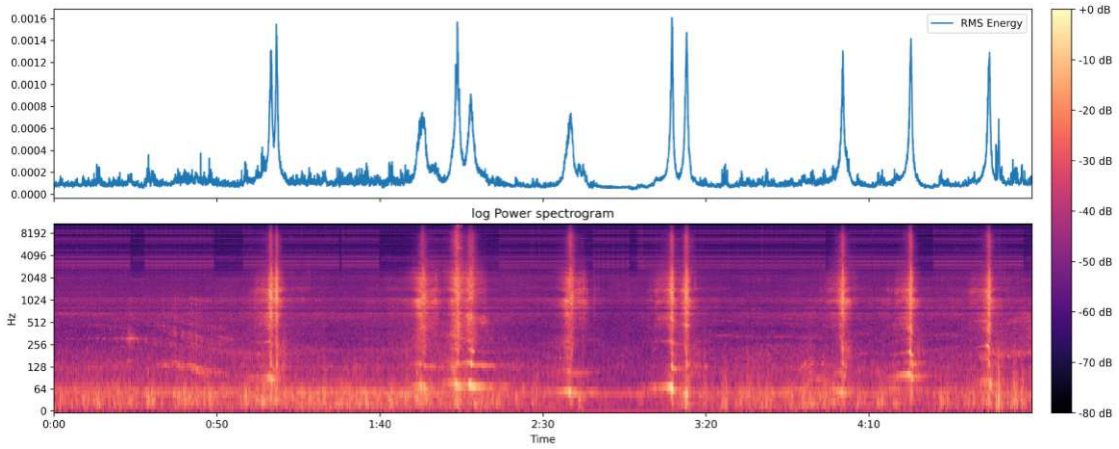

Figure 5.8: Input Signal RMS and spectrogram

The resulting RMS audio data is then denoised by utilizing the python noisereduce library [118] which utilises spectral gating as its method, which continously updates the noise threshold throughout the file, and filters out most noise. Below is a short explanation from the developer, and the result can be examined in figure 5.9.

> It works by computing a spectrogram of a signal (and optionally a noise signal) and estimating a noise threshold (or gate) for each frequency band of that signal/noise. That threshold is used to compute a mask, which gates noise below the frequency-varying threshold [118].

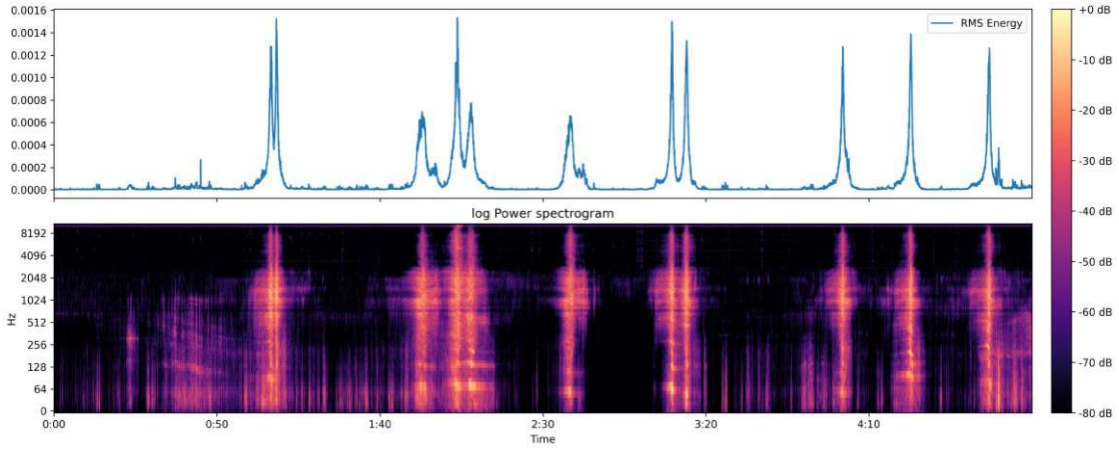

Figure 5.9: Input Signal RMS and spectrogram after noise reduction

Following denoising, the audio signal is smoothed using a Savitzky-Golay filter [119]. This particular type of low-pass filter is well-suited for data smoothing [108], [107].

Savitzky-Golay interpolation is a digital filter technique used to smooth data by fitting successive subsets of adjacent data points with a low-degree polynomial using the method of least squares. It preserves the features of the data, such as peak height and width, making it particularly effective for data smoothing and noise reduction.

This results in a smooth function from which various parameters can be more easily extracted, such as derivatives for identifying peaks or using peak detection by locating local maxima, which is shown on following figure 5.10.

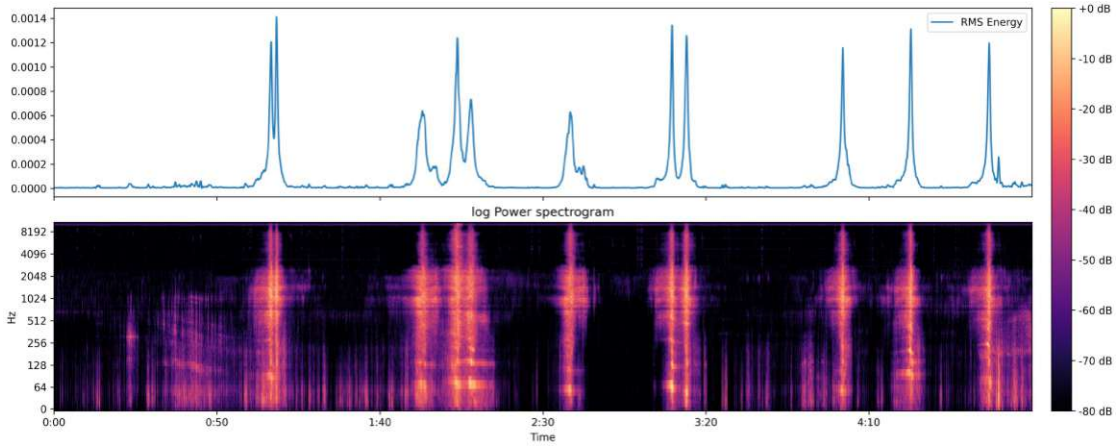

Figure 5.10: denoised and smoothed input signal RMS and spectrogram

To simplify the process of finding passing-by event timestamps, a threshold can be set to discard any part of the signal below a certain amplitude percentage on the Decibel (dB) scale to cut off noise at the bottom. This results in a signal containing only the peaks above the threshold, making it easier to identify the timestamps of passing-by events. After setting such a threshold and removing the noise floor, a signal with distinct peaks remains, as shown in figure 5.11

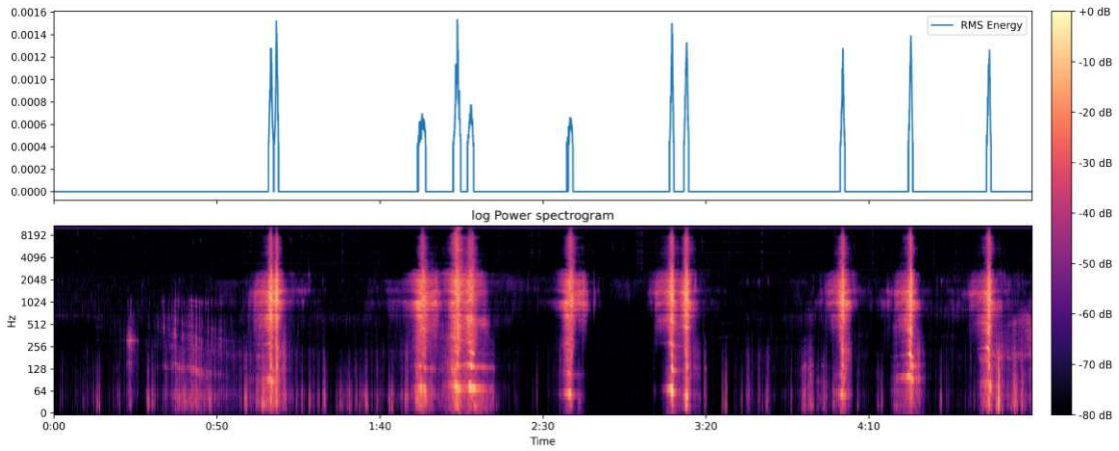

Figure 5.11: Input Signal RMS and spectrogram peaks

As the signal from figure 5.11 still contains some fluctuations at the peak of each timestamp location, this would lead to the detection of numerous timestamps at one location with the available audio data resolution of 44kHz. Therefore, it is necessary to smooth the signal a second time with a Savitzky-Golay filter [119] to provide a smooth signal where unique peaks can be detected, which can be seen in figure 5.12

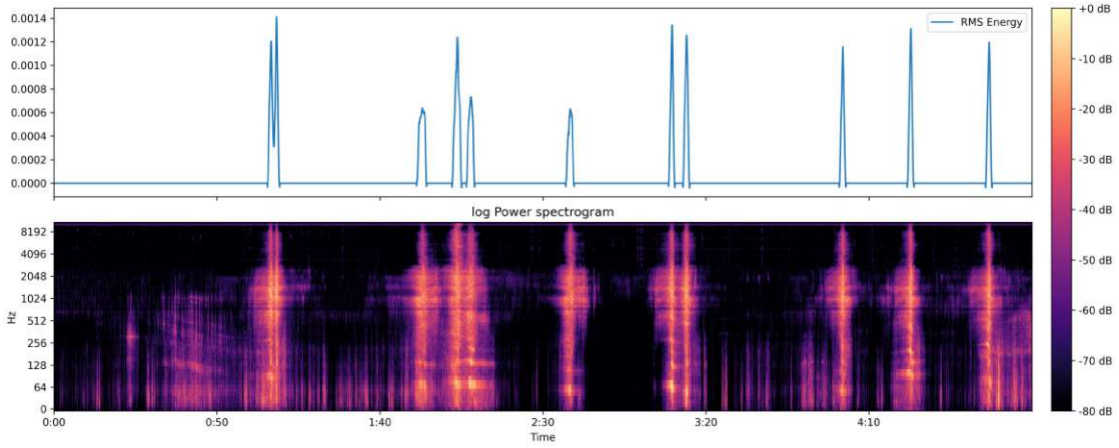

Figure 5.12: Input Signal RMS and spectrogram peaks smoothed

This now smooth signal containing distinct peaks then undergoes a peak detection algorithm which looks for the gradient next to a point, and when a point has a strictly rising gradient on its left, therefore before in time, and a strictly falling gradient on afterwards, or simply put is the highest point relative to neighbouring points, it is considered a peak and the corresponding timestamp is extracted.

These timestamps can then be overlaid with the original Root Mean Square (RMS) input data and the corresponding spectrogram, showing that each peak from the

input RMS signal is detected correctly, to be examined in figure 5.13.

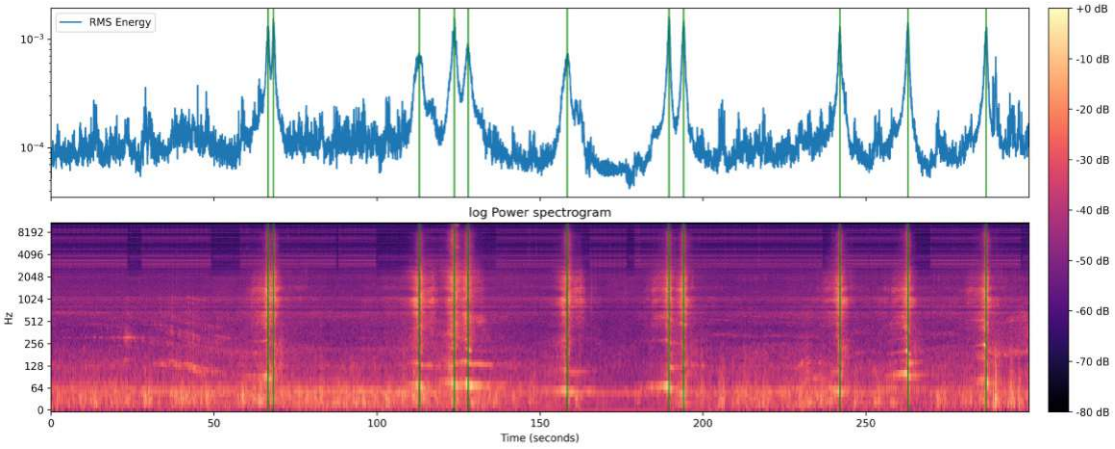

Figure 5.13: timestamp overlay for input Signal RMS and spectrogram

The same timestamps can also be shown as overlay to the original and unedited two-channel audio stream which represents the raw audio input data in figure 5.16. From this data, two seconds of audio are extracted as snippets for each timestamp with one second before and one second after each respective timestamp, resulting in as many audio snippets as timestamps are detected in a signal.

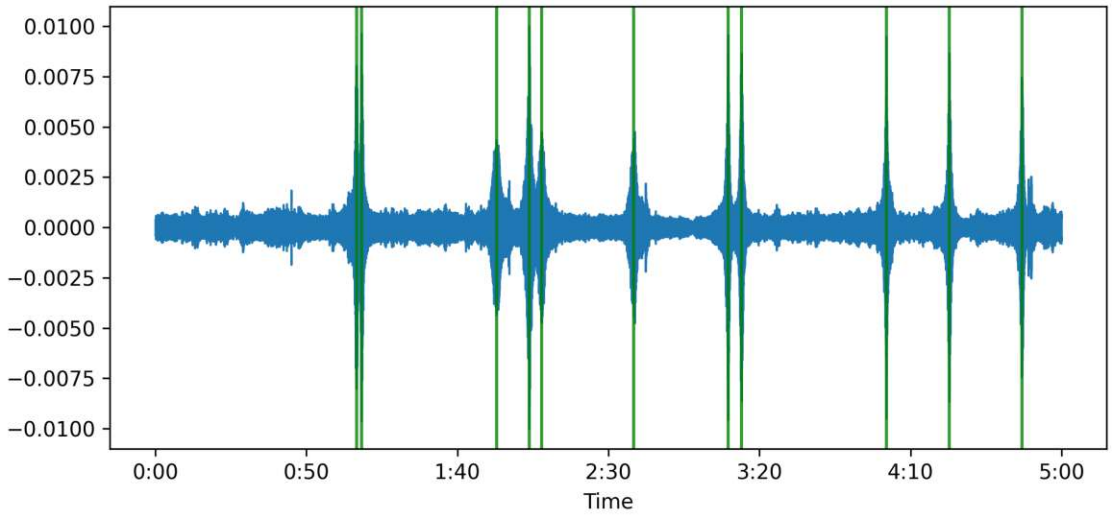

Figure 5.14: Timestamp overlay with two-channel input signal

The timestamps are then verified against reference data, and if there is a matching reference timestamp, the appropriate vehicle label will be attached to that snippet and it will be further processed to a spectrogram which is then used for ML model training and inference, examples shown in 5.17.

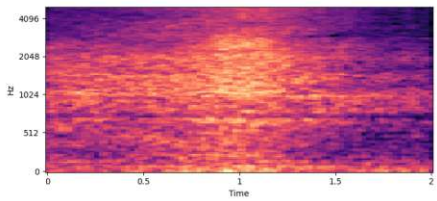

Figure 5.15: MEL spectrogram of a passing car

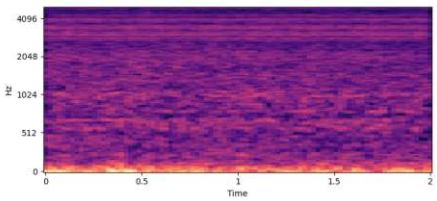

Figure 5.16: MEL spectrogram of noise

Figure 5.17: Exemplary car and noise MEL spectrograms

Two seconds proved to be a window where for the time a vehicle passes by at the testing location, all necessary data is recorded, as the whole signal of a passing vehicle is recorded. This is based on the average speed of vehicles there lying between 50km/h and 70km/h most of the time. With a buffer of again two seconds apart from a detected timestamp, noise snippets are also extracted between timestamps, thus if two timestamps lie at least eight seconds apart, a two second noise snippet will be extracted in the middle of them, as shown in 4.10. The same extraction applies between the start and end of the signal towards the nearest timestamp. This results in a dataset that contains timestamps with their correlating audio snippet data for detected passing-by events that have to be verified as well as for noise data extracted from original audio data between timestamps which are far enough apart.

This self-built approach to preprocessing was chosen over exploring and using prebuilt machine learning preprocessing steps for the sake of having control over the whole workflow, as well as keeping the algorithm easy to understand and modifiable and building the whole pipeline from scratch, which also leaves the computational modifiable by simplifying the algorithm.

5.3 Transformation Methods

Four different spectrogram transformation methods where implemented and tested utilizing scipy [114] and librosa [109], as well as TensorFlow [112], which provides a machine learning preprocessing step to make input data uniform and bring it to a scale better suited for model input.

5.3.1 Morlet Wavelet Transformation and Implementation in librosa cqt

The morlet wavelet transformation is a mathematical technique employed in signal processing and time-frequency analysis. It is particularly useful for analyzing the spectral content of signals, especially in cases where both time and frequency information are crucial [120]. The morlet wavelet is a complex exponential wave modulated by a Gaussian envelope, providing a balance between time and frequency localization [121].

Librosa's Constant-Q Transform (CQT) functionality emulates aspects of the Morlet wavelet transformation in the context of audio signal processing [109]. The CQT is specifically designed for analyzing audio signals at different frequencies with a constant ratio between them, mimicking the logarithmic scale of human auditory perception. It applies a series of Morlet wavelets at different center frequencies and logarithmically spaced bandwidths, capturing the signal's spectral content across both time and frequency dimensions [109]. The resulting CQT representation is beneficial for tasks such as music analysis, where understanding the harmonic structure and temporal evolution of audio signals is essential. [109], [122]

5.3.2 Librosa MEL spectrogram

A MEL spectrogram is a representation of the spectral content of an audio signal that emphasizes the perceptually relevant features for human hearing. It is derived from the MEL scale, which is a perceptual scale of pitches that approximates the human ear's response to different frequencies [123]. In a MEL spectrogram, the frequency axis is divided into MEL-frequency bands, and the energy within each band is computed using a series of overlapping windows along the signal [124].

Librosa's functionality for generating MEL spectrograms allows for the extraction of features that are more aligned with human auditory perception [109]. The process involves dividing the audio signal into short overlapping frames, applying a window function to each frame, and then computing the Discrete Fourier Transform (DFT) to obtain the magnitude spectrum [109], [124]. Subsequently, a filterbank based on the MEL scale is applied to these spectra, summing the energy in each MEL-frequency band. The result is a MEL spectrogram, a two-dimensional representation where time

is plotted on the horizontal axis, frequency on the vertical axis, and color represents the intensity of the signal's energy.

MEL spectrograms find extensive use in speech processing, music analysis, and various audio-related tasks, providing a more perceptually relevant representation of the signal's frequency content compared to a traditional spectrogram [109].

Examples are shown in figure 5.17.

5.3.3 Scipy Spectrogram

This functionality was implemented in a simple manner as well as with machine learning preprocessing to make the data more uniform.

Simple Approach to Scipy Spectrogram

Similar to the librosa approach described in section 5.3.2, the scipy library in Python provides a function called `scipy.signal.spectrogram` for computing spectrograms [110]. Spectrograms are a way to visualize the frequency content of a signal over time.

The output of this spectrogram transform is in turn used as input for the machine learning model, but due to bad preliminary results with average accuracy over multiple experiments below 60%, this approach was not further evaluated.

Scipy Spectrogram with Machine Learning Preprocessing

To improve on the method described in section 5.3.3, the preprocessing function `minmax_scale` from scikit is utilized to produce uniform data for the machine learning model [114].

> This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one [114].

`sklearn.preprocessing.minmax_scale` is a function provided by the scikit-learn library for feature scaling, specifically performing min-max scaling on numerical data. Feature scaling is a common preprocessing step in machine learning to standardize or normalize the range of independent variables or features. [114] The input data is thus re-transformed to a more uniform state, which benefits the machine learning model by using the full input range possible.

5.3.4 Transformation Methods Synopsis

Four different transformation methods were implemented and tested. From those, the simple approach with scipy described in section 5.3.3 was discarded after providing

unsatisfactory results. Its counterpart with ML pre-processing proved to provide better results, but relied on a pre-processing function from an external library. The best results could be achieved by utilising the morlet wavelet transformation with librosas CQT functionality.

5.4 Data Labeling

The crucial task of correctly labeling input data for the model provides the basis for accurate results and good model performance, and a short overview will be given here. With labeled data, extracted audio snippets and their corresponding timestamps and labels can then be passed to the machine learning model.

5.4.1 Overview of Data Labeling

To ensure correct results and reliability of the predictions made by the machine learning model, input data has to be correctly labeled, as this is crucial for the reliability of the model's predictions. The basis for this is reference data acquired from video recognition, which is automatically synchronized with results from audio pre-processing. Resulting labeled data is then analyzed for plausibility and to make sure that the data used for verifying the model is indeed correct in respect to the reference data. Data Labeling has to be automatized to be able to handle large input datasets, where manual labeling would be very time consuming and prone to error. The output of this processing step are labeled data snippets, both as audio samples and as corresponding spectrograms, which are used to train and test the model.

5.4.2 In-depth Data Labeling

Labeling starts with acquiring input data from the provided JSON files, as well as extracting event timestamps from the corresponding mp3 audio data. For both files, the exact starting time of the recording is known. Reference data in JSON form has a timestamp from which video recognition started, and contains crossing-line events for every detected vehicle that travelled across a defined boundary line, as shown in figure 4.6. Audio data has to be pre-processed to make timestamps of detected passing vehicles available, which is explained in detail in section 5.2.

Each of the timestamps detected during pre-processing has to be labeled to provide useful input for training the machine learning model. When a timestamp extracted from audio data coincides with an event timestamp from reference data, the associated label from reference data is also attached to the audio event, for which a two-second audio snippet it also extracted from raw data, which is then further processed into a spectrogram. These event spectrograms together with their associated

label then present the data used for training and evaluating the machine learning model.

To verify the correctness of appended labels, data can either be reviewed in a random sample survey, where an extracted and labeled audio sample can be listened to and then decided if it could be the sound of the labeled vehicle, which is a sufficiently exact proof for this usecase.

5.4.3 Synchronization between audio and json data

Since the exact starting time of each recording is known and available in a form of a timestamp such as `2022-11-10T14-11-00.000000`, as well as the duration of the recording, the results from video recognition can be accessed through a REST API provided by our partner company Bernard Group GmbH, to get JSON files containing timestamps and labels for events detected by the camera ML system for the same timeframe. Timestamps extracted from the audio files are then compared to the ones from the correlating JSON data, and closest matches accepted to pair the available vehicle type label to a timestamp as well as the corresponding audio data.

Since subclasses are not available for every detected event, "class" is the only extracted label, as it provides enough distinction for training and testing the model. The most common classes found in JSON files are listed as follows, with the noise class added from audio data during pre-processing.

- **Labels from reference data**
  - Car
  - Truck
  - Bus
  - Motorbike
- **Label generated during pre-processing**
  - Noise

With this labeling procedure, where extraction and matching of timestamps was described in section 5.2, data for training and inference can then be passed to the ML model. Spectrograms for each timestamp as shown in figure 5.17 are then combined with their timestamps as well as the corresponding raw audio snippet. The resulting dataset then contains all necessary data for model training and testing, and labeled spectrograms can then be passed to the model. Noise spectrograms are saved in the same dataset, also with their label as "noise", but without a timestamp as noise is

always extracted between events with enough buffer to the previous and next event, so saving a timestamp for extracted noise is pointless as it doesn't have reference data to be compared to.

This labeled data then functions as input data for the model and can either be passed as a train-test split, or a number of datasets can be used as training data, with a separate dataset for validation. The option of providing a separate dataset for validation instead of a train-test split of the same dataset provides a failsafe against overfitting the model, which could happen if the training and test data are too similar from a train-test split.

5.5 Implementation as a pipeline

The comprehensive methodology delineated in Sections 4.4 through 5.4.3 has been meticulously implemented as a cohesive Python script. Within this script, the diverse array of functions developed throughout the preceding sections are seamlessly integrated into a unified pipeline. Each function encapsulates specific aspects of the methodology, effectively abstracting the complex processes involved in data collection, preprocessing, and synchronization. This modular approach not only enhances code maintainability but also ensures that the implementation remains faithful to the theoretical foundations previously established, allowing for a robust and replicable execution of the proposed workflow.

5.5.1 Audio Processing and Spectrogram Generation

This section lists the crucial processes of audio-preprocessing and their associated functions, which lay the groundwork for subsequent analysis. It provides an overview of the implemented functions in order of their execution in the pipeline for a better understanding of the pre-processing workflow.

1. **Folder Creation and Data Restructuring**: Creates a structured folder hierarchy for organizing processed audio and spectrogram data. It also includes functions for copying raw audio files and external tags from JSON files into the appropriate folders, keeping the structure described in figure 4.9.
2. **Loading Audio Files**: Utilizes `librosa` to load audio files and obtain their sample rates.
3. **Noise Reduction**: Applies noise reduction using the `noisereduce` library to clean the audio data.
4. **Spectrogram Separation**: Separates the spectrogram into magnitude and phase components and computes the Root Mean Square (RMS) energy for each frame.

5. **Savitzky-Golay Interpolation**: Smoothes the observed RMS signal utilising Savitzky-Golay interpolation [107] to provide a smooth function where peaks can be detected. This is repeated after thesholding to ensure peaks are unique.
6. **Threshold Filtering**: Sets values below a specified threshold to zero in the array containing the smoothed RMS data to remove left over noise.
7. **Local Maxima Detection**: Identifies local maxima in the RMS signal to find peaks that may indicate significant events in the audio.
8. **JSON Processing**: Processes JSON files to extract event timestamps and associated data.
9. **Timestamp Matching**: Matches and compares timestamps extracted from audio files with those from JSON files to identify corresponding events.
10. **Audio Segment Extraction**: Extracts segments of audio data around detected peaks.
11. **Spectrogram Generation**: Generates spectrogram images from audio files in different modes
12. **Plotting Functions**: Includes several plotting functions using `matplotlib` to visualize RMS energy, spectrograms, and audio signals with timestamps.
13. **Command Line Interface**: Implements an argument parser to facilitate running the script from the command line, specifying input directories for audio files, JSON files, and the output directory, as well as additional parameters like maximum frequency (`fmax`) and transformation mode for spectrogram generation.
14. **Main Function**: Executes the data restructuring process by integrating the aforementioned functions, processing each audio file, and generating the corresponding spectrograms and noise-reduced audio segments.

5.5.2 Model Training and Evaluation

In this section, an overview of the code developed to train and evaluate the Machine Learning model is given, as well as mentioning the data split. All data used for machine learning was split in an 80/20 split as described in section 3.3.4 with the option to test for overfitting with a separate dataset as well.

The primary functionalities implemented in code include:

1. **Confusion Matrix Directory Creation**: Creates a directory to save confusion matrix images.

2. **Data Loading and Preprocessing**:
    - Traverses through subfolders in the specified data directory to load images.
    - Resizes images to 64x64 pixels and converts them to numpy arrays.
    - Assigns labels based on the image and reference data filenames depending on matched timestamps.
    - Splits the data into training and testing sets, or uses the entire dataset for testing against overfitting if specified.
    - Flattens the images for model training.
3. **Model Training and Evaluation**:
    - Trains a Support Vector Classifier (SVC) or a Convolutional Neural Network (CNN) model for a specified number of epochs.
    - Evaluates the model's accuracy on the test set for each epoch.
    - Selects the best performing model based on the highest accuracy achieved.
4. **Performance Metrics and Reporting**:
    - Generates a classification report and confusion matrix for the best model.
    - Plots the confusion matrix and saves it as an image file.
    - Saves the predicted and actual labels to a CSV file.
    - Prints classification metrics, including the number and percentage of files in each category, the number of correctly identified files, and the classification report.
5. **Saving Model Metrics**: Saves the best model's accuracy and classification report to a JSON file for later reference.
6. **Command Line Interface**: Implements an argument parser to facilitate running the script from the command line, allowing the specification of input directories for image data and output directories for confusion matrices, as well as the number of training epochs.
7. **Main Function**: Orchestrates the entire process by parsing command-line arguments, creating necessary directories, and calling the training and evaluation function.

5.5.3 Data Pre-Processing and Model Training Pipeline

A DVC pipeline is implemented which orchestrates the processes described in sections 5.5.1 and 5.5.2, and it's general setup is shown below:

- **Stage: structure_raw_data**
  - **Command**: Executes a Python script to structure raw audio data and reference timestamps into a combined format.
  - **Parameters**: Accepts `fmax` and `mode` as parameters to configure spectrogram generation.
  - **Dependencies**:
    - `audioclassification/scripts/structure_data.py`
    - `data/0_raw/raw_audio_01`
    - `data/0_external/reference_timestamps_01`
    - `data/1_intermediate/noise_sample`
  - **Outputs**: Generates structured data in `data/1_intermediate/combined_raw_data`.
- **Stage: train_and_evaluate_model**
  - **Command**: Executes a Python script to train a machine learning model using the structured data and evaluate its performance, outputting confusion matrices.
  - **Parameters**: Accepts `epochs` as a parameter to specify the number of training iterations.
  - **Dependencies**:
    - `audioclassification/scripts/train_model.py`
    - `data/1_intermediate/combined_raw_data`
  - **Outputs**: Saves confusion matrices and other evaluation metrics in `data/1_intermediate/confusion_matrices`.
  - **Metrics**: Stores model training metrics in `metrics/model_training.json`.
- **Plots**
  - Generates a plot from `data/1_intermediate/confusion_matrices/predictions.csv`, plotting `actual_class` against `predicted_class`, using a predefined template (`confusion_template.json`).

This configuration ensures a structured approach to processing audio data, training a model, and evaluating its performance, facilitating reproducibility and efficient management of data and results. An exemplary metrics report is shown in following listing 5.2.

Listing 5.2: Exemplary Metrics Output


```
{
    "accuracy": 0.8,
    "classification_report": {
        "accuracy": 0.8,
        "bus": {
            "f1-score": 0.0,
            "precision": 1.0,
            "recall": 0.0,
            "support": 1.0
        },
        "car": {
            "f1-score": 0.8695652173913044,
            "precision": 0.7692307692307693,
            "recall": 1.0,
            "support": 20.0
        },
        "macro avg": {
            "f1-score": 0.4173913043478261,
            "precision": 0.9145299145299145,
            "recall": 0.4318181818181818,
            "support": 35.0
        },
        "noise": {
            "f1-score": 0.7999999999999999,
            "precision": 0.8888888888888888,
            "recall": 0.7272727272727273,
            "support": 11.0
        },
        "truck": {
            "f1-score": 0.0,
            "precision": 1.0,
            "recall": 0.0,
            "support": 3.0
        },
        "weighted avg": {
            "f1-score": 0.7483229813664597,
            "precision": 0.8332112332112332,
            "recall": 0.8,
            "support": 35.0
        }
    }
}
```


As the whole pipeline was set up in a way that allows easy execution of experiments, experiment results for all experiments were not saved but can be reproduced at any time by running experiments with adjustable parameters as described in paragraph 4.6.1 and will produce output as shown in listing 5.3, where the achieved accuracy for that experiment is 98.08%

Listing 5.3: Exemplary Metrics Output

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 82, 85, 32)        320
```

```
max_pooling2d (MaxPooling2D)    (None, 41, 42, 32)          0

flatten (Flatten)               (None, 55104)               0

dense (Dense)                   (None, 128)                 7053440

dense_1 (Dense)                 (None, 3)                   387

=================================================================
Total params: 7054147 (26.91 MB)
Trainable params: 7054147 (26.91 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
4/4 [==============================] - 7s 81ms/step - loss: 80.1989 - accuracy: 0.5288
Epoch 2/10
4/4 [==============================] - 0s 23ms/step - loss: 47.1902 - accuracy: 0.5865
Epoch 3/10
4/4 [==============================] - 0s 19ms/step - loss: 23.9625 - accuracy: 0.5962
Epoch 4/10
4/4 [==============================] - 0s 20ms/step - loss: 9.1128 - accuracy: 0.7885
Epoch 5/10
4/4 [==============================] - 0s 19ms/step - loss: 5.6969 - accuracy: 0.8269
Epoch 6/10
4/4 [==============================] - 0s 19ms/step - loss: 5.5533 - accuracy: 0.7500
Epoch 7/10
4/4 [==============================] - 0s 20ms/step - loss: 2.8009 - accuracy: 0.8365
Epoch 8/10
4/4 [==============================] - 0s 19ms/step - loss: 0.8306 - accuracy: 0.9712
Epoch 9/10
4/4 [==============================] - 0s 19ms/step - loss: 1.0633 - accuracy: 0.9615
Epoch 10/10
4/4 [==============================] - 0s 30ms/step - loss: 0.4568 - accuracy: 0.9808
```

CHAPTER 6

Results

This chapter presents a comprehensive analysis of the results obtained throughout this thesis. A comparison with existing video recognition benchmarks is provided, utilizing confusion matrices to evaluate the model's accuracy in predicting vehicle types. The findings are discussed in the context of the research questions, offering insights into the model's performance and its implications.

The available data is split into an 80/20 split as described in section 3.3.4 and the following results are based on that split.

6.1 Overview of experiment Results SVM

This section provides a detailed analysis of the models predictions, presented mainly through confusion matrices. These matrices are an effective tool for visualizing the accuracy of the model, with the actual classes on the x-axis and the predicted classes on the y-axis as explained in section 3.5.3. By examining the distribution of predictions across these axes, we can assess how well the model distinguishes between different vehicle types, which in this case are cars only. Each cell in the matrix quantifies the number of instances for each prediction, allowing us to identify areas of strong performance as well as potential weaknesses. This analysis is crucial for understanding the overall effectiveness of the model and identifying any specific classes where the model may struggle, thereby offering insights into areas for potential improvement. With this, it also became apparent that the lack of busses, trucks and motorcycles in the available data made those almost impossible for the model to distinguish from cars, leading to the model behaving like a binary classifier.

6.1.1 Basic transformation, Librosa MelSpectrogram

The basic mode chosen to transform preprocessed audio snippets to image data for machine learning is the librosa function `librosa.feature.melspectrogram`, which computes a MEL-scale-spectrogram [109].It was chosen as it presented a good proof of concept, even if the results were not optimal.

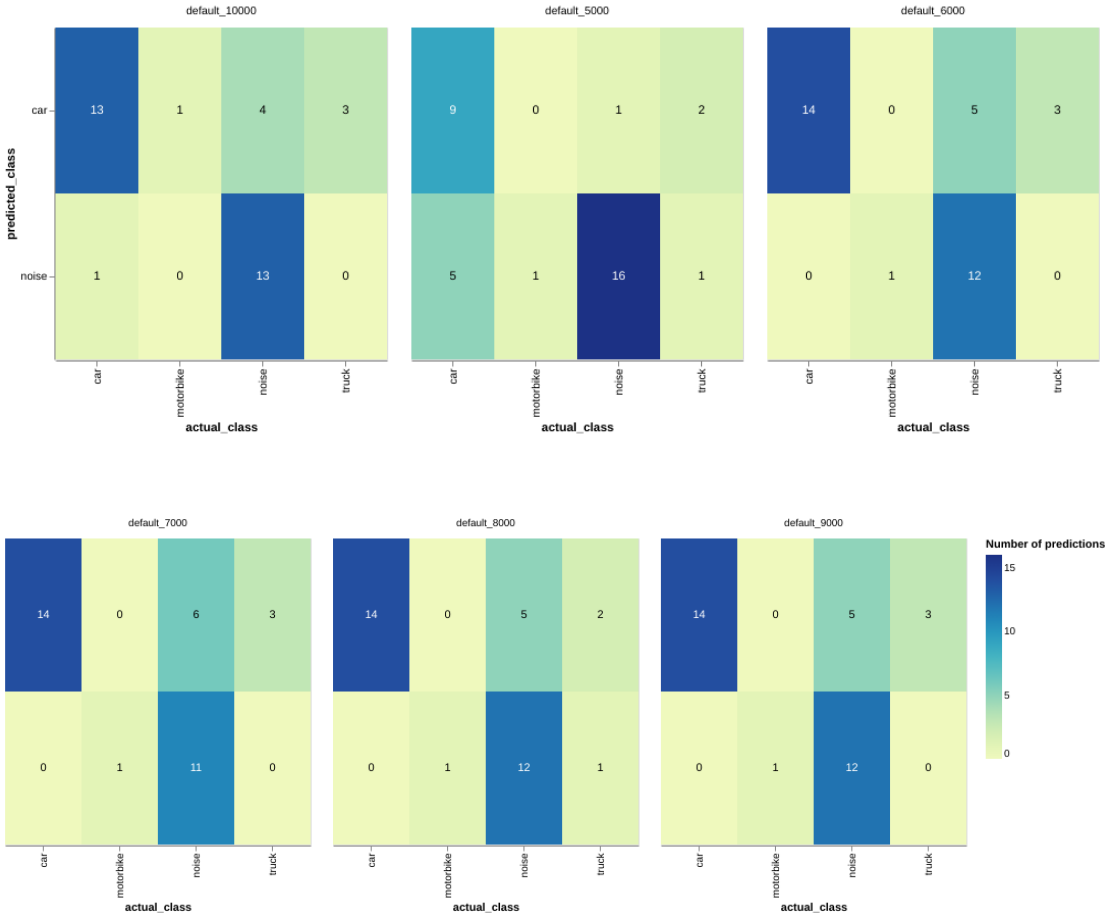

Figure 6.1: dvc confusion matrix results for basic transformation, $f_{max} = 5 - 10kHz$ for setting MEL frequency scales

The basic transformation is somewhat influenced by setting different frequencies for MEL frequency scales.

Best results are achieved for setting the MEL frequency scales at $6kHz$ and $8kHz$, as evident from figure 6.1. Accuracy is used as the defining metric for evaluating the model, as it is a fundamental and easy to understand method for showing the performance of a model. As the available datasets mostly lack data for motorbikes,

trucks and busses, the remaining main classes are "car" and "noise" for which the model classifies. Based on those two remaining classes, accuracy was deemed to be the most appropriate metric for comparison, as it is best suited for such binary classifiers with only two possible classes.

As only the classes `car` and `noise` are predicted here, formula 3.7 simplifies to:

$$Acc = \frac{TP_{car} + TP_{noise}}{Totalsamples} \tag{6.1}$$

which is also shown in formula 3.8 and yields an accuracy score of 74.3% at $6kHz$ and $8kHz$, which can also be checked by calculating accuracy based on figure 6.1. Motorbikes, trucks and busses are not detected due to their sparsity in the available dataset, with a support of only 1 motorbike, 3 trucks, and no busses.

Morlet wavelet transformation, Librosa MelSpectrogram

For the implementation of the morlet wavelet transform, Librosa's inbuilt constant-Q transform `librosa.cqt` yields the desired functionality. The model output of this approach is not influenced by setting different MEL frequency scales.

With equation 3.7, the accuracy for morlet wavelet transformation with subsequent machine learning is 71.4% on the same dataset.

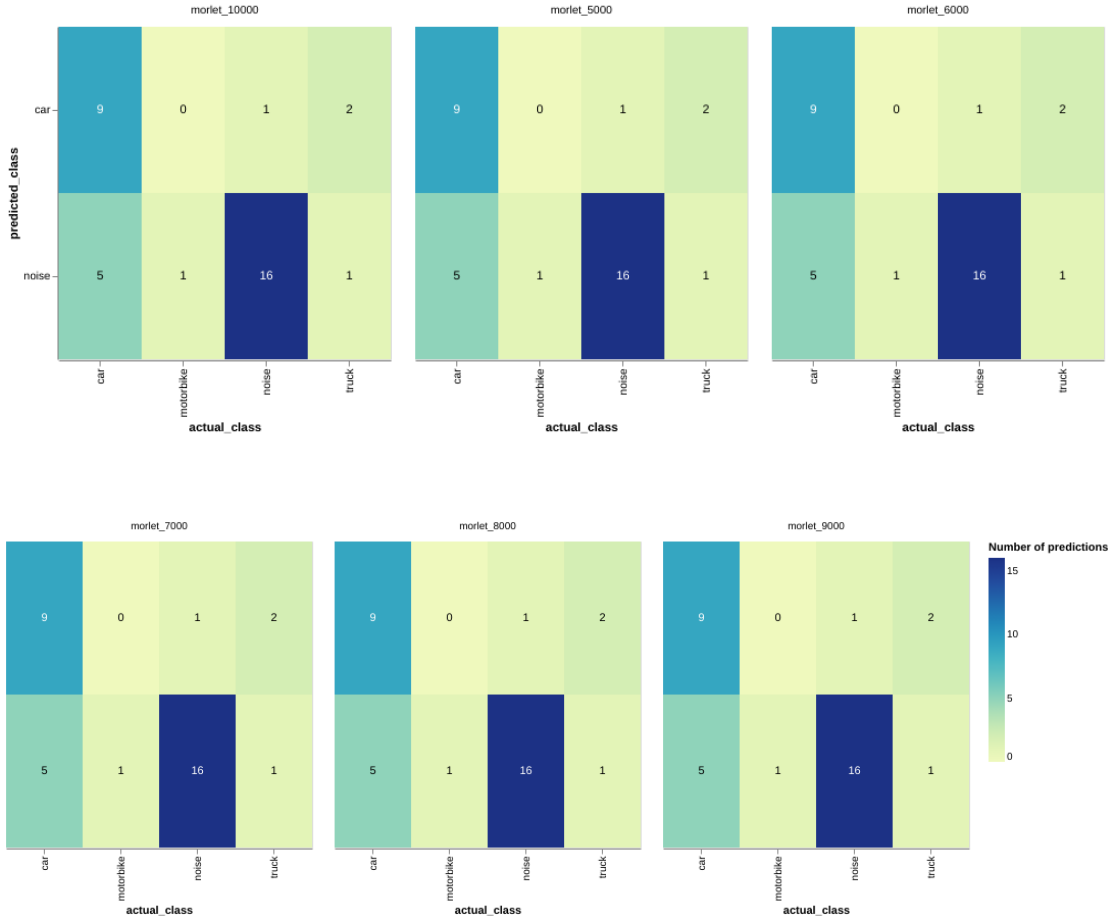

Figure 6.2: dvc confusion matrix results for morlet wavelet transformation, $f_{max} = 5 - 10kHz$ for setting MEL frequency scales

6.1.2 Spectrogram transformation with SciPy

For reference, the default scipy spectrogram `signal.spectrogram` was also utilized for preprocessing. The results show that this method does not yield good results, as the calculated accuracy from equation 3.7 yields only 40%. Detection accuracy for this approach was so low that the model could not differentiate between the two available classes, therefore this method was not further evaluated.

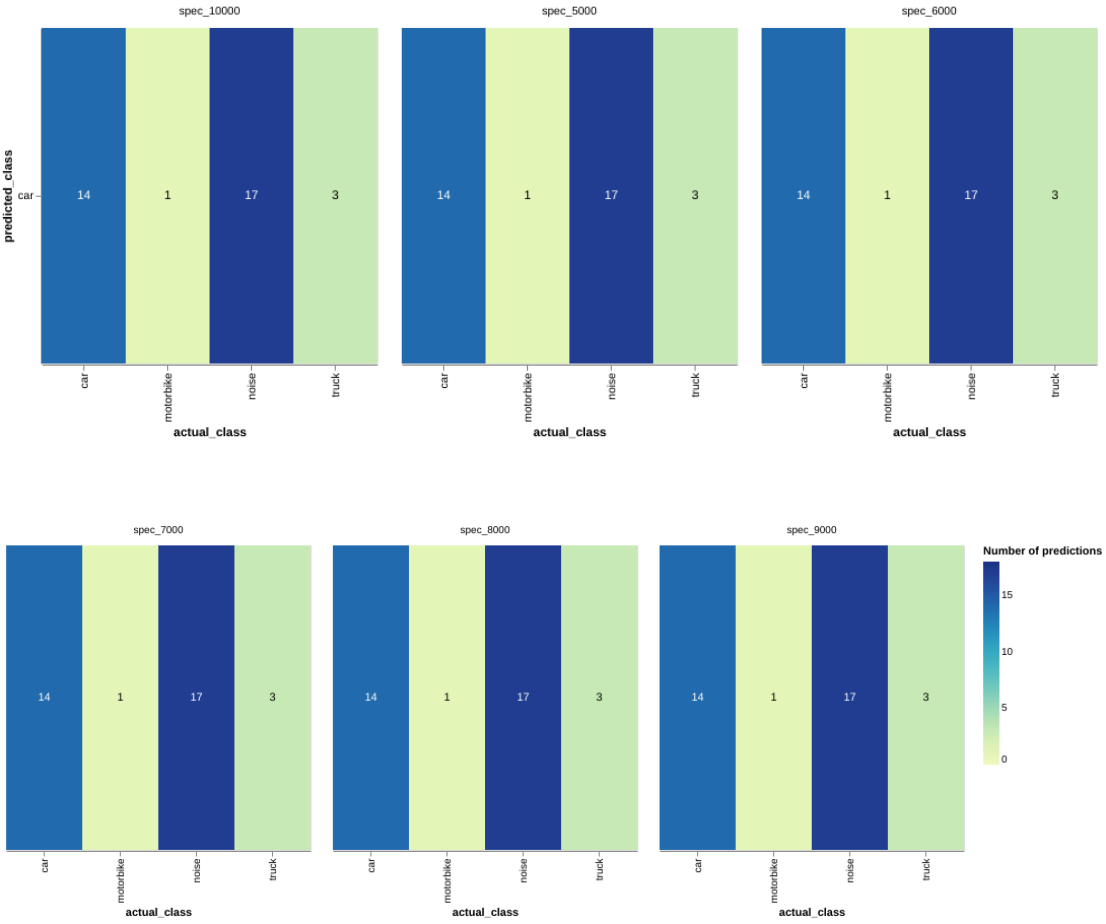

Figure 6.3: dvc confusion matrix results for simple scipy spectrogram, $f_{max} = 5 - 10kHz$ for setting MEL frequency scales

Spectrogram transformation with SciPy and ML preprocessing

For further reference, the default scipy spectrogram `signal.spectrogram` was enhanced with the function `minmax_scale` from `sklearn.preprocessing` for preprocessing. The results show that this method also yields better results, as the calculated accuracy from equation 3.7 yields 80%. The method was still not evaluated further as it needed more computational effort than the other methods and thus took significantly longer to compute while yielding only slightly better results, which were surpassed by later experiments.

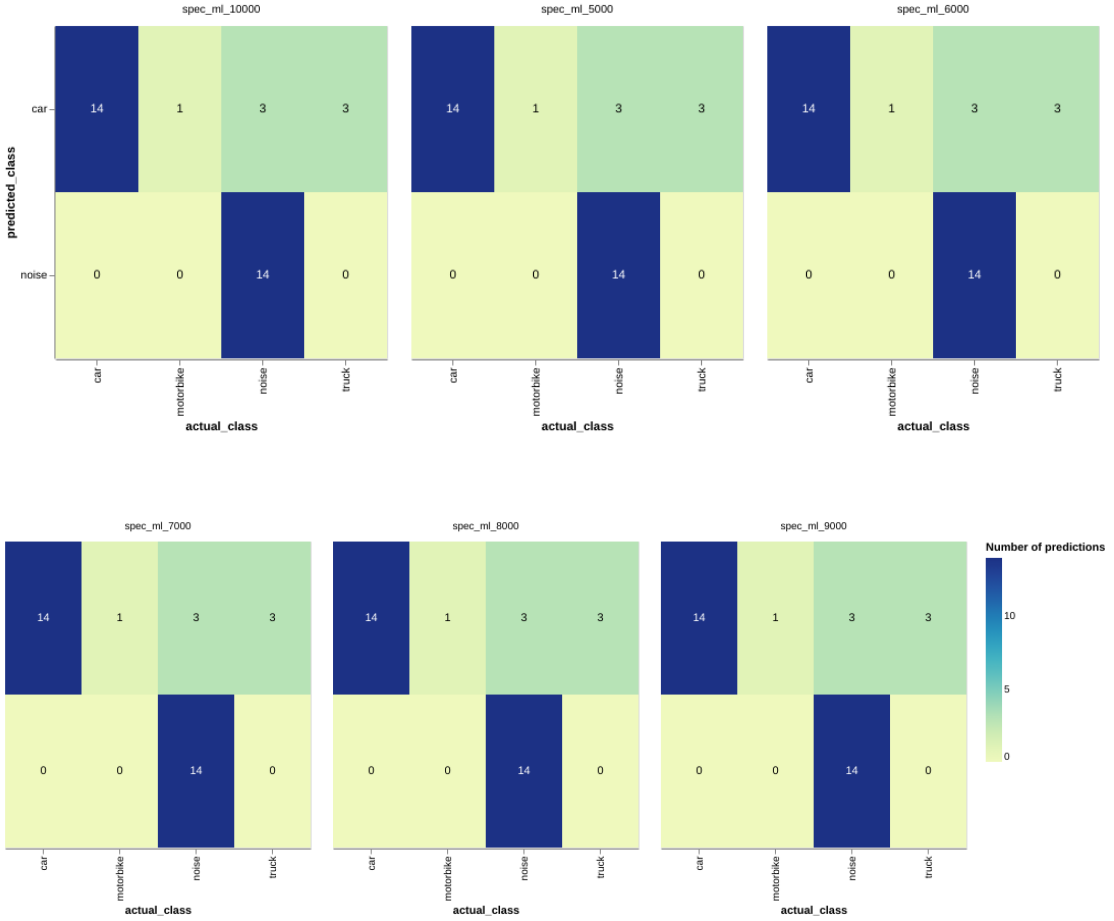

Figure 6.4: dvc confusion matrix results for scipy spectrogram enhanced by ML preprocessing, $f_{max} = 5 - 10kHz$ for setting MEL frequency scales

6.1.3 Further evaluation of SVM results

As a result of this overview, further experiments were only conducted with the default transformation MEL-spectrogram as well as with the morlet wavelet transformation MEL-spectrogram as these two methods performed the best. Based on these tests it was also concluded that the MEL-spectrogram is the ideal method of final preprocessing for the machine learning model. With these findings, further testing of both the basic mode as well as the morlet mode was conducted.

Comparison of default and morlet wavelet transformation model results

As concluded before, optimal results for both default and morlet method were achieved at $6kHz$ and $8kHz$. $8kHz$ was then chosen as the desired frequency for

further experiments, as with this frequency results seemed to be slightly better in some experiments, even when the average was very similar.

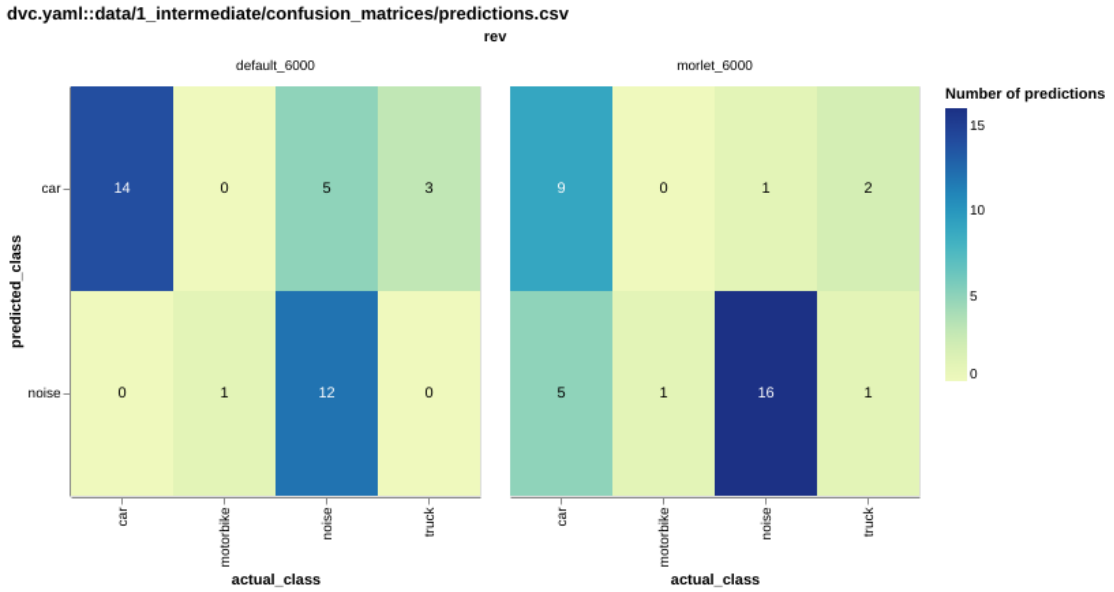

Figure 6.5: dvc confusion matrix results of default and morlet transformation ML results, $f_{max} = 6kHz$ for setting MEL frequency scales

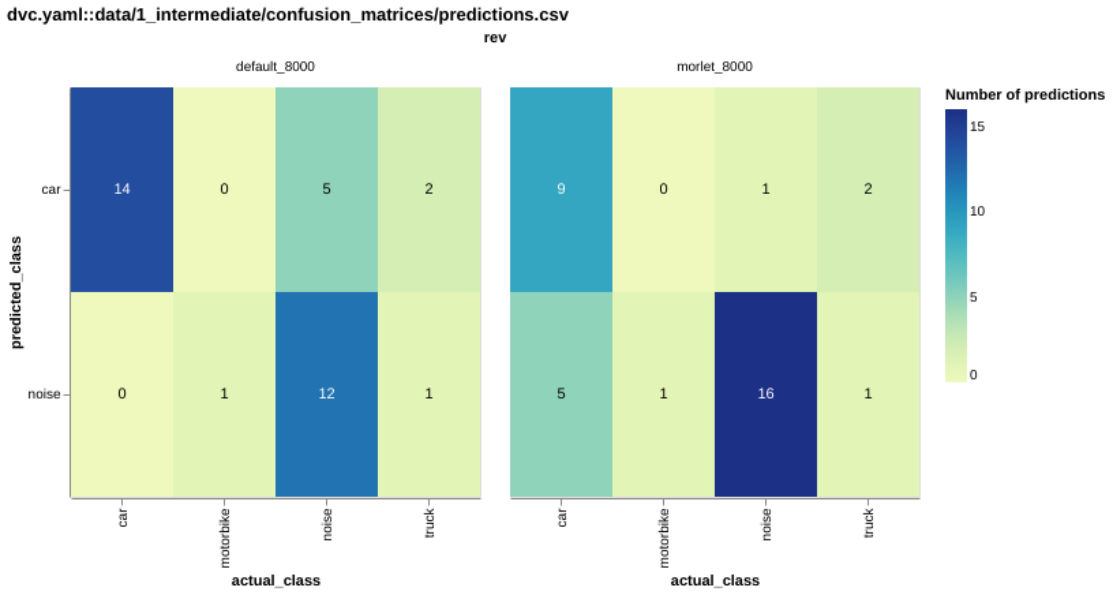

Figure 6.6: dvc confusion matrix results of default and morlet transformation ML results, $f_{max} = 8kHz$ for setting MEL frequency scales

As figures 6.5 and 6.6 show, in a direct comparison we can see that the default mode

is better at distinguishing cars from noise, and has no false negatives detected. On the other hand, 5 of 17 event were incorrectly classified as cars, when they were in fact noise.

The morlet model on the other hand shows a worse performance at detecting cars by 9 of 14, but only classifies 1 of 17 noise samples as car.

Both models struggle with motorbikes and trucks, and busses are not even in the test dataset, as there were too few samples for a train/test split.

In this regard, the classifier can be viewed as a binary classifier that is only classifying cars and noise, while disregarding the other three classes, as the necessary support data needed to provide reliable results for all possible classes does currently not exist, and would need to be accumulated by gathering a lot more data, which will be discussed in future work.

6.2 Overview of experiment Results Keras Model

With these preliminary results as basis, a Sequential Neural Network / KERAS model was built to replace the existing SVM model. The Sequential Neural Network / KERAS model setup is as follows:

Listing 6.1: Sequential Neural Network / KERAS model

```
model = tf.keras.Sequential(
            [
                layers.Input(shape=(mel_spectrogram_shape)),
                layers.Conv2D(32, (3, 3), activation="relu"),
                layers.MaxPooling2D((2, 2)),
                layers.Flatten(),
                layers.Dense(128, activation="relu"),
                layers.Dense(num_classes, activation="softmax"),
            ]
        )

        model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

        model.summary()
```

Layers used in this model are described in detail in section 4.8.4.

This model was then trained and tested with a train-test split on available data, and was then also evaluated with a separate testing dataset to test for overfitting, the reasoning behind that being explained in section 3.3.4.

From using equation 3.7 on the confusion matrix presented in figure 6.7, we can see that the default transformation yields an accuracy of 88.8%, which is significantly better than the SVM model. Here, 1 event is wrongly classified as noise, when it is in fact a car. Trucks, motorbikes and busses can still be disregarded as there are too few samples to make an impact or generate training data.

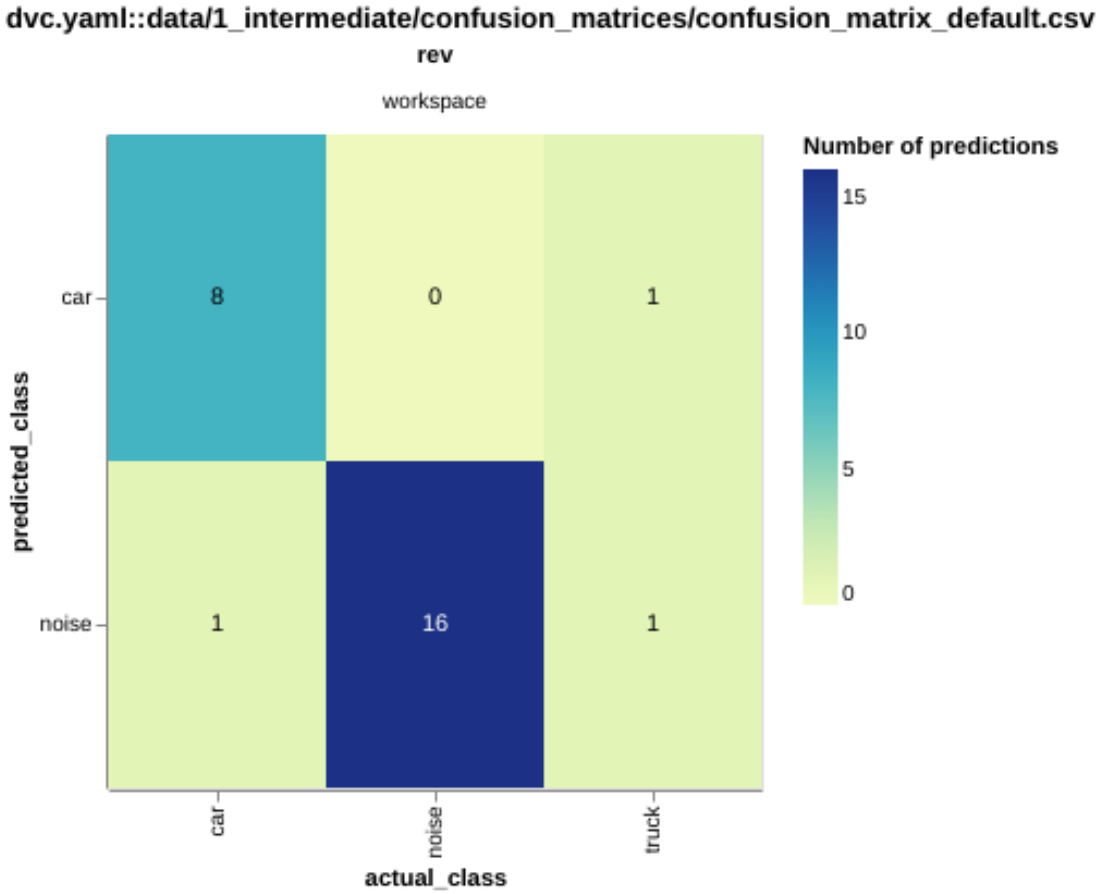

Figure 6.7: dvc confusion matrix results of default transformation ML results, $f_{max} = 8kHz$ for setting MEL frequency scales

From using equation 3.7 on the confusion matrix presented in figure 6.8 we can see that the morlet transformation yields an accuracy of 88.8% as well, which is also significantly better than the SVM model. This result shows that the morlet transformation approach does not recognize cars as noise, but does wrongly classify 1 noise sample as car. Such behaviour is slightly preferred, as detecting a car when there is none can be more important than not recognizing the car at all.

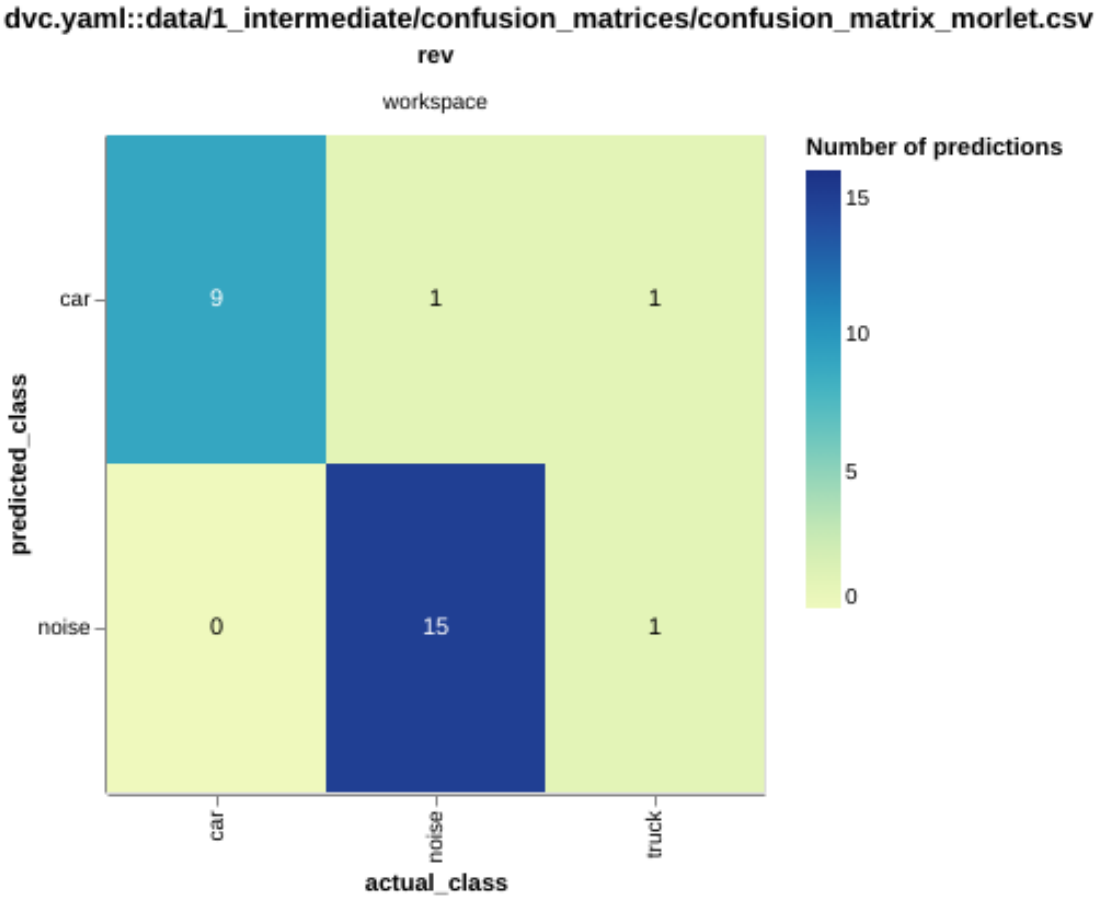

Figure 6.8: dvc confusion matrix results of morlet transformation ML results, $f_{max} = 8kHz$ for setting MEL frequency scales

6.2.1 Synopsis of ML Results

With these experiment results, it became clear that a lot of added detection accuracy could be gained from a larger training dataset for both the SVM approach as well as the keras CNN approach, as the results shown in listing 5.2 are based off of a support of only 35 events, whereas results shown in this section had a support of more than 300 events. This underlines the assumption that the model profits from more available and well labeled data, which is to be expected, but that the employed methods already delivered meaningful results with smaller dataset sizes as well. The KERAS model provides significantly better results than the SVM model, with the exemplary shown 88.8% accuracy shown as the average, while outliers reached results as good as 98% accuracy, but these results are not reliably reproducable on smaller datasets initially used by both models. For large datasets however, these excellent accuracy results are reproducible. Exemplary results shown in figures

6.1 to 6.8 rely on the same dataset to enable a direct and fair comparison of all approaches.

The maximum achieved accuracy of the Keras model of 98.08% is shown in listing 5.3, which is a definitive improvement over preceding experiments, and was in part due to adding more audio and reference data to the existing dataset, which lead to the increase in performance.

As shown by figures 6.9 and 6.10, the loss function as explained in section 3.5.4 of the morlet approach converges faster while also providing a higher ceiling for accuracy at the aforementioned 98%. Experiments were performed with up to 50 epochs, but after 10 epochs no changes were determined anymore, therefore 10 epochs was chosen as the evaluation epoch count. Both approaches show their best performance at 9 epochs, which is in both cases the model that was saved as the final result.

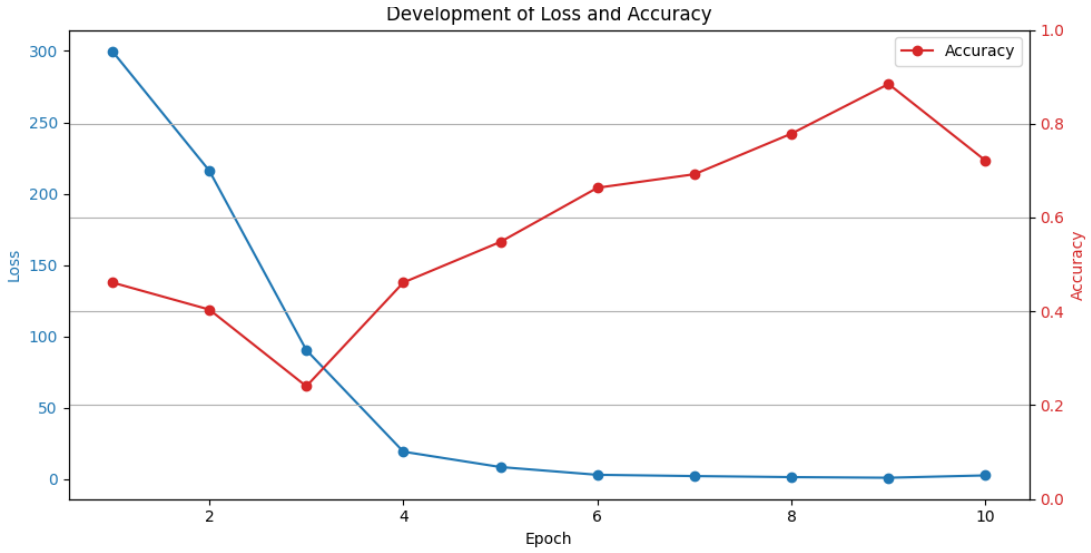

Figure 6.9: Loss and accuracy of the default transformation model

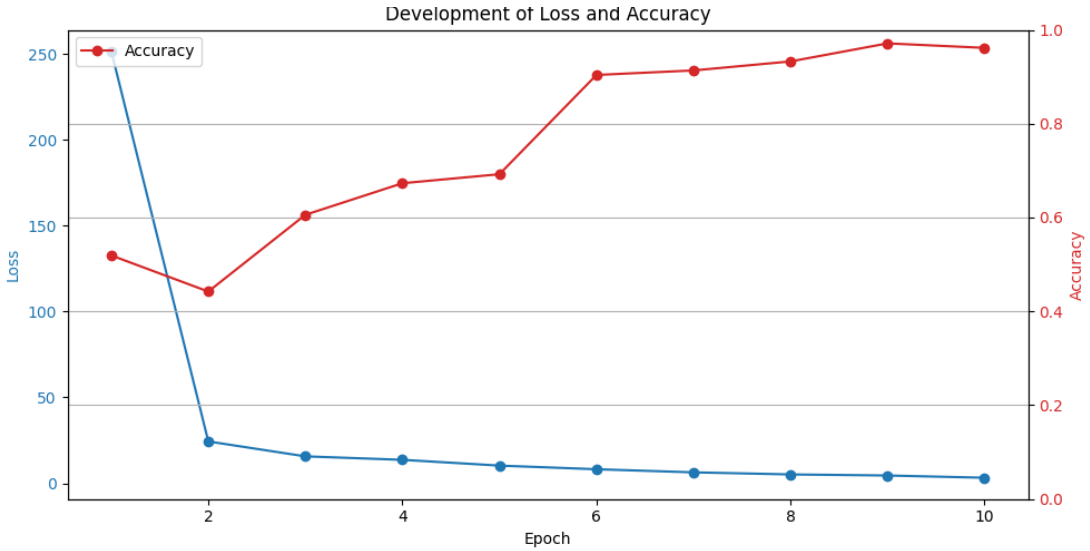

Figure 6.10: Loss and accuracy of the morlet transformation model

In conclusion, the results show that both the default- as well as the morlet transformation methods deliver reliable output data after machine learning with Keras.

With the morlet approach slightly outperforming the default approach, specifically regarding figures 6.9 and 6.10, the approach to use the morlet wavelet transformation along with a Keras CNN model is deemed to be the best model to use in this thesis, and is viewed as the final Machine Learning output of this work.

This also underlines the reason for the change to the Keras CNN model, as the originally implemented SVM model provided a good proof of concept, but as Keras CNN is a popular and effective choice with many advantages as outlined in section 3.5.2 it was the desired model to use in this thesis, which the achieved results validate.

6.2.2 Accuracy of detected events

In following figures 6.11 to 6.13 it becomes apparent that audio recognition sometimes registers events that are clearly audio events to be classified at times when reference data showed no event. This is most evident in the outtake in figure 6.11, and events detected by audio pre-processing can be listened to at that timestamp in original audio data and verified that they sound like cars passing, but the lack of reference data for that timestamp makes the sample unusable. Upon further evaluation it was concluded that these are in fact the sounds of vehicles passing by that for unknown reason did not register on video recognition, but it may be speculated that the camera was obscured.

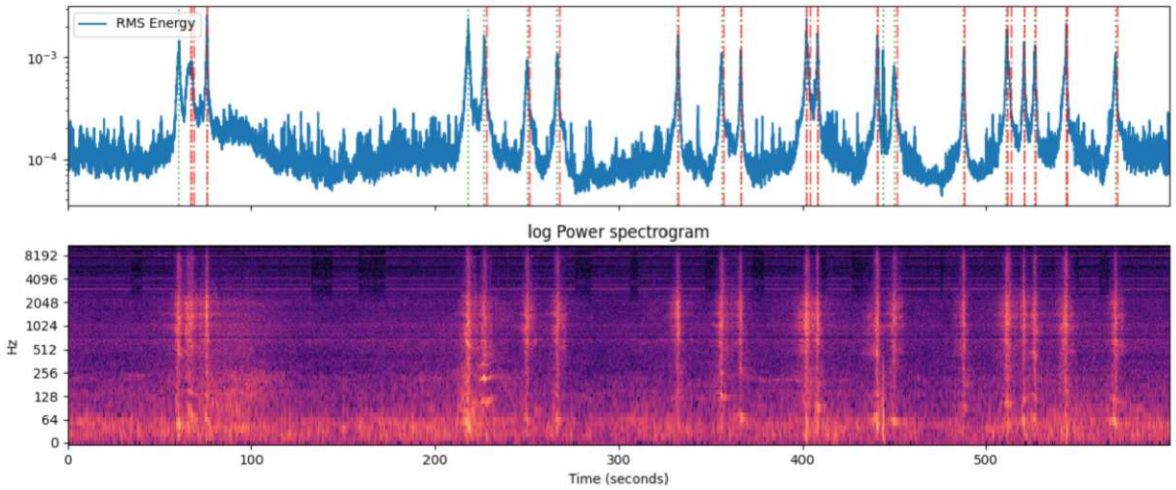

Figure 6.11: Accuracy of audio detection vs. video detection.
Event timestamps detected by audio pre-processing are shown as green lines, while event timestamps extracted from reference data are shown as red lines

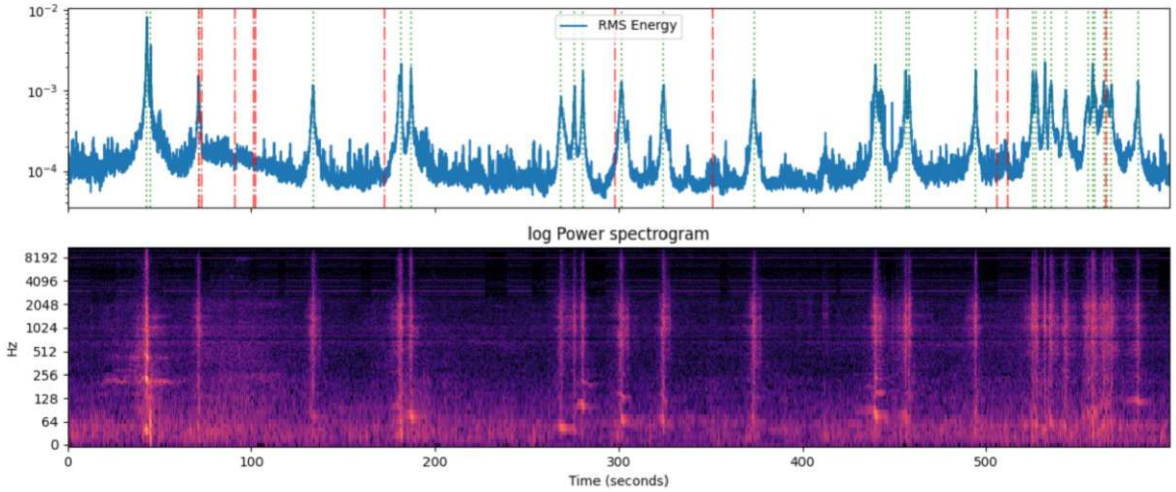

Figure 6.12: Accuracy of audio detection vs. video detection.
Event timestamps detected by audio pre-processing are shown as green lines, while event timestamps extracted from reference data are shown as red lines

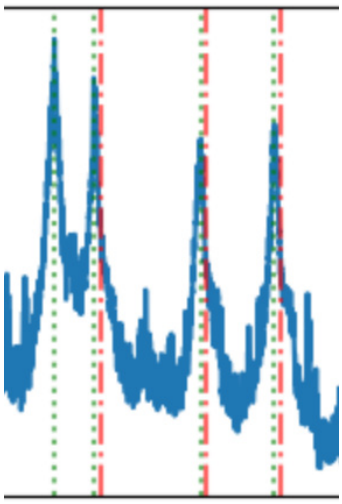

Figure 6.13: Accuracy of audio detection vs. video detection.
Event timestamps detected by audio pre-processing are shown as green lines, while event timestamps extracted from reference data are shown as red lines

CHAPTER 7

Discussion and future work

In this chapter, the findings of this thesis are discussed, as well as how research questions are answered and what the future outlook based on this thesis is.

Traffic classification is important because it enables the accurate computation of current traffic flow, congestions, diversity of vehicles and many more factors, as well as lay a foundation for future planning by presenting the development of traffic at the measured location. Audio-based methods provide an interesting and novel approach to currently existing systems, i.e. video based ones.

7.1 Research questions

The research questions already posed in section 2.1 are re-iterated here and also answered.

7.1.1 Calculation of Passing Vehicles (RQ1)

The first research question adresses the counting of vehicles from audio data.

Question Statement Can the number of passing vehicles be calculated accurately with respect to the number of lanes, or for a single lane only? Here, a software prototype will be built and tested against reference data obtained from video data. Research through development will help to refine the computations so that they can be verified by comparison with results from video analysis.

Empiric Findings Real-world data from a sensor box mounted at the same place as an existing video recognition system was collected. Different pre-processing

and feature extraction methods were tested along with different machine learning approaches. The number of passing vehicles could be calculated for the existing two-lane setup, even when higher traffic flow occurred, with an accuracy of up to 98% relative to reference data, as the number of passing vehicles detected correctly goes hand in hand with detecting their timestamp and type. However, this does not indicate the viability for a multi-lane setup like a highway, where this approach would have to be tested separately. For a multi-lane setup, data for each lane would need to be gathered separately, while ensuring that adjacent lanes are not detected.

Obtained results show that audio detection catches each peak in the presented audio data, whereas video detection does not capture every event, examples of this can be found in figures 6.11 and 6.12. Therefore concrete examples for situations where the accuracy of audio recognition for vehicle counting surpasses that of video recognition can be provided.

These audio files were verified by listening to each event audio snippet and concluding that they indeed sounded like passing cars and also sounded exactly like verified sound samples from timestamps where a video recognition event exists, therefore the accuracy of audio recognition for vehicle counting surpasses that of video recognition in this case.

7.1.2 Vehicle Type Deduction (RQ2)

The second research question adresses the correct detection of the vehicle type.

Question Statement Can the type of a passing vehicle be deduced accurately with machine learning? The results will be tested against video data obtained at the same location and time as the audio data, and photoelectric barriers can also be used to gather vehicle count and timestamps for verification.

Empiric Findings Utilizing machine learning, the type of passing vehicles could be deduced with a $95 - 97\%$ peak detection accuracy and an average of 88.7%, which is a very good proof of concept and shows that the audio recognition approach is a viable alternative to the existing video recognition setup. A known caveat is the prevalence of cars compared to the other tested types of busses, motorbikes and trucks. Cars make up about 85% of the passing vehicles at the testing site, which produces a bias towards detecting cars and noise very well, while the other classes are not detected often.

7.1.3 Influence of Preliminary Setup (RQ3)

Research question three adresses the influence of the hardware deployed in the sensorbox on the possibility of running the pipeline as described in section 4.5 also

on the Khadas VIM3 microcomputer.

Question Statement How does the hardware setup, which is based on a Khadas VIM3 as the microcomputer in use with a Neural Processing Unit (NPU) Machine Learning (ML) module for ML workloads, influence the computation, and is it possible to finish computations close to real-time, where results can be acquired within a maximum constant time window of a few seconds after each event? A window of five seconds is the desired maximum here, as the evaluation period on the sensor box has a sliding window of five seconds in length.

Empiric Findings

Preprocessing was implemented on the Khadas VIM3, however testing showed that the hardware was not appropriate to handle the computational load, as after the first few seconds of analyzing audio data, the system runs into memory issues and the process is shut down. This is further made more difficult as this audio pre-processing pipeline would only be one of many clients running on that microcomputer, and as the hardware is not capable of running it alone, this research question could not be answered, but will have to be moved to future work with the next generation of sensor boxes.

7.1.4 Overall Representation of Passing Traffic (RQ4)

The fourth research question adresses the accuracy of traffic representation regarding the available reference data.

Question Statement Can the workflow developed in this thesis lead to an accurate representation of passing traffic in regard to the existing video based detection system? Could this detection method be used in different locations as well, and how would that change in surroundings affect detection accuracy? This depends on the first three research questions and will be examined by research through development and comparing results from audio detection with reference measurements for the same period of time at the same location obtained from video analysis. All of these results will then be interpreted and described and will yield the final verdict.

Empiric Findings This thesis shows that an accurate representation of traffic can be made under certain circumstances, as the prevalence of cars means that their detection works great, whereas trucks, busses and motorbikes are sometimes not detected at all, leading to a skewed representation of actual traffic at the testing site. However, as cars pose about 85% of the passing vehicles, the methods employed in this thesis manage to model the number of passing cars very well. For the testing location, an accurate representation of passing traffic is possible, and for different

locations future field studies will have to be conducted. As the accuracy of the final model yielded an accordance of 98% with reference data, this research question can be answered as providing an accurate representation of traffic at the deployment location.

7.2 Discussion

In this thesis, accuracy, recall, precision and F1-score of different support vector machines and neural networks were evaluated, which is evident from examples as shown in listing 5.2. From these options, accuracy of these different models was chosen as the defining metric and compared between them. Combined with various preprocessing and feature extraction approaches, this leads to following takeaways:

In this thesis, it became apparent that an accurate alternative to the existing video recognition setup can be implemented with audio recognition and machine learning. The best model results came to an accuracy of 97% compared to video recognition being regarded as 100% and "true".

A big advantage of audio recognition is that it is less hindered by bad weather conditions or other adverse conditions as much as video recognition, as results show that events not captured on video can be found in audio data, examples shown in figures 6.11, 6.12 and 6.13. Overall, the SVM model approach provided a good proof of concept for further development, while the KERAS model approach showed that with a better model setup and a larger dataset, very accurate predictions of vehicle types can be made.

The biggest known caveat is that we have to regard the results of video recognition as "true", which for now is sufficient as this data is in use by the Austrian Autobahn and highway financing stock corporation (ASFINAG), which is a trusted source. With this in mind, the final result of this thesis is that accurate predictions of the types of passing cars can be made by audio recognition and machine learning, rivaling, and in certain conditions surpassing, those of video recognition.

This can be used to either try to replace video recognition for traffic monitoring in places where that would be helpful (i.e. no option to mount a camera), or to supplement an existing video recognition system with additional information, possibly yielding better results than one method alone.

The objective of this thesis to determine whether reliable results can be obtained through audio recognition and machine learning has been achieved. Consequently, additional testing of various configurations will be considered in future work.

7.3 Future Work

Based on the presented takeaways from this work, a plan for future endeavours to further explore these subjects can be formed.

As this thesis as a whole builds the proof of concept to utilize audio recognition in the existing setup with sensor boxes, a big part of further development will be the adaptation to the microcomputer, so classification can be done on site. For this to be possible, pre-processing will likely have to be reduced a little and made computationally more lightweight, as the microcomputer does not possess the necessary hardware capabilities to run it at full capacity, much less with other clients running on the same hardware simultaneously. A different approach would be the evaluation of alternative hardware, as a successor of the Khadas VIM3 microcomputer already exists as the Khadas VIM4, which poses more processing power and would be from the same manufacturer, or to look for different suppliers as well.

As all data for this thesis comes from a two-way street, in future work the possibility of analysing roads with more lanes or even highways will also have to be explored. Connected with this is the gathering of more data, as it became evident that with more available data, model results improved, which is to be expected from a Machine Learning approach. So one important part of further developments would be to acquire a lot more data than what was available for this thesis and further develop the models to be able to produce better results still.

The partner company Bernard Group is also working on providing a mobile measurement setup consisting of a camera as well as a sensor box, which will allow the generation of a golden dataset. This will lead to a verified model that no longer has to rely on data from a black box, as well as providing the option to test the setup in many different locations and under different conditions.

Once such a field study can be conducted, the model can be further developed with data where we know the exact conditions during recording as well as the exact vehicle correlated to an audio snippet, which can lead to even better classification. This is also directly related with simply gathering more data to improve the model, and would lead to possible testing of the same model in different locations, and how that affects detection accuracy, as well as providing the basis to train a model based on data from multiple different locations.

A model trained from that data should then be able to reliably detect events at changing locations, but would still need to be evaluated against a model specifically trained at one location to test the difference in performance, if there is one.

The detection of other events than passing cars is also a possibility, as this setup could be used to be trained on different signals like sirens of emergency vehicles to provide a reliable detection of those, which could help with traffic control as well.

Predictive maintenance is also a possible use case, as the project setup could be changed to be permanently listening to some input and watching for abnormal changes in the input signal, like a bearing giving out in an electric motor or generator.

There are many possible use cases that can be found and discusses, but this sums up the ideas that have come up during the course of this thesis.

Overview of Generative AI Tools Used

Use of Scite.ai for finding further relevant papers and where to download them for reading and referencing and ChatGPT for asking me questions about the thesis which lead to better and simpler written explanations and for translating abstract and acknowledgements.

Übersicht verwendeter Hilfsmittel

Scite.ai um weitere relevante Paper zu finden und herauszufinden wo diese verfügbar sind, ChatGPT als Konversationspartner der generelle Fragen zum besseren Verständnis der Arbeit stellt, was zu besseren und einfacheren Erklärungen geführt hat und zur Übersetzung von Abstract und Danksagung.

List of Figures

Acronyms

Acc Accuracy. 16, 92

API Application Programming Interface. 28, 29, 38, 50, 60, 92

ASFINAG Austrian Motorway and Expressway Financing Joint-Stock Company. 3, 92

CNN Convolutional Neural Network. 14, 38, 63, 76, 78, 92

CQT Constant Q Transform. 35, 57, 59, 92

CSV Comma Separated Values. 63, 92

CURL Client for URL. 29, 92

dB Decibel. 53, 92

DFT Discrete Fourier Transform. 57, 92

DVC Data Version Control. 33, 42, 64, 92

JSON JavaScript Object Notation. 23, 28, 29, 37, 46, 50, 51, 60–63, 92

MEL Melody or melody scale, a perceptual scale of pitches judged by listeners to be equal in distance from one another. 9, 10, 33, 35, 38, 56–58, 68, 91, 92

MEMS Micro-Electro-Mechanical Systems. 3, 49, 92

ML Machine Learning. 4, 15, 33, 35, 36, 41, 45, 49–51, 55, 59, 60, 62, 76, 78, 83, 85, 92

MP3 MPEG-1 Audio Layer III. 23, 28, 29, 46, 50, 92

NPU Neural Processing Unit. 4, 83, 92

PNG Portable Network Graphics. 23, 92

ReLU Rectified Linear Unit. 10, 38, 92

REST Representational State Transfer. 28, 50, 60, 92

RMS Root Mean Square. 35, 51–55, 61, 62, 91, 92

SPL Sound Pressure Level. 27, 92

SRB Sensorbox. 27, 91, 92

SVC Support Vector Classifier. 63, 92

SVM Support Vector Machine. xvi, 14, 36, 37, 42, 67, 69, 71, 73, 76, 78, 92

THD Total harmonic Distortion. 27, 92

TMPL Template. 92

Bibliography of Print Media

[1] J. Guerrero-Ibáñez, S. Zeadally, and J. Contreras-Castillo, „Sensor technologies for intelligent transportation systems“, *Sensors*, vol. 18, no. 4, p. 1212, 2018.

[2] G. Zhang, R. P. Avery, and Y. Wang, „Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras“, *Transportation research record*, vol. 1993, no. 1, pp. 138–147, 2007.

[3] X. Zhang, S. Hu, H. Zhang, and X. Hu, „A real-time multiple vehicle tracking method for traffic congestion identification“, *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 10, no. 6, pp. 2483–2503, 2016.

[4] K. Yousaf, A. Iftikhar, and A. Javed, „Comparative analysis of automatic vehicle classification techniques: A survey“, *International Journal of Image, Graphics and Signal Processing*, vol. 4, no. 9, p. 52, 2012.

[5] M. Najm and Y. H. Ali, „Automatic vehicles detection, classification and counting techniques/survey“, *Iraqi Journal of Science*, pp. 1811–1822, 2020.

[6] Y. Yang, T. Zhang, J. Hu, D. Xu, and G. Xie, „End-to-end background subtraction via a multi-scale spatio-temporal model“, *IEEE Access*, vol. 7, pp. 97 949–97 958, 2019.

[7] J. Yang, J. Yuan, and X. Shen, „Neuronal edge detection with median filtering and gradient sharpening“, in *2012 Fourth International Symposium on Information Science and Engineering*, IEEE, 2012, pp. 259–262.

[8] N. Buch, S. A. Velastin, and J. Orwell, „A review of computer vision techniques for the analysis of urban traffic“, *IEEE Transactions on intelligent transportation systems*, vol. 12, no. 3, pp. 920–939, 2011.

[9] H. Ouchra and A. Belangour, „Object detection approaches in images: A weighted scoring model based comparative study“, *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, pp. 268–275, 2021.

[10] J. Li, D. Lin, Y. Wang, *et al.*, „Deep discriminative representation learning with attention map for scene classification“, *Remote Sensing*, vol. 12, no. 9, p. 1366, 2020.

[11] G. Cui, S. Wang, Y. Wang, Z. Liu, Y. Yuan, and Q. Wang, „Preceding vehicle detection using faster r-cnn based on speed classification random anchor and q-square penalty coefficient“, *Electronics*, vol. 8, no. 9, p. 1024, 2019.

[15] E.-J. Kim, H.-C. Park, S.-W. Ham, S.-Y. Kho, and D.-K. Kim, „Extracting vehicle trajectories using unmanned aerial vehicles in congested traffic conditions“, *Journal of Advanced Transportation*, vol. 2019, no. 1, p. 9 060 797, 2019.

[16] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, „Deep learning in video multi-object tracking: A survey“, *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[17] X. Xie, H. van Lint, and A. Verbraeck, „A generic data assimilation framework for vehicle trajectory reconstruction on signalized urban arterials using particle filters“, *Transportation research part C: emerging technologies*, vol. 92, pp. 364–391, 2018.

[18] X. Tang, H. Song, W. Wang, and Y. Yang, „Vehicle spatial distribution and 3d trajectory extraction algorithm in a cross-camera traffic scene“, *Sensors*, vol. 20, no. 22, p. 6517, 2020.

[19] P. Gao, R. Guo, H. Lu, and H. Zhang, „Multi-view sensor fusion by integrating model-based estimation and graph learning for collaborative object localization“, pp. 9228–9234, 2021.

[20] P. Corcoran, A. Winstanley, P. Mooney, and R. Middleton, „Background foreground segmentation for slam“, *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1177–1183, 2011.

[21] M. Irani and P. Anandan, „A unified approach to moving object detection in 2d and 3d scenes“, *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 6, pp. 577–589, 1998.

[22] R. Hou and K.-S. Jeong, „3d reconstruction and self-calibration based on binocular stereo vision“, *Journal of the Korea Academia-Industrial cooperation Society*, vol. 13, no. 9, pp. 3856–3863, 2012.

[23] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, „Large-scale video classification with convolutional neural networks“, pp. 1725–1732, 2014.

[24] L. Rokach, „Ensemble-based classifiers“, *Artificial intelligence review*, vol. 33, pp. 1–39, 2010.

[25] F. He, „Intelligent video surveillance technology in intelligent transportation“, *Journal of Advanced Transportation*, vol. 2020, no. 1, p. 8 891 449, 2020.

[26] P. Foggia, A. Saggese, N. Strisciuglio, M. Vento, and V. Vigilante, „Detecting sounds of interest in roads with deep networks“, pp. 583–592, 2019.

[27] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, „Audio surveillance of roads: A system for detecting anomalous sounds“, *IEEE transactions on intelligent transportation systems*, vol. 17, no. 1, pp. 279–288, 2015.

[28] L. Lu, H.-J. Zhang, and S. Z. Li, „Content-based audio classification and segmentation by using support vector machines“, *Multimedia systems*, vol. 8, pp. 482–492, 2003.

[29] A. Wieczorkowska, E. Kubera, T. Słowik, and K. Skrzypiec, „Spectral features for audio based vehicle and engine classification“, *Journal of Intelligent Information Systems*, vol. 50, pp. 265–290, 2018.

[30] Y. Astuti, R. Hidayat, and A. Bejo, „A mel-weighted spectrogram feature extraction for improved speaker recognition system.“, *International Journal of Intelligent Engineering & Systems*, vol. 15, no. 6, 2022.

[31] S. Hu, Z. Liao, R. Hou, and P. Chen, „Characteristic sequence analysis of giant panda voiceprint“, *Frontiers in Physics*, vol. 10, p. 839 699, 2022.

[32] T. Jayasree and R. P. Ananth, „Sound signal based fault classification system in motorcycles using hybrid feature sets and extreme learning machine classifiers.“, *Sound and Vibration*, vol. 54, no. 1, pp. 57–74, 2020.

[33] J. K. Das, A. Chakrabarty, and M. J. Piran, „Environmental sound classification using convolution neural networks with different integrated loss functions“, *Expert Systems*, vol. 39, no. 5, e12804, 2022.

[34] M. Davy, A. Gretton, A. Doucet, and P. J. Rayner, „Optimized support vector machines for nonstationary signal classification“, *IEEE Signal Processing Letters*, vol. 9, no. 12, pp. 442–445, 2002.

[35] R. Sawata, T. Ogawa, and M. Haseyama, „Novel audio feature projection using kdlpcca-based correlation with eeg features for favorite music classification“, *IEEE transactions on affective computing*, vol. 10, no. 3, pp. 430–444, 2017.

[36] F. Ahmed, P. P. Paul, and M. Gavrilova, „Music genre classification using a gradient-based local texture descriptor“, pp. 455–464, 2016.

[37] L. Nanni, A. Rigo, A. Lumini, and S. Brahnam, „Spectrogram classification using dissimilarity space“, *Applied Sciences*, vol. 10, no. 12, p. 4176, 2020.

[38] Y. Zhang, D. Lv, and Y. Lin, „Environmental audio classification based on active learning with svm“, pp. 208–212, 2015.

[39] S. Zahid, F. Hussain, M. Rashid, M. H. Yousaf, and H. A. Habib, „Optimized audio classification and segmentation algorithm by using ensemble methods“, *Mathematical Problems in Engineering*, vol. 2015, no. 1, p. 209 814, 2015.

[40] R. E. Learned and A. S. Willsky, „A wavelet packet approach to transient signal classification“, *Applied and computational Harmonic analysis*, vol. 2, no. 3, pp. 265–278, 1995.

[41] I. Daubechies, „Ten lectures on wavelets“, 1992.

[42] H. Göksu, „Engine speed–independent acoustic signature for vehicles“, *Measurement and Control*, vol. 51, no. 3-4, pp. 94–103, 2018.

[43] N. E. Huang, Z. Shen, S. R. Long, *et al.*, „The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis“, *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, vol. 454, no. 1971, pp. 903–995, 1998.

[45] I. Daubechies, J. Lu, and H.-T. Wu, „Synchrosqueezed wavelet transforms: An empirical mode decomposition-like tool“, *Applied and computational harmonic analysis*, vol. 30, no. 2, pp. 243–261, 2011.

[46] M. Bernardini, G. Della Posta, F. Salvadore, and E. Martelli, „Unsteadiness characterisation of shock wave/turbulent boundary-layer interaction at moderate reynolds number“, *Journal of Fluid Mechanics*, vol. 954, A43, 2023.

[48] Y. Huang, J. Benesty, and J. Chen, „Analysis and comparison of multichannel noise reduction methods in a common framework“, *IEEE transactions on audio, speech, and language processing*, vol. 16, no. 5, pp. 957–968, 2008.

[49] J. Zürn and W. Burgard, „Self-supervised moving vehicle detection from audio-visual cues“, *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7415–7422, 2022.

[50] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, „Robust sound event classification using deep neural networks“, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 540–552, 2015.

[51] D. Jiang, D. Huang, Y. Song, *et al.*, „An audio data representation for traffic acoustic scene recognition“, *IEEE Access*, vol. 8, pp. 177 863–177 873, 2020.

[52] N. Almaadeed, M. Asim, S. Al-Maadeed, A. Bouridane, and A. Beghdadi, „Automatic detection and classification of audio events for road surveillance applications“, *Sensors*, vol. 18, no. 6, p. 1858, 2018.

[53] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, „Detection and classification of acoustic scenes and events“, *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[54] G. Guo and S. Z. Li, „Content-based audio classification and retrieval by support vector machines“, *IEEE transactions on Neural Networks*, vol. 14, no. 1, pp. 209–215, 2003.

[55] C. Cortes and V. Vapnik, „Support-vector networks“, *Machine learning*, vol. 20, pp. 273–297, 1995.

[56] J. VAVREK, J. Jozef, and A. Anton ˇCI ˇZM, „The svm binary tree classification using mrmr and f-score feature selection algorithms“, *Acta Electrotechnica et Informatica*, vol. 14, no. 2, pp. 8–14, 2014.

[57] V. Y. Kulkarni and P. K. Sinha, „Pruning of random forest classifiers: A survey and future directions“, pp. 64–68, 2012.

[58] A. Gupta and B. Kahali, „Machine learning-based cognitive impairment classification with optimal combination of neuropsychological tests“, *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, vol. 6, no. 1, e12049, 2020.

[59] P. Probst, M. N. Wright, and A.-L. Boulesteix, „Hyperparameters and tuning strategies for random forest“, *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, vol. 9, no. 3, e1301, 2019.

[61] J. Salamon and J. P. Bello, „Deep convolutional neural networks and data augmentation for environmental sound classification“, *IEEE Signal processing letters*, vol. 24, no. 3, pp. 279–283, 2017.

[62] Y. Liu, A. Neophytou, S. Sengupta, and E. Sommerlade, „Cross-modal spectrum transformation network for acoustic scene classification“, pp. 830–834, 2021.

[63] R. V. Sharan, H. Xiong, and S. Berkovsky, „Benchmarking audio signal representation techniques for classification with convolutional neural networks“, *Sensors*, vol. 21, no. 10, p. 3434, 2021.

[64] Q. Zhang, H. Lu, H. Sak, *et al.*, „Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss“, pp. 7829–7833, 2020.

[65] K. Avramidis, A. Kratimenos, C. Garoufis, A. Zlatintsi, and P. Maragos, „Deep convolutional and recurrent networks for polyphonic instrument classification from monophonic raw audio waveforms“, pp. 3010–3014, 2021.

[67] A. Ramalingam and S. Krishnan, „Gaussian mixture modeling using short time fourier transform features for audio fingerprinting“, pp. 1146–1149, 2005.

[68] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, „Semantic annotation and retrieval of music and sound effects“, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 467–476, 2008.

[69] H. Wang, „The generalized bayes method for high-dimensional data recognition with applications to audio signal recognition“, *Symmetry*, vol. 13, no. 1, p. 19, 2020.

[70] D. Bonet-Solà and R. M. Alsina-Pagès, „A comparative survey of feature extraction and machine learning methods in diverse acoustic environments“, *Sensors*, vol. 21, no. 4, p. 1274, 2021.

[71] A. A. Abro, E. Taşcı, and A. Ugur, „A stacking-based ensemble learning method for outlier detection“, *Balkan Journal of Electrical and Computer Engineering*, vol. 8, no. 2, pp. 181–185, 2020.

[72] M. Panda, D. Mishra, and S. Mishra, „Ensemble methods for improving classifier performance“, pp. 363–374, 2018.

[73] Y. Chen, J. He, W. Wei, N. Zhu, and C. Yu, „A multi-model approach for user portrait“, *Future Internet*, vol. 13, no. 6, p. 147, 2021.

[74] N. Sutton-Charani, A. Imoussaten, S. Harispe, and J. Montmain, „Evidential bagging: Combining heterogeneous classifiers in the belief functions framework“, pp. 297–309, 2018.

[76] G. E. Goutte C., „A probabilistic interpretation of precision, recall and f-score, with implication for evaluation“, *Losada, D.E., Fernández-Luna, J.M. (eds) Advances in Information Retrieval*, vol. ECIR 2005, 2005. doi: `https://doi.org/10.1007/978-3-540-31865-1_25`.

[77] M. M. Trivedi, T. L. Gandhi, and K. S. Huang, „Distributed interactive video arrays for event capture and enhanced situational awareness“, *IEEE Intelligent Systems*, vol. 20, no. 5, pp. 58–66, 2005.

[78] M. El-Helaly and A. Amer, „Synchronization of processed audio-video signals using time-stamps“, vol. 6, pp. VI–193, 2007.

[79] T. Wang, Z. Zhu, and C. N. Taylor, „Multimodal temporal panorama for moving vehicle detection and reconstruction“, pp. 571–576, 2011.

[80] M. Soleymani, J. Lichtenauer, T. Pun, and M. Pantic, „A multimodal database for affect recognition and implicit tagging“, *IEEE transactions on affective computing*, vol. 3, no. 1, pp. 42–55, 2011.

[81] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, „Multimodal machine learning: A survey and taxonomy“, *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 2, pp. 423–443, 2018.

[82] S. Petridis and M. Pantic, „Audiovisual discrimination between speech and laughter: Why and when visual information might help“, *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 216–234, 2010.

[83] Y. Wu, H. Mao, and Z. Yi, „Audio classification using attention-augmented convolutional neural network“, *Knowledge-Based Systems*, vol. 161, pp. 90–100, 2018.

[84] S. Bunrit, T. Inkian, N. Kerdprasop, and K. Kerdprasop, „Text-independent speaker identification using deep learning model of convolution neural network“, *International Journal of Machine Learning and Computing*, vol. 9, no. 2, pp. 143–148, 2019.

[85] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu, „Automatic traffic surveillance system for vehicle tracking and classification“, *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 2, pp. 175–187, 2006.

[86] Y. Zhou, H. Nejati, T.-T. Do, N.-M. Cheung, and L. Cheah, „Image-based vehicle analysis using deep neural network: A systematic study“, pp. 276–280, 2016.

[87] Z. Wang, X. Miao, Z. Huang, and H. Luo, „Research of target detection and classification techniques using millimeter-wave radar and vision sensors“, *Remote Sensing*, vol. 13, no. 6, p. 1064, 2021.

[88] Y. Chen, C. Wang, Y. Zhou, *et al.*, „Research on multi-source heterogeneous big data fusion method based on feature level“, 2023.

[89] Y. Li, J. Ren, Y. Wang, G. Wang, X. Li, and H. Liu, „Audio–visual keyword transformer for unconstrained sentence-level keyword spotting“, *CAAI Transactions on Intelligence Technology*, vol. 9, no. 1, pp. 142–152, 2024.

[90] X. Liu, Q. Li, J. Liang, *et al.*, „Advanced machine learning methods for autonomous classification of ground vehicles with acoustic data“, vol. 12113, pp. 524–533, 2022.

[91] G. Potamianos, C. Neti, G. Gravier, A. Garg, and A. W. Senior, „Recent advances in the automatic recognition of audiovisual speech“, *Proceedings of the IEEE*, vol. 91, no. 9, pp. 1306–1326, 2003.

[92] V. Shah, A. Aggarwal, and N. Chaubey, „Alert fusion of intrusion detection systems using fuzzy dempster shafer theory“, *Journal of Engineering Science and Technology Review*, vol. 10, no. 3, pp. 123–127, 2017.

[93] J. Gao, P. Li, Z. Chen, and J. Zhang, „A survey on deep learning for multimodal data fusion“, *Neural Computation*, vol. 32, no. 5, pp. 829–864, 2020.

[94] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency, „Tensor fusion network for multimodal sentiment analysis“, *arXiv preprint arXiv:1707.07250*, 2017.

[95] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian, „Deep modular co-attention networks for visual question answering“, pp. 6281–6290, 2019.

[96] K. Bayoudh, R. Knani, F. Hamdaoui, and A. Mtibaa, „A survey on deep multimodal learning for computer vision: Advances, trends, applications, and datasets“, *The Visual Computer*, vol. 38, no. 8, pp. 2939–2970, 2022.

[97] R. P. Loce, E. A. Bernal, W. Wu, and R. Bala, „Computer vision in roadway transportation systems: A survey“, *Journal of Electronic Imaging*, vol. 22, no. 4, pp. 041 121–041 121, 2013.

[98] M.-T. Yang, R.-K. Jhang, and J.-S. Hou, „Traffic flow estimation and vehicle-type classification using vision-based spatial–temporal profile analysis“, *IET Computer Vision*, vol. 7, no. 5, pp. 394–404, 2013.

[99] V. Mandal, A. R. Mussah, P. Jin, and Y. Adu-Gyamfi, „Artificial intelligence-enabled traffic monitoring system“, *Sustainability*, vol. 12, no. 21, p. 9177, 2020.

[100] W. Kang, D. Kim, and J. Park, „Dms: Dynamic model scaling for quality-aware deep learning inference in mobile and embedded devices“, *IEEE Access*, vol. 7, pp. 168 048–168 059, 2019.

[114] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, „Scikit-learn: Machine learning in Python“, *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[119] A. Savitzky and M. J. Golay, „Smoothing and differentiation of data by simplified least squares procedures.“, *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[120] R. Büssow, „An algorithm for the continuous morlet wavelet transform“, *Mechanical Systems and Signal Processing*, vol. 21, no. 8, pp. 2970–2979, 2007.

[121] L. Sheppard, A. Stefanovska, and P. McClintock, „Testing for time-localized coherence in bivariate data“, *Physical Review E*, vol. 85, no. 4, p. 046 205, 2012.

[122] C. Schörkhuber, „Constant-q transform toolbox for music processing“, [Online; accessed 13.06.2024], 2010.

[123] J. Shen, R. Pang, R. J. Weiss, *et al.*, „Natural tts synthesis by conditioning wavenet on mel spectrogram predictions“, pp. 4779–4783, 2018.

Bibliography of Online Sources

[12] R. Girshick, *Fast R-CNN Quick Overview*, https://blog.paperspace.com/faster-r-cnn-explained-object-detection/, [Online; accessed 13.06.2024].

[13] R. Kundu, *YOLO: Algorithm for Object Detection Explained*, https://www.v7labs.com/blog/yolo-object-detection, [Online; accessed 13.06.2024].

[14] A. Developers, *How single-shot detector (SSD) works?*, https://developers.arcgis.com/python/guide/how-ssd-works/, [Online; accessed 13.06.2024].

[44] Stevens and Davis, *MEL-scale*, https://www.sfu.ca/sonic-studio-webdav/handbook/Mel.html, [Online; accessed 13.06.2024].

[47] J. Brownlee, *A Gentle Introduction to the Rectified Linear Unit (ReLU)*, https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/, [Online; accessed 13.06.2024].

[60] IBM, *Random Forest Algorithm*, https://www.ibm.com/topics/random-forest, [Online; accessed 13.06.2024].

[66] Prof. Dr. Peter Roßbach, *NN vs Random Forest Algorithm*, https://blog.frankfurt-school.de/wp-content/uploads/2018/10/Neural-Networks-vs-Random-Forests.pdf, [Online; accessed 13.06.2024].

[75] F. Chollet *et al.*, *Keras*, https://github.com/fchollet/keras, 2015.

[101] Python TM, *Python 3.11.0*, https://www.python.org/downloads/release/python-3110/, [Online; accessed 13.06.2024].

[102] Khadas_VIM3, *Khadas VIM3*, https://www.khadas.com/vim3, [Online; accessed 13.06.2024].

[103] Bernard Gruppe ZT GmbH., *Bernard Gruppe*, https://www.bernard-gruppe.com/, [Online; accessed 13.06.2024].

[104] DrivenData, *Cookiecutter Data Science*, https://drivendata.github.io/cookiecutter-data-science/, [Online; accessed 13.06.2024].

[105] Data Version Control, *Data Version Control*, https://dvc.org/, [Online; accessed 13.06.2024].

[106] GitLab, *GitLab*, https://about.gitlab.com/, [Online; accessed 13.06.2024].

[107] SciPy, *scipy.signal.savgol_filter*, https://pypi.org/project/noisereduce/, [Online; accessed 13.06.2024].

[108] SciPy, *Savitzky Golay Filtering*, https://scipy.github.io/old-wiki/pages/Cookbook/SavitzkyGolay, [Online; accessed 13.06.2024].

[109] B. McFee, C. Raffel, D. Liang, *et al.*, *librosa: Audio and music signal analysis in Python*, https://librosa.org/doc/main/index.html, [Online; accessed 13.06.2024], 2015.

[110] SciPy, *scipy.signal.spectrogram*, https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.spectrogram.html, [Online; accessed 13.06.2024].

[111] scikitlearn, *sklearn.preprocessing.minmax_scale*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html, [Online; accessed 13.06.2024].

[112] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[113] PyTorch Team, *PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation*, https://pytorch.org/assets/pytorch2-2.pdf, [Online; accessed 13.06.2024]. doi: 10.1145/3620665.3640366.

[115] I. Shafkat, *Intuitively Understanding Convolutions for Deep Learning*, https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1, [Online; accessed 13.06.2024].

[116] N. Vishwakarma, *What is Adam Optimizer?*, https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/, [Online; accessed 13.06.2024].

[117] Land Tirol, Abteilung Landesstraßen und Radwege, *Webcam B 182 Brennerstraße*, https://www.tirol.gv.at/verkehr/strassenbau-und-strassenerhaltung/webcams/webcams-bezirk-innsbruck-land-mit-stadt/b-182-brennerstrasse/, [Online; accessed 13.06.2024].

[118] T. Sainburg, *Noisereduce 3.0.0*, https://pypi.org/project/noisereduce/, [Online; accessed 13.06.2024].

[124] L. Roberts, *Understanding the Mel Spectrogram*, https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53, [Online; accessed 13.06.2024].