LEUPHANA
UNIVERSITÄT LÜNEBURG

Institut für Wirtschaftsinformatik

Bachelorarbeit

Vergleich von Audioverarbeitungsmethoden für maschinelles Lernen am Beispiel der Verkehrszählung

Comparing Audio Processing Approaches for Machine Learning by the Example of Traffic Counting

Vorgelegt von:

Meier, Nils-Hendrik
nils-hendrik.meier@stud.leuphana.de
Studiengang: B. Sc. Wirtschaftsinformatik

Erstgutachter: Prof. Dr. rer. nat. Burkhardt Funk
Zweitgutachter: M. Sc. Jonas Scharfenberger

Eingereicht am: 04.05.2022

Zusammenfassung

Diese Arbeit verfolgt das Ziel, mit Hilfe von Machine Learning ein audiobasiertes System zur Verkehrszählung zu entwickeln und damit einen Beitrag für eine effizientere Verkehrssteuerung zu leisten. Dazu wurde ein Datensatz in Form von Videos erhoben und im Preprocessing in Feature und Label transformiert. Im Machine Learning Teil der Arbeit werden die Waveform und Spektrogramm Modelle trainiert und verglichen. Dabei konnten Abweichungen von unter 1% erreicht werden. Zur weiteren Evaluation wurden Testdaten mit verschiedenen Verkehrsdichten aufgenommen. Die Abweichungen hängen stark vom Verkehrsfluss ab und liegen zwischen 1% und 50%.

Abstract

This work pursues the goal of using machine learning to develop an audio-based system for traffic counting and thus contribute to more efficient traffic control. For this purpose, a data set in the form of videos was collected and transformed into feature and label in preprocessing. In the machine learning part of the work, the waveform and spectrogram models are trained and compared. Deviations of less than 1% could be achieved. For further evaluation, test data with different traffic densities were recorded. The deviations depend strongly on the traffic flow and lie between 1% and 50%.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der Straßenverkehr ist für einen großen Anteil der Luftverschmutzung im städtischen Raum verantwortlich. Je mehr Fahrzeuge unterwegs sind, desto größer ist die Umweltbelastung durch Feinstaubemissionen und Abgase der Motoren (Reek, 2019). Volle Straßen und insbesondere Abbremsen und Beschleunigen verstärken den Effekt. Unter dem Überbegriff *Intelligent Transportation Systems* werden Anwendungen entwickelt, die mit innovativen Ideen den Straßenverkehr effizienter gestalten und damit einhergehend die Nachhaltigkeit verbessern sollen (*Intelligent Transport*, 2017). Dazu zählen unter anderem auch Ampelanlagen mit integrierten intelligenten Systemen, die anfahrende Fahrzeuge über externe Sensoren erkennen und die Ampelschaltung an die Tageszeiten anpassen. Um den Verkehrsfluss weiter zu verbessern und die Verkehrssicherheit zu erhöhen, fordert der TÜV-Verband den weiteren Ausbau von intelligenten Verkehrssteuerungen (Roy & Shahd, 2020). Ein Problem genannter Systeme liegt in der Erfassung über in die Straße eingebettete induktive Kontaktschleifen, deren Installation und Wartung hohe Kosten verursacht (Augsburger Allgemeine, 2018). Diese Arbeit verfolgt daher das Ziel, eine audiobasierte Anwendung für die Verkehrszählung mit Hilfe von maschinellem Lernen zu entwickeln. Im Gegensatz zu fest integrierten Systemen lassen sich audiobasierte Systeme kostengünstig installieren und einfach warten. Die Ergebnisse können sowohl zur Analyse der Auslastung einzelner Straßen genutzt werden als auch als Input für weitere Verkehrssteuerungen dienen. Ein Use-Case wäre die Ergänzung der induktiven Fahrzeugerkennung durch audiobasierte Systeme an Zufahrtsstraßen von Ampelkreuzungen, um bereits frühzeitig über den ankommenden Verkehr informiert zu sein und Ampelschaltungen entsprechend anzupassen. Städte könnten auf Basis der gewonnenen Echtzeitdaten Staus vorhersagen und Fahrzeuge durch gezielte Hinweise um stark befahrene Strecken herumleiten.

Auf technischer Seite gibt es zwei übergeordnete Methoden der Audioverarbeitung für das maschinelle Lernen: Entweder werden die Audiodaten in ihrer Form als numerische Zeitreihe belassen oder alternativ in die Darstellung als Spektrogramm transformiert. Im Machine Learning Teil der Arbeit werden diese beiden Ansätze miteinander verglichen und es wird untersucht, ob es in Bezug auf die Verkehrszählung wesentliche Unterschiede in der Genauigkeit gibt. Innerhalb der Herangehensweisen wird durch die Verwendung verschiedener Modellarchitekturen ebenfalls ein Vergleich vorgenommen, ob State-of-the-Art Modelle bessere Ergebnisse erzielen als weniger komplexe Architekturen. Gerade in der Bildverarbeitung erzielen State-of-the-Art Modelle wie etwa ResNet50 deutlich bessere Ergebnisse als weniger komplexe Modelle (Zawadzka-Gosk, Wołk, & Czarnowski, 2019). Der Vergleich der Modelle soll daher darüber Aufschluss geben, ob diese Feststellung auch für die Audiodomäne gilt.

Zusammenfassend befasst sich diese Arbeit mit den folgenden Fragestellungen:

1. Mit welcher Genauigkeit kann ein audiobasiertes System vorbeifahrende Fahrzeuge im innerstädtischen Verkehr erkennen und ist diese ausreichend für eine Nutzung innerhalb der beschriebenen Use-Cases?

2. Welche Faktoren spielen bei der erfolgreichen Erkennung von Fahrzeugen eine wichtige Rolle und wie kann die Gegebenheit dieser Faktoren sichergestellt werden?

3. Welche Herangehensweise erzielt innerhalb der technischen Realisierung bzw. bei der Audioverarbeitung bessere Ergebnisse in Bezug auf die erste Fragestellung?

4. Gibt es innerhalb der Herangehensweise wesentliche Unterschiede zwischen verschiedenen Netzwerkarchitekturen oder spielt die Komplexität der Modelle eine untergeordnete Rolle?

Um die Fragestellungen zu beantworten und die Zielsetzung zu erreichen, wird zunächst die Vorgehensweise in Arbeiten mit ähnlicher Thematik untersucht. Daraus werden wichtige Inhalte abgeleitet und zusammengeführt, die in die eigene Ausarbeitung mit einfließen sollen. Im nächsten Schritt wurde ein Datensatz in Form von Videos erhoben. Zur Auswertung der Daten, also dem Erkennen von Fahrzeugen in den Videos, werden Methoden aus dem Feld Computer Vision in Form von YOLOv4 genutzt. Im Anschluss erfolgen das Feature Engineering entsprechend der beiden genannten Herangehensweisen und das Training der verschiedenen Modelle. Um die erste Fragestellung zu beantworten, wird die Genauigkeit der Erkennung von Fahrzeugen auf Testdaten erneut überprüft. Abschließend werden die Erkenntnisse der Arbeit zusammengefasst und die Grenzen des entwickelten Systems in Bezug auf mögliche Use-Cases diskutiert. Zudem wird ein Ausblick für den weiteren Forschungsbedarf gegeben, der im Anschluss an diese Arbeit erfolgen könnte.

2 Literature Review

In diesem Teil der Arbeit wird der aktuelle Stand der Forschung zum Thema Verkehrszählung untersucht. Daher wird zunächst die systematische Literaturauswahl einschließlich genutzter Stichworte sowie Auswahlkriterien erläutert. Anschließend werden die ausgewählten Arbeiten vorgestellt und die Vorgehensweise der Autoren in Teilen miteinander verglichen sowie grundlegende Erkenntnisse zusammengefasst.

Für die Suche nach Literatur wurden die Datenbanken IEEE-Explore, ArXiv und Springer Link sowie die Suchmaschine Google Scholar genutzt. Bei der Suche wurden im wesentlichen die folgenden Stichworte, unterteilt in drei Kategorien, in Kombination miteinander verwendet:

- **Audio:** Audio-Based, Acoustic, Audio Signal
- **Verkehr:** Traffic [Flow, Density], Vehicle, Car
- **Zählung:** Counting, Estimation, Measurement, Detection

Die Ergebnisse der Suche lassen sich in drei Themenfeldern zusammenfassen: Einen großen Teil bilden Arbeiten, die auf Basis von Bildmaterial bzw. Videos Fahrzeuge erkennen und damit eine Zählung vornehmen oder den Verkehrsfluss bewerten. Ein Beispiel dafür ist die Arbeit von Uke & Thool (2013), in der die Autoren auf Basis von Background-Substraction eine Erkennung der Fahrzeuge im fließenden Verkehr vornehmen. Diese Art der Verkehrszählung findet jedoch nicht mit Audiodaten statt und ist somit für diese Arbeit irrelevant. Eine zweite Gruppe bilden Arbeiten, die sich mit der Erkennung von bestimmten Geräuschen innerhalb des Straßenverkehrs befassen. Islam & Abdel-Aty (2022) haben bspw. ein System entwickelt, das auf Basis von Audiodaten die Sirenen von Einsatzfahrzeugen erkennt und damit zur Sicherheit im Verkehr beitragen soll. Auch diese Art von Arbeiten wird aussortiert, da sie sich nicht mit der Verkehrszählung befassen. Die dritte Gruppe bilden schließlich Arbeiten, die sich thematisch mit der audiobasierten Verkehrszählung befassen. Als weiteres Kriterium wurde die Veröffentlichung ab 2015 gewählt, um den Fokus auf aktuelle Entwicklungen der Forschung zu legen. Aus den verbleibenden Arbeiten wurde eine Auswahl getroffen, die verschiedene Ansätze umfasst und damit ein breites Spektrum an Methoden abdeckt.

Golovnin et al. (2021) nutzen den Urban8K-Datensatz, der über 8.000 Audiosamples mit städtischen Geräuschen aus 10 verschiedenen Klassen umfasst. Die Audiodaten werden in sich überlappende Fenster unterteilt, aus denen MFCCs (Mel Frequency Cepstral Coefficients) als Feature extrahiert werden. Auf Basis dieser Feature werden ein MLP-Modell sowie ein CNN-Modell trainiert und miteinander verglichen. Dabei erzielt das CNN eine Genauigkeit von etwa 92% und schneidet damit etwa 6% besser ab als das MLP. Um die Performance auf weiteren Daten zu evaluieren, wird ein

Datensatz aus Open-Source Daten in Kombination mit eigenen Aufnahmen zusammengestellt und von den Modellen vorhergesagt. Das CNN erzielt im Durchschnitt eine Genauigkeit von 84.2%, wobei bei den eigenen Aufnahmen teilweise nur etwa die Hälfte der vorbeifahrenden Fahrzeuge richtig erkannt wird. Zusammenfassend erzielen die Modelle also auf den Trainingsdaten gute Ergebnisse, die auf dem Testset nicht bestätigt werden konnten.

Einen anderen Ansatz verfolgen Djukanovic, Matas, & Virtanen (2020), die sich in ihrer Arbeit auf Straßen mit zwei Fahrspuren in die gleiche Richtung beschränken. Zur Datenerhebung wird eine Kamera genutzt, die Bilder sowie Audio mit einem Kanal aufnimmt. Auf Basis der Bilder wird annotiert, wann die Autos die geringste Distanz zur Kamera besitzen. Die Kamera ist während der Aufzeichnung in Richtung der ankommenden Fahrzeuge gedreht, daher ist dieser Moment bei Verlassen der rechten Bildseite erreicht. Basierend auf diesem Zeitpunkt stellen die Autoren eine Funktion auf, die unter Annahme einer konstanten Geschwindigkeit den Abstand zwischen ankommenden Fahrzeugen und Mikrofon modelliert. Sobald die Funktion ein lokales Minimum erreicht, gilt ein Fahrzeug als vorbeigefahren. Zur Vorhersage der Funktion wird ein Support-Vector-Regressor auf Basis von Audio Features (Short-Term-Energy, Top-Right-Frequency, High-Frequency-Power, Log-Melspectrogram) trainiert. Die Auswertung erfolgt, indem alle lokalen Minima unterhalb einer Schwelle gezählt werden. Diese Schwelle wird dabei so gewählt, dass sich die Anzahl der False-Positives und False-Negatives möglichst ausgleicht. Mit dieser Methodik konnten die Autoren eine relative Abweichung von unter 2% erzielen.

In der weiteren Arbeit an der vorherigen Methodik wurde die SVR zur Vorhersage der Distanzfunktion durch ein zweistufiges Neuronales Netz ersetzt. Eine weitere Änderung liegt in der Invertierung der Distanzfunktion, wodurch ein Fahrzeug beim Vorbeifahren ein lokales Maximum erzeugt. Die Auswertung der Distanz erfolgt nicht mehr durch das reine Zählen von lokalen Maxima, sondern durch ein 1D-CNN. Im Vergleich zur vorherigen Methodik konnten die Ergebnisse durch die Anpassungen weiter verbessert werden (Djukanović et al., 2020).

Im Gegensatz zur den vorherigen Arbeiten, die Audiodaten mit einem Kanal nutzen, verwenden Marciniuk, Szczodrak, & Czyzewski (2018) zwei Mikrofone zur Aufnahme der Straße. Das gewählte Setup basiert auf einem Arduino Mega, der mit zwei Mikrofonen sowie einem Beschleunigungsmesser mit integriertem Gyroskop verbunden ist. Durch die Verwendung von getrennten Mikrofonen mit seitlichem Versatz sind die Autoren in der Lage, die Richtung der vorbeifahrenden Fahrzeuge zu ermitteln. Sobald das Audiosignal eine bestimmte Schwelle überschreitet, gilt das als ein vorbeifahrendes Fahrzeug. Über den Beschleunigungsmesser ermitteln die Autoren, um welche Art von Fahrzeug es sich handelt. Dahinter steckt die Annahme, dass größere Fahrzeuge mehr Vibrationen und Luftbewegungen auslösen, die entsprechend von den

Sensoren erfasst werden können. Insgesamt können geringe Abweichungen in der Erkennung von Fahrzeugen erzielt werden, wobei die Autoren die Performance lediglich auf einer 30s langen Aufnahme validieren.

Shigemi Ishida et al. (2016) nutzen in ihrer Arbeit ebenfalls zwei seperate Mikrofone und haben ein System entwickelt, das aus den folgenden drei Komponenten besteht: Der Sound Retriever nimmt das Signal der beiden Mikrofone auf und wendet einen Low-Pass-Filter mit einer Cutoff-Frequenz von 2.5kHz an. Daran anschließend berechnet der Sound Mapper die Cross-Correlation zwischen den beiden Aufnahmen und erstellt damit eine sogenannte Soundmap. Damit ist der fortlaufende Unterschied gemeint, mit dem der Schall bei den Mikrofonen aufgenommen wird. Im dritten Schritt können durch Dynamic Time Warping (Template Matching) vorbeifahrende Autos in der Soundmap gefunden und gezählt werden. Die entsprechenden Templates wurden auf Basis von erhobenen Daten für beide Richtungen abgeleitet. Beim Template Matching trat vermehrt das Problem auf, dass fehlerhaft Autos erkannt wurden. Die Autoren stellten fest, dass das System mit dicht aufeinander folgenden Fahrzeugen Probleme hat.

Ein umfangreicheres Setup nutzen Na et al. (2015), die durch ein Array an Mikrofonen neben der reinen Anzahl der Fahrzeuge auch die Nutzung der verschiedenen Spuren und Geschwindigkeit sowie Typ der vorbeifahrenden Fahrzeuge bestimmen. Die Methodik basiert auf groben und feinen Erfassungsbereichen, die die Zuordnung der Fahrzeuge in eine Richtung und zu einer Fahrspur ermöglichen. Die Realisierung erfolgt mit zwei verschiedenen Modulen: Das Lane Detection Modul erkennt die Fahrspuren der Straße und das spurbezogene Modul zur Überwachung nutzt diese zur Erfassung der genannten Kennwerte. Die Autoren evaluieren ihre Methodik auf einem halbstündigem Testset und kommen zu dem Ergebnis, dass die korrekte Erkennung der Fahrzeuge stark von der Fahrspur abhängt. Die Fehlerrate der Fahrzeugerkennung liegt insgesamt bei etwa 13%, die Bestimmung der weiteren Kennwerte erzielt deutlich schlechtere Genauigkeiten.

Zusammenfassend wird aus den vorgestellten Arbeiten deutlich, dass es verschiedene Ansätze gibt, auf Basis von Audiodaten die vorbeifahrenden Fahrzeuge zu ermitteln. Die Ergebnisse hängen dabei stark von den gewählten Bedingungen und ggf. gesetzten Einschränkungen ab, wodurch ein übergeordneter Vergleich nicht sinnvoll wäre. Jedoch lassen sich aus den verschiedenen Arbeiten jeweils Inhalte ableiten, die in diese Arbeit übernommen werden sollen. Darunter zählt unter anderem die Feststellung von Djukanovic et al. (2020), dass sich False-Negatives und False-Positives ausgleichen und damit die Gesamtzahl der vorhergesagten Fahrzeuge stimmt. Das genauere Vorgehen wie Aufnahmesetup und Verarbeitung der Daten wird in den nachfolgenden Kapiteln näher thematisiert.

3 Datenerhebung

Im ersten Teil des Vorhabens geht es darum, einen geeigneten Datensatz zusammenzustellen, der die Basis für den Vergleich der verschiedenen Ansätze der Audioklassifikation bildet. Dazu wurden Rohdaten in Form von Videos erhoben und anschließend im Preprocessing (s. Kapitel 4) in eine nutzbare Form mit Features (reine Audiodaten) sowie Labels (wann Autos vorbeigefahren sind) transformiert. Die Entscheidung für eine eigene Datenerhebung lässt sich damit begründen, dass die Form der Daten für die entsprechenden Anforderungen gewählt sowie die Anzahl der Aufnahmen bei Bedarf angepasst werden kann. Gleichermaßen kann eine konstante Qualität der Daten sichergestellt werden, da für die gesamte Datenerhebung ein einheitliches Aufnahmesetup verwendet wird und auch die folgenden Schritte im Preprocessing darauf aufgebaut sind. Im Gegensatz dazu müsste bei Verwendung von externen Datensätzen wie bspw. von Kaggle oder Google Datasets darauf vertraut werden, dass eine ausreichende bis hohe Datenqualität vorhanden ist und nicht durch fehlerhafte Daten (z.B. ungenaues Labeling) die Ergebnisse in weiteren Teilen der Ausarbeitung beeinflusst und verfälscht werden. Ebenfalls zu beachten sind mögliche Urheberrechte, die bei der Verwendung fremder Daten eine Rolle spielen - diese entfallen bei einer eigenen Datenerhebung (D'agostino, 2022). Ein Nachteil der eigenen Datenerhebung liegt im erhöhten Zeitaufwand, jedoch überwiegen in diesem Fall die Vorteile.

In den folgenden beiden Abschnitten wird zunächst die Auswahl des Aufnahmesetups erläutert und begründet sowie anschließend eine Zusammenfassung der wichtigsten Zahlen zum finalen Datensatz vorgestellt.

3.1 Aufnahme-Setup

Wie bereits genannt erfolgt die Erhebung der Daten in Form von Videoaufnahmen. Die Aufnahmen und damit auch der endgültige Datensatz beschränken sich auf den fließenden Verkehr, da dieses Setting eine deutlichere Trennung der einzelnen Fahrzeuge ermöglicht und potenziell zu besseren Ergebnissen führt. Im Gegensatz dazu ist bei Aufnahmen mit stockendem oder sogar stehendem Verkehr, wie etwa an Kreuzungen, aus dem reinen Audiosignal nicht eindeutig zu identifizieren, wann ein Fahrzeug wirklich vorbeifährt oder ob es nur mit laufendem Moter steht. Generell gibt es zwar einen Unterschied im Geräusch zwischen Motor im Leerlauf und Anfahren, jedoch ist dieser nicht so eindeutig wie die Unterschiede im fließenden Verkehr. Die Videoaufnahmen der Straße erfolgen aus seitlicher Perspektive, damit innerhalb der Objekterkennung im Preprocessing Fahrzeuge gut erkannt werden können und die Qualität des Labeling Prozesses sichergestellt wird. Zudem kann bei seitlicher Perspektive gut festgelegt werden, ab wann ein Fahrzeug als vorbeigefahren gilt und dementspre-

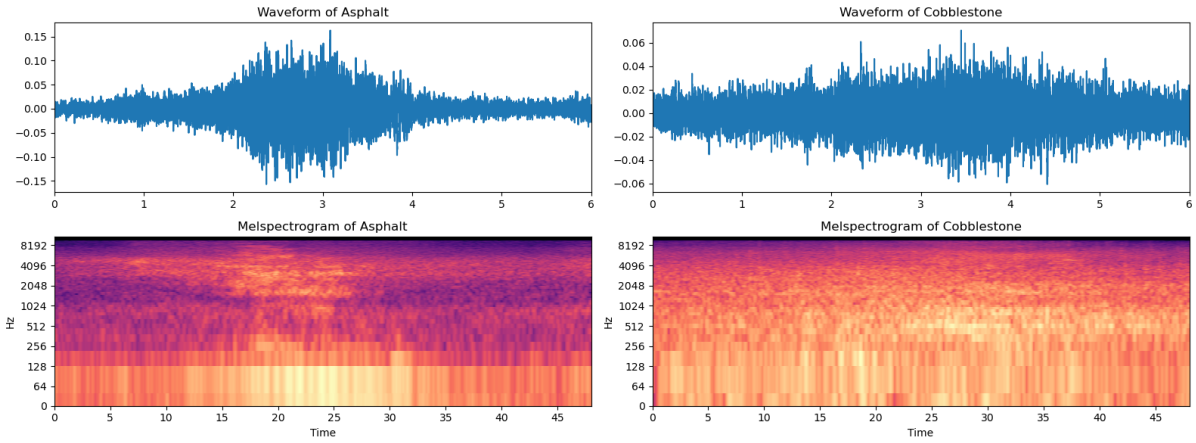

Abbildung 1: Vergleich des Audiosignals zwischen Asphalt und Pflaster

chend ein Label zu diesem Zeitpunkt gesetzt werden muss (s. Kapitel 4.1.2).

Zur Aufnahme der Videos wurde eine normale Handykamera mit Weitwinkel genutzt, wobei zur Reduzierung der Datengröße die Qualität der Videos auf 720p beschränkt wird. Die ersten Aufnahmen wurden mit dem internen Mikrofon des Smartphones erhoben, beim Auswerten der Videos ergab sich jedoch das folgende Problem: Aufgrund der Lage des Mikrofons im Smartphone zeigt es bei der Aufnahme der Straße zwingend in eine Richtung zur Straße und erfasst damit die Fahrzeuge aus dieser Fahrtrichtung eher als die der Gegenrichtung. Damit einhergehend ergibt sich für das Labeling das Problem, dass zwischen Markierung, wann ein Auto im Bild vorbeigefahren ist, und dem Ausschlag in der Audiospur eine zeitliche Differenz entsteht und damit Feature und Label nicht mehr genau übereinander passen. Dieses Problem wurde gelöst, indem ein externes Mikrofon verwendet und direkt auf die Straße ausgerichtet wird. Dadurch werden Fahrzeuge aus beiden Richtungen gleich erfasst und zeitliche Verschiebung zwischen Bild und Ton entfällt.

Die Orte der Aufnahmen verteilen sich auf das Lüneburger Stadtgebiet und wurden maßgeblich nach der Verkehrsflussdichte der Straße ausgewählt. Bei der Auswahl wurde darauf geachtet, dass insgesamt möglichst viele Fahrzeuge aufgenommen werden können, jedoch kein durchgehender Verkehrsfluss vorhanden ist und nicht dauerhaft Fahrzeuge in der Audioaufnahme zu hören sind. Dadurch soll ein möglichst ausgeglichener Datensatz geschaffen werden, der im weiteren Teil der Arbeit zu guten Ergebnissen führt. Ein weiterer Aspekt, der bei der Auswahl der Aufnahmeorte eine Rolle spielt, ist der Untergrund bzw. die Beschaffenheit der Fahrbahn. Die Geräusche vorbeifahrender Fahrzeuge unterscheiden sich je nach Untergrund, wie etwa zwischen Asphalt und Kopfsteinpfaster (s. Abbildung 1). Da die Modelle in der Lage sein sollen, Fahrzeuge unabhängig vom Untergrund der Fahrbahn zu erkennen, muss sich der Datensatz auch entsprechend zusammensetzen und verschiedene Untergründe abdecken. Gleichermaßen sollte auch der Abstand des Mikrofons zur Fahrbahn variiert

werden, um nicht in der Anwendung auf einen festgelegten Abstand zur Straße angewiesen zu sein.

An dieser Stelle soll kurz auf die vermeintliche Problematik von E-Autos eingegangen werden, die aufgrund des Elektromotors leiser sind als herkömmliche Fahrzeuge mit Verbrennermotor und daher zu keinem Ausschlag in der Audioaufnahme führen. Dieses Problem wird durch die Beschränkung der Aufnahmeorte auf Stellen mit fließendem Verkehr gelöst, da bei Fahrzeugen ab einer Geschwindigkeit von ca. 20km/h die Geräusche der Reifen lauter sind als das eigentliche Motorengeräusch (Zeller, 2018, S. 288–290). Damit stellen E-Autos kein Problem dar und werden normal in der Audioaufnahme erfasst.

3.2 Datenschutz

Die Videoaufnahmen der Straße enthalten unter anderem Kennzeichen und damit datenschutzrechtlich relevante Inhalte, die eine Betrachtung des Datenschutzes nötig machen. Die Aufnahmen werden ausschließlich für die weitere Verarbeitung im Preprocessing und nicht weiter genutzt. Die Verarbeitung erfolgt dabei automatisiert auf Basis des dafür geschriebenen Codes, eine weitere Betrachtung der Videos erfolgt nicht. Durch die verhältnismäßig geringe Auflösung der Videos ist es zudem nicht möglich, Personen in den Autos zu erkennen und damit personenbezogene Daten zu generieren. In Kombination mit der gewählten Perspektive seitlich zur Straße lassen sich Kennzeichen schwer bis gar nicht erkennen, was ebenfalls eine Zuordnung von Personen verhindert. Abschließend werden die Videos nach erfolgreicher Durchführung des Vorhabens gelöscht und während der Durchführung nicht mit Dritten geteilt, um eine anderweitige Nutzung durch Dritte ebenfalls auszuschließen und die erhobenen personenbezogenen Daten nicht weiter zu sichern.

4 Preprocessing

In Kapitel 3 wurde die Erhebung von Rohdaten in Form von Videos thematisiert, die in diesem Abschnitt der Arbeit weiter verarbeitet werden. Die Resultate der Datenerhebung liegen als `.mp4` Dateien vor und werden im Verlauf des Preprocessings in Feature und Label transformiert, damit nachfolgend darauf Machine Learning Modelle trainiert werden können. Die Features basieren auf der Tonspur der Videos, also wird diese zunächst extrahiert und zur weiteren Verarbeitung gesichert. Als Label werden in diesem Fall die Zeitpunkte innerhalb der einzelnen Aufnahmen verstanden, zu denen ein Fahrzeug an der Kamera des Smartphones vorbeigefahren ist. Daher wird aus den aufeinanderfolgenden Bildern der Videos abgeleitet, wann Fahrzeuge das Bild passiert haben.

In den nachfolgenden Abschnitten dieses Kapitels werden die Verarbeitung der Aufnahmen und das Feature Engineering auf den Audiodaten mit anschließender Data Augmentation behandelt und tiefergehend erläutert.

4.1 Verarbeitung der Aufnahmen

Wie bereits beschrieben ist das Ziel des Preprocessings, Feature und Label zu generieren. Daher ist die Verarbeitung der Videos im Wesentlichen aus zwei voneinander unabhängigen Schritten zusammengesetzt:

1. **Extraktion des Audiosignals**
   Die Extraktion des Audiosignals wird in Python mit externen Bibliotheken realisiert. Die Audiodaten werden aus den jeweiligen Videos extrahiert und als einzelne WAV-Dateien gesichert. Die Speicherung im WAV-Format ist darin begründet, dass die Daten in der unkomprimierten Form gesichert werden (nicht wie etwa beim `.mp3` Format) und damit die weitere Verarbeitung beschleunigt wird. Ebenfalls erfolgt in diesem Schritt ein Downsampling der Audiodaten von 48kHz auf ca. 22kHz, um den Speicherbedarf der unkomprimierten Dateien zu verringern. Zudem beinhalten die beim Downsampling entfernten Frequenzbereiche keinen hohen Informationsgehalt für die Thematik der Verkehrszählung, weil sich die Geräusche im fließenden Verkehr durch Reifen und Motor eher im tieferen Frequnzbereich abspielen (Zeller, 2018, S. 271–272).

2. **Markierung bzw. Labeling des Bildmaterials**
   Der zweite Teil der Verarbeitung besteht darin, Fahrzeuge im Video zu erkennen und zu markieren, wann diese den Aufnahmeort passieren. Dieser Schritt ist erneut zweigeteilt und besteht zum einen aus der reinen Erkennung der Fahrzeuge im Bild und zum anderen aus dem Vorgehen, die erkannten Fahrzeuge

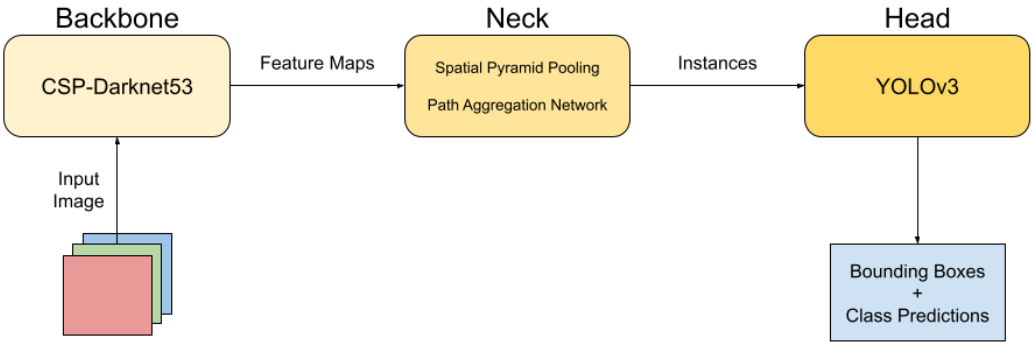

Abbildung 2: Vereinfacht dargestellter Aufbau von YOLOv4

als passiert einzuordnen. Diese beiden Schritte werden in den beiden folgenden Abschnitten im Detail erklärt und visuell dargestellt.

4.1.1 Objekterkennung mit YOLOv4

Der erste Schritt zum Labeling der Videos ist die Erkennung von Objekten innerhalb der einzelnen Bilder, in diesem Fall von Fahrzeugen (u.a. Autos, Busse und Motorräder). Zur Objekterkennung wurde in dieser Arbeit das YOLOv4 Framework in der DarkNet Implementierung genutzt, das in Python eine einfach Schnittstelle zur Verfügung stellt und verhältnismäßig schnell läuft im Vergleich zu anderen Frameworks. In diesem Abschnitt wird die Architektur von YOLOv4 näher betrachtet und anschließend ein Vergleich der Genauigkeit mit klassischen Methoden der Bildverarbeitung gezogen.

Abbildung 2 zeigt vereinfacht dargestellt den Informationsfluss durch die verschiedenen Komponenten der Objekterkennung, wie aus einem Inputbild Bounding Boxes sowie zugehörige Klassen abgeleitet werden. Die drei Abschnitte Backbone, Neck und Head erfüllen dabei verschiedene Aufgaben, die nachfolgend weiter spezifiziert werden. YOLOv4 nutzt im Backbone CSP-Darknet53, im Neck Spatial Pyramid Pooling sowie Path Aggregation Networks und im Head bestehende Komponenten von YOLOv3 (Bochkovskiy, Wang, & Liao, 2020, S. 7). Die Funktionen der einzelnen Komponenten sowie deren Zusammenspiel wird nachfolgend weiter spezifiziert.

Backbone

Allgemein werden die Netzwerke im Backbone dazu genutzt, mit Hilfe von Convolution und MaxPooling Schichten Features aus dem Input-Bild zu generieren und werden daher auch als Feature Extractor bezeichnet (Bochkovskiy et al., 2020, S. 2). Einige Beispiele sind VGG16, ResNet-50 oder ResNeXt50. YOLOv4 nutzt als Feature Extractor CSP-Darknet50, das zu den residualen Netzwerken zählt. Diese verbessern mit sogenannten Skipverbindungen den Informationsfluss und ermöglichen das Training von

großen Netzwerken (He et al., 2015). Die Abkürzung *CSP* steht für *Cross Stage Partial* und meint die Methodik, dass jeweils nur ein Teil des Bildes durch Convolution bearbeitet und anschließend mit dem verbleibenden Teil wieder zusammen geführt wird. Vorteil davon ist ein besserer Informations- bzw. Gradientenfluss, der das Training des Netzwerks erheblich beschleunigt und die Ergebnisse verbessert (Wang et al., 2019, S. 4–6). Output vom Backbone zum Neck sind Feature-Maps, also die Ausgabe von Convolution Schichten.

Neck

Das Neck stellt die Verbindung zwischen Backbone und Head dar und transformiert die Feature Maps weiter. Durch Path Aggregation Networks können einzelne zusammenhängende Instanzen und daraus die zugehörigen Bounding Boxes ermittelt werden. PANet nutzt eine Bottom-Up-Path-Aggregation sowie Adaptive-Feature-Pooling, um verschiedene Feature Ebenen miteinander zu kombinieren und daraus eine pixelgenaue Vorhersage der Instanzen treffen zu können (Liu et al., 2018, S. 3–5). Zweiter Bestandteil vom Neck ist das Spatial Pyramid Pooling, das Eingaben unabhängig von ihrer Größe durch 3-Level-Pooling zu Feature-Vektoren einheitlicher Länge transformiert. Hintergrund dieser Transformation ist, dass nachfolgende Dense-Schichten zur Klassifikation der gefundenen Instanzen eine konstante Größe benötigen, da die einzelnen Neuronen eine feste Anzahl an Gewichten haben (He et al., 2014, S. 1–2).

Head

Im letzen Schritt der Objekterkennung müssen die gefundenen Instanzen einer oder mehreren Klassen zugeordnet werden. YOLOv4 nutzt dafür die bestehende Methodik aus YOLOv3. Auf Basis eines Multilabel-Klassifikationsproblems wird für jede Instanz bzw. Bounding Box die Wahrscheinlichkeit ausgegeben, dass die Klasse zu finden ist. Ein Objekt kann daher keiner, einer oder mehreren Klassen zugeordnet werden (bspw. durch hierarchisch orientierte Klassen wie Apfel und Obst) (Redmon & Farhadi, 2018, S. 2). Endgültige Ausgabe vom Head ist daher ein Vektor je Objekt mit Wahrscheinlichkeiten für jede Klasse sowie den Koordinaten der zugehörigen Bounding Box.

Vergleich von YOLO und Frame Differencing

Eine weitere Variante der Objekterkennung ist das sogenannte Frame Differencing, bei dem die Veränderung zwischen zwei aufeinanderfolgenden Frames betrachtet wird. Über Erosion, Dilatation sowie Konturerkennung können daraus ebenfalls Bounding Boxes abgeleitet werden (I. Kartika & Shahrizat Shaik Mohamed, 2011). Abbildung 3 zeigt einen Vergleich von YOLO und Frame Differencing. Es ist ersichtlich, dass die Erkennung von YOLO wesentlich präsizer erfolgt. Ein weiterer Vorteil von YOLO liegt darin, dass nach den Klassen der Objekte gefiltert und damit etwa Radfahren-

Abbildung 3: Erkennung von Objekten über YOLO und Frame Differencing

de ignoriert werden können. Diese Option steht beim Frame Differencing nicht zur Verfügung, da die Art der Objekte nicht ermittelt wird. Insgesamt bietet YOLO damit mehr Möglichkeiten bei höherer Genauigkeit. Die erhöhte Verarbeitungsdauer spielt jedoch keine Rolle, da diese nicht in Echtzeit stattfinden muss und somit der zeitliche Aufwand keine Bedeutung hat.

4.1.2 Labeling der Videos

Im zweiten Teil der Videoverarbeitung geht es darum, die erkannten Objekte von YOLO weiter zu verarbeiten und die Zeitpunkte zu ermitteln, wann ein Fahrzeug die Aufnahme passiert hat. Abbildung 4 zeigt beispielhaft die verwendete Vorgehensweise beim Labeling. Für die Bounding Boxes der erkannten Objekte wird der Mittelpunkt berechnet, der sogenannte Centroid. Das zweite wesentliche Element ist eine Box in der Mitte des Bildes, nachfolgend als Center Box bezeichnet. Für jeden Frame, also jedes einzelne Bild, wird überprüft, ob sich der Centroid eines Objekts innerhalb der Center Box befindet (in der Abbildung durch die farblich veränderte Bounding Box dargestellt). Sobald der Centroid eines Fahrzeugs die Center Box wieder verlässt, hat es die Mitte des Bildes wieder verlassen und gilt damit als vorbeigefahren. Zu dem entsprechenden Zeitpunkt, der auf Basis der Frames bestimmt wird, kann dann ein Label gesetzt werden.

Ein Problem der gewählten Labeling Methodik liegt darin, dass Fahrzeuge auf der hinteren Spur verdeckt werden können. YOLO ist dann nicht in der Lage, das hintere Fahrzeug zu erkennen und dementsprechend gibt es für den Moment auch keine Bounding Box mit Centroid. Im ungünstigsten Fall passieren zwei Fahrzeuge zum exakt gleichen Zeitpunkt die Mitte des Bildes und somit wird nur das eine als vorbeigefahren wahrgenommen. Bei der Auswertung der Videos hat sich gezeigt, dass dieser Fall nur sehr selten auftritt und kaum Fahrzeuge nicht gelabelt werden. Einen Ansatz

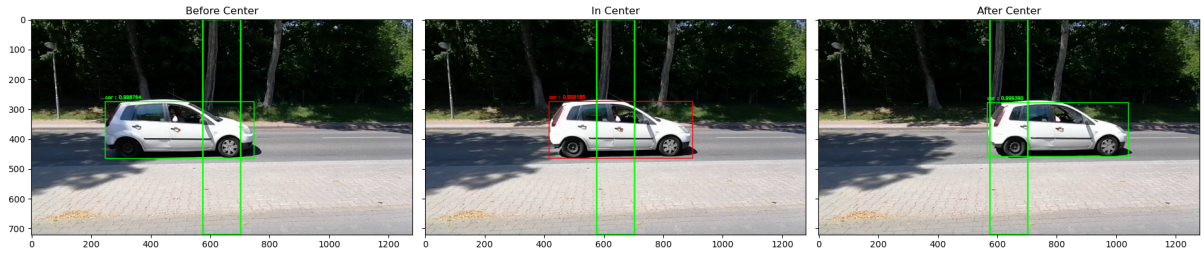

Abbildung 4: Vorgehensweise beim Labeling der Videos

zur Lösung dieses Problems könnte Objekt-Tracking bieten, das sich mit der Erkennung von gleichen Objekten in aufeinander folgenden Frames befasst (Dhiman, 2020). Die Methodik der Center Box müsste dann entsprechend angepasst werden, um alle Fahrzeuge zu erfassen. Damit einhergehend ergeben sich durch Ungenauigkeiten im Tracking weitere Probleme, die in Summe zu einer ähnlichen Fehlerquote beim Labeling führen könnten wie der gewählte Labeling-Prozess auf Basis der Center-Box.

Durch die Verarbeitung der Videos wurden diese transformiert und es liegen für die weiteren Schritte der Arbeit reine Audiodaten sowie Label in Form von Zeitpunkten zum Audio vor.

4.2 Feature Engineering & Data Augmentation

Im Anschluss an das Labeling der Videos werden die Audiodaten in kleinere Abschnitte unterteilt und mit dem entsprechendem Label kombiniert. Dadurch entstehen die Inputs `X` für die Machine Learning Modelle mit den Outputs `y`. Die Unterteilung des Audiosignals in Abschnitte sowie die Ermittlung des Labels ist in Abbildung 5 beispielhaft dargestellt und hängt von den folgenden Parametern ab:

- **Window Size:** Größe der einzelnen Abschnitte, die aus dem Audiosignal als Feature entnommen werden.
- **Step Size:** Verschiebung des Fensters bzw. Abstände der einzelnen Fenster innerhalb des Audiosignals.
- **Label Mode:** Modus, der für die Ermittlung des Labels genutzt wird. Entweder wird die Anzahl der Fahrzeuge in einem Abschnitt ermittelt oder ob überhaupt ein Fahrzeug vorbeigefahren ist.

Beispiel:
`window_size=1.5 & step_size=1.0 & label_mode='max'` steht dafür, dass jede Sekunde für ein 1.5s langes Fenster ermittelt wird, ob ein Fahrzeug vorbeigefahren

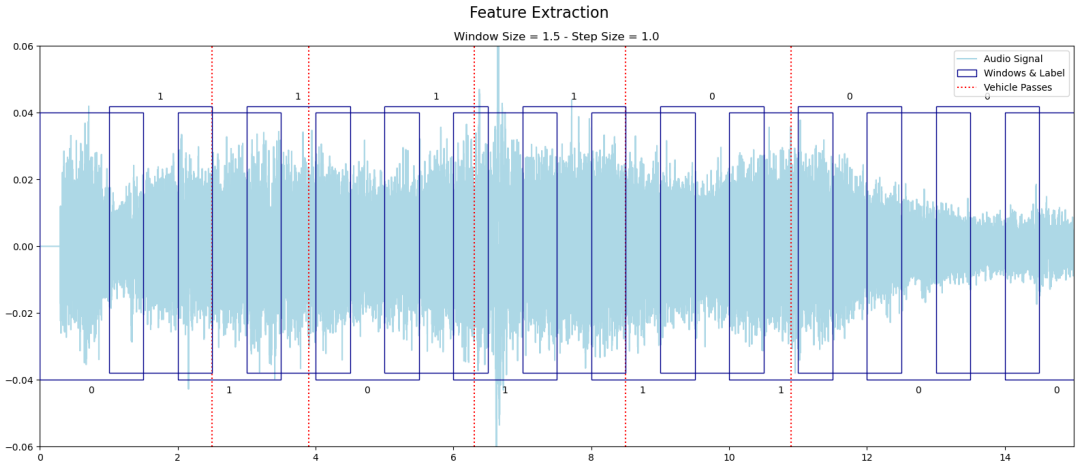

Abbildung 5: Unterteilung des Audiosignals in Abschnitte

ist oder nicht. Die Parameter des Feature Engineerings werden beim Training der Modelle als Hyperparameter betrachtet und dementsprechend optimiert.

Die Label sind als Klassen zu verstehen, wobei eine Klasse jeweils eine Anzahl von Fahrzeugen je Abschnitt repräsentiert. Generell handelt es sich bei der Problemstellung der Verkehrszählung um ein Regressions-Problem. Aus diesem Grund werden die Vorhersagen der Modelle in eine Regressions-Metrik (s. Kapitel 5) umgewandelt. Damit einhergehend können die Modelle später besser verglichen werden und die Auswertung der Ergebnisse erfolgt nicht nur auf Klassifikations-Metriken wie Accuracy oder F1-Score, die je nach Klassenverteilung ein falsches Bild der Vorhersagequalität liefern können (Yıldırım, 2020).

In den beiden folgenden Unterkapiteln wird das Feature Engineering für die verwendeten Ansätze über die Waveform des Audios sowie die Repräsentation durch Spektrogramme im Detail thematisiert. Dabei wird auch auf den Aspekt der Data Augmentation eingegangen, die im Training ein Overfitting der Modelle, also eine Überanpassung an die Trainingsdaten, verhindern und damit die Performance verbessern soll. Der Begriff Data Augmentation bezeichnet im Allgemeinen die künstliche Erzeugung von weiteren Trainingsdaten (van Dyk & Meng, 2001). Darunter fällt auf der einen Seite die synthetische Erzeugung neuer Daten auf Basis bestehender Datenpunkte, wie etwa durch das von Chawla et al. (2002) etablierte SMOTE Sampling Verfahren, auf der anderen Seite aber auch die Veränderung der bestehenden Samples innerhalb des Trainings. Im Falle von Neuronalen Netzwerken wird der Datensatz vor Beginn jeder Epoche durch Data Augmentation leicht verändert und ist dadurch nicht komplett identisch. Infolgedessen erzielen die Modelle nicht durch ausschließliches Auswendiglernen gute Ergebnisse, sondern müssen die tatsächlichen Merkmale ableiten und lernen (Takimoglu, 2021).

4.2.1 Spectrogram-Features

Audiodaten lassen sich durch verschiedene Formen repräsentieren, unter anderem durch Spektrogramme. Diese stellen die Intensität der verschiedenen Frequenzbereiche eines Audiosignals im Verlauf der Zeit dar und reduzieren das Signal somit auf seine wesentlichen Eigenschaften (Wyse, 2017). Die Berechnung der Spektrogramme erfolgt auf Basis der Fourier Transformation, die ein kontinuierliches Signal in seine einzelnen Frequenzbestandteile zerlegt und deren jeweilige Intensität ermittelt (Nussbaumer, 1981). Zur Berechnung von Spektrogrammen wird das Audiosignal in kleine Abschnitte unterteilt, auf denen die Fourier Transformation durchgeführt wird. Es gibt unterschiedliche Formen von Spektrogrammen, die je nach Anwendungsfall mehr oder weniger gut geeignet sind. Für diese Arbeit wurden die drei folgenden Feature ausgewählt:

- **Melspectrogram:** Bei Melspektrogrammen werden die Ergebnisse der Fourier Transformation auf die sogenannte Mel-Skala abgebildet. Diese basiert auf der menschlichen Wahrnehmung verschiedener Frequenzbereiche bei gleichbleibender Intensität und gewichtet Frequenzen entsprechend ihrer Wahrnehmung (Pedersen, 1965).

- **Chromagram:** Chromagramme bilden die Intensität der Frequenzen aus der Fourier Transformation auf einzelne Tonhöhen ab, die sich in Relation zueinander ordnen und in Oktaven zusammenfassen lassen. Dementsprechend geben Chroma Feature an, wie stark verschiedene Tonhöhen zu einem Zeitpunkt im Audiosignal vorhanden sind. (Shah et al., 2019)

- **Constant Q-Chromagram:** Das Constant Q-Chromagram (CQT) ist ähnlich zum Chromagram, nutzt jedoch bei der Zerlegung des Audiosignals in Frequenzen eine andere Methodik. Im Gegensatz zur Fourier Transformation, die eine lineare Einleitung der Frequenzen nutzt, verwendet die Constant-Q-Transformation eine logarithmische Skala und kann damit eine höhere zeitliche Auflösung in hohen Frequenzbereichen liefern. (Brown, 1991, S. 425–426)

Das genutzte Data Augmentation Verfahren basiert grundlegend auf SpecAugment von Google und wurde für die eigenen Zwecke leicht angepasst. Park et al. (2019) stellen in ihrer Arbeit das sogenannte *Frequency Masking* vor, bei dem zufällig ausgewählte Frequenzbereiche (mit Bezug auf Spektrogramme Zeilen der Matrix) 'abgedeckt' werden und daher im Training keine zusätzlichen Informationen liefern. Machine Learning Modelle sind daher gezwungen, alle Frequenzbereiche zu betrachten und nicht auf eine bestimmte Frequenz angewiesen zu sein. Die Methodik wurde dahingehend angepasst, dass die Abdeckung nicht auf Frequenzen oder Zeitpunkte beschränkt wird, sondern rein zufällig Bereiche der Spektrogramme abgedeckt werden.

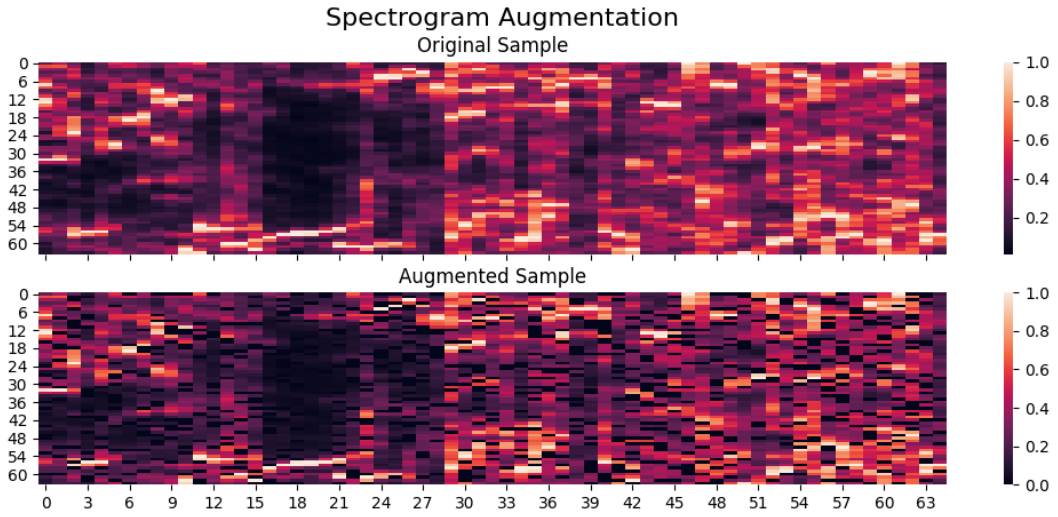

Abbildung 6: Data Augmentation für Spectrogram Features

In Abbildung 6 ist zu erkennen, wie das Sample nach der Data Augmentation abgedeckte Zellen bzw. Bereiche aufweist, die farblich schwarz markiert sind.

4.2.2 Waveform-Features

Für die Wave-Features bleiben die Audiodaten in ihrer bestehenden numerischen Form und werden lediglich wie beschrieben in Abschnitte unterteilt und mit einem Label versehen. Für die Data Augmentation wurden auf Basis der von Nanni, Maguolo, & Paci (2020, S. 4) vorgestellten Methoden drei verschiedene Varianten abgeleitet, die auf die Daten angewandt werden können:

- **Noise Injection:** Zum ursprünglichen Audiosignal werden normalverteilte Werte um 0 herum addiert. Dabei wird die Höhe der Ausschläge eines Samples mit einbezogen, indem der maximale Ausschlag mit in die Berechnung der Standardabweichung der Normalverteilung einfließt. Diese Methodik simuliert gewissermaßen leichte Hintergrundgeräusche, da die Differenzen der einzelnen Punkte des Signals durch die Addition direkt verändert werden.

- **Scaling:** Das Audiosignal wird mit einem um 1 normalverteilten Skalar s multipliziert. Durch die reine Multiplikation bleiben alle Verhältnisse und Muster im Signal generell unverändert, jedoch wird durch die Stauchung ($s < 1$) bzw. Streckung ($s > 1$) eine künstliche Änderung der Lautstärke erreicht.

- **Combined:** Diese Variante kombiniert beide vorherigen Methoden miteinander, d.h. das Audiosignal wird zunächst skaliert und anschließend mit Noise versehen.

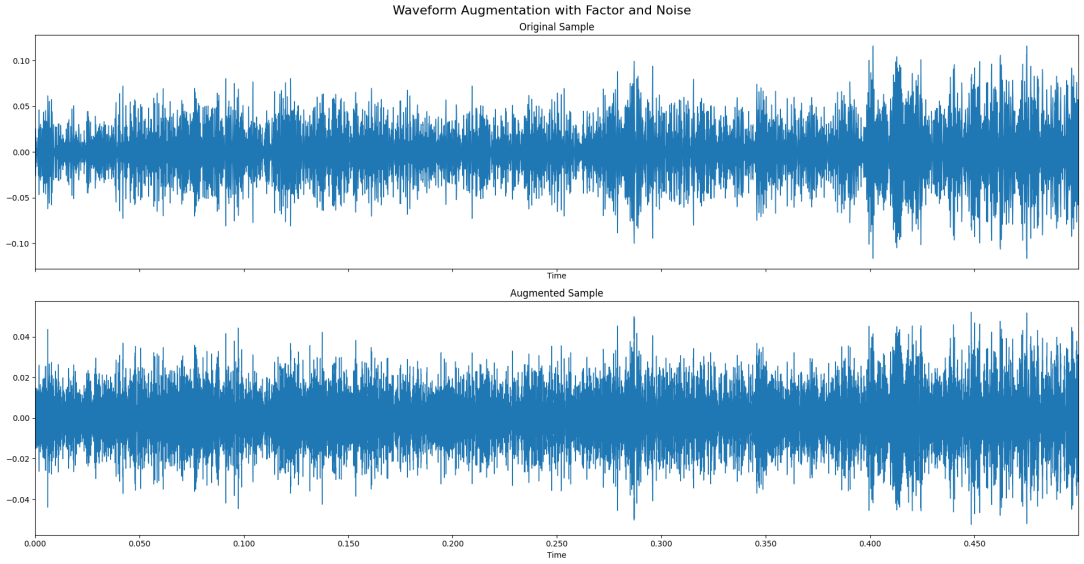

Abbildung 7: Data Augmentation für Waveform Feature

Abbildung 7 zeigt an einem Sample, wie die Data Augmentation unter Verwendung von Faktor sowie Noise das Signal verändert. Für diese Visualisierung wurde ein höherer Werte als `factor_scale` verwendet als beim Training der Modelle, um eine deutlichere Veränderung der Ausschläge zu erreichen.

5 Machine Learning

Nach Verarbeitung der Rohdaten und Feature Engineering erfolgt im nächsten Schritt der Arbeit das Training der Machine Learning Modelle. Dabei werden die Modelle der beiden Ansätze (Waveform und Spektrogramme) mit den entsprechenden Features trainiert und evaluiert. Es erfolgt sowohl innerhalb als auch zwischen den Vorgehensweisen ein Vergleich der verschiedenen Modellarchitekturen, um eine der Forschungsfragen beantworten zu können. Die Modelle innerhalb der Ansätze unterscheiden sich hinsichtlich ihrer Komplexität und dementsprechend auch der Anzahl der Gewichte innerhalb des Netzes. Der Hintergrund dabei ist die Fragestellung, ob die Erkennung von Fahrzeugen im wesentlichen Maße von Komplexität und Größe der Modelle abhängig ist. Die verwendeten Modelle werden in den entsprechenden Abschnitten näher erläutert (s. Kapitel 5.1.1 und 5.2.1). Nachfolgend wird das allgemeine Learning Setup hinsichtlich Vorgehen bei Training und Evaluation der Modelle erläutert.

Wie bereits in Abschnitt 4.2 erläutert, sind die Label als Klassen zu verstehen, die eine bestimmte Anzahl an Fahrzeugen je Abschnitt repräsentieren. Da es sich bei der Verkehrszählung jedoch um ein Regressionsproblem handelt, wurde eine eigene Metrik implementiert. Die Metrik ist basiert auf der Feststellung von Djukanovic et al. (2020), dass sich falsche Vorhersagen ausgleichen können und damit die Gesamtzahl an vorhergesagten Fahrzeugen wieder richtig ist. Der nachfolgende Code zeigt die Funktion zur Berechnung der Metrik:

```
def prediction_error(y_true, y_pred, normalize):
    cm = metrics.confusion_matrix(y_true, y_pred)
    true_count = np.dot(np.sum(cm, axis=1), np.arange(len(cm)))
    pred_count = np.dot(np.sum(cm, axis=0), np.arange(len(cm)))
    if normalize:
        return np.abs(true_count - pred_count) / true_count
    else:
        return np.abs(true_count - pred_count)
```

Die Klassenvorhersagen der Modelle werden in eine Konfusionsmatrix überführt, um nachfolgend die Anzahl der tatsächlichen Fahrzeuge sowie die Anzahl der vorhergesagten Fahrzeuge zu berechnen. Die Anzahl ergibt sich dabei aus dem Skalarprodukt der entsprechenden Anzahl von Fahrzeugen je Klasse und der Summe an Vorhersagen je Klasse. Die Ground Truth ist auf der y-Achse zu finden, daher wird für die tatsächliche Anzahl entlang Achse 1 aufsummiert und für die vorhergesagte Anzahl entlang Achse 0. Aus der absoluten Abweichung ergibt sich der absolute Vorhersagefehler (nachfolgend `abs_error`). Bei Division der absoluten Abweichung durch die tatsächliche Anzahl der Fahrzeuge ergibt sich der relative Vorhersagefehler (nachfol-

gend `rel_error`). Zur Bewertung der Ergebnisse werden der F1-Score und der relative Vorhersagefehler genutzt, um die Vorhersagen aus zwei Perspektiven zu betrachten und zu vergleichen.

Regularisierung wird beim maschinellen Lernen genutzt, um Overfitting zu verhindern und Modelle zu erhalten, die allgemeingültige Eigenschaften aus den Features ableiten können. Beim Training werden die folgenden Methoden der Regularisierung genutzt:

- **Early Stopping:** Falls sich der Validation Loss über 10 Epochen nicht weiter verringert, wird das Training gestoppt und die Gewichte des Zeitpunkts mit dem minimalen Validation Loss werden wiederhergestellt. Infolgedessen werden die Gewichte nicht weiter auf die Trainingsdaten optimiert und es kommt nicht zur Überanpassung. (*Keras documentation: EarlyStopping*, 2022; Mustafeez, 2020)

- **Dropout Layer:** In die einzelnen Modelle werden Dropout Schichten eingebaut, die jeweils einen bestimmten Anteil der Verbindungen zwischen den Neuronen deaktivieren. Als Resultat stehen weniger Pfade für den Informationsfluss zur Verfügung und die Modelle können die Trainingsdaten schwieriger auswendig lernen. (Brownlee, 2018; *Keras documentation: Dropout layer*, 2022)

- **Data Augmentation:** Auch die Data Augmentation, also die Veränderung der Trainingsdaten vor jeder Epoche, zählt zur Regularisierung. Die Vorteile der Data Augmentation und die genutzten Methoden für Waveform und Spektrogramme wurden bereits in Kapiel 4.2 thematisiert.

Ein weiterer wichtiger Aspekt beim Training von neuronalen Netzwerken ist die Learning Rate, die die Geschwindigkeit des Gradientenabstiegs und damit die Ergebnisse maßgeblich beeinflusst (Jordan, 2018). Der Auswahl der richtigen Learning Rate wird daher eine große Bedeutung beigemessen. Beim Training der Modelle werden die von Smith (2015) vorgeschlagenen Cyclical Learning Rates verwendet, wobei die Learning Rate während des Trainings in Zyklen zwischen einer unteren und einer oberen Schranke variiert. Durch die Variation der Learning Rate erreichen die Modelle schneller ein Optimum und es muss nicht manuell eine optimale statische Learning Rate ermittelt werden (Smith, 2015). Als Schranken werden 0.001 und 0.003 verwendet.

In den folgenden Unterkapiteln werden die genutzten Modellarchitekturen sowie die systematische Vorgehensweise zur Optimierung der verschiedenen Hyperparameter der Modelle dargestellt.

5.1 Spectrogram Modelle

5.1.1 Verwendete Modelle

Als Baseline Modell wird ein simples CNN verwendet, das durch drei aufeinander folgende Convolution Schichten Feature-Maps aus den Spektrogrammen extrahiert und anschließend mit einer Max-Pooling Schicht eine Aggregation vornimmt. Die Convolution Schichten nutzen dabei eine Kernel Größe von 3x3, gleichbleibendes Padding und die ReLu Aktivierungsfunktion. Daran anschließend folgt ein Multi-Layer-Perceptron (MLP), das die Klassifikation vornimmt. Das zweite Modell ist eine leicht erweiterte Variante des ersten Modells und unterscheidet sich im Wesentlichen durch die Anzahl der extrahierten Filter je Convolution Layer und die Komplexität des angeschlossenen MLPs (höhere Anzahl an Neuronen je Schicht). Zudem wird nach jeder Convolution durch Max-Pooling eine Aggregation vorgenommen.

Das dritte Modell orientiert sich an der Arbeit von Adavanne et al. (2017) und zählt zu den sogenannten Recurrent-CNN-Modellen, die die Funktionalität von Convolutional und Recurrent Neural Networks miteinander kombinieren. Die Autoren nutzen dabei eine Architektur, die zunächst durch Convolutional Blöcke mit BatchNormalization und MaxPooling Feature-Maps aus den Inputs ableitet, diese anschließend mit GRU Schichten weiter verarbeitet und durch ein MLP klassifiziert. Die GRU Schichten (bzw. generell rekurrente Schichten) können die zeitlichen Beschaffenheit der Daten, die bei Audiodaten und damit auch Spektrogrammen vorhanden ist, analysieren und daher entsprechende Zusammenhänge erkennen. Die in dieser Arbeit genutzte Architektur ist eine leicht veränderte Version, arbeitet jedoch nach der gleichen Funktionsweise.

Als viertes wird ein Residual Convolutional Neural Network verwendet. Der wesentliche Unterschied zu normalen CNNs liegt in den residualen Verbindungen, die parallel zu den eigentlichen Convolution Blöcken verlaufen. Der Vorteil ist ein erhöhter Informationsfluss, womit die Netzwerke leichter zu trainieren sind. Das Problem der 'sterbenden' Gradienten, das vermehrt bei tiefen Neuronalen Netzwerken auftritt, wird damit umgangen (He et al., 2015). Die Anzahl der residualen Blöcke ist ein wichtiger Hyperparameter, dessen Einfluss im nachfolgenden Abschnitt thematisiert wird. Ein Block besteht dabei jeweils aus dem Convolutional und Residual Path, die im Anschluss addiert und durch MaxPooling aggregiert werden. An die residualen Blöcke ist erneut ein MLP angeschlossen, das entsprechend die Klassifikation vornimmt.

5.1.2 Experimente

Im ersten Schritt wurde die Auswirkung der Fenster- und Schrittgröße beim Feature Engineering auf die Genauigkeit der Erkennung untersucht. Die Schrittgröße wurde jeweils nur bis zur Fenstergröße gewählt, da andernfalls Abschnitte des Audiosignals

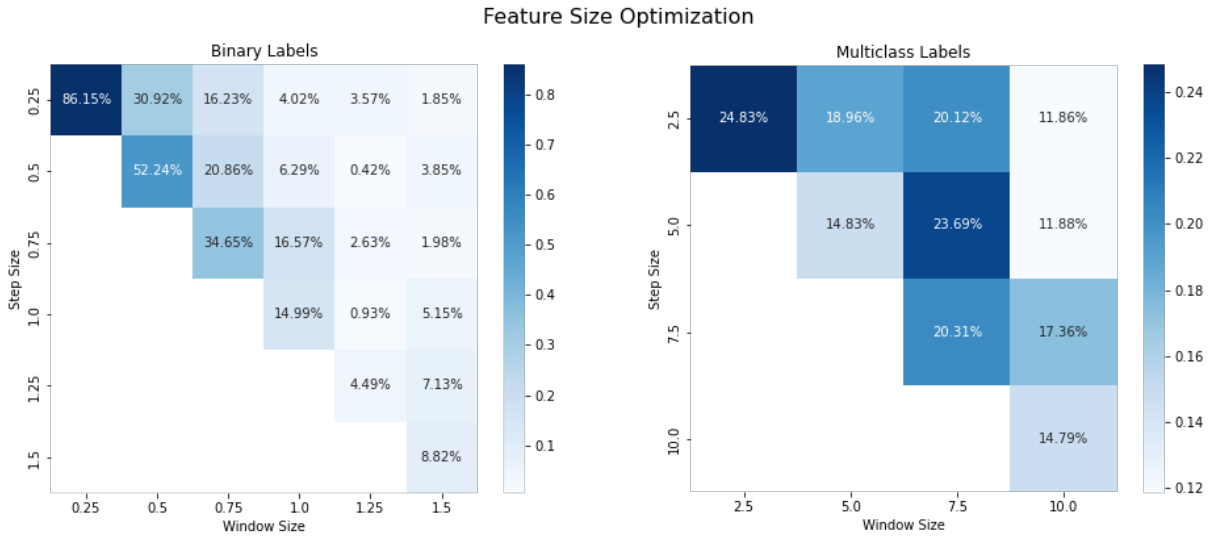

Abbildung 8: Relative Abweichung bei Variation der Fenster- und Schrittgröße

übersprungen werden. Abbildung 8 zeigt den relativen Vorhersagefehler für Binary (links) und Multiclass (rechts) Label unter Verwendung des erweiterten CNN Modells. Im Falle der Multiclass Label (Label gibt Anzahl der Fahrzeuge je Abschnitt an) liegen die Abweichungen alle oberhalb von 10%. Auch andere Modellarchitekturen erzielen keine besseren Ergebnisse, sodass die Verwendung von Nicht-Binären Labels nicht weiter verfolgt wird. Im Gegensatz dazu werden im Falle der binären Label (Fahrzeug fährt vorbei oder nicht) deutlich genauere Ergebnisse erzielt. Durch die Verwendung von binären Labeln kommt es vor, dass innerhalb eines Fensters zwei Fahrzeuge passieren, jedoch nur als eins gezählt werden. Diese Auswirkung wird in Kapitel 6 erneut aufgegriffen. Schlussendlich wird eine Fenstergröße von 1.25s und eine Schrittgröße von 1.0s gewählt. Zwar ist die geringste Abweichung bei 1.25s und 0.5s zu finden, jedoch kann diese durch die deutliche Überlappung der Fenster und damit ähnlichen Samples einhergehen. Die Überlappung ist bei einer Schrittgröße von einer Sekunde deutlich geringer, daher wird das Ergebnis als zuverlässiger eingestuft.

Bei der Auswahl der Feature stehen drei verschiedene Arten von Spektrogrammen (Melspektrogram, Chromagram STFT & Chromagram CQT) zur Verfügung. Diese können einzeln oder miteinander kombiniert zum Training der Modelle verwendet werden. Abbildung 9 zeigt die Auswirkung der Auswahl der Feature auf die Performance der verschiedenen Modelle. Die wesentlichen Erkenntnisse sind:

- Allgemein betrachtet ist die Abweichung eher gering, wenn das Melspektrogramm als Feature verwendet wird
- Das Recurrent-CNN kann mit einzelnen Spektrogrammen gute Ergebnisse erzielen, verschlechtert sich jedoch beim Zusammenfügen der Spektrogramme

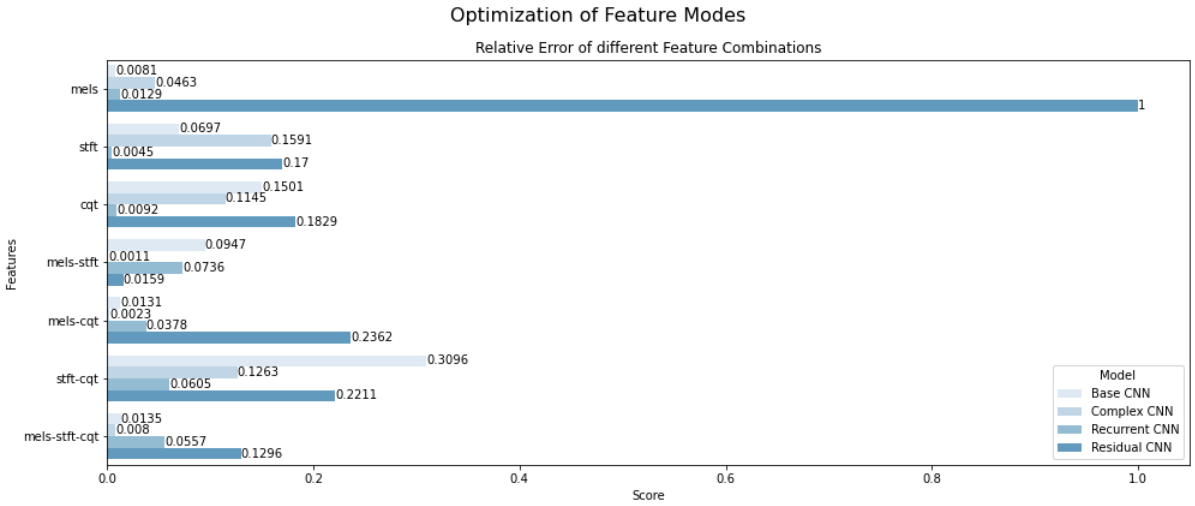

Abbildung 9: Relative Abweichung unter Variation der verfügbaren Spektrogramme

- Insgesamt erzielt die Kombination von Melspektrogramm und Chromagram STFT die geringsten Abweichungen, während die Kombination der beiden Chromagramme die ungenauesten Ergebnisse liefert

Für das finale Training der Modelle werden jeweils die Feature-Modi genutzt, die für ein spezifisches Modell die besten Ergebnisse liefern. Generell zeigt sich, dass nicht jedes Modell mit mehr Informationen auch bessere Ergebnisse erzielt.

Weitere Untersuchungen werden zur Auswirkung der Data Augmentation durchgeführt. Dabei wird der Prozentsatz der zufällig gewählten Felder innerhalb der Spektrogramme, die vor Beginn jeder Epoche abgedeckt werden, schrittweise erhöht. Die Ergebnisse weichen im Wesentlichen bis zu einer Rate von 80% nicht übermäßig stark voneinander ab und erst bei nahezu 100% sind nicht mehr genug Informationen zur Klassifikation vorhanden. Zum Training der finalen Modelle wird eine Rate von 20% genutzt, um eine ausreichende Veränderung der Daten je Epoche zu erreichen und Overfitting der Modelle zu vermeiden.

Als letztes wird die Anzahl der Blöcke in den residualen Netzwerken betrachtet. Die leicht angepasste Architektur erzeilt mit drei Blöcken und etwa 6% Abweichung die besten Ergebnisse, bei der herkömmlichen ResNet Architektur ist die Abweichung mit zwei Blöcken am geringsten und liegt bei etwa 1%. Daran zeigt sich, dass die Architekturen mit vielen residualen Blöcken aufgrund der hohen Anzahl an Parametern viele Samples auswendig lernen und eher overfitten.

5.2 Waveform Modelle

5.2.1 Verwendete Modelle

Bei den Waveform-Features handelt es sich im Gegensatz zu den Spektrogrammen um eindimensionale Daten, daher werden 1D-Convolutional Modelle verwendet. Sowohl das genutze Baseline Modell als auch die Sample-CNN-Architektur nach Allamy & Koerich (2021) kombinieren 1D-Convolution Schichten mit Max-Pooling, unterscheiden sich jedoch in der Komplexität bzw. Tiefe. Ähnlich zum vorgestellten Modell für die Spektrogramme wird auch für die Waveform ein R-CNN implementiert, das im Anschluss an die Convolution Schichten mit LSTM Zellen zeitliche Zusammenhänge aus den Feature-Maps extrahiert.

Als weitere Architektur werden 1D-Convolution Blöcke mit verschiedenen Kernel-Größen parallel genutzt, um bei der Extraktion der Feature-Maps unterschiedlich lange Zeiträume bzw. Abschnitte im Audiosignal zu betrachten. Diese Methodik wird unter anderem von Lidy & Schindler (2016) genutzt, um verschiedene Feature parallel zu verarbeiten und anschließend miteinander zu verbinden. Beim Zusammenführen der Feature-Maps stehen zwei Aggregationsmöglichkeiten zur Verfügung: Entweder werden die Feature-Maps aneinander angehängt (Concatenate-Layer) oder aufaddiert (Add-Layer). Die Auswirkung der Kernelgrößen sowie der Aggregationsmethode wird im folgenden Kapitel näher thematisiert.

Bei der letzten Architektur handelt es sich um WaveNet, das von van den Oord et al. (2016) vorgestellt wurde. WaveNet ist ursprünglich zur Audioerzeugung entwickelt worden, kann jedoch auch zur Audioklassifikation genutzt werden. Ähnlich zum parallelen CNN werden verschieden lange Zeitabschnitte betrachtet, allerdings unterscheidet sich die Realisierung. Innerhalb des Modells wird das Audiosignal von k 1D-Convolution Blöcken verarbeitet, die durch Dilation immer größere Zusammenhänge erfassen können. Eine weitere Besonderheit liegt in den Skip-Verbindungen, durch die die Outputs aller Blöcke am Ende aufaddiert werden. Ähnlich zu residualen Netzwerken wird dadurch der Informationsfluss erhöht und damit das Training vereinfacht.

5.2.2 Experimente

Auch zu den Waveform basierten Modellen werden einige Experimente zu den Hyperparametern durchgeführt, die einen wesentlichen Einfluss auf die Ergebnisse haben. Bei der Variation der Skalierung der Data Augmentation wird deutlich, dass zu große Unterschiede zu verschlechterten Ergebnissen im Vergleich zum Training ohne Data Augmentation führen. Durch viel Stauchung bzw. Streckung des Signals (Parameter `factor_scale` und Hinzufügen von viel Noise (Parameter `noise_scale`) wird das Signal zu stark verändert und die Modelle können keine allgemeinen Regeln mehr ableiten. Gute Ergebnisse konnten erzielt werden, wenn lediglich Faktor oder Noise

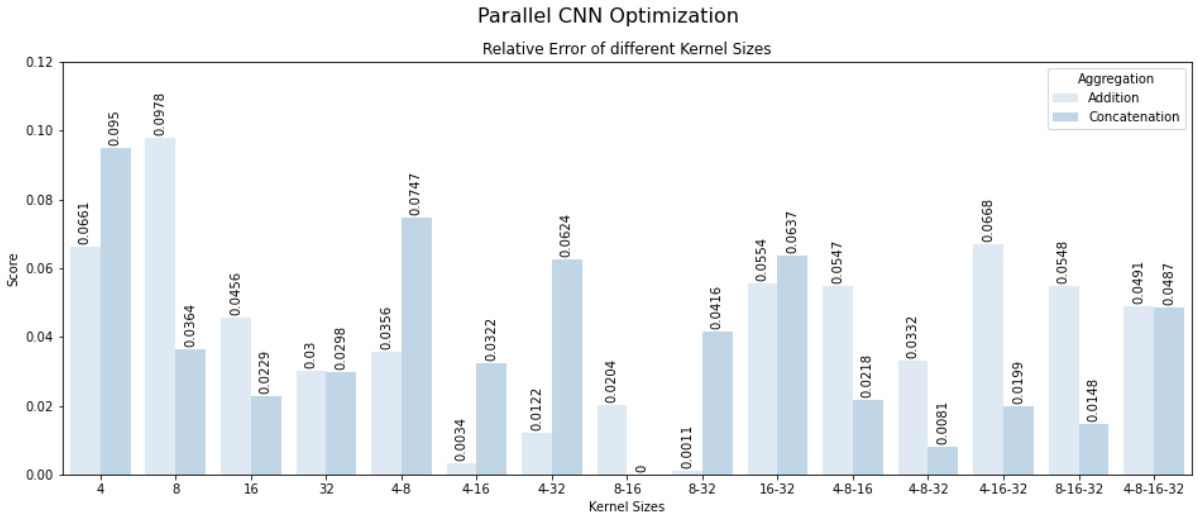

Abbildung 10: Relative Abweichung unter Verwendung verschiedener Kernel Größen

verwendet wird. Für das weitere Training wird die Data Augmentation mit einer Faktor Scale von 0.1 ohne Hinzufügen von Noise verwendet.

Innerhalb der WaveNet Architektur sind die wichtigsten Hyperparameter die Anzahl der Schichten und die Anzahl der Filter innerhalb der Schichten. Eine hohe Anzahl an Schichten bewirkt, dass eine höhere Dilation erreicht wird und damit große Zusammenhänge mit einbezogen werden. Die Anzahl der Filter beeinflusst, wie viele verschiedene Feature-Maps aus dem Audiosignal innerhalb jeder Schicht extrahiert werden. Beim Training zeigt sich, dass die WaveNet Architektur sehr stark zum Overfitting neigt und die Trainingssamples schnell auswendig lernt. Die geringsten Abweichungen in der Erkennung der Fahrzeuge werden mit 4 Schichten und 32 Filtern erzielt. Jedoch sind auch bereits mit 2 Schichten und 8 Filtern gute Ergebnisse möglich, was die Position der WaveNet Architektur als State-of-the-Art zur Audioverarbeitung unterstreicht.

Als drittes wird die Auswirkung verschiedener Kernel-Größen innerhalb von 1D-Convolution Schichten untersucht. Für jede Größe werden fünf Convolution Schichten jeweils gefolgt von MaxPooling genutzt. Wie bereits beschrieben werden die Outputs der Blöcke entweder kombiniert oder addiert. Abbildung 10 zeigt die Ergebnisse für alle Kombinationsmöglichkeiten der Kernel-Größen 4, 8, 16 und 32. Es zeigt sich, dass die Aggregation der parallelen Blöcke keinen signifikaten Einfluss aus die Ergebnisse der Modelle hat. Generell ist jedoch der Trend erkennbar, dass Concatenation unter Verwendung mehrerer Filter bessere Ergebnisse erzielen kann. In Bezug auf die Kernelgrößen ist kein eindeutiges Muster zu erkennen, welche Größen zu geringeren Abweichungen führen. Als finale Kombination wurden die Kombination 4-8-32 ausgewählt, da diese insgesamt die zuverlässigsten und stabilsten Ergebnisse erzielt.

Approach	Model	Metric		
		F1	Recall	rel_error
Spectrogram	Base CNN	0.790	0.812	0.057
	Complex CNN	0.779	0.775	0.010
	Recurrent CNN	0.582	0.577	0.018
	Adapted Residual CNN	0.605	0.613	0.028
	Residual CNN	0.780	0.773	0.018
Waveform	Base CNN	0.765	0.791	0.068
	Sample Level CNN	0.730	0.732	0.006
	Recurrent CNN	0.782	0.776	0.014
	Parallel CNN	0.763	0.766	0.008
	WaveNet	0.725	0.738	0.034

Tabelle 1: Vergleich der Ergebnisse der genutzten Modelle

5.3 Vergleich der Ergebnisse

Nach der Optimierung der Hyperparameter werden die finalen Modelle trainiert und miteinander verglichen. Die Ermittlung der Scores erfolgt durch wiederholtes Training der Modelle mit jeweils verschiedener Unterteilung der Daten in 70% Trainings- und 30% Validation-Daten. Aus den Wiederholungen wird der geringste relative Vorhersagefehler mit zugehörigem F1-Score sowie Recall gewählt. Diese Methodik liefert ein Optimum, das ggf. nicht die wahre Genauigkeit der Modelle auf Testdaten widerspiegelt. Jedoch werden andere Methoden der Evaluation, wie etwa Cross Validation Scoring, stark durch Probleme innerhalb des Trainings (z.B. Local Minimum Problem) beeinflusst und liefern einen tendenziell zu niedrigen Score.

Zur Bewertung der Modelle werden die folgenden Metriken verwendet:

- **F1-Score:** Da die Verkehrszählung als binäres Klassifikations-Problem behandelt wurde (Auto oder kein Auto), gibt der F1-Score Aufschluss über die generelle Genauigkeit bei der Erkennung der Klassen. Im Gegensatz zur Accuracy hat der F1-Score den Vorteil, dass er auch bei Ungleichgewicht der Klassen aussagekräftige Werte liefert (Yıldırım, 2020).

- **Recall:** Der Recall wird genutzt, um den Anteil der richtig erkannten Fahrzeuge aus allen Fahrzeugen zu bestimmen. Ein hoher Wert bedeutet dabei, dass ein hoher Anteil der Fahrzeuge richtig erkannt wurde.

- **rel_error:** Wie bereits zu Beginn dieses Kapitel beschrieben, wandelt der relative Vorhersagefehler die vorhergesagten Klassen in ein Regressions-Problem um. Er gibt Aufschluss darüber, wie weit die Anzahl der vorhergesagten Fahrzeuge im Verhältnis zur tatsächlichen Anzahl abweicht.

Die finalen Ergebnisse lassen sich nach Waveform und Spectrogram Modellen unterteilt aus Tabelle 1 entnehmen. Innerhalb der Herangehensweisen schneiden die Baseline Modelle in Bezug auf den relativen Vorhersagefehler am schlechtesten ab, weisen jedoch insgesamt einen guten F1-Score auf. Generell ergibt sich ein geringer relativer Vorhersagefehler, wenn F1-Score und Recall relativ nah beieinander liegen (gegeben durch ausgeglichene Klassenverteilung und der Annahme von Djukanovic et al. (2020), dass sich False-Positives und False-Negatives aufheben). Infolgedessen kann, wie im Fall des Recurrent CNN der Spektrogramm Modelle, auch ein geringer F1-Score von 0.582 zu einem geringen relativen Vorhersagefehler von 1.8% führen. Insgesamt fällt auf, dass die Modelle mit einem durchschnittlichen F1-Score von ca. 0.75 keine eindeutige Unterteilung zwischen den Klassen finden bzw. sich die Klassen in einigen Fällen sehr ähneln. Dies könnte mit der gewählten Methodik im Feature Engineering (s. Abschnitt 4.2) begründet werden, durch die Sample den Anfang eines vorbeifahrenden Fahrzeugs umfassen, jedoch nicht den tatsächlichen Moment des Vobeifahrens einschließen und damit als Klasse *Kein Fahrzeug* gewertet werden. Unter Betrachtung von F1-Score und relativem Vorhersagefehler erzielt für die Spektrogramm basierten Modelle das Complex CNN die besten Ergebnisse, bei den Waveform basierten Modellen das Recurrent CNN Modell.

Merkmal	Traffic Flow		
	Low	Moderate	High
Fahrzeuge $[1/min]$	<10	10-20	>20
Länge $[mm:ss]$	15:23	15:45	14:42
$\sum$ Fahrzeuge	89	174	338
$\varnothing$ Fahrzeuge $[1/min]$	5.79	11.05	22.99
$\sum$ Binary Labels	87	161	279

Tabelle 2: Grundlegende Eigenschaften der Testdaten

6 Anwendung

Um die finalen Modelle nach Optimierung der Hyperparameter noch einmal zu testen, werden zusätzliche Testdaten zur weiteren Evaluation herangezogen. Diese wurden im Training nicht verwendet und sind den Modellen daher vollständig unbekannt. Die Erhebung und Verarbeitung erfolgt nach dem gleichen Prinzip wie in Abschnitten 3 und 4 beschrieben. Die Testdaten umfassen die drei verschiedenen Verkehrsdichten niedrig, moderat und hoch. Damit soll untersucht werden, für welche Situationen die Modelle zuverlässige Ergebnisse liefern. Weitere Details lassen sich Tabelle 2 entnehmen. Das Merkmal $\sum$ *Binary Labels* gibt dabei an, wie viele Fahrzeuge insgesamt durch die binären Label erfasst werden. Gerade im dichteren Verkehrt kommt es vor, dass innerhalb eines Abschnittes von 1.25s mehr als ein Fahrzeug passiert. Jedoch werden diese nur als eines gewertet, wodurch die Abweichungen zur tatsächlichen Anzahl der Fahrzeuge ($\sum$Fahrzeuge) zu Stande kommen.

Die Audiodaten werden wie bereits im Training in Abschnitte zum Feature Engineering unterteilt, jedoch ohne Überlappung der Fenster. Andernfalls würden Abschnitte des Audiosignals doppelt in die Erkennung mit einfließen und es könnte zu doppelten Erkennungen eines einzigen Fahrzeugs kommen. Zur Evaluation werden im weiteren Vorgehen die Ausgaben der Modelle aufsummiert und die Abweichung zur tatsächlichen Anzahl der Fahrzeuge berechnet. Mit den binären Labels werden keine weiteren Metriken berechnet, da diese keine Aussage über die tatsächliche Performance der Modelle im realen Straßenverkehr geben würden. Tabelle 3 zeigt die Anzahl der erkannten Fahrzeuge sowie die Abweichung zur tatsächlichen Anzahl je Verkehrsfluss für die genutzten Modelle.

Es ist ersichtlich, dass bei geringem Verkehrsaufkommen die größten Abweichungen vorliegen. Es werden von allen Modellen deutlich zu viele Fahrzeuge erkannt, im Falle des Spektrogramm basierten Recurrent CNN liegt die Abweichung bei fast 260%. Bei der Analyse der Videos in Kombination mit den Ausgaben der Modelle wird deutlich, dass im geringen Verkehrsfluss das Geräusch der vorbeifahrenden Fahrzeuge recht frühzeitig vom Mikrofon erfasst wird und diese daher doppelt oder teilwei-

Approach	Model	Traffic Flow					
		Low		Moderate		High	
		$\sum$	Error	$\sum$	Error	$\sum$	Error
Spectrogram	Base CNN	226	1.539	217	0.247	302	0.107
	Complex CNN	201	1.258	155	0.109	300	0.112
	Recurrent CNN	318	2.573	169	0.029	345	0.021
	Adapted Residual CNN	203	1.281	123	0.293	301	0.109
	Residual CNN	217	1.438	207	0.190	320	0.053
Waveform	Base CNN	140	0.573	156	0.103	383	0.133
	Parallel CNN	132	0.483	148	0.149	393	0.163
	Recurrent CNN	139	0.562	133	0.236	379	0.121
	Sample Level CNN	136	0.528	159	0.086	361	0.068
	WaveNet	149	0.674	168	0.034	339	0.003

Tabelle 3: Ergebnisse der Modelle auf den Testdaten

se dreifach gezählt werden. Auffällig ist dabei ebenfalls, dass die Spektrogram Modelle weitaus mehr Fahrzeuge im Audiosignal erkennen als die Waveform Modelle. Ein Lösungsvorschlag, der dem doppelten Zählen von Fahrzeugen entgegenwirken könnte, wird im Fazit der Arbeit vorgestellt. Bei moderatem und hohem Verkehrsfluss wird die Abweichung geringer, da es aufgrund der verringerten Abstände zwischen den Fahrzeugen zu weniger doppelten Erkennungen kommt. Bei moderatem sowie hohem Verkehrsfluss erzielen das Spektrogram-basierte Recurrent CNN und die WaveNet Architektur die geringsten Abweichungen. Bei den Waveform Modellen folgt an zweiter Stelle das Sample Level CNN mit einer geringfügig höheren Abweichung. Damit zeigt sich für die Waveform Modelle, dass die State-of-the-Art Architekturen gute Ergebnisse erzielen und eine bessere Performance zeigen als die weiteren Modelle.

Insgesamt zeigt die Evaluation auf Basis der Testdaten, dass die Modelle im geringen Verkehrsfluss durch doppelte Erkennungen zu viele Fahrzeuge ausgeben und es zu hohen Abweichungen kommt. Dafür liegen die Abweichungen bei modereratem und hohen Verkehrsfluss teils deutlich unterhalb von 5% und damit in einem annehmbaren Bereich für den Einsatz zum echtzeitbasierten Verkehrs-Tracking.

7 Fazit

Als übergeordnete Zielsetzung dieser Arbeit wurde festgehalten, mit Machine Learning ein audiobasiertes System zur Verkehrszählung zu entwickeln. Dazu wurden im Laufe der Arbeit ein Datensatz erstellt und verarbeitet sowie die entsprechenden Machine Learning Modelle trainiert. Innerhalb der Audioverarbeitung wurden zwei verschiedene Methoden genutzt und miteinander verglichen. Abschließend wurden weitere Testdaten aufgenommen, um die Genauigkeit der Fahrzeugerkennung noch einmal zu validieren. In diesem Teil der Arbeit werden die wesentlichen Erkenntnisse zusammengefasst und Beschränkungen sowie Probleme des Systems thematisiert. Zudem werden die anfänglich gestellten Forschungsfragen beantwortet und es wird ein Ausblick gegeben, welchen weiteren Forschungsbedarf es gibt.

Innerhalb der Datenerhebung war ein wichtiger Punkt, ein sinnvolles Aufnahmesetup und passende Stellen an Straßen zu finden, um eine solide Datengrundlage zu schaffen. Beim Labeling der Daten hat sich gezeigt, dass durch simple Computer Vision Anwendungen wie YOLOv4 ein nahezu fehlerfreies Erkennen der Fahrzeuge möglich ist und auch ohne Object Tracking kaum Fahrzeuge nicht gelabelt werden. Nachfolgend wurden die Machine Learning Modelle auf den jeweiligen Features trainiert und im Anwendungsteil der Arbeit validiert. Eine wesentliche Erkenntnis liegt in der Problematik, dass die Modelle im geringen Verkehrsfluss deutlich zu viele Fahrzeuge vorhersagen und es damit einhergehend zu hohen relativen Abweichungen zur tatächlichen Anzahl an Fahrzeugen kommt. Allerdings konnte insbesondere WaveNet bei moderatem und hohem Verkehrsfluss gute Ergebnisse mit Abweichungen von unter 5% erzielen. Somit ist das audiobasierte System zur Verkehrszählung insgesamt funktionsfähig, unterscheidet sich jedoch hinsichtlich der Genauigkeit je nach Verkehrsfluss. Damit ist jedoch die übergeordnete Zielsetzung erreicht.

Wie bereits genannt, liegt eine wesentliche Einschränkung bzw. ein wesentliches Problem des Systems im geringen Verkehrsfluss mit weniger als zehn Fahrzeugen pro Minute. Das Geräusch der nahenden Fahrzeuge wird bereits frühzeitig vom Mikrofon erfasst, was zu vielen doppelten Erkennungen führt. Gerade im geringen Verkehrsfluss bedeuten doppelte Erkennungen schnell eine hohe Abweichung (je nach Modell bis zu 250%, s. Tabelle 3). Ein Ansatz wäre die Nutzung eines Richtmikrofons, um eine Beschränkung der aufgenommenen Audiosignale auf einen kleinen Abschnitt der Straße vorzunehmen. Alternativ wäre auch eine Abschirmung des Mikrofons denkbar, um den seitlich eintreffenden Schall der näherkommenden Fahrzeuge abzublocken. Damit einhergehend könnte die Anzahl der doppelten Erkennungen verringert und die generelle Abweichung verbessert werden.

Im Folgenden werden die in der Einleitung festgehaltenen Forschungsfragen beantwortet:

1. Audiobasierte Systeme sind in der Lage, vorbeifahrende Autos zu erkennen. Die Genauigkeit hängt dabei vom gewählten Modell und im wesentlichen Maße von der Verkehrsdichte ab. Es hat sich gezeigt, dass WaveNet für moderaten und hohen Verkehrsfluss geringe Abweichungen erzielt, die für die Detektion von Fahrzeugen an Zufahrtsstraßen zu Kreuzungen ausreichend sind. Für eine exakte Verkehrszählung sind die Modelle eher weniger geeignet.

2. Eine wesentliche Rolle bei der genauen Erkennung von Fahrzeugen spielt das Aufnahmesetup. Es wurde deutlich, dass das gewählte Mikrofon anfahrende Fahrzeuge frühzeitig erfasst und diese doppelt zählt. Im Falle der Spektrogramm basierten Modelle ist zudem ein Faktor, welche Arten von Spektrogrammen als Feature verwendet werden.

3. Im Training werden keine signifikanten Unterschiede zwischen der Waveform und Spektrogrammen festgestellt (Differenzen der relativen Abweichung unterhalb von 1%). Auf den Testdaten dagegen erweisen sich die Waveform Modelle als robuster und sind weniger anfällig für das doppelte Erkennen von Fahrzeugen. Mit WaveNet und dem Sample-Level-CNN gibt es zwei Architekturen, die bei moderatem und hohem Verkehrsfluss gute Ergebnisse erzielen.

4. Beim Vergleich der Modelle innerhalb der Herangehensweisen fällt im Training auf, dass die Baseline Modelle mit geringer Komplexität die höchsten Abweichungen aufweisen. Dies wird in der Anwendung mit den Testdaten jedoch nicht weiter bestätigt und die Baseline Modelle erzielen ähnliche Ergebnisse wie Modelle mit einer höheren Komplexität.

Insgesamt sollte in der weiteren Forschung der Fokus auf der Problematik der hohen Abweichungen im niedrigen Verkehrsfluss liegen. Eine Variante wäre die Anpassung des Aufnahmesetups, um die vermehrten doppelten Erkennungen zu vermeiden. Alternativ könnte die Verwendung von Multiclass Labels oder die direkte Formulierung als Regressionsproblem untersucht werden. Neben den Problemen auf technischer Seite sollte das System zudem nutzbar gemacht werden, um in einem der anfangs beschriebenen Use-Cases genutzt zu werden. Auf Basis von Daten aus der realen Verwendung des Systems können zudem weitere Schwachstellen und Probleme gefunden werden, die in dieser Arbeit noch nicht aufgetreten sind. Weiterhin könnten neue Use-Cases eines audiobasierten Systems zur Verkehrszählung entwickelt werden, um mit dieser Arbeit einen Beitrag zu Intelligent Transportation Systems zu liefern, den Straßenverkehr effizienter zu gestalten und langfristig gesehen die Lebensqualität in Städten zu verbessern.

Literatur

Adavanne, S., Drossos, K., Çakır, E., & Virtanen, T. (2017). *Stacked convolutional and recurrent neural networks for bird audio detection.* Retrieved from https://arxiv.org/pdf/1706.02047

Allamy, S., & Koerich, A. L. (2021). *1d cnn architectures for music genre classification.* Retrieved from https://arxiv.org/pdf/2105.07302

Augsburger Allgemeine. (2018). Intelligente verkehrssteuerung: Mehr als rot-gelb-grün: Was ampeln alles können — augsburger allgemeine. *Augsburger Allgemeine*. Retrieved 15.04.2022, from https://www.augsburger-allgemeine.de/themenwelten/auto-verkehr/Intelligente-Verkehrssteuerung-Mehr-als-Rot-Gelb-Gruen-Was-Ampeln-alles-koennen-id43732191.html

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *Yolov4: Optimal speed and accuracy of object detection.* Retrieved from https://arxiv.org/pdf/2004.10934

Brown, J. (1991). Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, *89*, 425–434. doi: 10.1121/1.400476

Brownlee, J. (2018). A gentle introduction to dropout for regularizing deep neural networks. *Machine Learning Mastery*. Retrieved 24.04.2022, from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. Retrieved from https://arxiv.org/pdf/1106.1813 doi: 10.1613/jair.953

D'agostino, A. (2022). Building your own dataset: Benefits, approach, and tools. *Towards Data Science*. Retrieved 07.03.2022, from https://towardsdatascience.com/building-your-own-dataset-benefits-approach-and-tools-6ab096e37f2

Dhiman, A. (2020). Object tracking using deepsort in tensorflow 2. *Analytics Vidhya*. Retrieved 09.06.2021, from https://medium.com/analytics-vidhya/object-tracking-using-deepsort-in-tensorflow-2-ec013a2eeb4f

Djukanovic, S., Matas, J., & Virtanen, T. (2020). Robust audio-based vehicle counting in low-to-moderate traffic flow. In *2020 ieee intelligent vehicles symposium (iv)* (pp. 1608–1614). IEEE. doi: 10.1109/IV47402.2020.9304600

Djukanović, S., Patel, Y., Matas, J., & Virtanen, T. (2020). *Neural network-based acoustic vehicle counting.* Retrieved from https://arxiv.org/pdf/2010.11659

Golovnin, O., Privalov, A., Stolbova, A., & Ivaschenko, A. (2021). Audio-based vehicle

detection implementing artificial intelligence. In O. Dolinina et al. (Eds.), *Recent research in control engineering and decision making* (Vol. 337, pp. 627–638). Cham: Springer International Publishing and Imprint Springer. doi: 10.1007/978-3-030-65283-8{\textunderscore}51

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision - eccv 2014* (Vol. 8691, pp. 346–361). Cham: Springer. doi: 10.1007/978-3-319-10578-9{\textunderscore}23

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition.* Retrieved from https://arxiv.org/pdf/1512.03385

I. Kartika, & Shahrizat Shaik Mohamed. (2011). Frame differencing with post-processing techniques for moving object detection in outdoor environment. In *2011 ieee 7th international colloquium on signal processing and its applications* (pp. 172–176). doi: 10.1109/CSPA.2011.5759867

Intelligent transport. (2017). Retrieved 15.04.2022, from https://www.intelligenttransport.com/

Islam, Z., & Abdel-Aty, M. (2022). *Real-time emergency vehicle event detection using audio data.* Retrieved from https://arxiv.org/pdf/2202.01367

Jordan, J. (2018). Setting the learning rate of your neural network. *Jeremy Jordan*. Retrieved 07.08.2021, from https://www.jeremyjordan.me/nn-learning-rate/

Keras documentation: Dropout layer. (2022). Retrieved 24.04.2022, from https://keras.io/api/layers/regularization_layers/dropout/

Keras documentation: Earlystopping. (2022). Retrieved 24.04.2022, from https://keras.io/api/callbacks/early_stopping/

Lidy, T., & Schindler, A. (2016). *Parallel convolutional neural networks for music genre and mood classification*. Retrieved from http://www.ifs.tuwien.ac.at/~schindler/pubs/mirex2016.pdf

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). *Path aggregation network for instance segmentation.* Retrieved from https://arxiv.org/pdf/1803.01534

Marciniuk, K., Szczodrak, M., & Czyzewski, A. (2018). An application of acoustic sensors for the monitoring of road traffic. In *Spa 2018* (pp. 208–212). Piscataway, NJ: IEEE. doi: 10.23919/SPA.2018.8563406

Mustafeez, A. Z. (2020). What is early stopping? *Educative*. Retrieved 24.04.2022, from https://www.educative.io/edpresso/what-is-early-stopping

Na, Y., Guo, Y., Fu, Q., & Yan, Y. (2015). An acoustic traffic monitoring system: Design and implementation. In *2015 ieee 12th international conference on ubiquitous intelligence and computing, 2015 ieee 12th international conference on advanced and trusted computing, 2015 ieee 15th international conference on scalable computing and commu-*

nications, 2015 ieee international conference on cloud and big data computing, 2015 ieee international conference on internet of people and associated symposia/workshops (pp. 119–126). Piscataway, NJ: IEEE. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.41

Nanni, L., Maguolo, G., & Paci, M. (2020). Data augmentation approaches for improving animal audio classification. *Ecological Informatics*, *57*. doi: 10.1016/j.ecoinf.2020.101084

Nussbaumer, H. J. (1981). *Fast fourier transform and convolution algorithms* (Vol. 2). Berlin and Heidelberg: Springer. doi: 10.1007/978-3-662-00551-4

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *Proc. Interspeech*, 2613–2617. Retrieved from https://arxiv.org/pdf/1904.08779 doi: 10.21437/Interspeech.2019-2680

Pedersen, P. (1965). The mel scale. *Journal of Music Theory*, *9*(2), 295–308. Retrieved from http://www.jstor.org/stable/843164

Redmon, J., & Farhadi, A. (2018). *Yolov3: An incremental improvement.* Retrieved from https://arxiv.org/pdf/1804.02767

Reek, F. (2019). Feinstaub: Die abgase sind nicht das hauptproblem. *Süddeutsche Zeitung*. Retrieved 15.04.2022, from https://www.sueddeutsche.de/auto/feinstaub-verkehr-bremsen-reifen-1.4427241

Roy, L., & Shahd, M. (2020). *Tüv-verband fordert flächendeckende intelligente verkehrssteuerung.* Retrieved 15.04.2022, from https://www.tuev-verband.de/pressemitteilungen/intelligente-verkehrssteuerung

Shah, A., Kattel, M., Nepal, A., & Shrestha, D. (2019). Chroma feature extraction. In *Chroma feature extraction using fourier transform.*

Shigemi Ishida, Song Liu, Kohei Mimura, Shigeaki Tagashira, & Akira Fukuda. (2016). Design of acoustic vehicle count system using dtw. In *23rd its world congress melbourne.* Retrieved from https://www.researchgate.net/profile/shigemi-ishida/publication/329254580_design_of_acoustic_vehicle_count_system_using_dtw

Smith, L. N. (2015). *Cyclical learning rates for training neural networks.* Retrieved from https://arxiv.org/pdf/1506.01186

Takimoglu, A. (2021). What is data augmentation? techniques & examples in 2022. *AIMultiple*. Retrieved 23.03.2022, from https://research.aimultiple.com/data-augmentation/

Uke, N. J., & Thool, R. C. (2013). Moving vehicle detection for measuring traffic count using opencv. *Journal of Automation and Control Engineering*, 349–352. doi: 10.12720/joace.1.4.349-352

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., …

Kavukcuoglu, K. (2016). *Wavenet: A generative model for raw audio.* Retrieved from https://arxiv.org/pdf/1609.03499

van Dyk, D. A., & Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, *10*(1), 1–50. doi: 10.1198/10618600152418584

Wang, C.-Y., Liao, H.-Y. M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., & Hsieh, J.-W. (2019). *Cspnet: A new backbone that can enhance learning capability of cnn.* Retrieved from https://arxiv.org/pdf/1911.11929

Wyse, L. (2017). Audio spectrogram representations for processing with convolutional neural networks. *DLM*. Retrieved from https://arxiv.org/pdf/1706.09559

Yıldırım, S. (2020). How to best evaluate a classification model - towards data science. *Towards Data Science*. Retrieved 24.04.2022, from https://towardsdatascience.com/how-to-best-evaluate-a-classification-model-2edb12bcc587

Zawadzka-Gosk, E., Wołk, K., & Czarnowski, W. (2019). Deep learning in state-of-the-art image classification exceeding 99% accuracy. In Á. Rocha, H. Adeli, L. P. Reis, & S. Costanzo (Eds.), *New knowledge in information systems and technologies* (Vol. 930, pp. 946–957). Cham: Springer. doi: 10.1007/978-3-030-16181-1{\textunderscore}89

Zeller, P. (2018). Fahrgeräusch. In P. Zeller (Ed.), *Handbuch fahrzeugakustik* (pp. 267–290). Wiesbaden: Springer Fachmedien Wiesbaden GmbH. doi: 10.1007/978-3-658-18520-6{\textunderscore}9

Anhang

Github Repository:
https://github.com/NilsHMeier/Audio_Classification_Approaches

Kaggle Dataset:
https://www.kaggle.com/datasets/nilshmeier/road-traffic-audio

YOLOv4
Weights-File:
https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
Config-File:
https://raw.githubusercontent.com/AlexeyAB/darknet/master/cfg/yolov4.cfg

YOLOv4-Tiny
Weights-File:
https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.weights
Config-File:
https://raw.githubusercontent.com/AlexeyAB/darknet/master/cfg/yolov4-tiny.cfg