Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Audio-Based Vehicle Recognition

Masters's thesis

Bc. Andrii Yermakov

Study program: Open Informatics
Supervisor: Ing. Vojtech Franc, Ph.D.

This work was supported by the Technology Agency of Czech Republic
project FW03010571 / MultiDO

Prague, May 2022

Thesis Supervisor:
Ing. Vojtech Franc, Ph.D.
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo namesti 13
12135 Praha 2
Czech Republic

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Yermakov** Jméno: **Andrii** Osobní číslo: **478153**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Rozpoznávání vozidel z audia

Název diplomové práce anglicky:

Audio-based vehicle recognition

Pokyny pro vypracování:

The goal of the project is to design and implement a system for audio-based vehicle recognition. The input is an audio recorded by a low-end consumer device with an inbuilt microphone placed on the side of road. The system outputs statistics about passing vehicles like their counts, direction of movement, categories, and other attributes recognizable from audio. The system will be trained on examples prepared by an imperfect commercial system recognizing the vehicles based on video only. The performance of the designed audio-system will be statistically evaluated on real data and compared with the performance of the video-based system.

Seznam doporučené literatury:

- Slobodan Djukanoivic et al. Neural network-based vehicle counting in low-to-moderate traffic flow. CoRR. 2021.
- Slobodan Djukanović et al. Robust Audio-Based Vehicle Counting in Low-to-Moderate Traffic Flow. 31st IEEE Intelligent Vehicles Symposium, October 2020, Las Vegas, NV, United States.
- Alicja Wieczorkowska et al. Spectral features for audio based vehicle and engine classification. Journal on Intelligent Information Systems. 50:265-290. 2018.
- Rejla Arandjelovic et al. Look, listen, and learn. ICCV 2017.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Vojtěch Franc, Ph.D. Strojové učení FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **02.02.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

Ing. Vojtěch Franc, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Declaration

I hereby declare I have written this master's thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2022

.....
Andrii Yermakov

Abstract

In this thesis, we have designed and implemented an audio-based vehicle recognition module that takes an audio signal and outputs statistics about passing vehicles. The proposed method is versatile, besides the number of passing vehicles, it can be configured to predict an arbitrary event from the audio, such as directions of movement and vehicle types. The proposed approach is based on a classical convolutional neural network architecture such as ResNet and a number of methods used to preprocess the audio signal. We learned the proposed neural network in a semi-supervised fashion using imprecise annotations obtained automatically from a commercial vision-based vehicle recognition module. We propose a novel method that aggregates predictions from overlapped time windows. The method helps to handle events that occur on the boundary of fixed-size time windows that are processed by the neural network. The performance of the proposed audio-based module in the vehicle counting task is comparable to that of the vision-based module tested under good light conditions with mean RVCE of 7% and 3%, respectively.

Keywords: Vehicle recognition, vehicle counting, traffic estimation, multimodal data, audio processing, deep neural networks.

V této práci jsme navrhli a implementovali modul pro rozpoznávání vozidel na základě zvuku, který přijímá zvukový signál nahraný na okraji silnice a poskytuje statistické údaje o projíždějících vozidlech. Navržená metoda je univerzální, kromě počtu projíždějících vozidel ji lze nakonfigurovat tak, aby ze zvuku odhadovala libovolnou událost, například směr pohybu a typy vozidel. Navrhovaný přístup je založen na použití konvoluční neuronové sítě s klasickou architektuou, jako je ResNet, a na řadě metod použitých k předzpracování zvukového signálu. Neuronovou síť jsme naučili s využitím nepřesných anotací získaných automaticky z komerčního modulu pro rozpoznávání vozidel na základě obrazu. V práci také navrhujeme novou metodu, která agreguje předpovědi z překrývajících se časových oken. Metoda pomáhá zpracovávat události, které se vyskytují na hranici časových oken pevné velikosti, jež jsou vstupem neuroné sítě. V úloze počítání vozidel je výkonnost navrženého modulu založeného na zvuku srovnatelná s výkonností modulu založeného na vidění, testovaného za dobrých světelných podmínek, kdy zvukový modul dosahuje průměrnou hodnotou RVCE 7% zatímco video modul 3%.

Klíčová slova: Rozpoznávání vozidel, počítání vozidel, odhad dopravy, multimodální data, zpracování zvuku, hluboké neuronové sítě.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Vojtěch Franc, who guided me through this research project and provided me with valuable feedback. Without his advice and help, achieving the desired results would have taken much longer.

I am very grateful to the Eyedea Recognition team [1] who spent many hours and resources on data acquisition and manual annotation and also provided me with financial support and server infrastructure to train neural network models.

The completion of this work would have been more difficult, confusing, unpleasant, and less productive without the visualization tools developed by Weights & Biases [2]. I appreciate that they made them available for free for academic purposes.

I gratefully acknowledge that my work was supported by the Technology Agency of Czech Republic project FW03010571 / MultiDO.

[1] https://eyedea.cz/
[2] https://wandb.ai/

List of Abbreviations

CBCE	Class-Balanced Cross-Entropy
CE	Cross-Entropy
CNN	Convolutional Neural Network
DNN	Deep Neural Networks
EM	Expectation–Maximization
LSTM	Long Short-Term Memory
MMR	Make and Model Recognition
PCP	Pitch Class Profile
RVCE	Relative Vehicle Counting Error
STE	Short-Term Energy
STFT	Short-Time Fourier Transform

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

Traffic monitoring systems play an important role in increasing the safety of existing road infrastructure. The most common system is based on vehicle recognition from visual images. A vision-based system can be extended with an audio input that helps or corrects an image classifier when it makes mistakes, for instance, due to occlusions, changes in illumination, bad weather conditions, etc. Another potential usage of the audio recognition system is to detect failures of the vision system and vice versa.

Data acquisition faces a legislative challenge when it comes to collecting videos in public places using cameras to monitor traffic situation. It becomes an ethical question when authorities track subjects using cameras placed in every corner of the city. Fortunately, for some purposes, there is another solution; for example, the audio signal from a microphone can be used to monitor traffic without disturbing drivers or people passing by the camera. In addition, microphones are cheaper and work under any light conditions.

The emergence of deep neural networks (DNN) has pushed the boundaries of state-of-the-art results in different problems like classification, detection, or segmentation. In recent years, much attention has been paid to solving computer vision problems. The computer vision community has developed techniques that work best in this area: network architectures, image preprocessing, data augmentation, etc. A much smaller community of researchers working in the audio domain have attempted to adapt the knowledge gained from solving computer vision tasks. The standard approach to classify the audio signal relies on representing the audio signal by spectrograms. However, using spectrograms and treating them as images, as classical CNN architectures do, assumes that both axes represent spatial information rather than time and frequency. Having more knowledge of the problem will help us develop better approaches that work best for the audio domain.

1.2 Goals

The goal of the project is to design and implement a system for audio-based vehicle recognition. The input is an audio recorded by a low-end consumer device with an inbuilt microphone placed on the side of road. The system outputs statistics about passing vehicles like their counts, direction of movement, and vehicle types. The system will be trained on imperfect annotations obtained using commercial vision-based vehicle recognition system. The performance of the designed audio-system will be statistically evaluated on real data and compared with the performance of the vision-based system.

1.3 Contributions

In this thesis, we design and implement audio-based vehicle recognition software that takes an audio signal recorded by a low-end consumer device and outputs statistics about passing vehicles, such as counts, directions of movement, and vehicle types. The proposed system is based on a neural network that is learned in a semi-supervised manner. The annotation of the audio signal used to train the network does not involve a human annotator. Instead, we propose a method to obtain the audio annotations from weak labels produced by an imperfect pre-trained vision-based system. Inference in the proposed system involves splitting the long audio record into a sequence of fixed-size time windows, predicting the audio events of each window independently, and aggregating the individual predictions into the output summary statistics about the record. We also propose a novel method to aggregate the predictions from a sequence of overlapping time windows. The method helps to solve the situation when the audio events occur close to the boundary of the time window. We carried out a large number of experiments to empirically measure the influence of individual components on the proposed architecture. We use the results obtained to identify the best configuration of the prediction model and to statistically evaluate its performance. The performance of the proposed audio system in predicting vehicle counts is comparable to that of the vision-based system. We also discuss the best practices that can be useful for the design of other audio-based systems.

Chapter 2

State-of-the-art

To our knowledge, the first attempt to address the problem of vehicle classification from an audio signal using ANN was made in 1998 [19]. They were motivated by the benefits of the acoustic approach, which works in low-light conditions, such as fog or at night, when visual systems fail. They showed that from the acoustic signature of vehicle passes, it is feasible to reliably identify vehicle categories such as: 1. busses and lorries (trucks); 2. small and large saloons (sedans); 3. various types of motorcycles; 4. lightweight vehicles and vans.

In 2013, the authors of [21] described a method for counting vehicles and estimating the direction of movement. Their method uses three equidistantly spaced microphones to measure the delay in sound which allows to infer the angle of the sound waves. Despite the fact that the method uses specialized hardware, it still suffers from the wind and noise of other vehicles. They tested their setup on two-lane roads and observed that the approach does not work well in situations where two cars meet in front of microphones at the same time.

Researchers from Graz University of Technology together with Klagenfurt University proposed an audio visual system for intelligent traffic monitoring that involves the detection and classification of passing vehicles [2]. They train an audio classifier from manually labeled examples to predict whether the scene contains no vehicles, a single vehicle, or multiple vehicles. The audio classifier then serves as a teacher for a visual classifier that is used to find blobs in the scene that correspond to the passing vehicles. The authors assume that the audio classifier can resolve typical ambiguities in distinguishing the vehicle types, e.g. a car versus a truck, which are hard to resolve for visual classifiers but are easy for acoustic classifiers, and hence they train two different vision-based detectors, one for tucks and one for cars. To obtain the final predictions, they aggregate the outputs of both the audio and visual classifiers, which works better than using either of the two modalities independently.

The authors of [10] addressed the problem of counting vehicles in the audio signal in low to moderate traffic scenarios. Their method is based on modeling the so-called vehicle-to-microphone distance function using the Support Vector Regression (SVR) algorithm. They estimate the number of vehicles passing by detecting local minima in the vehicle-to-microphone distance function. A limitation of the method is the assumption of one-directional traffic and a single vehicle passing in time. In addition, training of the SVR model requires exact annotations of the audio with the time the vehicle passes the microphone. Another issue is related to the appropriate configuration of the detection threshold, which is resolved in the follow-up paper of the authors [11] where the SVR is replaced by a fully connected two-stage ANN. The ANN model speeds up the computations, and, in contrast to the previous paper, it estimates the number of vehicles directly from the predicted distance, without detecting local minima in a postprocessing step.

Another work addressing audio-based vehicle classification was proposed in [26]. The proposed method classifies passing vehicles into categories such as car, tractor, motorcycle, van, small / large truck, and bus. The authors use manually designed features extracted from the raw

audio signal and compare several classification models, including SVM [14], random forests [4] and ANN classifiers. They found that the deep ANN consistently outperforms other classification models.

In [6], a novel neural network architecture and audio features were proposed for vehicle type identification. The authors extract audio characteristics with different methods, such as the mel frequency cepstrum coefficient (MFCC), the pitch class profile (PCP), and the short-term energy (STE), and consequently fuse all the characteristics together to the novel feature representation. The features are then used as input to the novel hybrid neural network architecture, which consists of a canonical CNN followed by long-short-term memory (LSTM) units. The authors collected a data set for 10 different types of vehicle audio, such as: YUTONG bus, JAC truck, HAVAL H5, SGMW, RANGE ROVER, JETTA, HYUNDAI IX35, BYD e6, JAC V7 and train. Each class contains 100 samples. The 3 second long audio sampled at 44,100 Hz frequency contains the vehicle's engine, brake, tire friction, and horn sounds. The authors reported the classification accuracy of 100%. However, it is not known how the system will work in the presence of a large number of classes of similar sounds and a small number of representative samples.

In [12], the authors classify traffic flow into two categories: free, when traffic flows without stopping, and congested, characterized by vehicles passing at very low speeds and stopping frequently. The authors propose to represent 20-second long windows of the audio signal using features constructed from MFCC. The features then serve as input to the Random Forest classifier, which predicts traffic flow as free or congested. They used a data set that consists of 13 video files, 7 of which were recorded directly by the authors (local instances), and the other 6 were downloaded from YouTube (world instances). The records were labeled manually on the basis of visually inspecting the traffic flow. The authors compared the model performance in local vs. local, local vs. world, and world vs. local settings. That means, they trained the model on local instances and tested on world instances and other way around. This way they checked how the system performs in unknown and distinct places. The average classification accuracy for local vs. world was 89% while that for world vs. local was 93%.

The authors of [1] presented a data set consisting of 15,706 2-second audio clips recorded with high-quality and medium-quality microphones. Because the data set consists of 2-second long videos, there is no need to solve the problem of detecting and localizing a passing car. Audio clips were recorded at four different locations including three locations in the city and one on a country road located in and around Ilmenau, Germany. Recordings were made during dry and wet weather conditions on roads with three different speed limits: 30, 50, and 70 km/h, and also at two different times of the day: morning and afternoon. For each microphone type, they recorded around 2.5 hours of footage with a total of 4718 annotated vehicle passages in which there are: 3903 cars, 511 trucks, 53 busses, and 251 motorcycles. For the class representing the absence of vehicles passing by the microphone, they added audio clips containing typical background sounds along the road without any audible vehicle sounds. The authors evaluated how four different CNN architectures perform in the vehicle type classification problem and in the direction of movement estimation. They achieved high accuracy in both tasks.

The authors of [25] presented a method to align the visual and acoustic signatures of moving vehicles using a multimodal temporal panorama. In the next paper [24] they applied the method together with the SVM classification algorithm and conducted a comprehensive study of how visual and audio features, as well as their combinations, influence the precision of two classification tasks. In the first task, they classified vehicles into 5 classes: sedan, van, pickup, truck, and bus, while in the second task they had three classes corresponding to the size of a vehicle, such as light, medium, and heavy. The authors evaluate the influence of different types of sound-based and vision-based features for each task. They concluded that for the first task, vision-based features have better performance because vehicles were labeled based on their visual appearance. However, in the second task sound-based features show better results because

vehicles were labeled based on their audio signatures. Finally, they showed that combining features from different modalities can further improve the model performance in both tasks.

In summary, from 1998 [19], different attempts were made to classify vehicle types using an audio signal [2, 26, 1, 6] and to estimate the number of vehicles passing by the microphone [21, 10, 11, 12]. Most of the works consider data collected by a single recording device, usually using dedicated hardware, which was placed at a single fixed location. In contrast, in this thesis we consider vehicle recognition in audio signal recorded at multiple different locations using multiple low-end consumer devices. Our setup is more realistic, however, at the same time much more challenging. The challenge of using different recording devices has previously been recognized [1].

Chapter 3

Proposed method

3.1 Problem definition

The input is an audio signal recorded by a low-end consumer device that is placed on the side of the road. The task is to learn an audio module that takes an audio record and outputs statistics about passing vehicles such as their count, direction of movement, categories, and other attributes recognizable from audio. The length of the input record can range from several minutes to several hours. The audio module should be trained on examples prepared by an imperfect system that recognizes vehicles based only on video [1].

3.2 Data set

We use a data set captured by a simple setup that consists of a low-end consumer device with a camera and an inbuilt microphone mounted on a tripod [2]. The tripod was placed on the side of a road so that the camera could capture the front of approaching vehicles and the rear part of vehicles going in the opposite direction. The distances between the recording device, the side line of the road and the ground are variable. The recording setup is shown in Figure 3.1.

The data were recorded simultaneously by 3 different devices: Sony RX100 camera, iPhone XS, Mobius camera. Figure 3.2 shows sample frames captured by the three devices.

The database consists of 102 audio-visual recordings. The videos were recorded in 35 different locations around Prague that are restricted to two-lane roads (one lane in each direction). The recordings are around 20 minutes long, which is equivalent to 34.5 hours of raw footage in total. The basic statistics of the database are summarized in Table 3.1.

There is a large variation in the traffic density, vehicle occlusions, weather conditions and quality of the road surface. The number of vehicle passages ranges from hundreds to thousands per hour. The weather conditions range from sunny, windless to rainy and windy weather. The road surface is covered with asphalt, but its state is largely variable. In each recording, the same number of vehicle passages can be deduced from the video and the audio. That is, we do not have recordings where we can only hear the vehicles, but they never appear in the video, or vice versa.

recording length		# passages per recording		# locations
average/std	total	average/std	total	
20.3/2.1 [min]	34.5 [h]	166/95	16917	35

Table 3.1: Basic statistics of the recordings in the used audio-visual road traffic database.

[1] We use a vision-based module for traffic analysis provided by Eyedea Recognition (www.eyedea.cz).

[2] The data set was captured between Autumn 2021 and Spring 2022 by Eyedea Recognition.

Figure 3.1: Setup used to capture the audio-visual records of road traffic.

(a) Sample from iPhone

(b) Sample from Mobius

(c) Sample from RX100

Figure 3.2: Example of frames captured by the 3 recording devices used.

3.3 Challenges

3.3.1 Environmental noise

Videos are recorded in unconstrained environments, which does not guarantee that audio quality will be free from defects, such as ambient noise. The most common situation is a disruption due to weather conditions, such as wind or rain. In order to work properly under such conditions, the recognition system has to be trained from the samples containing an appropriate noise or should be properly augmented. Another type of noise is background noise that comes from objects that are contained in the data set at the training stage. For example, when placing a recognition system near a railroad or airport, one should verify how events such as a passing train or flying plane will influence the system.

3.3.2 Out-of-distribution sound events

Systems that were trained on audio signatures from one distribution may not behave well in the presence of unknown sounds. The recognition accuracy of a system can be corrupted by sound events that do not occur in the training set. If we train the system on samples of passing sedans or trucks, but are interested in the number of passing tractors or tanks, we may face the problem of presenting unknown audio signatures to the system, which will lead to unexpected results. Another example of out-of-distribution events is when drivers artificially adjust the loudness of their vehicles.

3.3.3 Sound superposition

Detection and annotation of events that overlap in time, such as two cars passing by the microphone in different lanes, is extremely challenging for a human being. The problem becomes easier when we consider videos instead of just a raw audio signal. Nevertheless, another problem arises which is the localization of such events.

3.3.4 Event localization ambiguity

It is difficult to determine the start and end times of events, such as a car passage. The onset and offset duration of such an event are ambiguous and differ between cars, which makes it complicated to localize it. In addition to the fact that manual annotation is laborious work, each human annotator will have different judgments on how to localize such events due to differences in the perception of the audio signal. Automatic annotation process using thresholding is also complicated in this scenario due to the differences in energies for each event as well as the disruption by background noises.

3.3.5 Traffic congestion

While vision-based systems suffer from low-light conditions, audio-based systems have similar drawbacks when it is hard to recognize a vehicle due to a quiet passage. There are a few situations where it is hard or even impossible to distinguish a single vehicle passage such as when cars move slowly or do not move at all due to traffic congestion or when the system is placed on the crossroad or pedestrian crosswalk. In such places, it is very complicated not only to localize such events but also to distinguish the audio signature of the event from all other sounds.

3.3.6 Differences in recording hardware

Just like different cameras capture photos using their unique sensors and post-processing algorithms, different audio recording hardware processes the sound in a different manner. The output of the recording is stored as the waveform, which has its own unique signature for each device. While it is not a problem for human perception to recognize sound recorded using different devices, it poses a problem for machine learning algorithms.

3.4 Obtaining video annotations

In this work, we consider data captured from two modalities: video and audio. The video modality is used to extract weak annotations of the audio signal, which then serve as the training set for the audio module we aim to train. To this end, we use commercial software [3] to extract information about passing vehicles from the video. Given a video record, the vision-based module produces a CSV file with each row representing a track of a single passing vehicle. Columns of the CSV file corresponding to attributes extracted from the tracks, out of which we used the following attributes: start and end timestamps of the vehicle track, direction of movement (outgoing, incoming), and vehicle type (car, van, truck, motorcycle, etc.).

The annotations produced by the vision-based module are noisy. In order to evaluate the accuracy of the vision-based module, all recordings in the database were manually annotated. To facilitate manual annotation, we cut each video into shorter clips that are on average 40 seconds long. The human operator plays the audible clip and annotates the number of vehicles passing in both directions.

[3] We use Make and Model Recognition (MMR) software provided by Eyedea Recognition.

3.5 Converting video annotations to audio annotations

To train the audio module, we need the audio signal to be annotated by moments in time the vehicles are passing the camera. Vehicle passage time is defined as the time when the vehicle is closest to the camera. To obtain the audio annotations, we use the output of the vision-based module, which for each passing car outputs a time interval determined by the start and end timestamps of the vehicle tracking (see the previous section). We propose an automated procedure to convert the time intervals from the vision-based module to the single points in time that correspond to the vehicle passages. The procedure is based on the estimate of a time shift between the tracking timestamps and the vehicle passage time. Because we do not restrict ourselves to roads with one-way traffic, we need to consider two possible time shifts: first, the time shift between the moment when the incoming car disappears from the scene and the moment when it passes the recording device, and second, the scenario which happens on the opposite lane, when we first hear the passing vehicle, and then the camera captures it. These two time shifts are illustrated in Figure 3.3.

To obtain the time shifts, we use the simplifying assumption that all cars travel at approximately the same speed. Under this assumption, we need to estimate two positive scalars for each video record, which correspond to the time shifts for the incoming and outgoing direction, respectively. The time shifts are then estimated independently of each other using an audio signal and annotations of the direction and time when the vision-based module starts or stops tracking the vehicle. To find these two values robustly, we use the following procedure. We perform the Short-Time Fourier Transform (STFT) on the audio signal. By squaring the magnitude of the STFT we obtain a spectrogram corresponding to a visual representation of the spectrum of frequencies of a signal as it varies in time. By summing rows of the spectrogram, we obtain the sum of energies across all frequencies $\mathbf{E} \in \mathbb{R}^N$, where N is the number of discrete time steps. With the assumption that the highest energy is reached when the car passes the camera, we can find the shifts for incoming Δ^*_{in} and outgoing Δ^*_{out} vehicles by solving the following optimization problem:

$$\Delta^*_{\text{in}} = \frac{l}{2} + \underset{\Delta_{\text{in}} \in \mathbb{N}}{\arg\max} \sum_{t \in T_{\text{in}}} \sum_{i=t+\Delta_{\text{in}}}^{t+\Delta_{\text{in}}+l} \mathbf{E}_i$$

$$\Delta^*_{\text{out}} = -\frac{l}{2} - \underset{\Delta_{\text{out}} \in \mathbb{N}}{\arg\max} \sum_{t \in T_{\text{out}}} \sum_{i=t-\Delta_{\text{out}}-l}^{t-\Delta_{\text{out}}} \mathbf{E}_i$$

where l is the length of the window, T_{in} and T_{out} are sets of timestamps for incoming and outgoing vehicles when they leave or enter a scene, respectively.

Ideally, we should model the shift for each passing vehicle separately. However, the constant speed assumption turned out to provide sufficiently accurate estimates of the passage times. We note that for audio module training, we need the annotation of the number of vehicle passages in windows that are several seconds long; hence, if the vehicle passage times estimated from the video annotation does not fall outside the boundary of the input window, it still counts as a correct annotation. The other source of problems comes from the fact that the video tracker misses some vehicles when the input video is of low quality or the vehicles are occluded due to heavy traffic. The proposed procedure cannot deal with this type of error.

3.6 Alignment of signals from multiple devices

We noticed that the quality of the annotations obtained from the vision-based module differ significantly across the recording devices used. We objectively measure the quality of the video annotations using the manual annotations (see Section 4.15 for details). For that reason, we

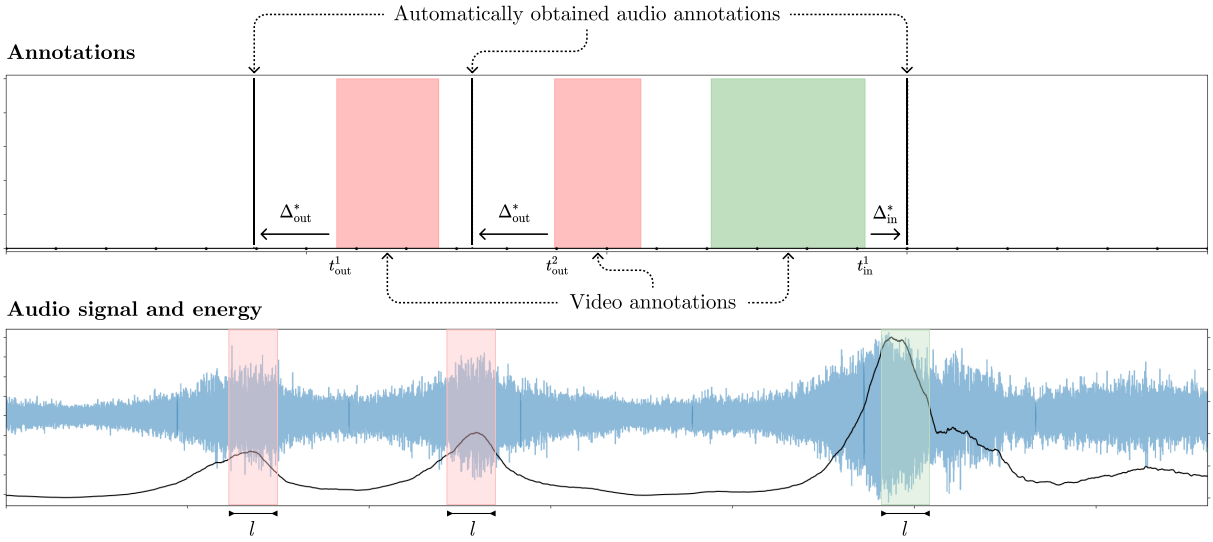


Figure 3.3: Visualization of the method proposed in Section 3.5 to convert the interval annotation from the vision-based module into the point-in-time annotation of the audio signal. The top image shows annotations obtained from the vision-based module as rectangles representing the time period during which the vehicle is appeared in the scene. Red and green colors describe the direction of the vehicle as outgoing and incoming, respectively. The vertical black lines are the audio annotations obtained by the proposed method. The bottom image visualizes the audio signal, blue color, and the extracted energy shown as black curve.

selected one device as a reference, namely RX100, for which the vision-based module provides consistently accurate annotations. The video annotations from RX100 are converted to the audio annotations (by the method described in the previous section) and then the obtained annotations are transferred to the audio signals of the remaining two devices, i.e. Mobius and iPhone. The transfer is achieve by aligning the audio signals of the three devices by simply cross-correlating two audio arrays and finding the displacement that maximizes the cross-correlation. The displacement is used to shift the signal such that it is aligned with the signal of the reference device. Figure 3.4 illustrates the signals before and after the alignment.

In summary, using the described procedure, we obtain three different audio signals with exactly the same annotation. This setup allows a fair comparison of the quality of the audio signals.

3.7 Feature extraction

The purpose of the feature extraction process is to extract task-relevant information from a long audio signal of variable length and to represent it in a compact fixed-size form. We propose splitting the audio signal into time windows of a fixed size and representing each time window independently by features. The feature representation of the individual time windows serves as an input of the CNN which is in the core of the audio module, i.e. the predictions are made for the time windows independently and consequently aggregated to the final prediction for the whole audio record. The optimal length of the time window (in the text reported in seconds) was found experimentally (see Section 4.12).

It is common to use a technique such as short-time Fourier transform (STFT) to analyze the signal. It shows how the frequency content of a signal changes over time. We experimented with two types of feature that can be obtained using the STFT, namely, we use spectrograms and mel spectograms that are described below.

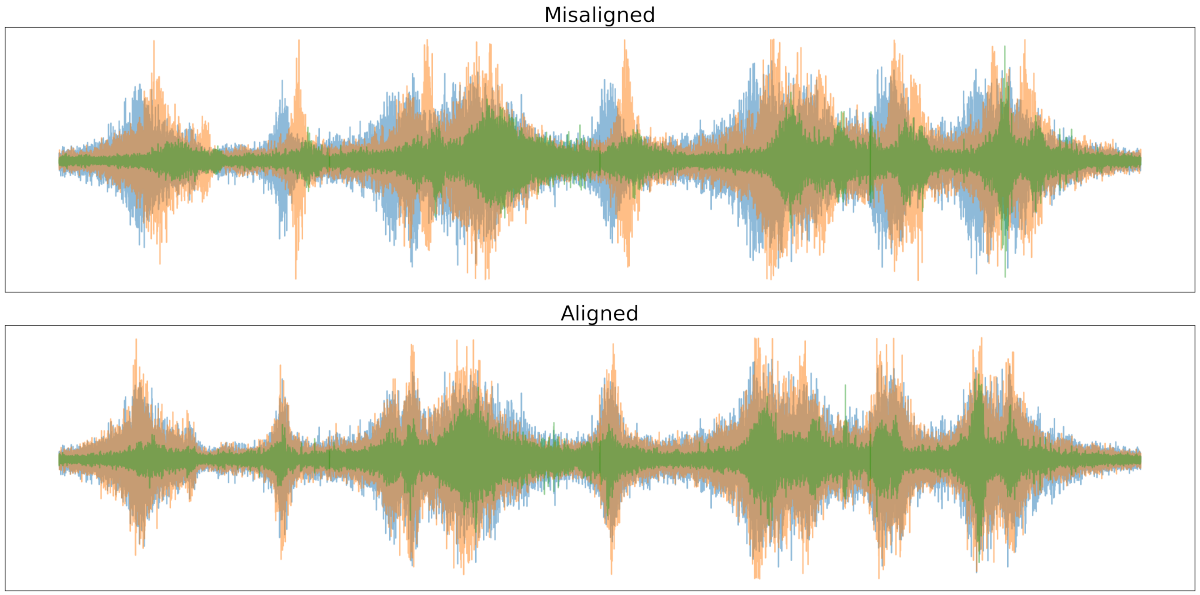


Figure 3.4: Illustration of the signal alignment procedure where the audio from RX100 (blue) is used as the reference and the audio from iPhone (green) and Mobius (orange) are the target signals to be aligned.

3.7.1 Short-time Fourier transform

We convert a one-dimensional audio signal to two-dimensional images using the STFT. Each dimension of such an image corresponds to features in either the time domain or the frequency domain. The STFT algorithm has two parameters that we can tune. The first parameter is the **frame length**, it defines how many samples will be used to analyze the frequency content of some time segment. For example, the typical sampling frequency is 44,100 Hz and commonly used frame length is 2,048 samples. For each STFT column we will have a frequency content of 46 millisecond time interval. The second parameter, which is called the **hop length**, defines how many samples are between STFT columns. For examples, given the same sampling rate and the same frame length, with hop length of 1,024 samples, we will have 50% of overlap between frames. Mathematically, STFT is written as follows:

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-i\omega t} \, dt$$

where $w(\tau)$ is the window function, commonly a Hann window [3] or Gaussian window centered around zero, $x(t)$ is the signal value in time t. $X(\tau, \omega)$ is a complex-valued number where real and imaginary part represent an amplitude and a phase of a signal over time τ and frequency ω.

3.7.2 Spectrogram

The most common machine learning models work with read-valued inputs. Fourier transform gives us a complex-valued result, so the phase information can be discarded and only the magnitude of the spectrum is used. After the transfomation, we get linearly spaced frequencies. The sound events we are interested in have more information at lower energies. It also seems natural to apply the logarithmic transformation to magnitudes to obtain log magnitude spectrograms. Authors of [7] argue that logarithmic scaling of the magnitude result in significant improvements in accuracy.

3.7.3 Mel spectrogram

Another commonly used features are mel spectrogram [23]. Mel spectrograms are obtained by applying STFT on a signal with a subsequent rescaling of the squared magnitude of STFT to the mel scale [23]. Wikipedia defines the mel scale as *a perceptual scale of pitches judged by listeners to be equal in distance from one another*. For this reason, there is no single mel-scale formula and different rescaling were proposed by many authors in the 20th century. By default, PyTorch implementation of mel rescaling uses the formula from [27]:

$$\text{Mel}(f) = 2595 \log_{10}(1 + \frac{f}{700})$$

3.8 Data augmentations and features normalization

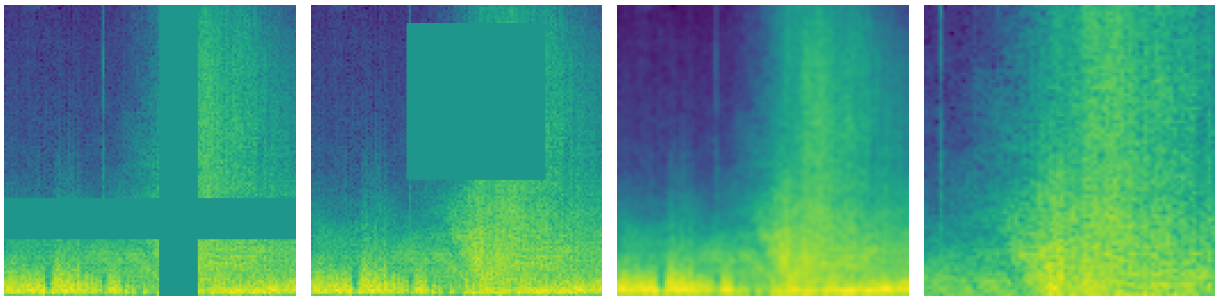
Given a raw audio signal, we split it sequentially into fixed-length audio windows. For example, if we are given 30 minutes video and 6-second long windows, we obtain 300 training samples. Because we work with temporal data, in each epoch we augment audio data by offsetting the signal and extract unseen data that differ only by this offset time which is always smaller than the window length.

When we train neural networks, it is a well-established practice to normalize features and augment them [22, 15] to increase the amount of data by adding slightly modified copies of already existing data. It has been experimentally proven that data augmentation can often compensate for the insufficient amount of training data and improve the generalizing ability of the trained neural network. Formally, most data augmentation methods can be thought of as a transformation of the input sample that preserves its class assignment.

Since we use temporal data, we can apply feature scaling to each image dimension separately. For instance, we can standardize data globally or independently across time or frequency dimension. Feature standardization causes feature values to have zero mean and unit variance.

There are two conceptually different data augmentation techniques: augmentations that are applied directly to the signal and augmentations that are applied to features extracted from the signal. The most common image augmentation techniques include random gaussian blurring, random erasing, random resized crop, random time or frequency masking, etc. Examples of such augmentations are illustrated in Figure 3.5a. Another set of augmentation are applied directly to the raw audio signal, these augmentations are: random colored noise, random gain, random high pass filter, random low pass filter, random pitch shift, etc.

Another feature preprocessing that can be applied is the resize. Sizes of both image dimensions depend on parameters like sampling rate, length of the hop between STFT windows, size of FFT, number of mel filterbanks, etc. To make input images smaller, we can scale them by the standard image resizing technique, i.e. default in PyTorch bilinear interpolation.



(a) Masking (b) Random erasing (c) Gaussian blurring (d) Random resized crop

Figure 3.5: Examples of image augmentations

3.9 Architecture of the proposed audio predictor

3.9.1 Neural network model backbone

One of the most used neural network architectures for image classification today is ResNet [13]. There are three variants of the network that we have considered, namely, ResNet18, ResNet34, and ResNet50. The three variants differ in the depth of the network and thus in the number of trainable parameters which are 11.2M, 21.3M, and 23.6M, respectively. Architectures of three variants of the network are illustrated in Figure 3.6. The architecture mainly consists of convolutional, pooling, and nonlinear layers acting as feature extraction components. At the end of the network, there is a fully-connected layer that gives scores to each class. In the task of vehicle counting, each class represents the number of passing vehicles in a given time interval defined by the window length.

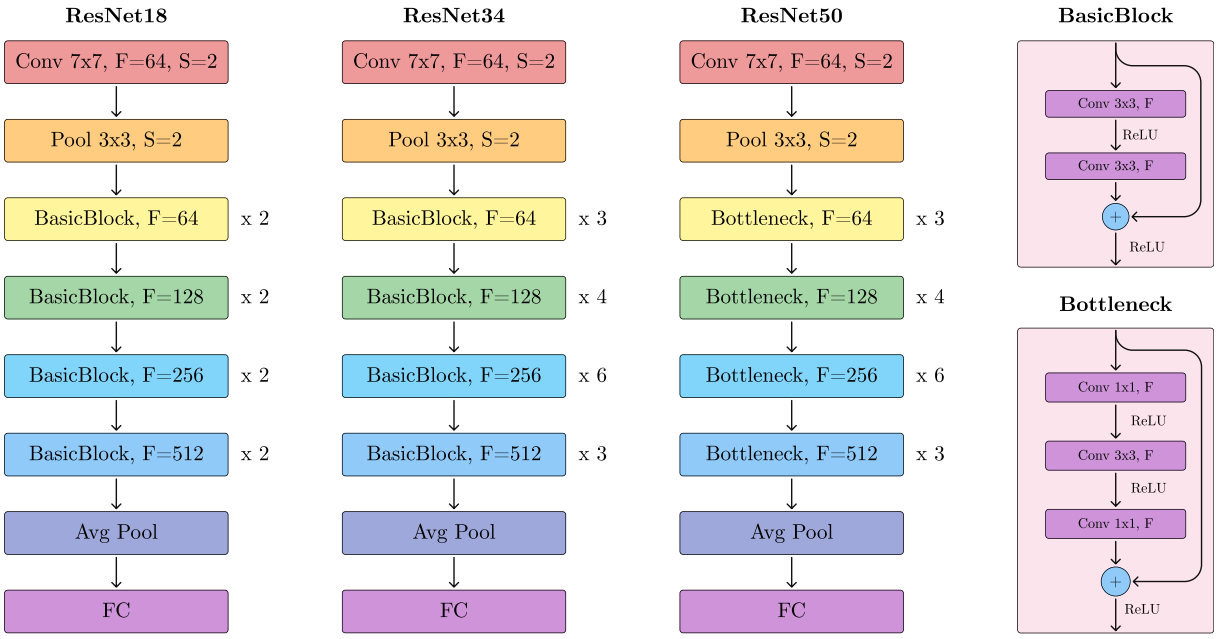


Figure 3.6: Variants ResNet architectures used as the backbone of the proposed audio module.

3.9.2 Learning parameters of the neural network

The model is learned using AdamW [18] optimizer by minimizing cross-entropy loss which is defined as

$$\mathcal{L}_{\mathrm{CE}}(p, y) = -\sum_{c=1}^{C} y_i \log p_i$$

where C is the number of classes, y_i equals to 1 if the sample belongs to class i and 0 otherwise and p_i are probabilities for each class given by the softmax activation function.

Because the data set is very unbalanced, we also tried Class-Balanced Cross-Entropy [8] (CBCE) loss which utilizes re-weighting scheme that uses the effective number of samples for each class to re-balance the loss. CBCE loss has a single hyperparameter $\beta \in [0, 1)$ and can be written as

$$\mathcal{L}_{\mathrm{CBCE}}(p, y) = \frac{1-\beta}{1-\beta^{n_y}} \mathcal{L}_{\mathrm{CE}}(p, y)$$

where n_y is the number of samples in the ground-truth class y.

3.9.3 Multi-headed neural network

The proposed audio module is versatile. Besides the number of passing vehicles, it can be configured to predict an arbitrary event from the audio. For example, we experimented with the prediction of the movement direction and the vehicle type. The predictor outputs the number of the specific events in the input time window. In case of the movement direction, we have two predictors: the first is counting the number of incoming vehicles while the second one predicts the number of vehicles in outgoing direction. In case of the vehicle type prediction, we have a separate predictor for each category: one predictor counts the passages of the personal cars while the other counts all other types besides personal cars. All predictions are performed simultaneously by a single NN with multiple heads. Each head is a fully-connected classification layer at the end of the network which solves one task. The optimized criterion for multi-headed neural network consists of a weighted sum of cross-entropy losses, i.e. one cross-entropy loss per head.

To predict the class, we use a Maximum A Posteriori (MAP) rule: $\arg\max_y p(y|x)$, where y is a class, x is a sample and p is gives a conditional probability which is modeled by the weights of the neural network with a softmax function in the end.

3.10 Evaluation metrics and protocols

To measure the performance of the proposed method and to perform a model selection, we use two evaluation metrics such as Relative Vehicle Counting Error (RVCE) that was defined in [10] and Mean Absolute Error (MAE). RVCE gives us a relative error for a single video recording

$$\mathrm{RVCE}(v) = \frac{|n_v^{\mathrm{true}} - n_v^{\mathrm{pred}}|}{n_v^{\mathrm{true}}} \tag{3.1}$$

where n^{true} and n^{pred} represent the true and predicted number of vehicles in a video v.

Besides this, we also keep track of the MAE estimated for each time window in a video

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{3.2}$$

where n is the number of windows in the video, y_i is the ground true and x_i is the predicted number of events in i-th time window.

An important part of the training is the model selection [20]. To prevent the model from the overfitting to the training data set, we use k-Fold Cross-validation. It gives us a more accurate estimate about the generalization abilities of the classifier on unseen data. This technique splits our data set to 3 parts: training, validation and test. The validation part is used to select the model during the training process on the training data. In literature, this method is called the early stopping [5]. We select the model that minimizes the sum of RVCE errors

$$\sum_{h}^{H} w_h \sum_{v}^{V} \mathrm{RVCE}(v; h)$$

where H and V are the numbers of classification heads and videos respectively and w_h is the weight of h-th head.

3.11 Inference on overlapping windows

The neural network used takes a fixed-size time window and outputs a prediction for this window, e.g. it predicts the number of passed vehicles. However, the goal is to get the total number of vehicles passed in the entire video record. A simple strategy to get the total number of passages is

to slide the input window over the input signal and aggregate the predictions on individual time windows. In case of non-overlapping time windows, the aggregation is a simple summation of counts predicted from the individual windows. However, we found that the prediction accuracy drops when the distance of the audio event is close to the borders of the input window. We measured this dependency experimentally and the results are shown in Figure 3.7.

To resolve the problem, we propose a novel method to aggregate the predictions from overlapping time windows. Let us explain the method using the simplified setup visualized in Figure 3.8. For example, let us have 6 seconds long time windows placed over the input signal with 3 second long overlaps. Let $c_{i,i+1}$, $i = 1, \ldots, L-1$, denote a sequence of event counts in the overlapping time windows. Let x_i, $i = 1, \ldots, L$, be the event counts in a sequence of imaginary non-overlapping windows covering the same input signal. The neural network applied to the overlapping windows outputs distributions $\hat{p}(c_{i,i+1}) \in \Delta(N+1)$, $i = 1, \ldots, L-1$, over the events counts in these windows, where $\Delta(M)$ denotes M-dimensional probability simplex. The goal is to recover the distributions $p(x_i) \in \Delta(N/2+1)$, $i = 1, \ldots, L$, of the event counts in the imaginary non-overlapping windows with doubled time resolution.

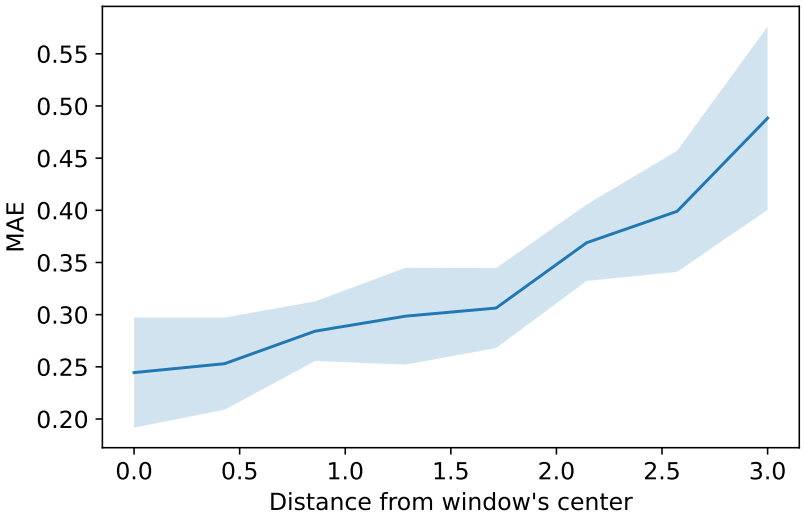


Figure 3.7: Dependence of mean absolute error on event's distance from the window's center. If there are more than 1 event in the window, we take take the maximal distance to describe the window.

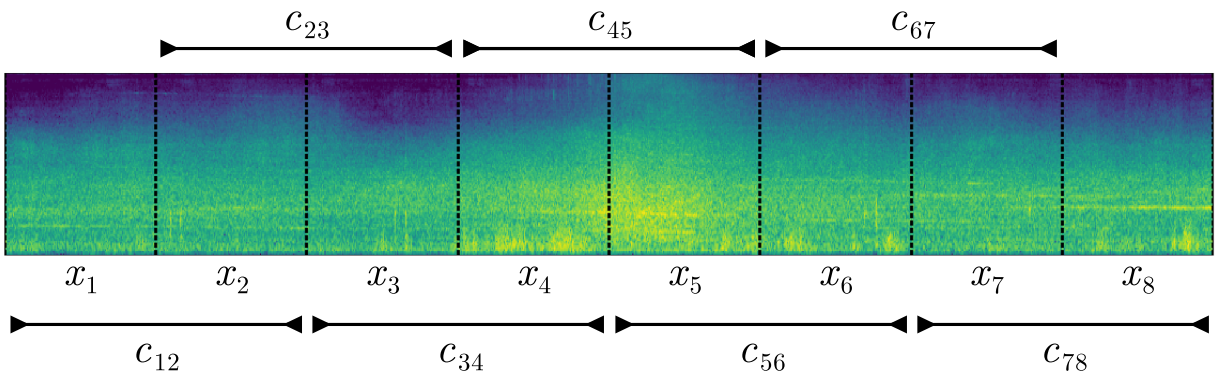


Figure 3.8: Visualization of a novel method that aggregates counts in a sequence of overlapping time windows $c_{i,i+1}$, $i = 1, \ldots, L-1$ to recover counts in a non-overlapping time windows x_i, $i = 1, \ldots, L$

Algorithm 1 EM Algorithm

1: init α^0
2: $t \leftarrow 0$
3: **while** converge **do**
4: $\quad t \leftarrow t + 1$
5: $\quad p^t \leftarrow \arg\max_p F(p, \alpha^{t-1})$
6: $\quad \alpha^t \leftarrow \arg\max_\alpha F(p^t, \alpha)$
7: **end while**

Let us consider a single time window described by the distribution $\hat{p}(c_{12})$ obtained from the neural network. Let $p(x_1)$ and $p(x_2)$ be the unknown distributions of two non-overlapping time windows covering the same input. Distributions $p(x_1)$ and $p(x_2)$ are found by maximizing the negative Kullback-Leibler (KL) divergence between the known $\hat{p}(c_{12})$ and the unknown $p(x_1, x_2) = p(x_1)\, p(x_2)$. To solve the maximization problem, we propose a technique inspired by the Expectation-Maximization (EM) algorithm [9]. The idea is to introduce an auxiliary distribution α and to replace the KL-divergence $\mathrm{KL}(\hat{p} \mid\mid p)$ by a tight lower bound $F(p, \alpha)$ derived from Jensen's inequality. The algorithm uses a fixed number of iterations in which it alternates maximization of the lower bound $F(p, \alpha)$ w.r.t. p and α which have an analytical solution. It is easy to show that the algorithm monotonically increases the KL-divergence $\mathrm{KL}(\hat{p} \mid\mid \alpha)$ being our target objective. The proposed optimization technique is described by Algorithm 1. The lower bound to be optimized is obtained as follows:

$$
\begin{aligned}
\mathrm{KL}(\hat{p} \mid\mid p) &= \sum_{c_{12}} \hat{p}(c_{12}) \log \sum_{c_{12}=x_1+x_2} p(x_1)p(x_2) \\
&\geq \sum_{c_{12}} \hat{p}(c_{12}) \sum_{c_{12}=x_1+x_2} \alpha(x_1, x_2) \Big[\log p(x_1) + \log p(x_2) \Big] \\
&= \sum_{c_{12}} \hat{p}(c_{12}) \sum_{c_{12}=x_1+x_2} \alpha(x_1, x_2) \log p(x_1) \\
&\quad + \sum_{c_{12}} \hat{p}(c_{12}) \sum_{c_{12}=x_1+x_2} \alpha(x_1, x_2) \log p(x_2) \\
&= \sum_{x_1} q_1(x_1; \alpha) \log p(x_1) + \sum_{x_2} q_2(x_2; \alpha) \log p(x_2) \\
&= F(p, \alpha)
\end{aligned}
$$

$$
\begin{aligned}
q_1(x_1, \alpha) &= \sum_{x_2} \alpha(x_1, x_2) \hat{p}(c_{12} = x_1 + x_2) \\
q_2(x_2, \alpha) &= \sum_{x_1} \alpha(x_1, x_2) \hat{p}(c_{12} = x_1 + x_2)
\end{aligned}
$$

The optimization of the whole recording that combines all overlapping windows $c_{i,i+1}$, $i = 1, \ldots, L-1$ boils down to the maximization of the following criterion

$$
\sum_{i}^{L-1} \sum_{c_{i,i+1}} \hat{p}(c_{i,i+1}) \log \sum_{c_{i,i+1}=x_i+x_{i+1}} p(x_i)p(x_{i+1})
$$

Notice, the method can be extended further to work on arbitrary number of overlapping windows, e.g. if we have 6 second long window, 3 equally placed overlapping windows will have 2 second overlap. Though, the method grows exponentially in complexity with respect to the number of overlaps between windows.

3.12 Ensembling

When training data are a limiting factor, we can make model predictions more stable by learning more models and aggregating their outputs. When our data set is small and we select the model using the validation set, we do not have to pick the optimal one due to the estimation error. To reduce the variance, we can shuffle the training and the validation set so that each new model is learned and validated using a different subset of examples. The models are trained independently of each other. After training is complete, we can aggregate the probabilities given by each model on examples in the test set. A simple averaging aggregation rule reads as follows:

$$p(y|x) = \frac{1}{n} \sum_{i}^{n} p_i(y|x)$$

where n is the number of learned models, p_i gives the probability returned by the model i that sample x belongs to the class y.

3.13 Results visualization

To better analyze the failures of the audio recognition module being developed, we need an appropriate visualization method that shows all relevant information as it evolves in time. Namely, the most important information we want to visually inspect includes the ground true audio and video annotations, predictions of the audio module, the raw audio signal along with the spectogram and the energy extracted from the signal. In this section, we describe a compact visualization of all the mentioned data that proved to be very useful for development of the method.

An example of the visualization we have been using is shown in Figure 3.9. The top image shows the source annotations from the vision-based module and the extracted annotation of the audio signal. Namely, the green and the red rectangles correspond to the periods of time the incoming and outgoing vehicles were in the field of view of the camera and were tracked by the vision-based module. The black dotted vertical lines mark the automatically extracted annotation of the vehicle passage times, i.e. moments when the vehicle is closest to the recording device. The second image from the top shows the neural network predictions for a given time window. The black dotted line corresponds to the ground number of vehicle passages, while the green line represents the predicted number of passages. In addition, each cell of the figure contains to numbers: the number of vehicle passages in a given window, and the probability of that number predicted by the neural network. The third image from the top visualizes the raw audio signal (blue line) and the extracted energy (black line) which is used to find the vehicle passages. Finally, the bottom figure shows the spectrogram of the audio signal which is used to compute features for the neural network and where patterns corresponding to vehicle passages are visible by a naked eye.

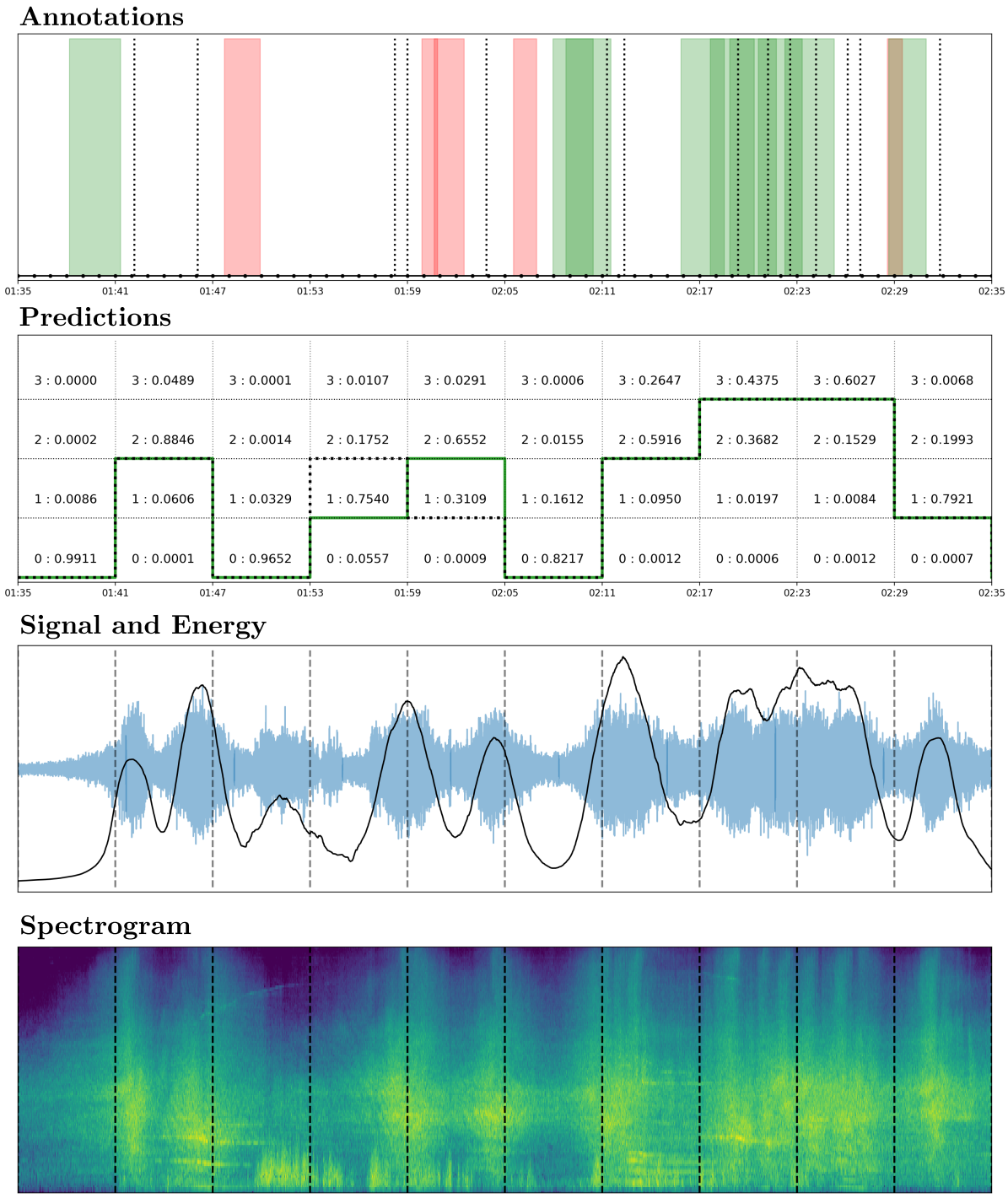


Figure 3.9: Results visualization from one minute long audio signal. The top picture shows two types of annotations: video annotations which correspond to vehicle tracking in incoming (green) and outgoing (red) directions and audio annotations which are obtained by automatic alignment described in Section 3.5. The second image shows neural network predictions for a given window in green and dotted for the ground true. The third image shows audio signal in blue and energy of the signal in black. The bottom plot shows the spectrogram for this audio signal.

Chapter 4

Experiments

This chapter presents experiments that were carried out to find a good configuration of the neural network architecture and the training procedure, and to thoroughly evaluate the performance of the found set-up.

4.1 Evaluation protocol

If not stated otherwise, the models are trained to predict the number of passed vehicles using the data recorded by the RX100 camera.

RX100 data set consists of 36 videos, which is 12.5 hours of footage in total. For each experiment, we trained five models using the 5-fold cross-validation protocol. We split the data set into training, validation, and test files using 60%, 20%, 20% of data, respectively.

Training samples are generated by sequentially cropping short clips without overlap from the entire audio signal. Using 6-second long clips, we generated 4.4k training and 1.5k validation samples in each epoch. We augment the training data by offsetting the moment from which the sequential cropping started. We increase the offset in each epoch by 0.25 seconds and reset the offset to zero after each 23 epochs completed.

When training the network, we use randomly initialized weights that were created by a random generator starting from the same seed, by which we make sure that the initial weights are the same for each experiment. When comparing different configurations of the prediction model, we fix all the parameters except for the one we are evaluating. Therefore, tuning the model parameters can be seen as using a coordinate-wise optimization strategy, which may not be an optimal but usually yields reasonable results and, mainly, it is time efficient.

In most experiments, we visualize the results using box plots [17]. The main metric is RVCE defined in the equation (3.1). The RVCE metric is evaluated for each test video separately. The box plots then visualize the RVCE for individual videos as colored dots, and aggregated statistics like the mean, which is shown as a black cross, and the quartiles visualized by boxes with the horizontal line in the center corresponding to the median. Notch is used to show the 95% confidence interval for the median. The visual description of the parts of the box plot is presented in Figure 4.1

4.2 Baseline

In this section, we describe the training of a baseline model that is used in follow-up experiments as a reference point to compare with.

The baseline model is trained as follows. First, we resample the input signal from the original 44,100 to 22,050 Hz. The input audio record is divided into 6-second long clips, each of which is described by a sequence of $22,050 \times 6 = 132,300$ values. Then, we extract the features from the sequence using the STFT with a window size of 1024 samples and a hop length of

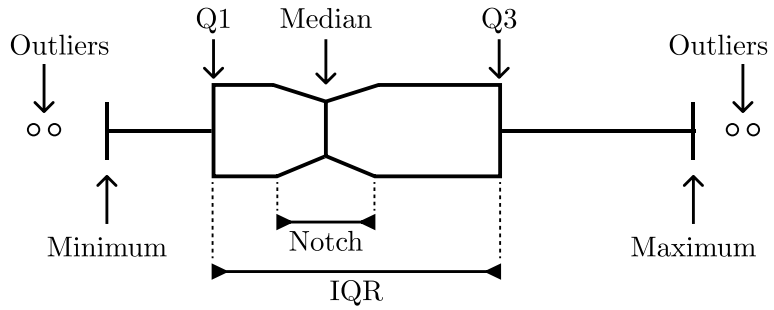


Figure 4.1: Detailed description of box plot used to visualize the experimental results. Q1 and Q3 quartiles show that 25% and 75% of measurements are below its value. Interquartile range (IQR) covers the central 50% of the measurements. Notch is used to show the 95% confidence interval for the median. Minimum and maximum represent the lowest and the highest measurement excluding any outliers

512 samples. The obtained spectrogram is rescaled by applying the logarithm to each of its values. Consequently, we resize the spectrogram to an image of 128×128 pixels using bilinear interpolation. Finally, we normalize the pixel values of the 128×128 image to have zero mean and unit variance.

As the backbone architecture of the neural network used for the baseline, we selected ResNet18 [13], which has 11.2M trainable parameters. We optimize the parameters by minimizing cross-entropy loss using the AdamW [18] optimizer with the standard PyTorch setting: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, learning rate of 10^{-4} and weight decay of 10^{-2}. Early stopping [5] is achieved by selecting the model that has the lowest RVCE on the validation set, as discussed in Section 3.10. The performance of the baseline model is shown in Figure 4.2. The mean and median RVCE of the baseline model are 0.08 and 0.06, respectively.

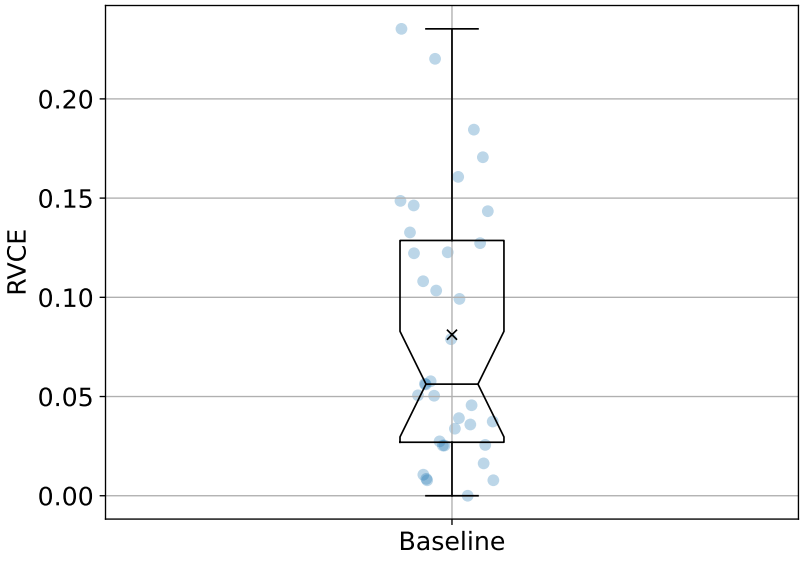


Figure 4.2: The performance of the baseline model.

4.3 Features from different recording devices

Motivation Different recording hardware processes the sound in a different way. This influences the way how the waveforms, obtained using different recording devices, look. In our experiments, we use three devices, namely: RX100, iPhone, and Mobius. If we look at Figure 4.3, which shows the features as audio spectrograms, we can clearly see the visual differences. This experiment compares the influence of these visual differences on the final RVCE. We train three prediction models. Each model is trained and evaluated on the audio signal from one of the three recording devices. Except for the audio signal, we use the exact same setup for the three models.

Results The results are shown in Figure 4.4. The model using the audio signal from the the RX100 camera performs the best, namely, it has the lowest mean and median RVCE as well as the results on individual test records have the variance. The second best performance was observed using the Mobius and, worst, performed the model using audio from iPhone.

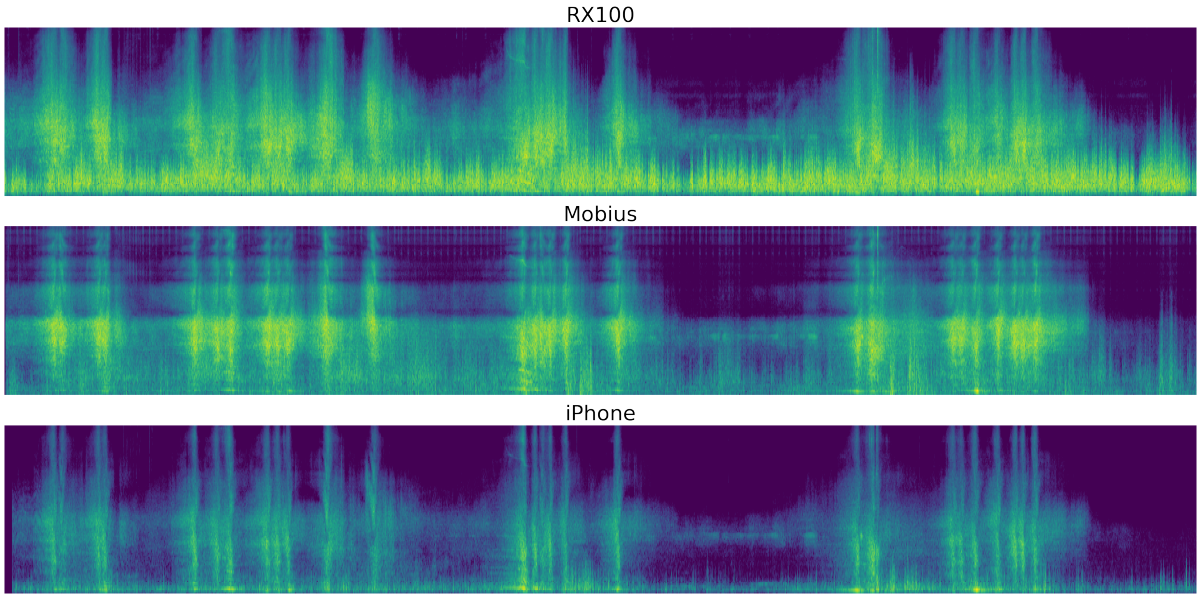


Figure 4.3: Comparison of spectrograms extracted from aligned audio signals captured from three recording devices: RX100, iPhone and Mobius.

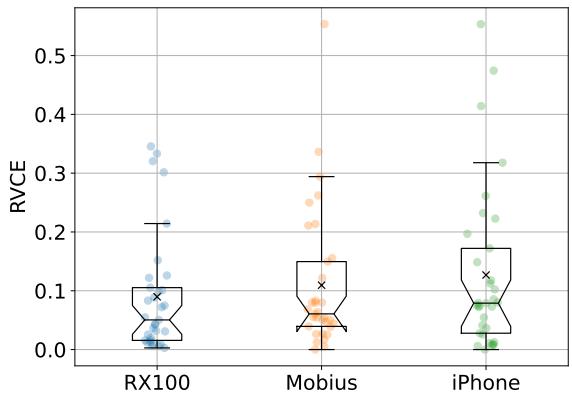


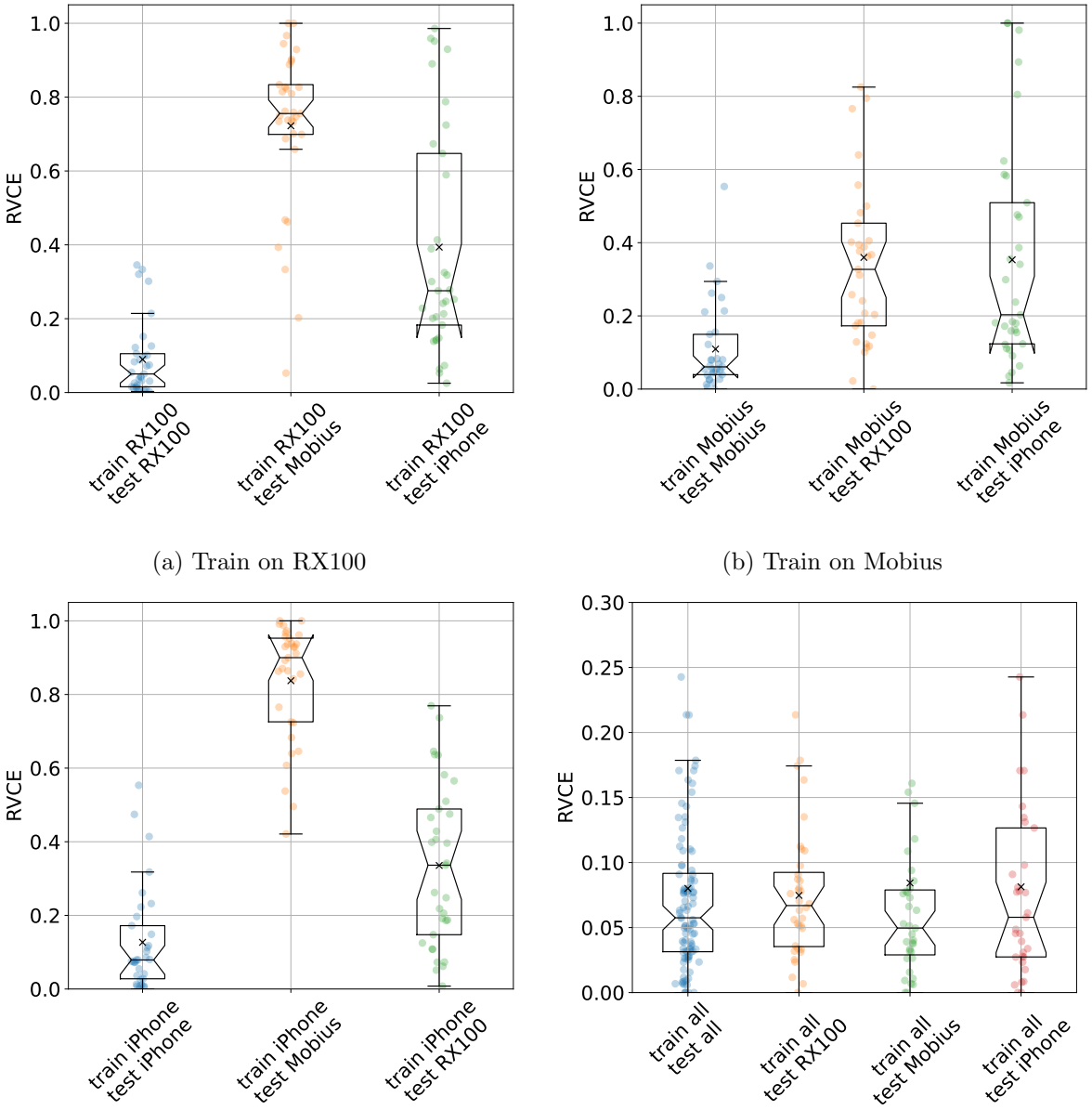
Figure 4.4: Comparison of the performances of prediction models trained and tested on audio signal from three different devices: RX100, Mobius and iPhone

4.4 Cross-device generalization

Motivation In this experiment, we train the prediction models on signal from one recording device and test their performance on signal from other devices. We make sure that the recording locations in the training set and the testing set are always different. In other words, we want to evaluate how the prediction model generalizes not only to different recording devices but also to different locations.

In addition, we train a single prediction model using data from all recording devices and test its performance on signals from each device separately. The goal is to check whether the neural architecture used has sufficient capacity to accommodate multiple devices or whether it is necessary to train a separate prediction model for each device.

Results The results obtained are summarized in Figure 4.5. First, we can notice a significant drop in performance when the model is trained on features extracted from the audio signal of one recording device and tested on the others. The same observation was made by the authors of [16] who studied the problem of generalization in audio classification. They showed that classification accuracy drops significantly when the model is trained on samples from one recording device and tested on samples from a different one. Second, the Mobius signal appears to be more difficult to classify when using a model trained on other devices; see Figures 4.5a and 4.5c. However, when a model is trained on the Mobius signal, it performs better on other devices; see Figure 4.5b. This suggests that training using the signal from certain recording devices might lead to better generalization than using the features of other devices. Finally, Figure 4.5d shows that when training a model on data from all recording devices, it performs well on all devices. It shows that we can use a single prediction model for all recording devices without compromising performance, which can be beneficial for a real-life application of the method.



(a) Train on RX100

(b) Train on Mobius

(c) Train on iPhone

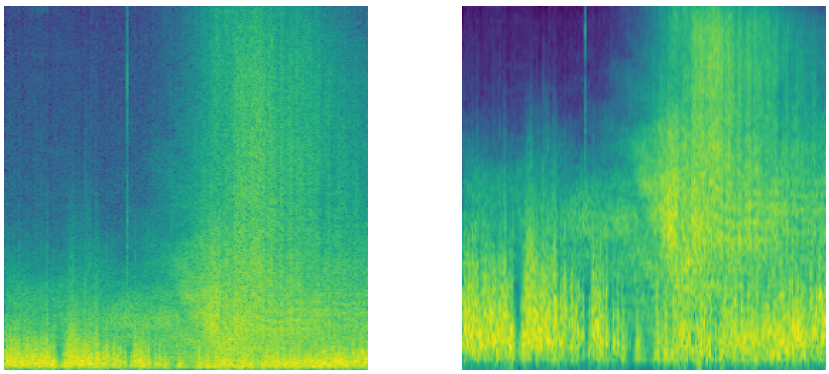
(d) Train on all

Figure 4.5: The following box plots compare the perpormence of models trained on features from one device and tested on features from the different one

4.5 Spectrogram versus mel spectrogram

Motivation In Section 3.7 we empirically evaluate two variants of the spectrogram-based features. Namely, we compare the (ordinary) spectorgram and the mel spectrogram, the main difference of which is the nonlinear transform of the extracted frequency features (cf. Section 3.7). To this end, we train two prediction models each using one of the feature representations. Visually, the difference between the two feature representations can be seen in Figure 4.6.

Results The results obtained are shown in Figure 4.7. It can be seen that the Mel transformation of the spectrogram features does not bring any significant improvement. The most probable reason for this observation is that the neural network can capture the non-linearities, if necessary, automatically from the data as the architecture used has enough non-linear layers.



(a) Spectrogram (b) Mel spectogram

Figure 4.6: Visual difference between the ordinary audio spectrogram (left) and the Mel spectrogram (right).

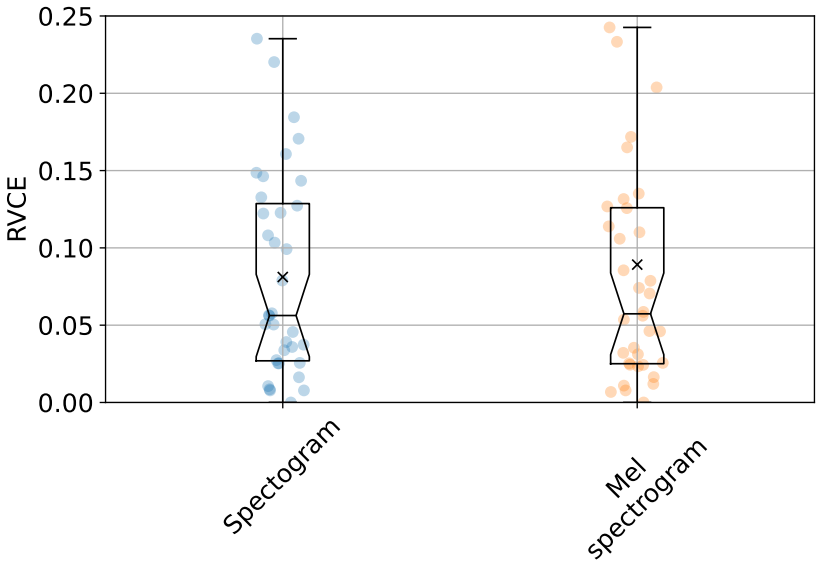


Figure 4.7: Performance comparison of the prediction model using either the ordinary Spectogram (blue) and the Mel spectrogram (orange).

4.6 Data augmentations

Motivation In Section 3.8 we describe the most common data augmentation techniques used for image and audio data augmentation. In this section, we experimentally evaluate the impact of the augmentations on the performance of the audio module. In case of image augmentation, the augmentation is applied to the spectrogram extracted from the audio signal. In case of audio augmentation, the augmentation is applied to the raw audio signal. We compare the following image augmentations: random erasing, random Gaussian blur and random resized crop that are implemented in the PyTorch library, and the following audio augmentations: random colored noise, random gain, random high pass filter, random low pass filter, random pitch shift that re implemented in Torch-Audiomentations. We evaluated the models trained using just one of the augmentations, and the models trained using all the augmentations. In the latter case, each training sample is processed by a single augmentation selected uniformly at random from the pool of options. Figure 3.5 shows examples of how some of the methods considered augment the input image.

Results The results obtained are summarized in Figure 4.8. We see that the best results are obtained by using the audio and image augmentations simultaneously, as it leads to the performance improvement and at the same time it most reduces the variance of the results. A single best-performing image augmentation method is the random erasing, while best-performing audio augmentations are the random colored noise and the random loss-pass filter method.

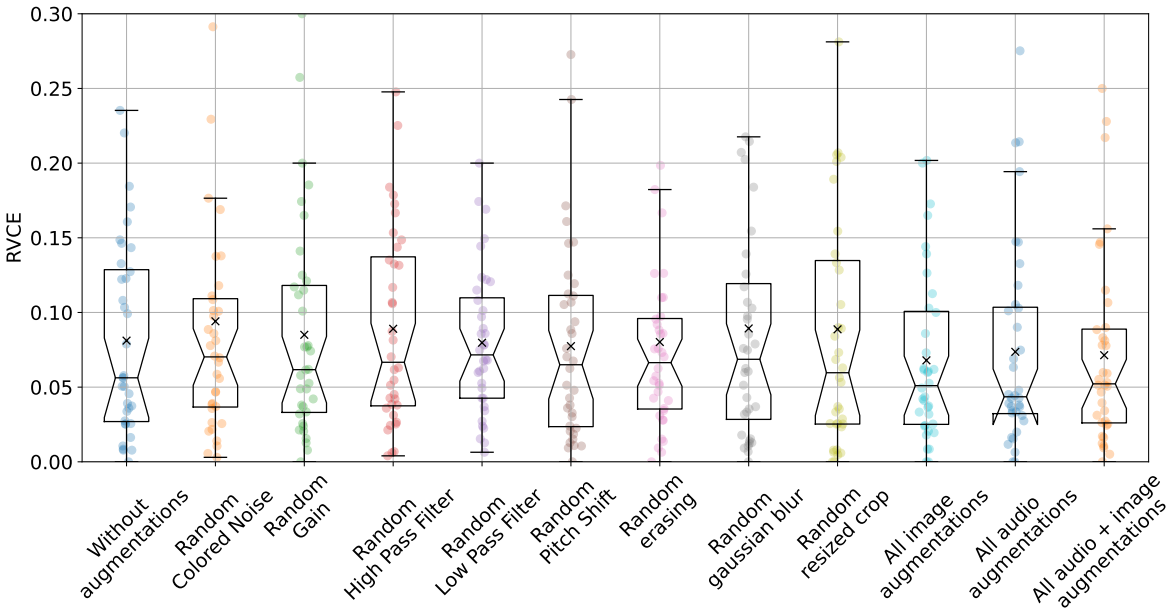


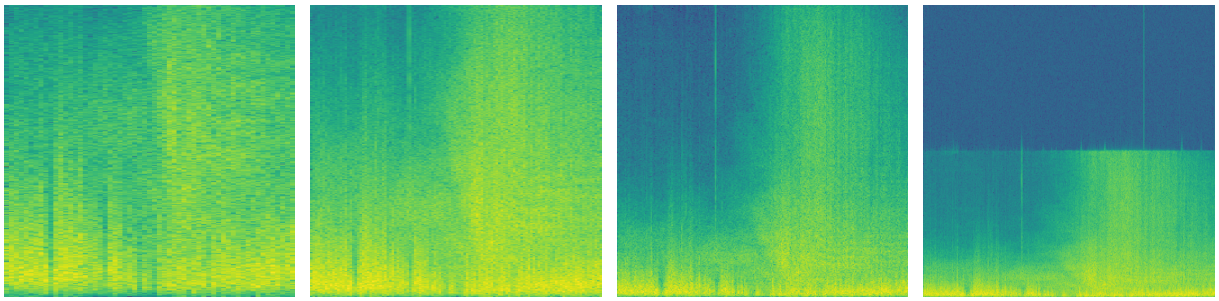
Figure 4.8: Performance comparison of models trained with different augmentation techniques

4.7 Influence of the sampling rate

Motivation One possible application of the audio module that is being developed in this thesis is its deployment on low-power embedded systems. Embedded systems may put constraints on the sampling rate used for the audio recording. With this in mind, we experimentally evaluate how the prediction performance changes when the sound is recorded with different sampling rates.

It is worth noting that when we transform the signal sampled with a specific sampling frequency into a spectrogram, which is described in Section 3.7, we use fixed STFT parameters, such as hop length and frame length. Because of this, the size of the output spectrogram, i.e. the 2d image, will depend only on the sampling rate. For instance, if we fix the frame length and the hop length to be 1024 and 512 samples, respectively, the 6-second signal with sampling rates of 5000 and 44100 Hz will be transformed into spectograms with time dimensions equal to 98 and 862 features, respectively. Thus, having a higher sampling rate requires more memory and computations for training and inference, leading to a longer training time. Examples of features extracted using various sampling rates are presented in Figure 4.9

Results The experimental results presented in Figure 4.10 indicate that increasing the sampling rate steadily improves the performance measured by RVCE. This might be explained by the fact that spectrograms obtained with higher sampling rates contain more information because of their sizes. Nevertheless, we also see that there is a negligible performance difference when we use sampling rates of 10000, 22050 and 44100 Hz.



(a) 5000 Hz (b) 10000 Hz (c) 22050 Hz (d) 44100 Hz

Figure 4.9: Examples of audio spectrograms obtained when using different sampling rates.

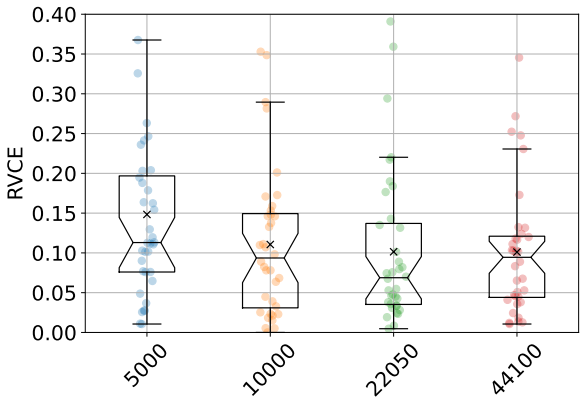
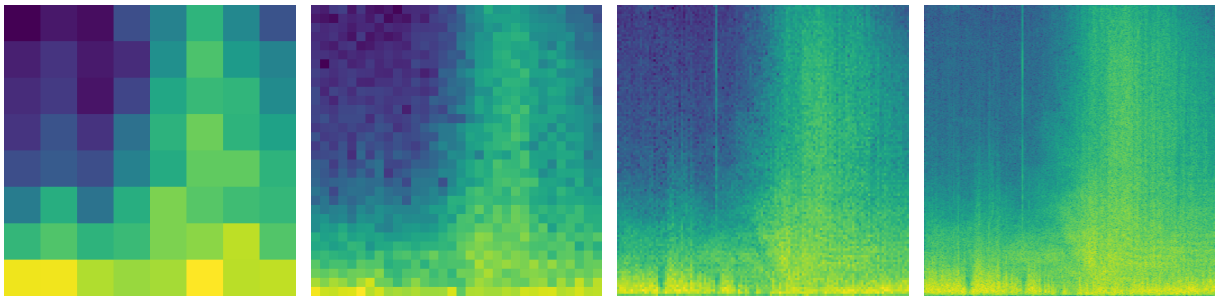


Figure 4.10: The prediction performance obtained when using different sampling rates of the input audio signal.

4.8 Size of the input spectrogram

Motivation As we showed in the previous section, the sampling rate influences the size of the spectrogram. When we see the spectrogram as an image, an obvious question is whether, by decreasing its dimension by resizing, we can improve the model performance. To answer this question, we downscale the spectrogram image size by a bilinear transformation before it is fed to the neural network and measure how it influences the prediction performance. We also note that the spectrogram size has an impact on the processing time because, by making the input images smaller, the forward and backward passes will need fewer computations, which speeds up training as well as inference. In this experiment, we used the sampling rate 22,050 Hz, which defines the base spectrogram size as 513 × 431 pixels.

Results The results obtained are summarized in Figure 4.12. We see that even small 64 by 64 spectrogram images have enough information to guarantee the same performance as the original 513 by 431 images that are not resized. Interestingly, even when using 8 by 8 pixel images that do not carry much information and the mean, and lead to a noticeable increase of the overall RVCE, some videos are still classified almost with 0 RVCE.



(a) 8 × 8 (b) 32 × 32 (c) 128 × 128 (d) 513 × 431

Figure 4.11: Examples of spectrogram images obtained by downscaling the original 513 × 431 spectrogram (the right most image).

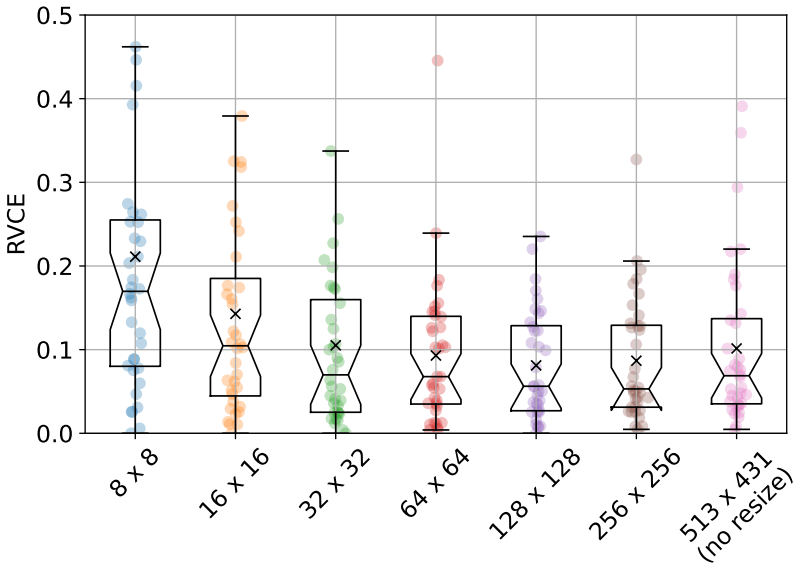


Figure 4.12: Performance comparison of models trained on differently resized spectrograms

4.9 Effect of changing the ResNet depth

Motivation This section investigates how the depth of the neural network, i.e. the number of layers, influences the performance. We compare three variants of ResNet architecure: ResNet18, ResNet34, ResNet50, which differ in the depth of the network and in the number of parameters, which is 11.2M, 21.3M and 23.6M, respectively. The compared variants of the ResNet architecture are shown in Figure 3.6

Results The results are summarized in Figure 4.13. It can be seen that increasing the depth of the ResNet architecture does not lead to any significant performance improvement.

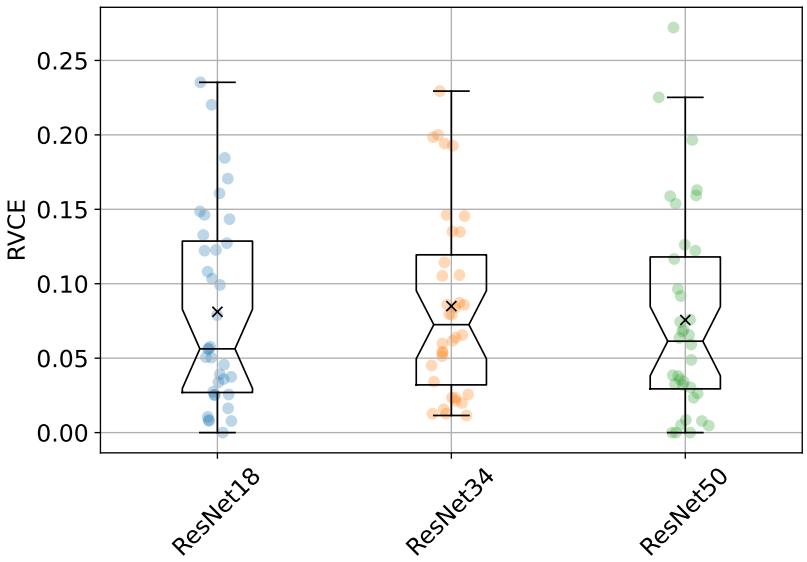


Figure 4.13: Comparison of the performance when using different variants of the ResNet architecture.

4.10 Evaluation of the Class-Balanced Cross-Entropy loss

Motivation Figure 4.14 shows that the distribution of predicted classes, corresponding to the number of vehicle passages, is highly unbalanced. In Section 3.9.2 we presented the Class-Balanced Cross-Entropy (CBCE) loss as a replacement for the usual cross-entropy loss, which tries to balance the contribution of the classes by assigning a weight to each class corresponding to its relative frequency. The loss function has a single hyperparameter β.

Results For our data set, we tried two values for the hyperparameter β. Figure 4.15 indicates that the CBCE loss with neither $\beta = 0.999$ nor $\beta = 0.9$ has a noticeable positive influence on the performance of the model.

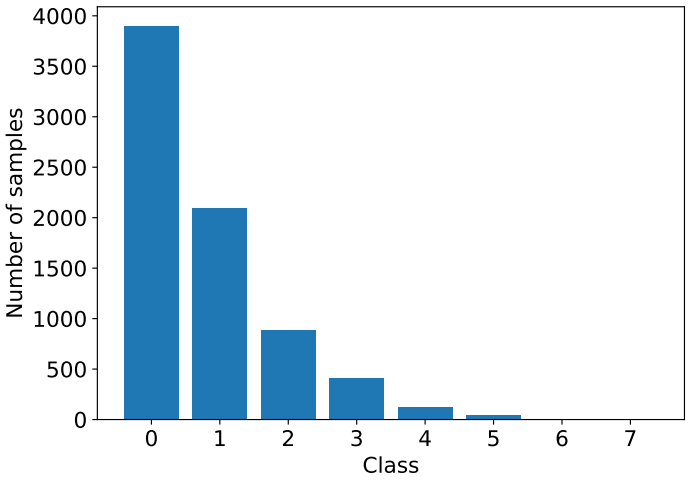


Figure 4.14: Distribution of samples in the classes. The classes correspond to the number of passed vehicles.

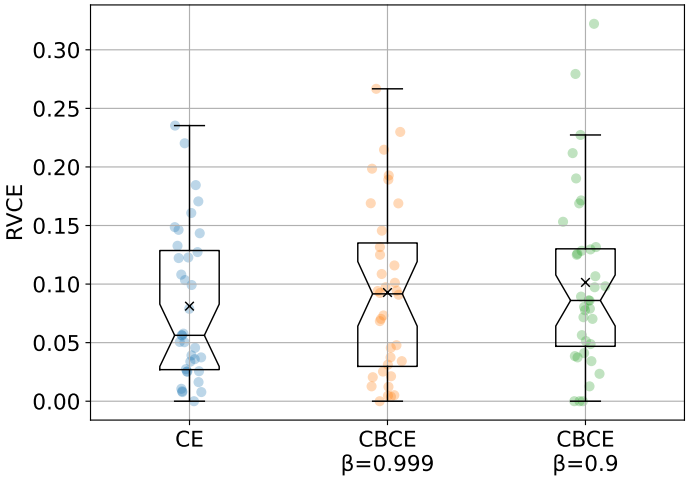


Figure 4.15: Performance evaluation of two variants of the loss function: the standard cross-entropy loss and the class-balanced cross-entropy with parameters $\beta = 0.999$ and $\beta = 0.9$, respectively.

4.11 Inference on overlapping windows

Motivation As we shown in Figure 3.7, the prediction performance decreases when audio events occur close to the boundaries of the input time window. In Section 3.11, we developed a method that attempts to solve the problem by applying inference on overlapping windows and reconstructing a distribution corresponding to (virtual) non-overlapping windows with half the size of the original ones. This section presents experimental results for the newly introduced inference method.

Results A quantitative evaluation of the results is presented in Figure 4.16. It is seen that applying the inference that uses overlapping windows only very slightly improves the mean RVCE, however, it significantly reduces the variance of the results and the number of outliers with very high RVCE error.

A qualitative evaluation of the method shown on a particular example presented in Figure 4.17. At the bottom of Figure 4.17 we can see a spectrogram divided into eight 3-second-long time intervals. The plot above shows a waveform in blue and the energy of the signal in black. The maximum of energy, shown as a red vertical line, marks an automatically found passage of a single vehicle.

We compare the inference on non-overlapping windows placed on the signal in two different ways. In the first case, we start from window 1 and obtain statistics for windows (1,2), (3,4), (5,6), and (7,9). In the second case, we start the inference from window 2 and obtain it for windows (2,3), (4,5), and (6,7). The statistics for both inferences are shown in the two tables above.

We can observe two situations that occurred: in the first case, a model made a mistake by classifying both windows (3,4) and (5,6) as having one vehicle pass, which was influenced by the fact that the event occurred near the border of a window (5,6). In the second case, the event occurred in the center of the window (4,5), and the entire sequence was correctly classified.

Our goal is to aggregate the results from these two inferences on non-overlapping windows and make it invariant to the place from which the inference starts. We call it the inference on overlapping windows and describe how it works in Section 3.11. The result of the method is a reconstructed distribution on virtual windows, which is shown at the top of the figure. Each cell contains probability values corresponding to the predicted number of evens. The green color marks the times where the ground-truth number of passages matches the predicted one, while the red color represents the errors.

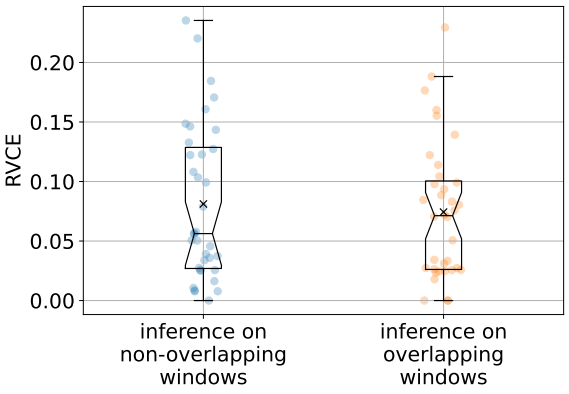


Figure 4.16: Performance evaluation of two variants of inference. The first one uses nonoverlapping windows while the second one uses overlapping windows.

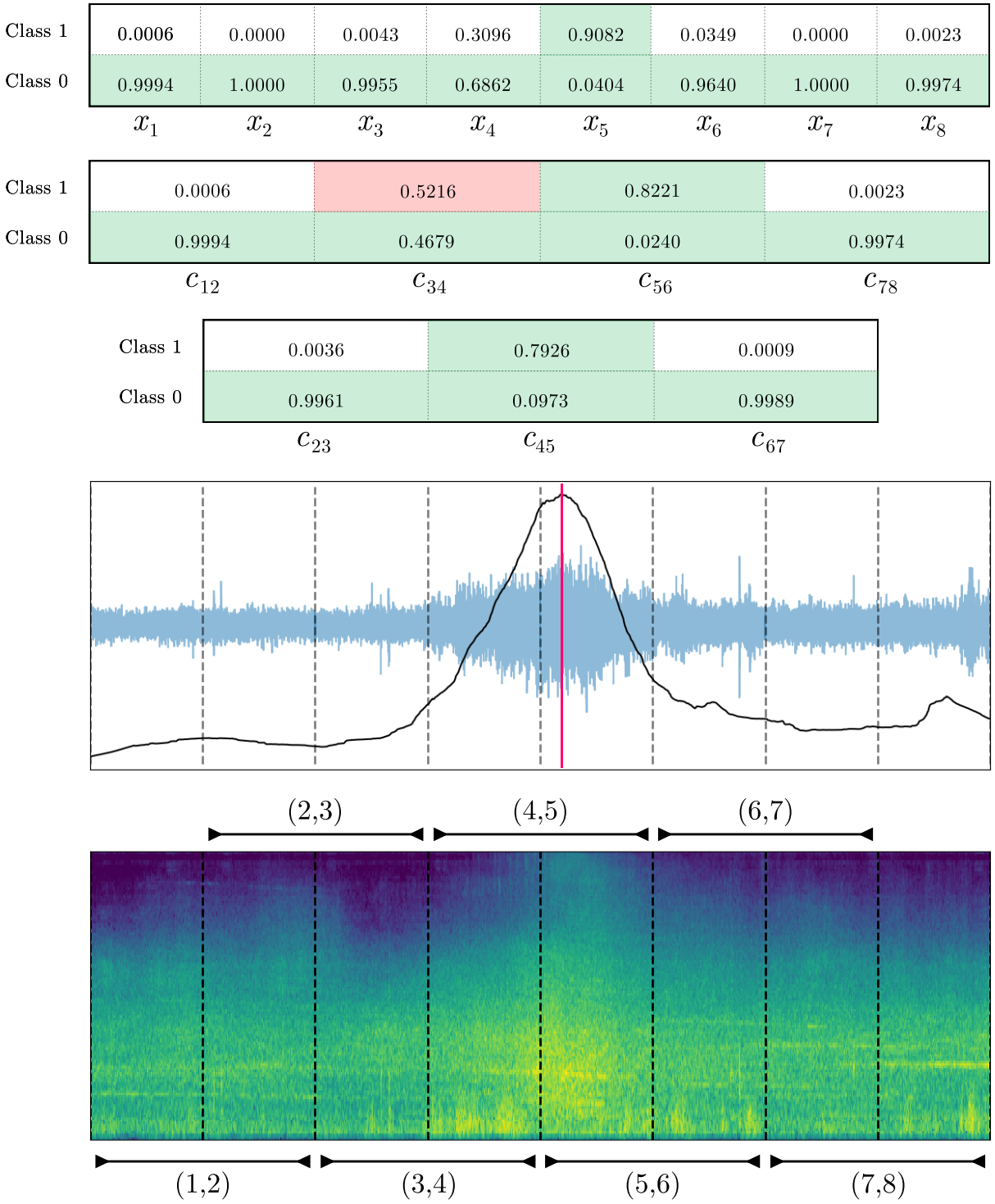


Figure 4.17: Vizualization of the issues occured when using inference on non-overlapping windows and the proposed solution using the overlapping ones.

4.12 Influence of the time window length

Motivation Our prediction model splits the input signal into a sequence of fixed-sized time windows, which are then processed by the neural network independently. This experiment evaluates how the length of the time window influences the performance of the model.

Results The results obtained are presented in Figure 4.18. As can be seen, the window length does not appear to be a key factor in determining the performance of the model. However, experiments suggest that the model trained with a 7-second long window length achieves the best performance.

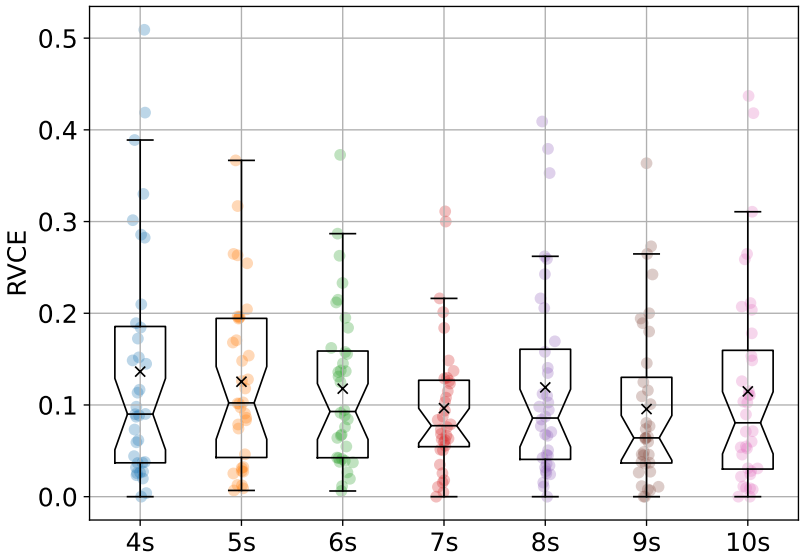


Figure 4.18: Performance comparison of models trained on windows with different length which was varied from 4 to 10 seconds.

4.13 Evaluation of the multi-task prediction

Motivation This section investigates the influence of adding multiple classification heads on the model performance. Each head corresponds to a different counting task. Note that the vision-based module provides additional information for each detected vehicle, such as direction of movement and vehicle type. We can use this information to define other counting tasks to be solved by the model. In particular, we consider the estimation of the number of vehicles that passed in the incoming and the outgoing directions and the number of vehicles of a particular type that have passed.

Results Figure 4.19 shows the RVCE of the vehicle counting task for four different configurations of the prediction heads. We can see that adding new counting tasks has a negligible influence on the performance of the base prediction problem. Using all four heads seems to reduce the variance of the results.

We tried to estimate the number of vehicles passing in the incoming and outgoing directions and their type, namely, the category of personal cars and the category of other than personal cars. However, we did not have enough time to experiment with the model configurations. For that reason, the results presented in Table 4.1 should be considered as the baseline.

		movement direction		vehicle type	
	counting	incoming	outgoing	car	not car
Mean RVCE	0.09	0.1	0.18	0.1	0.5

Table 4.1: Performance of the prediction model in different counting tasks. Namely, the prediction of the count of the vehicles passed, the vehicles passed in the incoming direction, the vehicles passed in the outgoing direction, the number of personal cars, the number of vehicles other than personal cars.

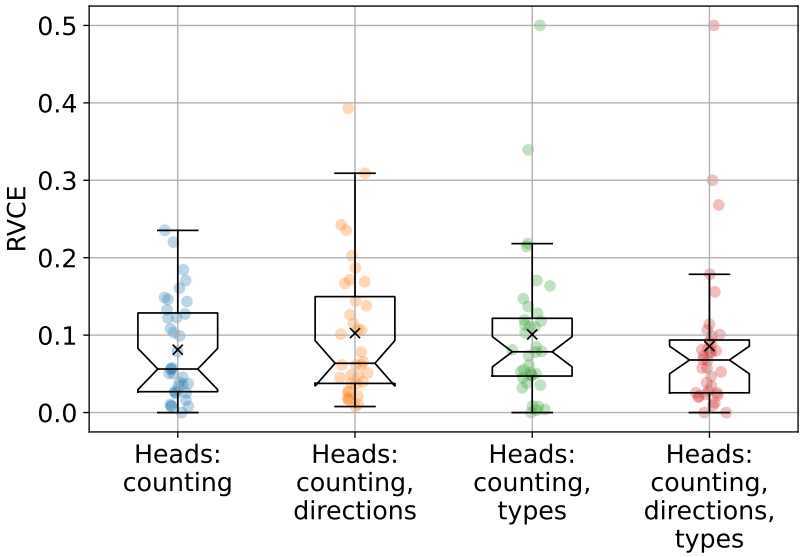


Figure 4.19: Performance of the vehicle counting prediction with respect to the different configuration of prediction heads.

4.14 Ensembling

Motivation When evaluating the models from previous experiments, we noticed that some videos are hard to classify for some models. However, the same videos could be easier for other models. In this experiment, the difference between models is that each model is trained on a randomly generated subset of recordings from the training and validation sets. The test set is fixed for every model. We combine the resulting models into a single ensembling model by averaging the probabilities from every model, as described in Section 3.12.

Results The results are presented in Figure 4.20. The fact that we have such a large variance of errors when we train models that have the same number of training and validation samples but are shuffled for each model might signify that our training set is not large enough yet. However, we can observe that by combining outputs from more models we can reduce the probability that recording will be extremely misclassified as it happened in model trained with seeds 42, 46 and 47, see Figure 4.20.

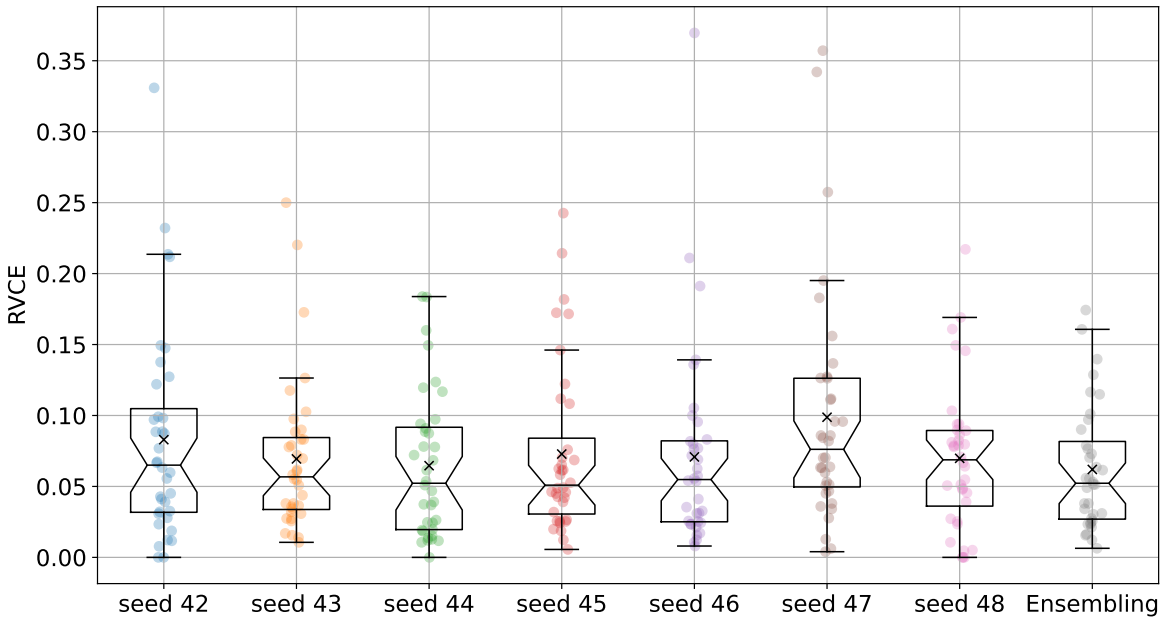


Figure 4.20: Results from ensembling of models trained and validated using different data.

4.15 Performance of vision-based versus audio module

Motivation Recall, that in addition to the fact that a recording device, such as a microphone, is cheaper than a camera and requires a less powerful device to process the input data online, the obvious advantage of microphones is that they can work properly in the absence of light, in fog, or when it rains. Unfortunately, we have no data recorded in low light conditions to compare how the performance of the vision-based module decreases in such scenarios.

Thanks to the available manual annotations that are in the form of the counts of passing vehicles for each recording, we can compare the RVCE of the proposed audio model with the RVCE of the vision-based module when using the manual annotations as the ground truth. We know that the accuracy of the vision-based module depends on the device from which the data come from. It can be explained by the fact that each recording device has a different frame rate, i.e. the frequency at which consecutive images are captured, or other parameters like the resolution, exposure, etc. Note that the audio module was trained on noisy labels obtained from the vision-based module. All models were evaluated on the same manually created labels. The manual annotation process is described in Section 3.4.

Results The results presented in Figure 4.21 show both the performance of the audio module and the performance of the vision-based module. The experiment shows that depending on the camera, mean RVCE of vision-based module varies from 0.03 for RX100 and Mobius to 0.25 for iPhone. The sharp drop in performance on iPhone videos can be explained by the fact that the video-based module tracks cars using their license plates, which were heavily blurred due to the low frame rate on the iPhone. Audio module has mean RVCE of 0.07, which can be considered a good performance for a vehicle counting task.

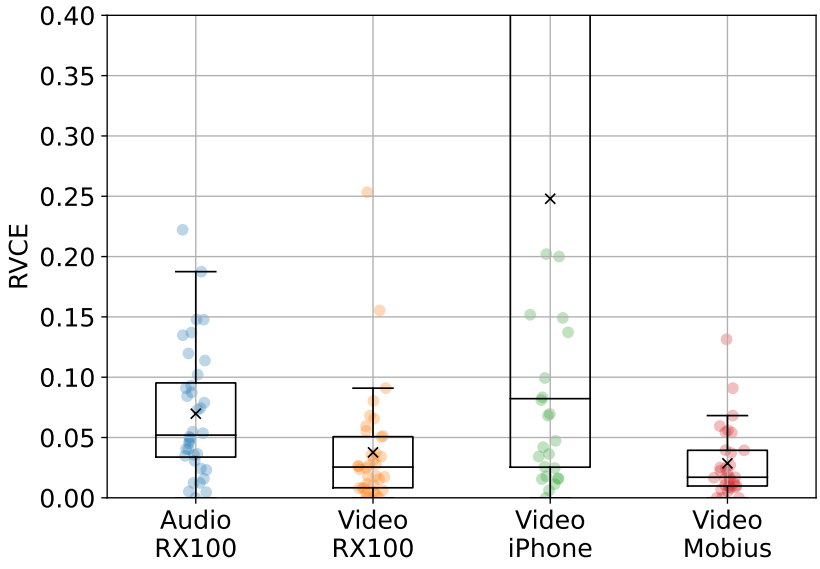


Figure 4.21: Comparison of the performance of the audio module trained on noisy annotations obtained from the video-based module and the performance of the video-based module using videos recorded with three different devices. All errors were evaluated using the same manually created labels as the ground-truth.

4.16 Data set size

Motivation The size of the data set is considered a crucial part that influences the performance of a machine learning model. Typically, large data sets lead to better generalization, whereas small data sets are prone to overfitting. Unfortunately, in our application, data acquisition turned out to be a slow and laborious process that requires additional cost. This experiment aims to measure the impact of the training set size on the model performance in order to estimate how adding additional data will reduce the test error.

To this end, we compared the performance of the three prediction models from training sets of different sizes, while the validation and the test set were the same. The training sets were generated as follows. In each of the five folds, we have 21 training, 7 validation, and 7 testing files. To create smaller training data sets, we randomly remove files, leaving only half (11 files) and a quarter (6 files) of those.

Results According to Figure 4.22, the size of the data set has a great influence on the performance of the model. The addition of more training data is believed to probably lead to a significant improvement in performance.

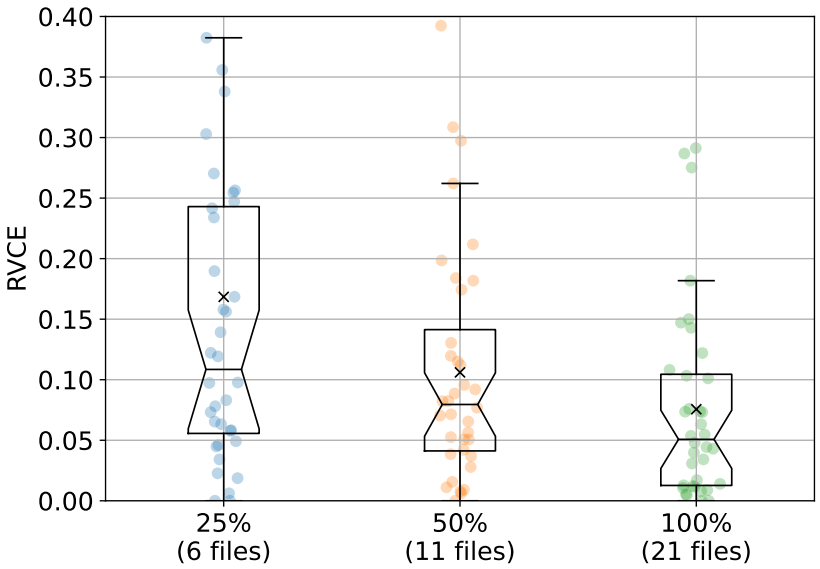


Figure 4.22: Dependence of model performance on the training data set size.

Chapter 5

Discussion

In this chapter, we summarize the experimental results, present the best model configuration, and discuss the methods that were the most effective and least effective for the performance of the model. We also discuss further improvements and future work.

5.1 The best model configuration

We conducted a comprehensive study of how a CNN such as ResNet can be applied to features extracted from a one-dimensional audio signal to compute statistics about passing vehicles. We explained the whole process from data acquisition and annotation, feature extraction, and model training, to obtaining the final statistics about the recording. In each step, we tried and compared the available methods and followed the state-of-the-art practices developed in this area so far. During our experiments, we gradually developed the model by adding and fine-tuning its different parts. We came to the conclusion that the model with the following configurations shows the best results:

- The best length of the time window is equal to 7 seconds but the model is not sensitive to the length of chosen window and show good results for window sizes between 6 and 9 seconds.
- The sampling rate of 22,050 Hz shows the best trade-off between processing speed and classification accuracy.
- The audio signal is represented by a spectrogram that is normalized to have zero mean and unit variance.
- Training input data are augmented using random erasing, random Gaussian blur, random colored noise, and random loss-pass and high-pass filters.
- Early stopping is used; the best model that minimizes RVCE in the validation data set is selected.
- The proposed inference method, described in Section 3.11, which works on overlapped time windows is used.

5.2 Efficiency of different factors

From our experiments, it follows that the most important factors that have the greatest influence on the performance of the model are the following:

- The size of the data set plays a crucial role in the performance of the model. Section 4.16 shows the empirical evidence by comparing the model performance trained in 6, 11, and 21 files. The performance of the model steadily improves with the training set size, and with the current 21 training files we are far from the diminishing returns regime.
- To obtain good results on different devices, it is important that samples collected from these devices are present in the training set; for more details, see Section 4.4. That is, we found that the generalization of the model to unseen devices is weak.
- Data augmentation techniques, described in Section 3.8, significantly help improve performance.
- Inference on overlapping windows, proposed in Section 3.11, helps to overcome the problem of near-boarder events, and it consistently improves the performance of the model; see the results in Section 4.11.

It is worth noting that things such as the ResNet model architecture backbone (see Section 4.9), the specialized class-balanced loss function (see Section 4.10), mel spectrogram (see Section 4.5) or ensembling (see Section 4.14) did not show noticeable benefits for the performance of the model.

5.3 Future work

An important ingredient in learning a generalizable neural network model is the number of training samples. From our experiments, it was clear that the amount of data we have at our disposal is a limiting factor. We believe that the performance of the model can be further improved by simply adding more data.

A promising direction for future work would be to develop an architecture that does not require the application of STFT to extract features from the audio signal. It might be beneficial to extract the features end-to-end using a suitable architecture that can be applied directly to the audio signal. In other words, the aim is to replace an STFT with a trainable part of a neural network.

In addition to that, another issue that remains unsolved is the performance of the audio module on recordings coming from devices that did not appear in the training set. We believe that the problem can be partially solved by applying techniques developed in the area of domain adaptation.

Finally, we also acknowledge that there is tremendous room for improvement in the model performance when predicting direction of vehicle movement and classification of the vehicle types.

Chapter 6

Conclusions

We have developed audio-based vehicle recognition software that takes an audio signal and outputs statistics about passing vehicles, such as counts, directions of movement, and vehicle types. The proposed approach is based on a classical convolutional neural network architecture such as ResNet and a number of methods used to preprocess the audio signal. We learned the proposed neural network in a semi-supervised fashion using imprecise annotations obtained automatically from the vision-based vehicle recognition model. An equally important part is a novel method that aggregates outputs from overlapped time windows. It helps our software to handle events that occur on the boundary of time windows. We used manual annotations to compare the performance of the proposed audio-based model with the vision-based model in vehicle counting task. We showed that the performance of audio module is comparable to that of vision-based module tested in good light conditions with mean RVCE of 7% and 3% respectively. We believe that the performance difference will be more pronounced at night, when classical vision-based models usually fail. Another benefit of the system lies in the speed with which the task of counting passing vehicles is solved compared to the vision-based approach. During development, we conducted extensive experiments that empirically measured the effects of the different parts of a recognition software. We described what worked best and discussed the results we have obtained. Lastly, because we have recordings produced by three different low-end consumer devices, we evaluated the model performance across different devices.

Bibliography

[1] Jakob Abeßer, Saichand Gourishetti, András Kátai, Tobias Clauß, Prachi Sharma, and Judith Liebetrau. Idmt-traffic: an open benchmark dataset for acoustic traffic monitoring research. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 551–555. IEEE, 2021.

[2] Horst Bischof, Martin Godec, Christian Leistner, Bernhard Rinner, and Andreas Starzacher. Autonomous audio-supported learning of visual classifiers for traffic monitoring. *IEEE intelligent systems*, 25(3):15–23, 2010.

[3] Ralph Beebe Blackman and John Wilder Tukey. The measurement of power spectra from the point of view of communications engineering—part i. *Bell System Technical Journal*, 37(1):185–282, 1958.

[4] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] Rich Caruana, Steve Lawrence, and C Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems*, 13, 2000.

[6] Haoze Chen and Zhijie Zhang. Hybrid neural network based on novel audio feature for vehicle type identification. *Scientific Reports*, 11(1):1–10, 2021.

[7] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. A comparison of audio signal preprocessing methods for deep neural networks on music tagging. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1870–1874. IEEE, 2018.

[8] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.

[9] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[10] Slobodan Djukanović, Jiří Matas, and Tuomas Virtanen. Robust audio-based vehicle counting in low-to-moderate traffic flow. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1608–1614. IEEE, 2020.

[11] Slobodan Djukanović, Yash Patel, Jiři Matas, and Tuomas Virtanen. Neural network-based acoustic vehicle counting. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 561–565. IEEE, 2021.

[12] Rubens Cruz Gatto and Carlos Henrique Quartucci Forster. Audio-based machine learning model for traffic congestion detection. *IEEE Transactions on Intelligent Transportation Systems*, 22(11):7200–7207, 2020.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[15] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.

[16] Khaled Koutini, Hamid Eghbal-zadeh, Florian Henkel, Jan Schlüter, and Gerhard Widmer. Over-parameterization and generalization in audio classification. *arXiv preprint arXiv:2107.08933*, 2021.

[17] Martin Krzywinski and Naomi Altman. Visualizing samples with box plots. *Nature methods*, 11(2):119–120, 2014.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[19] AY Nooralahiyan, Howard R Kirby, and D McKeown. Vehicle classification by acoustic signature. *Mathematical and Computer Modelling*, 27(9-11):205–214, 1998.

[20] Sebastian Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018.

[21] Aivars Severdaks and Martins Liepins. Vehicle counting and motion direction detection using microphone array. *Elektronika ir Elektrotechnika*, 19(8):89–92, 2013.

[22] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[23] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 8(3):185–190, 1937.

[24] Tao Wang and Zhigang Zhu. Multimodal and multi-task audio-visual vehicle detection and classification. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, pages 440–446. IEEE, 2012.

[25] Tao Wang, Zhigang Zhu, and Clark N Taylor. Multimodal temporal panorama for moving vehicle detection and reconstruction. In *2011 IEEE International Symposium on Multimedia*, pages 571–576. IEEE, 2011.

[26] Alicja Wieczorkowska, Elżbieta Kubera, Tomasz Słowik, and Krzysztof Skrzypiec. Spectral features for audio based vehicle and engine classification. *Journal of Intelligent Information Systems*, 50(2):265–290, 2018.

[27] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book. *Cambridge university engineering department*, 3(175):12, 2002.