FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Convolutional Neural Network-Based Object Detection in Night-Time Traffic Scenes

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Visual Computing

eingereicht von

Astrid Unger, B.Eng.
Matrikelnummer 00720278

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof.Dipl.-Ing.Mag.Dr. Margrit Gelautz
Mitwirkung: Dr. Michael Hödlmoser

Wien, 1. Oktober 2024

Astrid Unger	Margrit Gelautz

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
Bibliothek
Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Convolutional Neural Network-Based Object Detection in Night-Time Traffic Scenes

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Visual Computing

by

Astrid Unger, B.Eng.
Registration Number 00720278

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof.Dipl.-Ing.Mag.Dr. Margrit Gelautz
Assistance: Dr. Michael Hödlmoser

Vienna, 1st October, 2024

Astrid Unger

Margrit Gelautz

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Astrid Unger, B.Eng.
Columbusgasse 11/8, 1100 Wien

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

Wien, 1. Oktober 2024

Astrid Unger

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Ich möchte mich zunächst bei Frau Professor Margrit Gelautz von der Technischen Universität Wien für ihre Betreuung und ihr Feedback sowie ihre wertvollen wissenschaftlichen Anregungen während des Entstehungsprozesses dieser Arbeit bedanken.

Des Weiteren möchte ich mich bei der emotion3D GmbH für das Sponsoring meiner Arbeit und die Bereitstellung eines Arbeitsplatzes und der notwendigen Hardware bedanken. Insbesondere bei Michael Hödlmoser, meinem Betreuer und Projektleiter bei emotion3D, für die freundliche und aufgeschlossene Betreuung, die Klärung meiner Fragen und Zweifel und die Ermöglichung der Arbeit an dieser Masterarbeit. Weiters möchte ich mich bei Matej Nezveda und Florian Seitner für ihr Feedback und ihre Unterstützung bedanken. Diese Arbeit wurde durch das Projekt CarVisionLight unterstützt, das von der Österreichischen Forschungsförderungsgesellschaft (FFG) und dem österreichischen Bundesministerium für Verkehr, Innovation und Technologie (BMVIT) im Rahmen des Programms “IKT der Zukunft” (Projektnummer 861251) gefördert wurde.

Vor allem möchte ich mich bei meiner Familie Michaela Unger und Maria Skrein-Bumballa für die Unterstützung während meines Studiums bedanken, für ihre Geduld und Freundlichkeit in all diesen Jahren und dafür, dass sie mich zu der Person geformt haben, die ich heute bin. Schließlich möchte ich mich bei meinem Lebensgefährten Tobias Sippl bedanken, der mir wertvolle Anregungen gegeben hat und mich stets unterstützt hat.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to thank Professor Margrit Gelautz of the TU Wien for her supervision and feedback, as well as her valuable scientific input throughout this thesis' formation process.

I want to express my gratitude for emotion3D GmbH for sponsoring my work and providing me with a workplace and the necessary hardware, especially Michael Hödlmoser, my advisor and project manager at emotion3D, for his friendly and open-minded supervision, resolving my questions and doubts and enabling work on this master thesis. Additionally, I would like to thank Matej Nezveda and Florian Seitner for their feedback and constant support.

This work was supported by the CarVisionLight project, which was funded by the Austrian Research Promotion Agency (FFG) and the Austrian Federal Ministry BMVIT under the program "ICT of the Future" (project no. 861251). I gratefully acknowledge the funding provided by the FFG, without which this work would not have been possible.

Above all, I would like to thank my family Michaela Unger and Maria Skrein-Bumballa for their support through my studies, for their patience and kindness throughout these years and shaping me into the person that I am today. Finally, I would like to thank my partner, Tobias Sippl, for providing me with valuable input and for his never-ending support and supply of coffee.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Mit dem wachsenden Bedarf an robusten Objekterkennungsalgorithmen in selbstfahrenden Systemen ist es entscheidend, die vielfältigen Licht- und Wetterbedingungen zu berücksichtigen, denen Fahrzeuge das ganze Jahr über ausgesetzt sind. Es ist unerlässlich, einem selbstfahrenden System einen zuverlässigen Objekterkennungsalgorithmus bereitzustellen, der unter diesen unterschiedlichen Bedingungen arbeiten kann. Deep-Learning-Methoden, insbesondere Convolutional Neural Networks (CNNs), haben sich hierbei als effektiv erwiesen, um diesem Bedarf gerecht zu werden. Um ein CNN in die Lage zu versetzen, diese unterschiedlichen Lichtverhältnisse zu bewältigen, ist eine sorgfältige Auswahl des Trainingsdatensatzes erforderlich. Öffentlich zugängliche Datensätze, welche für das Training von Deep-Learning-Algorithmen verwendet werden, weisen jedoch oft eine Verzerrung zugunsten von Tageslichtszenarien auf. Das Ziel dieser Masterarbeit ist es daher, effektive Strategien zur Auswahl und Zusammenführung von Trainingsdaten aus öffentlich zugänglichen und selbst aufgenommenen Datensätzen zu erforschen, um einem CNN die Erkennung von Objekten in nächtlichen Verkehrsszenen zu ermöglichen. Zu diesem Zweck haben wir das bestehende CNN YOLOv3 neu trainiert, um die Auswirkungen verschiedener Kombinationen von Trainingsdatensätzen auf die endgültigen Objekterkennungsergebnisse zu untersuchen. Unsere Ergebnisse zeigen, dass eine geeignete Auswahl von Trainingsdaten, kombiniert mit selbst aufgenommenen Nachtszenen, eine mittlere durchschnittliche Präzision (mean average precision, mAP) von 63,5% erreichen kann, was eine Verbesserung von 16,7% im Vergleich zur Leistung des ursprünglichen YOLOv3-Netzwerks darstellt.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

As the demand for robust object detection algorithms in self-driving systems grows, it is crucial to consider the diverse lighting and weather conditions that vehicles encounter year-round. It is essential to provide a self-driving system with a reliable object detection algorithm, capable of operating under these varying conditions. Deep learning methodologies, particularly Convolutional Neural Networks (CNNs), have proven effective in meeting this need. To equip a CNN with the ability to handle these varying lighting challenges, a careful selection of the training dataset is necessary. However, publicly available traffic scene datasets used for training deep learning algorithms often have a bias towards daylight scenarios and optimal visibility conditions. Therefore, the goal of this master thesis is to explore effective strategies for selecting and merging training data from publicly available and self-recorded datasets, to enable a CNN to detect objects in night-time traffic scenes. To achieve this, we retrained the existing CNN YOLOv3, to study the impact of different training dataset combinations on the final object detection results. Our findings show that an appropriate selection of training data, combined with self-recorded night scenes, can achieve mean average prevision (mAP) of 63.5%, which is an improvement of 16.7%, compared to the performance of the original YOLOv3 network.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung **xi**

Abstract **xiii**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Statement 2
1.3 Methodological Approach 3
1.4 Structure of the Work 4

2 Background and Related Work **5**
2.1 Machine Learning 5
2.2 Object Recognition 6
2.3 Neural Networks 11
2.4 Convolutional Neural Networks 15
2.5 Object Recognition CNNs 17
2.6 YOLOv3 19
2.7 Intersection Over Union 22
2.8 Object Detection in Night Scenes Using CNNs 25

3 Datasets **27**
3.1 Publicly Available Datasets 28
3.2 Self Recorded Dataset 30

4 Implementation **33**
4.1 Software Description 34
4.2 Initial Test Run 37
4.3 Training and Evaluation Pipeline 37
4.4 Data Handling 46

5 Evaluation and Results **57**
5.1 Evaluation Setup 57
5.2 Results and Discussion 62
5.3 Critical Reflection and Future Work 74

6 Conclusion **81**

Overview of Generative AI Tools Used **83**

List of Figures **85**

List of Tables **91**

Bibliography **93**

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

The demand for robust object detection algorithms has been continuously increasing as autonomous driving technology seeks to evolve from advanced driver assistance systems (ADAS) to autonomous, self-driving systems. To this end, cars must be able to operate under varying environmental influences throughout the year, at any time of day. Therefore, it is vital to provide the cars' self-driving systems with robust object detection algorithms that can operate under these varying conditions. While Convolutional Neural Networks (CNNs) are able to fulfil this need for robust object detection, their detection success is highly dependent on the dataset used for training. When using publicly available datasets, one should be aware that these datasets have a bias towards day-time scenes with ideal driving environments, therefore excluding many use cases.

To overcome this issue, this master thesis focuses on methods that allow a Convolution Neural Network (CNN) to detect objects under varying environmental conditions, focussing on night-scenes. This chapter will serve as an introduction by first giving insight into the challenges one faces when implementing object detection algorithms for varying lighting conditions in Sections 1.1 and 1.2. This is then followed in Section 1.3 by explaining the methodology used for this master thesis, which includes literature research, dataset selection and development of the software, followed by evaluation. Finally, Section 1.4 gives an overview of the subsequent chapters of this master thesis.

1.1 Motivation

The field of camera-based object detection during daytime has been extensively explored and evaluated, as evidenced, for example, by the works of Miller et al. [SBM06], Zhaowei et al. [CFFV16], and Sivaraman et al. [ST13]. However, a comprehensive survey conducted by Almagambetov et al. [AVC15] has highlighted the need for more rigorous investigation into vision-based techniques for object detection during night-time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

In light of this, the focus of this master thesis is to delve into the realm of object detection in night-time traffic scenes. This was tackled by retraining a Convolutional Neural Network (CNN) using a combination of publicly available datasets and self-recorded data. As Ulmer et al. [UGL$^+$18] have pointed out, publicly available datasets that provide data for deep learning algorithms are typically skewed towards daylight and optimal visibility conditions. This necessitates careful selection of datasets when developing self-driving systems. The incorporation of night scenes into training datasets for CNNs has led to improved object detection results, with notable contributions from Sivaraman et al. [ST13] and Jensen et al. [JNM17]. To overcome the limited availability of publicly accessible datasets featuring night scenes, it may be necessary to resort to self-recorded data. The manual annotation of these images, however, can be a laborious and time-consuming process, particularly due to the challenging and variable illumination conditions. To keep the need for annotating self-recorded data to a minimum, this master thesis explores the results of progressively incorporating simulated night-scenes from the video game GTA 5 with existing and self-recorded datasets.

1.2 Problem Statement

Deep Learning (DL) techniques have emerged as one of the fundamental building blocks of contemporary object detection networks, with Faster R-CNN [RHGS15] coming through as one example. However, these algorithms encounter significant challenges when tasked with detecting objects in driving scenes, due to issues such as variations in illumination in RGB images or the degradation of image quality at high velocities. One potential solution to the problem of illumination variations involves the use of diverse training images, which could facilitate a smooth transition from day to night images, as described by Wei et al. [WHZ$^+$19].

Convolutional Neural Networks (CNNs) for object detection applications are typically optimized for standard RGB cameras [KSH12]. To enable a neural network to handle varying lighting conditions, it may be necessary to adapt the training data and retrain the entire network. The training data can be generated through several means, including the use of images created in a virtual environment (e.g.: video games), the explicit recording and labelling of images, or the utilization of publicly available datasets. We composed a training dataset by progressively incorporating simulated datasets from the video game GTA5 into existing and self-recorded datasets. The results derived from the incremental expansion of the training dataset across various test cycles will be discussed in detail in this master thesis.

This master thesis was carried out in cooperation with emotion3D [e3d], within the FFG-funded project CarVisionLight (CVL): Combined 3D-Vision and Adaptive Front-lighting Systems for Safe Autonomous Driving [cvl]. The CVL project strived to generate a general vision system for situational awareness and environmental understanding under shifting illumination conditions. The contribution of this master thesis consisted of providing a more in-depth understanding of meaningful strategies for selecting and merging training data with existing datasets.

1.3 Methodological Approach

After a review of the background and related work, this master thesis will present the datasets used for the training and evaluation cycles. Ultimately, the results derived from the incremental expansion of the training dataset across various test cycles will be presented and subjected to detailed analysis.

Background and Related Work

The first step in writing this master thesis was a literature review, which served to establish a theoretical understanding of the fundamentals of object detection. To elucidate this process, Section 2 will present an analysis of machine learning and object recognition methods, with a focus on the YOLOv3 [RF18] network.

Dataset Selection and Generation

From the insights gained during the Literature Research, the datasets for training the CNN were selected accordingly. Using only publicly available datasets introduced a bias towards daylight scenarios with good visibility conditions. The challenge was to find suitable combinations of existing and new datasets for training. The datasets used in this master thesis are further discussed in Chapter 3.

The selected publicly available datasets include: VIPER [RHK17], Microsoft COCO: Common Objects in Context [LMB$^+$14], German Traffic Signs Recognition Database [SSSI11], Berkeley DeepDrive [YXC$^+$18] and CityScapes [COR$^+$16]. They will be discussed in more detail in Section 3. The self-recorded sets include data recorded by project partner ZKW Group GmbH and labelled by TU Wien under the CarVisionLight Project. These labelled images were then segmented into training, evaluation, and test sets. These topics will be discussed further in Section 4.4.5.

Algorithmic Development

After taking the insights gained from the previous analysis into account, the learning algorithm was developed by utilizing TensorFlow v1.13 [ABC$^+$16], Python, C++, and YOLOv3 [RF18]. The technical details will be discussed in Section 4. The main focus of this master thesis' implementation lies in retraining the existing CNN YOLOv3 [RF18], to enable it to detect objects in street-specific night scenes. This was achieved by successively augmenting the training dataset, which resulted in fifteen distinct training and evaluation cycles.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

1.4 Structure of the Work

This master thesis is structured into several chapters. Given that object detection via Convolutional Neural Networks (CNNs) incorporates techniques from computer vision and other computer science disciplines, a theoretical foundation is essential prior to delving into the implementation. Therefore, Chapter 2 introduces the reader to neural networks, the functionality of CNNs, and their historical development in relation to object detection. This is then followed by a detailed examination of YOLOv3, the CNN used for this master thesis, as well as an explanation of the evaluation methods. This chapter culminates in a review of current object detection approaches in night scenes, providing the reader with an explanation for the approaches selected for this master thesis. Chapter 3 provides an in-depth explanation of the datasets used for training, testing, and evaluating the CNN. The following Chapter 4 then describes the implementation done for this master thesis. This consists of the design concept, a initial test run, the training and evaluation pipeline, as well as how the data has to be adapted in order to be usable. This discussion on implementation is succeeded by Chapter 5, which presents and analyzes the results that were achieved for each training phase. This chapter also involves a critical reflection of the work and obtained results. The final Chapter 6 contains the conclusion of this master thesis.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Background and Related Work

This chapter introduces machine learning, object recognition, Neural Networks (NNs), and Convolutional Neural Networks (CNNs). It begins with a brief overview of various object recognition techniques in Sections 2.1 and 2.2. Then, it outlines the classification techniques used in this master thesis by introducing Neural Networks in Section 2.3 and presenting CNNs in Section 2.4. Once the reader understands the basic functionalities used for object detection - a topic discussed in Section 2.5 - the chapter examines the CNN used in this thesis, YOLOv3 [RF18], in Section 2.6. Section 2.7 discusses the evaluation method used to measure the accuracy of the object detector, Intersection over Union (IoU). Finally, Section 2.8 reviews related works on object detection in night scenes.

2.1 Machine Learning

Conventionally, computer programs are manually programmed to execute a specific task. However, machine learning introduces a paradigm shift by substituting a portion of the human input with a learning algorithm, as stated by Goodfellow et al. [GBC16]. The surge in computational power coupled with the widespread adoption of machine learning algorithms has led to an increased use of these techniques in the realm of computer vision [GBC16]. As described by Bishop et al. [Bis06], supervised and unsupervised learning should be distinguished when talking about different machine learning techniques:

Supervised learning refers to a type of learning algorithm that requires human interaction before the program can proceed. This process is called labelling or data annotation. It describes the manual process of marking, for example, the location of an object on a given image and adding the class to which this object belongs, annotating the image. Tools like LabelImg [Lab] can be used to achieve this, as illustrated in Figure 2.1. These annotated images are then used for training the learning algorithm. After learning from these examples, the algorithm can then predict the locations and classes of previously

TU WIEN Bibliothek Your knowledge hub

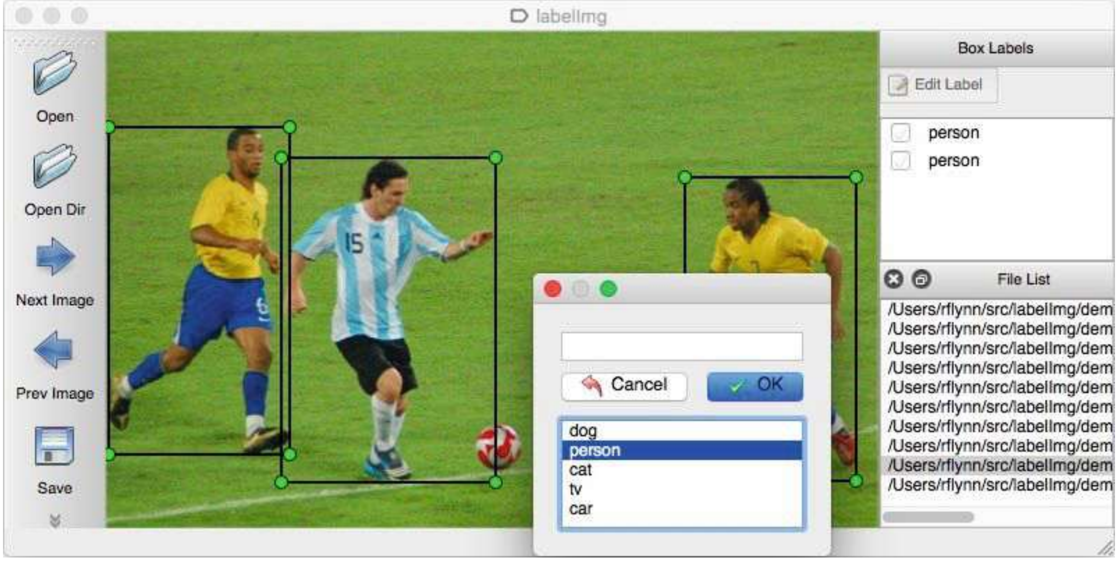

labelImg
Open
Open Dir
Next Image
Prev Image
Save
Box Labels
Edit Label
person
person
File List
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
Cancel
OK
dog
person
cat
tv
car

Figure 2.1: LabelImg being used for image annotation. The user can draw a rectangle along the object and attaches a class to it - in this case "person" to the football player. Source: [Lab]

unseen data. [Bis06]

Unsupervised learning on the other hand, describes the method of learning data properties without human interaction. In principle, unsupervised learning tries to extrapolate algorithmic relationships between categories of data. This involves data that has not been labelled, classified, or categorized. Instead, the machine learning algorithm groups similar entities together to find common attributes. Popular examples include clustering methods like ask-means, hierarchical clustering, and some forms of neural networks. [Bis06]

The focus of this master thesis lies within supervised learning, as the basis for the learning algorithm involves labelled images.

2.2 Object Recognition

Object recognition involves identifying and locating specific objects within images or videos, as stated by Goodfellow et al. [GBC16]. The term "object recognition" encompasses both object detection and object classification. Object detection refers to the task of locating objects present within an image, while object classification focuses on identifying the class of the objects. The objects and their classes used for object recognition must be defined in advance. The possible applications for this are manifold and highly dependent on their final use [GBC16]. In the context of this master thesis, these objects and their corresponding class names are: car, person, bus, bicycle, truck, train, rider, motorbike, traffic light, and traffic sign.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Upon successfully identifying an object, the subsequent step involves marking the object's location within the image [GBC16]. Optionally, the object's location can be visually highlighted by either drawing a bounding box around the object or through segmentation. Segmentation is a technique that involves marking every pixel in the image that contains the object. This master thesis will focus on the bounding box technique, in which a rectangular box encapsulating the object is represented by a four-element vector. This vector can either denote two corner coordinates of the box $(x1, y1, x2, y2)$ or the box's center location, width, and height (x, y, w, h). Each bounding box can optionally be accompanied by a confidence score, indicating the likelihood of the box containing the identified object. Figure 2.2 provides an example of bounding box object detection, with the identified classes (dog, bicycle, and truck) and their corresponding confidence scores displayed above the box. Prior to delving into the details of Neural Networks and CNNs in Sections 2.3 and 2.4, a brief review of object detection in the field of computer vision will be provided in the following sections.

Before the onset of CNNs, feature extraction for object detection was implemented using methods such as the histogram of oriented gradients (HOG) and scale-invariant feature transform (SIFT) [Bis06]. A feature can generally be defined as information needed to solve the computational tasks associated with a specific application. Feature detection methods identify specific image points and subsequently calculate a specific image feature [Bis06]. It is noteworthy that in the realm of computer vision, each object class possesses its own set of features that define the class to which the object belongs. These feature sets, along with the object class, are stored in a database, and subsequent object class detection algorithms can utilize this information for object recognition. Prominent feature extraction examples include the Viola-Jones object detection framework based on Haar features [VJ04], scale-invariant feature transform (SIFT) [Low99], and histogram of oriented gradients (HOG) [DT05], which will be reviewed in Sections 2.2.1 - 2.2.3.

2.2.1 Viola-Jones Object Detection Framework

The Viola-Jones object detection framework, as described by Viola and Jones [VJ04], was the first framework providing object detection rates in real-time, with the primary focus on face detection. It exploited the fact that human faces generally share similar proportions, therefore a basic set of rules was defined and matched using Haar features. Haar features calculate the difference of the sum of pixels of areas inside a predefined rectangle, which can be at any position and scale within the original image. For the human face, the defined regions are around the eyes, nose, and mouth. Haar features are constructed accordingly, with the nose bridge being the brightest region in the human face, followed by the upper cheeks, and the region around the eyes being the darkest. An example of these feature regions applied to a human face can be seen in Figure 2.3. This detection framework is an example for how much a detection system depends on the nature of the object it tries to identify.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

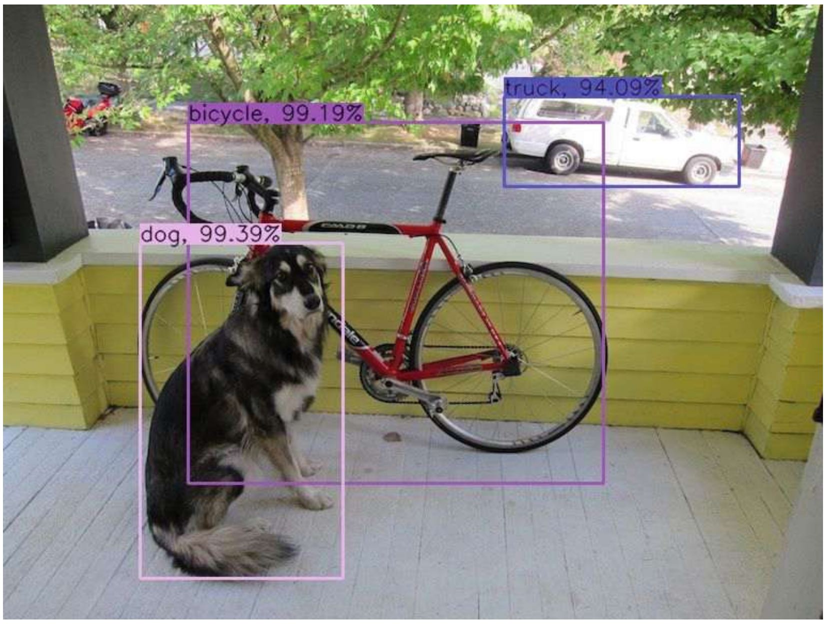


Figure 2.2: This example for object detection shows three detected objects in the image: a dog, a bicycle and a truck. Around each detected object, a bounding box, with the corresponding confidence values is drawn. These confidence values (99.39% for the dog, 99.19% for the bicycle and 94.09% for the truck) indicate the likelihood that this box contains the detected class. A value of 100% represents absolute certainty that the box contains the detected object. Source: [RF18].

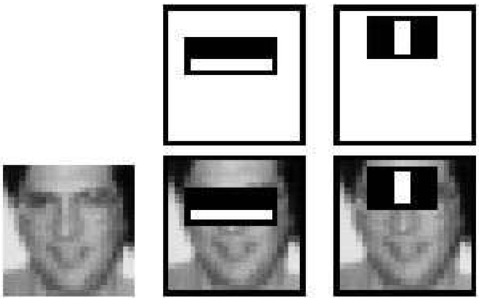


Figure 2.3: Haar feature applied to the image of a human face. The first feature measures the difference in intensity between eyes and upper cheeks, the second feature shows the intensity in the eye regions compared to the bridge of the nose. Source: [VJ04]

2.2.2 SIFT

Scale-invariant feature transform (SIFT) was first published by Lowe [Low99] in 1999. The basic functionality involves transforming a given image into a large collection of feature vectors at predefined key locations. In the case of the original paper, these key locations were defined as min and max values of a Difference of Gaussians (DoG) function applied to the image. The DoG or edge detection filter functions as a band-pass filter

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

by discarding high-frequency information representing noise. It removes low-frequency components representing homogeneous areas like the color, leaving the edges in the image. In the next step, keypoint locations are defined, and dominant orientations are assigned to these detected key points. This ensures that the found key points are invariant to image translation, scaling and rotation, and partially invariant to illumination changes. Each SIFT keypoint is specified by a 2D location (scale and orientation), which gets stored in the database accordingly. A visualization of feature points found in an image can be seen in Figure 2.4. [Low99]

2.2.3 HoG

Histogram of oriented gradients (HOG) is a feature descriptor developed by Dalal et al. [DT05], which counts occurrences of gradient orientations in an image. Objects in an image can be described by the distribution of edge directions - with similar objects having similar edge directions. Therefore, the image is divided into small cells and for each pixel within each cell, a histogram of gradient directions is created. These image gradients describe intensity changes across each cell and can be calculated in any direction. In the next step, a histogram - with the histogram bins representing the different directions and the value of each bin corresponding to the magnitude of intensity changes in that direction - is created. In the last step, the feature descriptor of the found object as the concentration of these histograms is created. A visualization of the HOG feature descriptor can be seen in Figure 2.5. [DT05]

2.2.4 Classification

Once these features have been extracted, the next step is to classify the objects in images unknown to the system. A commonly used classification method is Bag of Words (BoW) or Bag of Visual Words (BoVW). This method is inspired by text classifications from Joachims [Joa98] and McCallum [MN98], where the number of occurrences of words in a document gets counted. A frequency histogram of the words counted was created and stored for future reference. This process was repeated for different texts in distinct categories. The category of a new text was then classified by calculating its frequency histogram. For a Bag of Visual Words, an image is represented as a set of features, key points, and descriptors. These are then used to construct a “visual vocabulary”, to represent each image as a frequency histogram of features. From this frequency histogram, similar images can be found, or the category of the image can be predicted.

For a computer vision system to detect and classify objects, a considerable amount of training data is required. Since it is not possible to cover every instance of an object in every shape and form, the system has to generalize the training data to handle previously unseen data points. As described by Bishop [Bis06], a method for generalization which is too simple can fail to capture aspects of the model (called underfitting), while too complex methods tend to detect small details and noise as objects (named overfitting).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

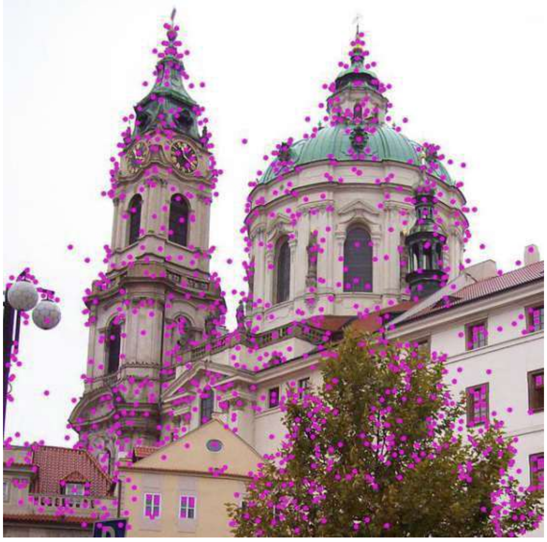


Figure 2.4: Visualization of SIFT feature points found in an image, after difference of Gaussians has been applied. Source: [sif]

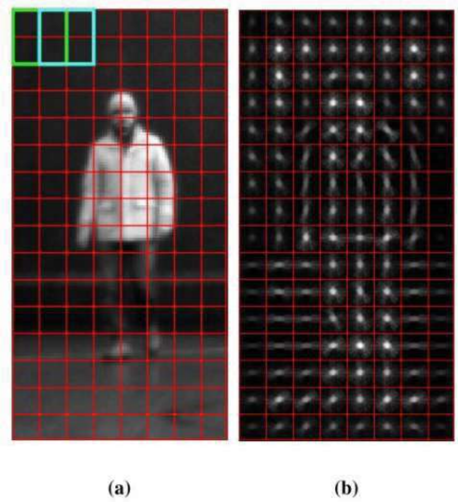


Figure 2.5: Visualization of HoG feature points. Subfigure (a) shows the image cells in red, in which HoG features are calculated. Subfigure (b) visualizes the HoG descriptor computed for the same image. Source: [FGB13]

How an algorithm performs under various conditions can be measured by both quantitative and qualitative evaluation. Qualitative evaluation involves the observation of the results by human standards. Quantitative evaluation, on the other hand, assesses the method utilizing predefined mathematical functions, which will be discussed in more detail in Section 2.7.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.3 Neural Networks

Neural networks - originally called artificial neural networks (ANN) [GBC16] - are computational models that use training sets to develop a classification system. The use cases for these classification systems vary, although this master thesis will focus solely on networks used for object detection in images. The term artificial neural network originally stems from researchers trying to find a mathematical representation of the human brain, with the added prefix artificial, making the distinction explicit. Pioneering research works start from McCulloch and Pitts in 1943 [MP43], and Rosenblatt in 1958 [Ros57].

Generally, neural networks are composed of input, output, and at least one intermediate layer called the hidden layer. These layers consist of elements called neurons or perceptrons. The concept of a perceptron was first introduced in 1958 by Rosenblatt [Ros57], as a method for pattern recognition. It builds upon the structure of the McCulloch-Pitts-Cell [MP43], which takes an input signal, compares it to a threshold and produces a binary output. The Rosenblatt perceptron developed this further by taking several inputs $x_0, x_1, ...x_n$ and mapping them onto a single binary output value. This is calculated by concatenating each input while multiplying it with the corresponding weight $w_0, w_1, ...w_n$ - a real number that expresses the importance of the respective input to the output. An example for a perceptron can be seen in Figure 2.6. The perceptron's binary output is calculated by comparing the weighted sum $\sum_{j=0}^{n} w_j x_j$ to a predefined threshold. By combining several perceptrons, a network can be created (as seen in Figure 2.7). [Bis06]

As described by Grossi and Buscema [GB07], a neural network is a combination of several neurons. These neurons are grouped into layers, with the output of one layer being fed as input to the neurons of the next layer. A visualization of this can be seen in Figure 2.7. Here, the leftmost part in red shows the input layer. Here, individual input values $x_1, ..., x_4$ are given to the network. This first layer usually passes data along without modifying it, providing the input for the network. Next are the hidden layers (depicted in blue in Figure 2.7). The weighted connection is calculated, and the output is passed onto the next layer as the new input. A network typically consists of one or more hidden layers, depending on the application. Hidden layers are not visible to an outside observer; only the input and the output layer are. The last layer in green shows the output layer, which converts the hidden layer values to an output value, which correlates to a predefined class. [GB07]

Neurons per definition only provide one single output [GB07] - the example in Figure 2.7 shows the output being used as input for several neurons. While the input and output of a perceptron are binary, this is not the case for artificial neurons, whereby input and output can take on any value between 0 and 1. In a neural network, each layer of neurons is connected to the preceding and succeeding layer through weighted connections. Each neuron y_j computes a weighted sum of its inputs x_i using weights w_{ij} and a bias b_j:

$$y_j = \sum_i x_i w_{ij} + b_j$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

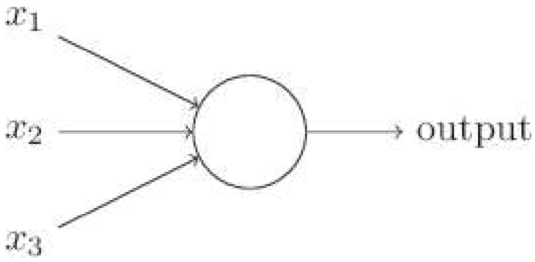


Figure 2.6: Concept of a perceptron by Rosenblatt [Ros57], which takes several inputs and maps them onto a single output. Source: [akc]

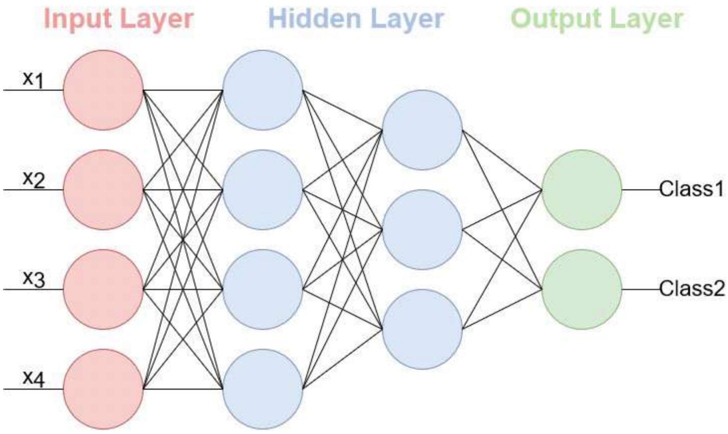


Figure 2.7: Example of a fully-connected multi-layer neural network: Starting with several input values, from x_1 to x_4, which are fed into the input layer (red nodes). This is followed by processing the input in several hidden layers (blue nodes) and finally creating two outputs (green nodes), Class1 and Class2.

2.3.1 Backpropagation

As described by Rojas [Roj96], the learning process of a neural network is based on a back-propagation algorithm. It involves two iterative steps called the forward and the backward pass. First, an input vector of the training set is propagated forward through the nodes until it reaches the end of the network. The network output is then compared to the desired output (also called the ground truth), and the difference between these two values is the “error” of the network. In the next phase, the error is then propagated back through the network, starting from the output until it reaches the start of the network. While propagating back, the weights (associated with each decision) are updated, depending on their influence on the error. [Roj96]

In order to update the weights, a loss function is utilized, as described by Janocha et al. [JC16] and Goodfellow et al. [GBC16]. This function, in general, optimizes the parameters of a neural network. With this optimization problem solved by the loss function, this “loss” is calculated by matching the target with the predicted values. The

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

gradient descent method, a first-order iterative optimization algorithm for finding a local minimum, is then used to update the weights such that the loss is minimized.

2.3.2 Loss Function

During neural network training, the model's error is calculated as part of the weight optimization process. For this, a loss function must be chosen. Some options include maximum likelihood estimation, cross-entropy, mean squared error, and categorical cross-entropy - to name a few. [WMZT20]

Maximum Likelihood Estimation (MLE) is a method for finding the best statistical estimates of parameters from historical training data. In a neural network, the training data includes one or more input variables. As stated by Bishop [Bis96], MLE aims to find the optimal parameter values by maximizing likelihood functions derived from the training data. These input variables are weighted by using a model to estimate the parameters that best map the inputs to the output during training. The model aims to make predictions that match the data distribution of the target variable. Under maximum likelihood, a loss function measures how closely the model's predictions align with the distribution of target variables in the training data [Gas19].

When tackling a classification problem, which involves mapping input variables to a class label, the task can be viewed as predicting the probability of an example belonging to a class. In the training dataset, the probability of an example belonging to a specific class is either 1 or 0, since the class of each sample is known. Therefore, using maximum likelihood estimation, the objective is to find a set of model weights that minimize the difference between the model's predicted probability distribution and the actual distribution of probabilities in the training data. This function is called ***Cross-Entropy Loss***.

Mean Squared Error (MSE) is calculated by taking the difference between the predicted and actual value. MSE Loss is calculated by taking the mean squared differences between the target and predicted values, which indicates how well the network is performing. [SW11]

Categorical Cross-entropy (log loss) is useful for multi-class classification tasks. The final layer is passed through a softmax activation function (see below), so each node output has a probability value between $[0, 1]$. [SSA17]

2.3.3 Activation Function

As described by Sharma et al. [SSA17], in the last step of each layer, an activation function φ determines whether a neuron should be "activated" or not. Several possibilities for an activation function have to be selected for each network individually, depending on the use case [SSA17]. Previously in this chapter, the most basic activation function, the step function, has been discussed implicitly. This step function, also called threshold-based

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

activation function, selects the output depending on its position above or below a certain threshold.

$$A = \begin{cases} 1, x \geq val \\ 0, x < val \end{cases}$$

The value A represents the activation function, x the input to a neuron, and val the threshold. The function can be seen in Figure 2.8.

This step function is often insufficient since it only allows for binary classification. Other functions which create a softer transition between values include the sigmoid and tangens hyperbolicus functions. These functions can be seen in Figures 2.9 and 2.10.

Another function which allows for easy computation and differentiation (a requirement for back-propagation) is the Rectified Linear Units (ReLUs) function. The most basic form of this returns the output value x, if x is positive and 0 otherwise, which can be seen in Figure 2.11. Due to the low computational costs, ReLUs have largely replaced sigmoidal activation functions [KSH12], [sta].

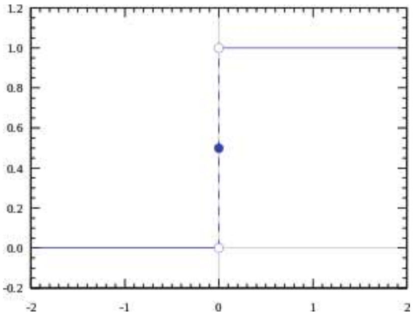


Figure 2.8: Visualization of a step function, which shows a change in the input value from 0 to 1, once a threshold has been met. In this case, the threshold is set at 0.

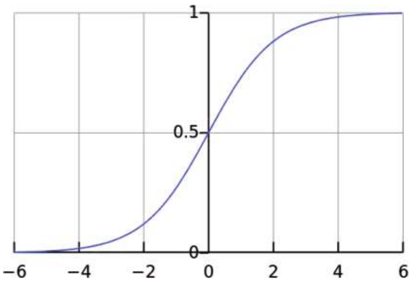


Figure 2.9: Visualization of a Sigmoid function, $A = \frac{1}{1-e^{-x}}$. The value change is more gradual when compared to the step function shown in Figure 2.8

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

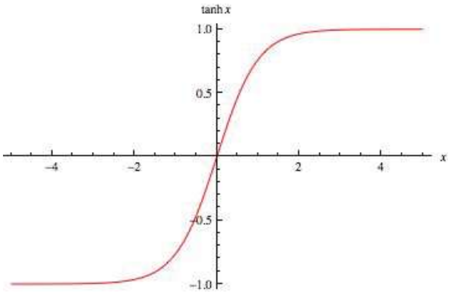


Figure 2.10: Visualization of a tangens hyperbolicus function, $A = \frac{2}{1+e^{-2x}} - 1$.

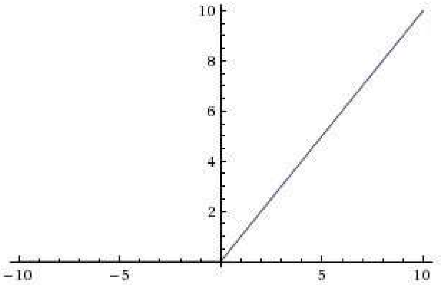


Figure 2.11: Visualization of a Rectified Linear Units (ReLUs) function, $A = max(0, x)$. The value of the function is 0 below a predefined threshold and from this threshold forward (in this case 0), the value gets mapped directly.

2.4 Convolutional Neural Networks

Within the context of a deep learning system, a Convolutional Neural Network (CNN) is a class of neural networks, most commonly applied to analyzing images. CNNs are fully connected networks, in which each neuron in one layer is connected to all neurons in the next layer, as shown in Figure 2.7. In the context of computer vision research, a CNN takes an input image and assigns weights and biases to various components of it. While traditional methods for object detection were usually engineered to fit the specific use case, CNNs learn characteristics in the image, allowing them to recognize these features in new images.

The connectivity pattern of neurons in the visual cortex of animal brains inspired the architecture of a CNN [Fuk80]. Individual neurons respond to stimuli in a restricted visual field region known as the receptive field instead of processing the whole input image at once. The receptive field, in general, was previously described by Robert Burke [Bur07] as the area of the body where a stimulus elicits a reflex. Specifically, the receptive field of a sensory neuron is the region of the visual field in which a stimulus will trigger the firing of that neuron.

This design made object recognition much more feasible and was therefore taken as inspiration for CNN design. Unlike traditional neural networks, each pixel intensity in an

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

image would require its own weight and neuron. A 410x410 pixel image would therefore lead to 168.100 weights per color channel for a learning network. A model of this size is prone to overfitting and slow to perform. Therefore, convolution was added to the basic neural network structure. [sta]

Since not every pixel in the image can be considered, dimensionality must be reduced before further processing is possible. Instead of using every value in an image, it is possible to single out relevant data (edges, colors, and many more) by utilizing convolution matrices. Images can be filtered using a convolution matrix to produce visible effects, like edge detection, sharpening, or blurring. This decreases the complexity and number of required neurons since the number of free parameters is reduced for training. [sta]

The general expression of a convolution is

$$g(x,y) = w * f(x,y) = \sum_{s=-a}^{a} \sum_{s=-b}^{b} w(s,t) f(x-s, y-t)$$

whereas $g(x,y)$ is the filtered image, $f(x,y)$ is the original image, and w is the filter kernel. Every element of the filter kernel is considered by $-a <= s <= a$ and $-b <= t <= b$.

The theoretical concept of a CNN was first introduced by Fukushima [Fuk80] in 1979 as Neocognitron and further developed by LeCun et al. [LBD+89], who introduced backpropagation for training increasingly complex neural networks with convolution layers. This research was applied to the recognition of hand-written digits. Due to lack of computational resources for training deeper and more complex networks, the first ground-breaking result was achieved in 2012 by Krizhevsky et al. in AlexNet [KSH12], which will be further discussed in Subsection 2.5.1.

The typical structure of a CNN can be seen in Figure 2.12. It consists of two sections: feature extraction and classification. Feature extraction contains several convolution layers, combining several small two-dimensional filters or feature detectors. The first layer captures low-level features such as edges, color, and gradient orientation by applying a filter kernel on the receptive field (the currently observed image section, which is highlighted in Figure 2.12 by a black circle). The architecture adapts to high-level features like larger shapes with added layers, providing the CNN with an overall understanding of the image [sta].

These convolution layers are alternated by subsampling or pooling layers which perform a combination of dimensional reduction. The pooling layer is responsible for reducing the spatial size of the convolved feature to decrease the computational power required to process the data. Depending on the network, max-pooling (which returns the maximum value of the convolved feature and performs de-noising along with dimensionality reduction) or average pooling (which returns the average of all values) can be chosen. Finally, a fully connected neural network is included at the end, which results in a classification of the found objects in the image.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

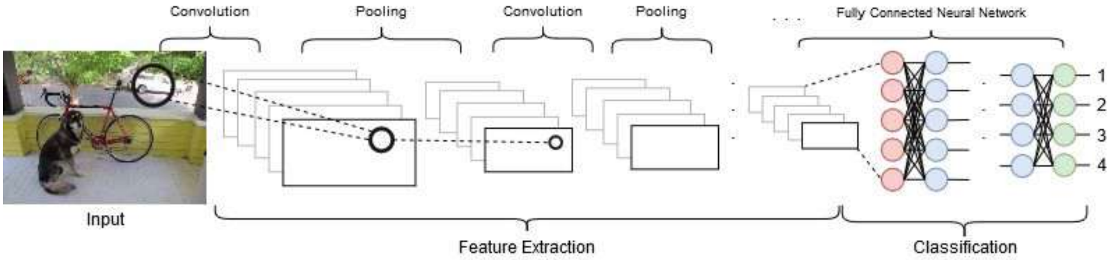


Figure 2.12: CNN example structure. An input image is processed by the network, starting with feature extraction through alternating convolution and pooling layers. The objects in the image are then classified by the classification layer, using a fully connected neural network.

2.5 Object Recognition CNNs

As described in Section 2.2, object recognition refers to the computer vision problem that deals with identifying and locating a set of predefined objects in images or videos. In a CNN, a fixed-size sliding window moves across the input image to extract features. These features are then fed into the neural network for classification, identifying and returning the object's class found within the sliding window.

In the following section, several CNNs and CNN-like concepts will be described. In particular, YOLOv3, the CNN used for this master thesis, will be discussed in more detail in Section 2.6.

2.5.1 AlexNet

One of the earliest successful object detection algorithms using CNNs was achieved with AlexNet by Krizhevsky et al. [KSH12]. AlexNet solved the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by training a CNN for 1000 categories, with eight learned layers - five convolutional and pooling layers, followed by three fully-connected layers and overall 650,000 neurons [KSH12]. The standard way to model a neuron's output f with input x is with $f(x) = tanh(x)$ [Wu17], but Krizhevsky et al. [KSH12] chose $f(x) = max(0, x)$, referring to the neurons utilizing this functionality as Rectified Linear Units (ReLUs). While CNNs have been used for image classification before, AlexNet was the first one to utilize Rectified Linear Units (ReLUs). [KSH12]

Figure 2.13 shows the AlexNet architecture, with five convolution layers, followed by max-pooling, and finally, three fully connected layers, which classify the found object. The original image is split between two GPUs to reduce processing time, with communication occurring only at specific layers. [KSH12]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

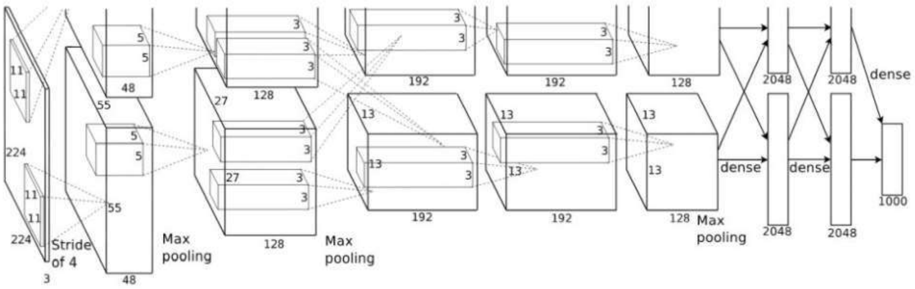


Figure 2.13: AlexNet CNN Architecture. Source: [KSH12]

2.5.2 Regions with CNN Features (R-CNN)

When replacing traditional object recognition methods with CNNs, one often encounters the problem of them being slow and computationally expensive. This issue was addressed with the development of Regions with CNN features (R-CNN) [GDDM14], which combines a region-wise classifier with CNNs to localize and segment objects. The object proposal algorithm described in the original paper by Girshick et al. [GDDM14] is called “Selective Search” and uses local cues like texture, intensity, color, and other features to generate possible object locations, significantly reducing computational time.

2.5.3 Spatial Pyramid Pooling Net (SSPNet)

Spatial Pyramid Pooling Net (SSPNet) by He et al. [HZRS15] developed these ideas further by introducing adaptive-sized pooling. SSPNet utilizes “Selective Search” proposed by R-CNN and calculates the CNN representation for each possible object location only once. Therefore the convolution is only calculated for an image section, not the whole input image. Besides that, as the name suggests, spatial pooling instead of max pooling is being used. Instead of returning the maximum value of the convolved feature, a region of arbitrary size is reduced into a constant number of bins, producing a feature vector. The problem with this system is that back-propagation is not trivial and therefore takes more computational time. [HZRS15]

2.5.4 Faster R-CNN

Faster R-CNN [RHGS15] introduced an improvement in execution time through the use of a region proposal network. The paper by Richter et al. [RHK17] introduces the concepts of anchor boxes and bounding box regression. Anchor boxes are predefined bounding boxes designed to capture the scale and aspect ratio of objects to be detected, based on the sizes of objects in the training dataset. These are often calculated beforehand using k-means clustering and then passed to the CNN. Bounding box regression involves scoring each selective search detection and predicting a new bounding box for detection. This process is applied within the anchor boxes, resulting in bounding boxes of various sizes with different probabilities for each class. [RHK17]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.5.5 You Only Look Once (YOLO)

So far, all previously discussed CNNs follow the same pipeline: first, generating object proposals, and then classifying these proposals. However, object detection can also be approached using regression-based methods. A prime example of this is You Only Look Once (YOLO) [RDGF16]. In YOLO, an image is first divided into several KxK grids, and each grid predicts N bounding boxes, which can be seen in Figure 2.14. For instance, in this example, the image is divided into 13x13 grid cells and each cell predicts one of three possible classes (dog, bicycle, car).

Additionally, YOLO calculates a confidence score, which reflects the accuracy of the bounding box and whether or not it contains an object. So $KxKxN$ boxes are predicted, and if their confidence score is too low, the box is ignored. During runtime, YOLO only needs to process the image once, allowing it to operate in real-time [RDGF16].

2.6 YOLOv3

More specifically, YOLO in its third iteration - YOLOv3 - was chosen for this master thesis. Its architecture can be seen in Figure 2.15 and the functionalities will be discussed in more detail within this section.

2.6.1 Feature Vectors

YOLOv3 uses Darknet53, a 53-layered network trained on ImageNet, to which 53 more layers are added, resulting in the 106-layered architecture of YOLOv3. One layer describes one mathematical operation (detection, upsampling, or convolution). In contrast to the original implementation of YOLO, this third iteration now detects objects at three different image scales, shown in Figure 2.15 as light brown layers. The object detection is done by applying a 1x1 detection kernel on the feature maps of three different sizes at three different places in the network. [yola]

The shape of the detection kernel is:

$$1x1x(B * (5 + C))$$

whereby B describes the number of bounding boxes a cell on the feature map can predict (with YOLOv3, this number is 3) and C the number of classes. 5 stands for the number of bounding box attributes. These are: the confidence followed by either (x_1, y_1, x_2, y_2), the corner points of the bounding box, or (x, y, w, h), the center of the box, width, and height. For example, the detection kernel for the COCO dataset would be calculated as follows: $B = 3$, $C = 80$, which results in a kernel of 1x1x$(3 * (5 + 80))$ =1x1x255 [yola].

YOLOv3 extracts features at three different image scales. The first extraction happens at the 82nd layer. For the preceeding 81 layers, the image is downsampled and processed through several convolutional layers, shown in magenta in Figure 2.15,resulting in a

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

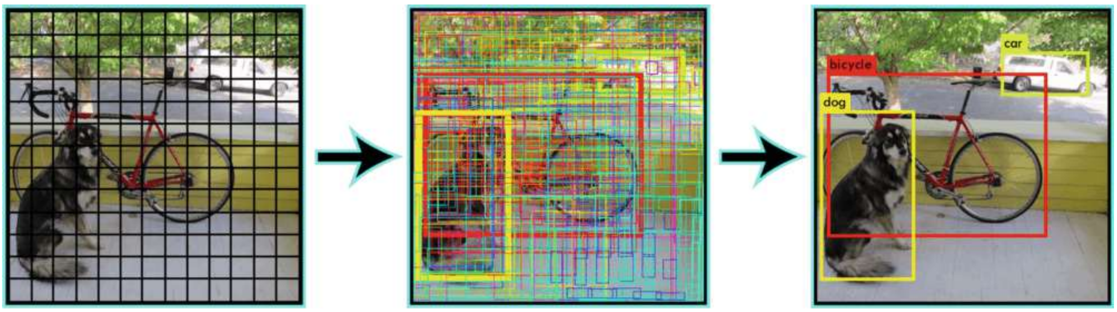


Figure 2.14: Example image of how the YOLO object detector [RDGF16] calculates bounding boxes. The image gets segmented into grids, in which each grid predicts N bounding boxes. These are then filtered, resulting in the final classification. Source: [RDGF16]

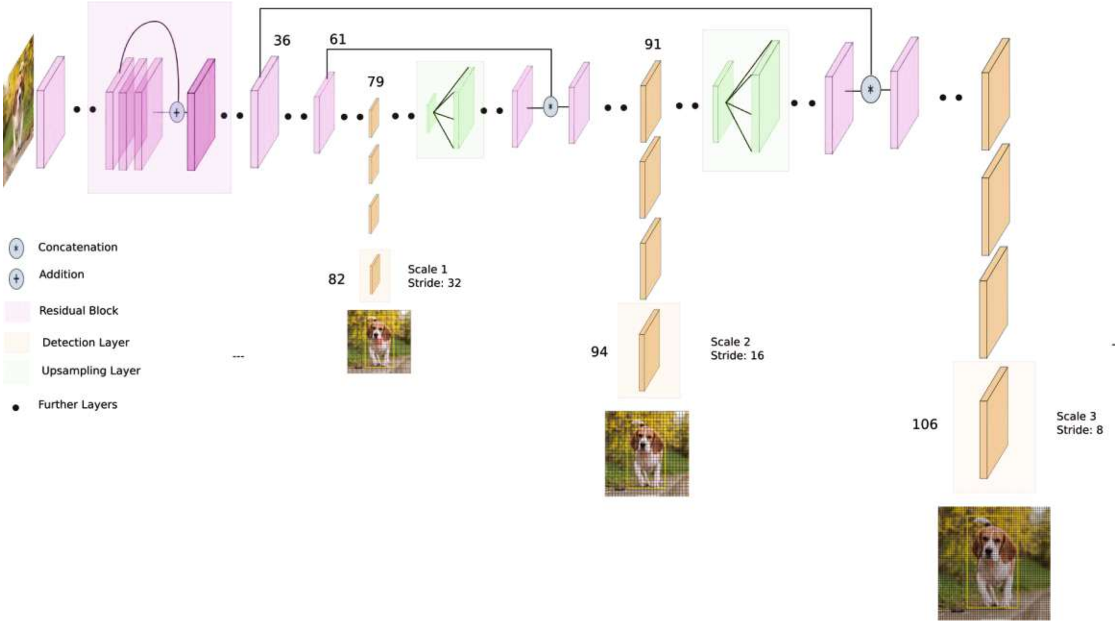


Figure 2.15: The YOLOv3 network architecture, as described on the TowardsDataScience webpage [yola], processes an image of a dog by first passing it through a residual block to reduce the image scale. This is followed by several detection layers that attempt to identify the object at the current scale. After this, upsampling layers resize the image to a larger resolution, and the detection process is repeated. Source: [yola]

downsampled feature map. For instance, if the original image has a resolution of 416x416, the downsampled feature map will be 13x13x255. Next, the feature map from layer 79 undergoes convolution, before being upsampled twice (in the example used before, this results in a 26x26 image). The upsampling is depicted in green in Figure 2.15. The upsampled image is then concatenated with the output from layer 61 and convoluted once more. The second feature extraction occurs at layer 94, producing another feature map (taking the example from before, the result is a feature map of size 26x26x255). Finally,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the output from layer 91 is upsampled, convoluted, and concatenated with the output from layer 36, resulting in a final feature map at layer 106 (in this example, 52x52x255).

Bounding boxes are detected at those three different scales (levels), and only the boxes with the highest confidence are selected as the final result. These different scales enable YOLOv3 to detect small objects more effectively than the previous YOLO versions, v1 and v2. [yola]

2.6.2 Anchor Boxes

As proposed in Faster-RCNN by Richter et al. [RHGS15], YOLOv3 also utilizes anchor boxes as well. Due to the significant variance in the scale and aspect ratio of bounding boxes within an image, using anchor boxes improves object detection rates. These anchor boxes have pre-defined aspect ratios, determined before training by running k-means clustering on the training dataset [RF18].

Traditionally, object detection algorithms used a sliding window approach, scanning the entire image and performing image classification on each window. However, this method was inefficient. YOLOv3 improves on this by using convolutional outputs to create a grid of feature values with resolutions of 13x13, 26x26, and 52x52. Each cell in this grid is assigned anchor boxes. Instead of predicting boxes for each sliding window, YOLOv3 predicts boxes based on these anchor boxes. [RF18]

YOLOv3 has three anchor boxes per grid cell, as shown in Figure 2.16. The exact scale of these anchor boxes is calculated by utilizing k-means clustering. With each object in the anchor box, the offset (x, y, width, height) has to be predicted and a score added, which indicates if the box contains an object.

2.6.3 Loss Function

Using the final detection output, which includes the offset from the anchor box, the score, and the confidence, the loss function against the ground truth can be calculated. The loss function consists of four components: centroid loss, width and height loss, score loss, and classification loss. During training, these values are minimized, and the weights are adjusted accordingly until training is complete. [RF18], [intb]

Bounding Box Loss

The loss for the bounding box includes centroid (x, y) loss and width and height loss. To minimize the centroid loss, the center box location is compared to the ground truth location. Redmon and Farhadi [RF18] suggest using mean square error for this regression problem. The same approach is recommended for the width and height of the box.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

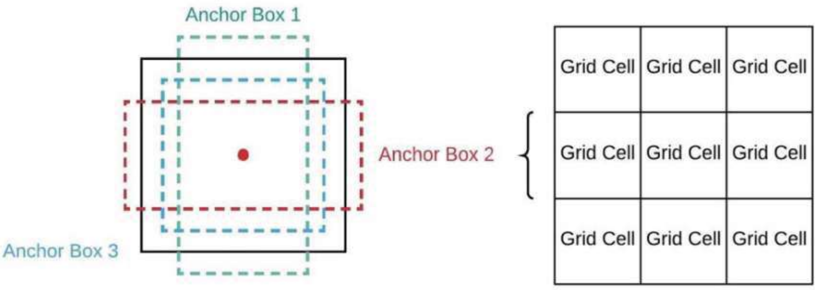


Figure 2.16: The YOLOv3 anchor box detection as described by the TowardsDataScience webpage [yolb]. The image gets split into several grid cells (right side of the figure), and each grid cell gets assigned three anchor boxes (left side of the figure). Within these boxes, objects are detected.

Score Loss
Generally, the score indicates the likelihood of an object being in the current cell, with 1 meaning the cell contains an object and 0 meaning it does not. By measuring this score loss, the network gradually learns to detect regions of interest. Redmon and Farhadi propose using binary cross-entropy for this purpose [RF18].

Classification Loss
In YOLOv3 [RF18], a dataset can contain related labels, such as the object class “woman” and “person”, which have a hierarchical relationship. Each output cell can be associated with more than one class, allowing for multi-label classification. Therefore, the application of binary cross-entropy for each class is required, followed by a summation of all class values, since they are not mutually exclusive.

2.7 Intersection Over Union

After training a CNN, the next step is to evaluate its performance and accuracy. One common evaluation metric is Intersection over Union (IoU), which uses bounding boxes as described by Rezatofighi et al. [RTG+19]. IoU requires predefined ground truth bounding boxes from the training and validation sets, and it compares these with the predicted bounding boxes from the model being evaluated. Figure 2.17 illustrates the comparison between the ground truth bounding box (in green) and the CNN’s predicted bounding box (in red). To calculate the IoU, the Area of Overlap is divided by the Area of Union:

$$IoU = \frac{A_{Overlap}}{A_{Union}}$$

Area of Overlap describes the overlap of the predicted bounding and the ground truth box, whereas the Area of Union describes the area encompassed by at least one of the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

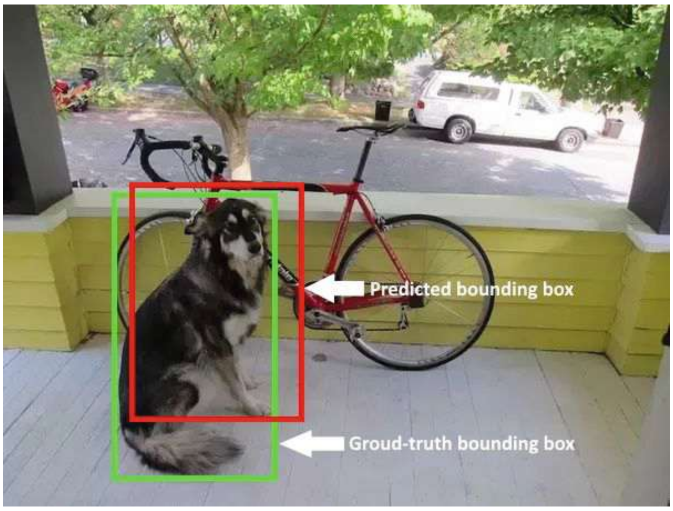


Figure 2.17: The image shows an example of Intersection over Union (IoU) with predicted and ground truth boxes. It shows a dog sitting in front of a bicycle, with two colored boxes: one red and one green. The red box represents the predicted bounding box after the image was processed by the CNN, while the green box represents the ground truth bounding box. The IoU is then calculated by comparing these two boxes.

two boxes. The division of those two results in the IoU score. It shall be noted that an IoU score above 0.5 is considered a “good” prediction, and an IoU score of 1 is nearly impossible to achieve since it requires a pixel-perfect match between the two boxes. A visualization of this can be seen in Figure 2.18.

Classification Evaluation

Evaluating the accuracy of an object detector is done by defining a threshold for the IoU, in which, above a specific value, an object is counted as “detected” [sta]. These thresholds usually range from 0.4 to 0.8, depending on the network and application. These values are calculated by comparing the found objects by the object detector to the “ground truth” of the validation dataset. Depending on whether an object was correctly detected or not, it is labeled differently [inta]:

- True Positive, TP: There is an object, object detector finds it
- True Negative, TN: There is no object, object detector finds nothing
- False Positive, FP: There is no object, object detector finds one or finds a wrong class
- False Negative, FN: There is an object, object detector finds nothing

Examples of this can be seen in Figure 2.19.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

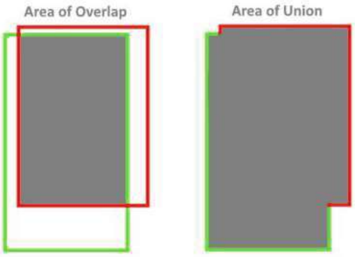


Figure 2.18: Area of Overlap and Area of Union, needed for IoU calculation. The left image shows the Are of Overlap (filled gray), calculated from the overlap of the predicted bounding box (red) and ground-truth bounding box (green). The right image shows the Area of Union, calculated as the union of predicted and ground-truth bounding boxes.

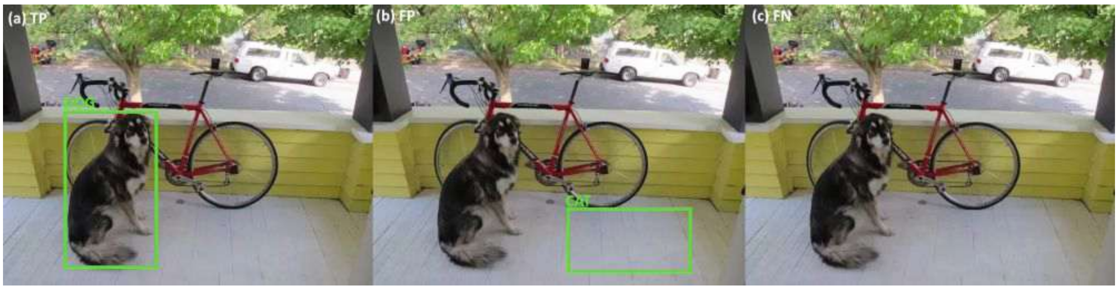


Figure 2.19: Examples for (a) True Positive (TP): the network correctly predicts that an object (in this case, a "Dog") was found. Subfigure (b) shows a False Positive (FP): The network predicts a positive find for an object that is not actually there (in this case a "Cat"). Finally, (c) shows a False Negative (FN): There are objects in the image that should be detected, but the network fails to find them.

It is then possible to evaluate the network's performance using various measures:

Sensitivity evaluates the network's ability to detect true positives (TP).

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Accuracy describes how well the network is able to differentiate between objects.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Specificity describes if and how well the network is able to identify true negative (TN) cases.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.8 Object Detection in Night Scenes Using CNNs

As Lim et al. [LHCB17] state, the training data used for a CNN has to be tailored to allow object detection for a specific problem. For example, Kim et al. [KSK17] study the detection of humans, and Lim et al. [LHCB17] concentrate on traffic sign recognition under illumination variations. With regards to the YOLO network, Tung et al. [TKS$^+$19] examine YOLO's ability to detect objects in shifting illumination conditions. However, the authors do not retrain the network, which served as a starting point for this master thesis.

While at the starting time of this project, work on object detection in night-time traffic scenes was rather scarce, more recent investigations have also targeted solutions to ensure stable object detection during night-time for self-driving systems and smart headlights. Song et al. [SPL24] propose an algorithm which utilizes a candidate search algorithm instead of an anchor box-based method. Their solution was reported to be 22ms faster compared to other state-of-the-art methods, while their mAP (mean average precision) improved by 12% [SPL24]. Other recent works put their focus on related pre-processing methods. For example, Ben-Shoushan and Brook [BSB23] use a combination of multi-sensoral RGB and thermal camera data to process the image before it is fed into the CNN for object detection. Other works seek to provide the CNN with more information regarding the observed object - for example, in the case of Chacon-Murguia and Guzman-Pando [CMGP23] the moving car. In that work, the authors propose the use of a so called “Two-Frame CNN” [CMGP23], which estimates the motion of an object using the temporal movement information. These recent results show that focussing on different aspects of a CNN allows for further improvement of object detection rates. This master thesis aims to improve object detection rates in night-RGB image scenes by using a combination of publicly available and self-recorded datasets.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Datasets

This section describes the datasets used for this master thesis as a basis for training the YOLOv3 network and provides insight into why they were chosen. The selected datasets are discussed in more detail in Section 3.1. This chapter is concluded by Section 3.2, which introduces the datasets which were recorded and annotated for this master thesis.

Publicly available datasets displaying traffic scenes that provide training data for deep learning algorithms are traditionally biased towards daylight and environments with good visibility [RHK17]. This bias, in turn, raises the need for a manual annotation, since available datasets often prove not sufficient for solving the required task [RHK17]. It has to be kept in mind that manual annotation of night scenes with demanding illumination conditions is error-prone and time-consuming, so the challenge moves to finding a suitable combination of existing and new datasets for training, which allow the network to detect objects in night-scenes.

The datasets used in this master thesis include: Microsoft Common Objects in COntext (COCO) [LMB+14], the German Traffic Sign Recognition Database (GTSRD) [SSSI11], VIsual PERception benchmark (VIPER) [RHK17], and Berkeley Deep Drive 100k (BDD) [YXC+18], as well as recordings taken within the CarVisionLight (CVL) project. The following section will review the main characteristics of these datasets and explain the motivation for incorporating them into the training set.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.1 Publicly Available Datasets

Microsoft Common Objects in Context (COCO)

Microsoft COCO is a large-scale object detection and segmentation dataset, which consists of images of complex everyday scenes showing common objects in their natural context, as exemplary shown in Figure 3.1. The COCO dataset contains 91 object types with a total of 2.5 million labelled instances in 328k images [LMB$^+$14]. This dataset was not used for training, but due to its extensive coverage of everyday images, with over 80 classes, it was taken as a basis for evaluation, with YOLOv3 being initially trained with the COCO dataset.

German Traffic Sign Recognition Database (GTSRD)

The first dataset used in the final training setup is the German Traffic Sign Recognition Database (GTSRD), which is a combination of the German Traffic Sign Recognition Benchmark (GTSRB) and the German Traffic Sign Detection Benchmark (GTSDB) datasets. An example can be seen in Figure 3.2. It was created by Stallkamp et al. [SSSI11]. It is a multi-class, single-image dataset, in which each image contains one traffic sign. It comprises over 40 classes of different street signs and over 50k images and shows these signs in various lighting conditions. While the dataset is called “German Traffic Signs”, it should be pointed out that these road signs are subject to the Vienna Convention on Road Signs and Signals [UNC], which has been widely adopted in Europe and Russia and can therefore be used to solve traffic sign recognition problems in European and Russian street conditions.

Visual Perception Benchmark (VIPER)

The second dataset is the VIsual PERception benchmark (VIPER), which was assembled by Richter et al. [RHK17] and can be seen in Figure 3.3. It was created while driving, riding, and walking 184km in the virtual world of the Rockstar video game Grand Theft Auto 5 (GTA 5). It consists of over 250k high-resolution video frames, split into training, validation, and test sets, containing 134K, 50K, and 70K frames, respectively. Each set contains a balanced distribution of data acquired at different times of day and in different lighting conditions (day, sunset, rain, snow, night). The subset used for this master thesis consists of over 15k labeled images showing night scenes with 30 classes, out of which ten were deemed relevant and therefore incorporated for training. The simulation model contains, for the most part, US-American street conditions.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Figure 3.1: An example from Microsoft's COCO [LMB+14] dataset, showing various common objects in their natural context.

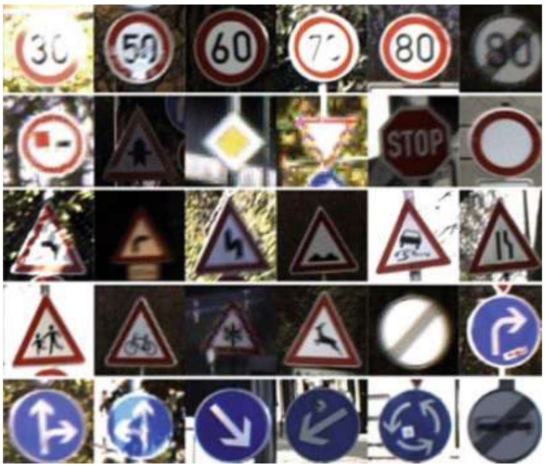


Figure 3.2: Example of German Traffic Sign Recognition Database (GTSRD) [SSSI11], showing several street signs in different lighting situations.

Berkeley Deep Drive 100k (BDD)

The Berkeley Deep Drive (BDD) [YXC+18] dataset is a large-scale, diverse driving video database. It contains annotated images of different scenarios, day, night, and dusk/dawn of 100k driving videos from more than 50k rides, an example of which can be seen in Figure 3.4. These videos were filmed in New York, San Francisco, the San Francisco Bay Area, and Berkeley, in diverse weather conditions (sunny, rainy, snowy). The recordings comprise six weather conditions, six scene types, and three different times of day for each image, with an approximately equal distribution between night and day images. This dataset provides bounding box annotations for ten different categories, all chosen for the classification in this master thesis' object detector.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

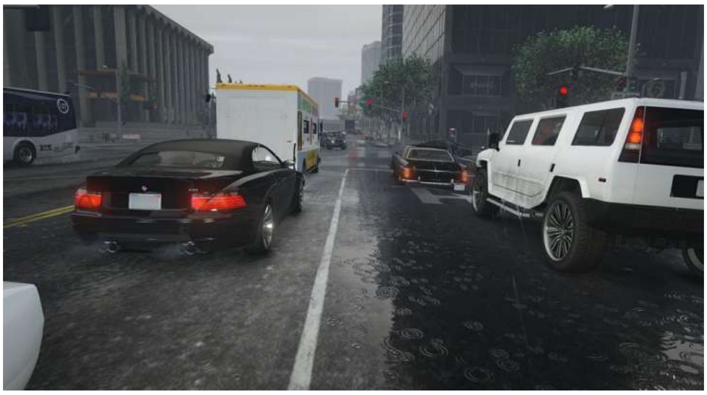


Figure 3.3: An example of the VIPER dataset [RHK17], showing several simulated cars in a simulated city street.

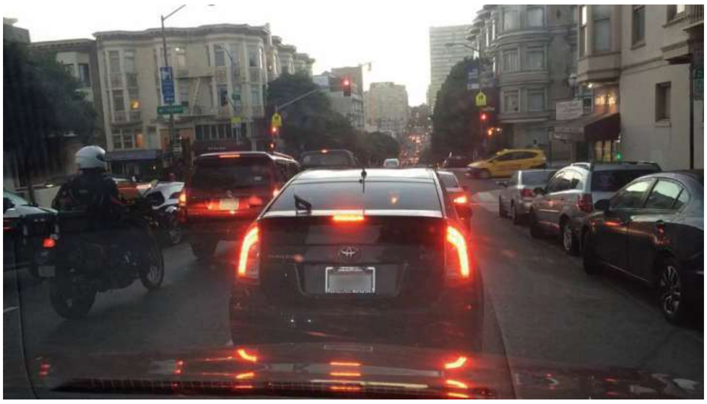


Figure 3.4: Example of Berkeley Deep Drive dataset [YXC$^+$18], showing several cars and a motorbike in the city of San Francisco, waiting in front of a red traffic light.

3.2 Self Recorded Dataset

It is not enough to rely solely on publicly available datasets. While the mentioned datasets include night scenes, they often suffer from light pollution or are recorded in city streets, which means they lack true rural-street conditions. This reduces their effectiveness for object detection in night scenes. Therefore, we decided to add our own self-recorded and self-annotated data, specifically tailored to our use case.

CarVisionLight Dataset (CVL)

The CarVisionLight (CVL) dataset was recorded at dusk and during the night in the countryside of rural Austria. The recordings contain over 50 videos, of which over 9k images were annotated and then split into training and validation datasets. The bounding box annotations contain three classes: person, car, and street sign. An example can be seen in Figure 3.5.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

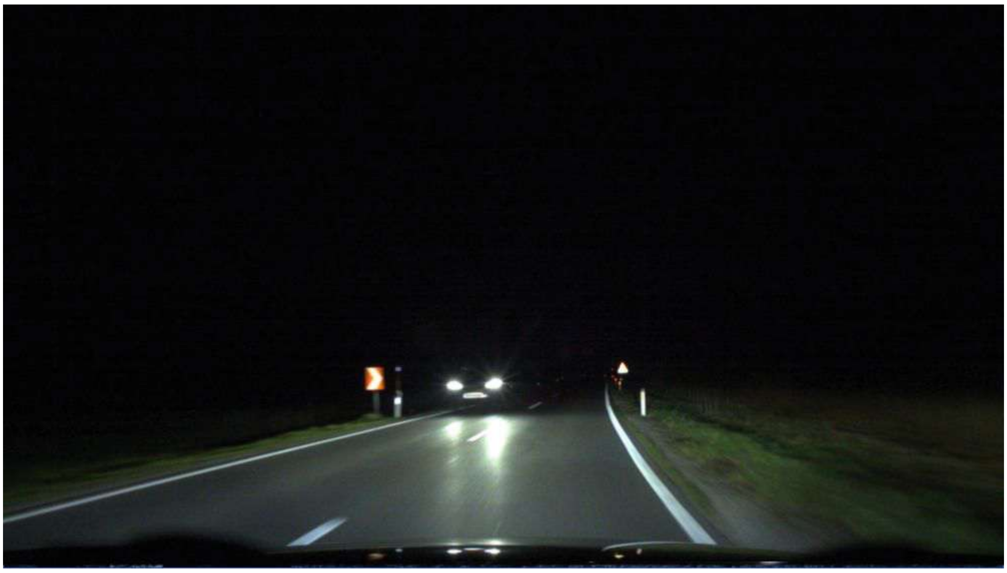


Figure 3.5: Example of the CarVisionLight (CVL) dataset, showing a car and two street signs at night in the rural streets of Lower Austria.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Implementation

This chapter discusses the design and implementation of the software developed for this master thesis. It begins with an overview of the design concept, followed by Section 4.1, which details the software itself and provides guidelines for using the network to ensure future reproducibility. The following Section 4.2 offers insights into the initial test run. Next, Section 4.3 describes the training and evaluation pipelines. Finally, the chapter concludes with Section 4.4, explaining how the datasets' metadata was adapted before training the network.

The implementation design followed a few key steps, which are illustrated in Figure 4.1, with the first step being a system setup. This setup included installing the TensorFlow v1.13 [ABC⁺16] framework on a Ubuntu 18.4 system, which enabled the retraining of YOLOv3. How to set up the system will be described in more detail in Section 4.1. As shown in Figure 4.1, after the system has been set up, the first step was to run an initial test run to gain a better understanding of the YOLOv3 network, which will be explained in Section 4.2. The centerpiece of the implementation consists of the following training and evaluation steps, shown in Figure 4.1. Both training and evaluation are composed as individual pipelines, which were developed for this master thesis. These pipelines will be explained in detail in Section 4.3.

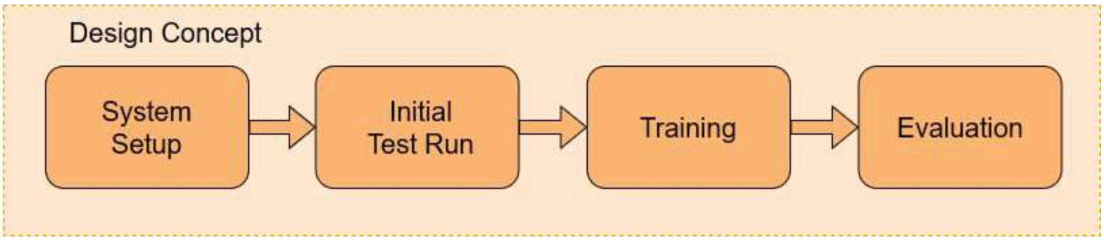


Figure 4.1: System pipeline. First, the system has to be set up, which is then followed by an initial test-run, to confirm it was set up correctly. After this, training and evaluation follows.

4.1 Software Description

This section describes the custom software developed specifically for this master thesis, as well as the external software utilized. To achieve this, the setup of the system will be discussed: how to configure the system to run the developed Convolutional Neural Network, as well as the necessary installation steps.

System Setup

The software was developed and run on Ubuntu 16.04, Xenial Xerus, using a NVIDIA GeForce GTX1090ti, for all graphic calculations. Initially, Python 2.5 was used, but it was later upgraded to Python 3.6, to ensure the re-usability of this project.

Additionally, CUDA 10.0, cuDNN 7.4.2, the OpenCV Python Package, TensorFlow 1.13, NVIDIA Collective Communications Library (NCCL) 2.3.5 and YOLOv3 [RF18] were used. Since YOLOv3 does not natively incorporate TensorFlow, an additional implementation, written by the user wizyoung on GitHub [git], was used.

To get the system running, several components have to be installed in the right order. This is vital, since installations like OpenCV and TensorFlow can compromise the executability of the code. Installation order should be: Ubuntu 16.04, OpenCV, CUDA, cuDNN, NCCL, TensorFlow and finally YOLOv3.

Software Packages and Installation

In the following section, the software packages used will be briefly described. This should give the reader sufficient insight into how the system was set up and which parts of the different software packages were used to run the final CNN.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CUDA

CUDA - previously an acronym for Compute Unified Device Architecture - is an API developed by the US-American company NVIDIA, which grants access to its graphics processing units (GPU) [cuda]. In contrast to other APIs like OpenGL, Vulcan or Direct3D, which focus on graphics programming, CUDA allows the GPU to be used strictly as a data processing resource. It provides additional computing capacities, which allow for parallel programming, significantly faster than a standard CPU. In the context of this project, it was used to run the CNN. While it was not directly programmed, other programs used the access to the GPU provided by CUDA. Therefore it has to be installed on the computer before running or training the CNN. In the scope of this master thesis, CUDA version 10.0 was used.

cuDNN

The CUDA Deep Neural Network library (cuDNN) is a library, provided by NVIDIA, which enables to use CUDA for Neural Networks [cudb]. It offers implementations for forward and backward convolution, pooling, as well as normalization. cuDNN provides an API that allows programs to directly access multi-threading on the GPU. In this master thesis, cuDNN was used to enable TensorFlow to perform batch-normalization and other necessary functions. Similar to CUDA, it was not directly programmed, but accessed through the CNN via TensorFlow.

YOLOv3

The object detection algorithm YOLOv3 [RF18] allows for real-time detection and identification of objects in videos or images. YOLOv3 is a machine learning algorithm that utilizes features learned by a CNN for object detection. Its functionality was previously described in more detail in Section 2.6.

TensorFlow

TensorFlow [ten] is an open-source framework developed by Google that enables effective machine learning. It allows users to develop and use Convolutional Neural Networks and is known for its scalability. While the framework itself is developed in C++, the Keras (high level) API can be accessed in different languages, such as Python. The basic operational element of TensorFlow is a graph, which is a directed diagram that provides an abstracted depiction of a mathematical problem. In this use case, the graph represents the mathematical operations used while calculating a CNN. It consists of edges and nodes. The edges represent data and the nodes are the mathematical operations themselves. The final graph represents the sequential process of all operations carried out by TensorFlow.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The following short code shows an example of such an operation in Python:

```
simple_math.py

# load the tensorflow library
import tensorflow as tf

# start a new tensorflow session
sess = tf.Session()

# x = 5, y = 4
x = tf.constant(5, dtype=tf.int8)
y = tf.constant(4, dtype=tf.int8)

# function f should multiply x by y
f = tf.multiply(x,y)

# OUTPUT
sess.run(f)
20
```

The above example depicts an implementation of TensorFlow in Python. After the TensorFlow library was imported, a new "session" can be started. This session defines the above mentioned setting of a graph, whereby different operations can be executed. In the case above, first the different values are loaded and the function then multiplies the two variables. After the session has been executed, the result - in this example 20 - is displayed. Although it is not essential to build TensorFlow from source, doing so can provide a deeper understanding of the system and is recommended if additional adjustments need to be made. To ensure that the system has been set up correctly, it is recommended to test the installation after TensorFlow has been installed by importing the module into Python and starting it, after YOLOv3 has been installed. This master thesis used a Python implementation of TensorFlow. Specifically, TensorFlow was used in conjunction with YOLOv3. However, since TensorFlow does not natively incorporate YOLOv3, it hinders the time-effective development of a CNN. Therefore, for this master thesis, an implementation written by the user wizyoung on GitHub [git] was used.

NCCL 2.3.5

NCCL stands for NVIDIA Collective Communications Library [ncc]. This library implements multi-GPU and multi-node communication for NVIDIA GPUs and is used to accelerate deep learning training methods. It is integrated with deep learning frameworks such as PyTorch or - in case of this master thesis - TensorFlow. NCCL can be downloaded as part of the NVIDIA HPC SDK.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.2 Initial Test Run

After the system has been set up, an initial test run was started to gain a more in-depth understanding of the existing CNN and its usefulness for the target application. For this, we provided YOLOv3, pre-trained on the COCO dataset, with new input in order to evaluate the network's ability to deal with previously unknown data.

Considering how the final version of YOLOv3 should be able to identify objects in images depicting street scenarios, the Cityscapes dataset [COR$^+$16] was chosen for the initial test tun. It was not included in the final testing, training, or evaluation dataset, but served as a initial test to see how the YOLOv3 network handles data specific for this use case. The Cityscapes dataset is a large-scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities during the daytime. It does contain semantic, instance-wise, and dense pixel annotation but no bounding box annotation [COR$^+$16]. It proved itself useful for an initial test run since it includes over 50 cities, different seasons, and weather conditions.

As shown in Figure 4.2, the Cityscapes dataset was processed through the YOLOv3 network, to gain a basic understanding of its analytical capabilities. Since the execution time and quality of the results were deemed sufficient, the test dataset— comprising a combination of the GTSRD, VIPER, BDD, and CVL datasets —was also sent through the network, following the same structure as depicted in Figure 4.2. The analysis and evaluation setup of the YOLOv3 network on the test dataset can be found in more detail in Section 5.1.

From these first evaluations, an initial training run was started using the final test dataset. It consisted of over 64.000 day and night images - more details on their distribution can be found in Section 4.4. This run was evaluated with qualitative and quantitative measures, which will be further discussed in Sections 5.2.1 and 5.2.2.

4.3 Training and Evaluation Pipeline

After evaluating the intial test run, the first training round was started, a visualization of which can be seen in Figure 4.3. As the first step of each training session, a new dataset was chosen and then processed through the CNN for retraining. The result of each run, which consisted of the retrained CNN, as well as the trained weights and the parameters, was stored, ensuring repeatability. Qualitative and quantitative evaluation was performed for each session, and the results were stored for later comparison. The following sections describe both training and evaluation pipelines - as depicted in Figure 4.3 - in detail.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

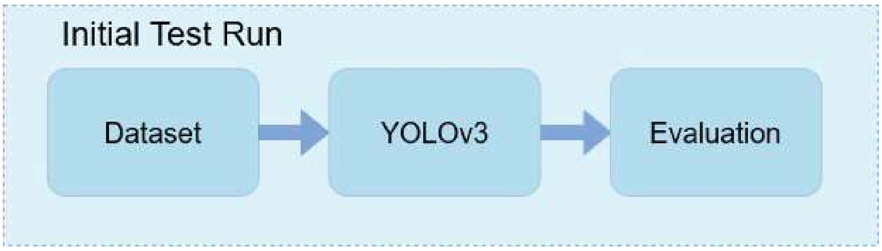


Figure 4.2: Initial test run of YOLOv3 using the Cityscapes dataset. First, the Cityscapes dataset is processed through the YOLOv3 CNN. This is then followed by an evaluation of the training results.

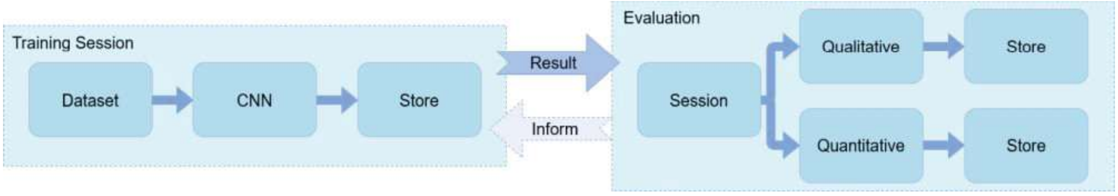


Figure 4.3: Pipeline for CNN development. The left box shows the Training Session, in which the dataset is loaded into the CNN. Then, the CNN is retrained and the result gets stored. This result is then sent to the Evaluation (the right side of the image), in which each session is evaluated with qualitative and quantitative methods. The results of each evaluation round are stored for further analysis.

4.3.1 Training Session

Each training session starts with the selection of the dataset by the user, followed by retraining the CNN, and is concluded by the storing of the results.

Step 1: Dataset Selection

First, the dataset for the current training session has to be selected by the person who decides to start a new training round. For the purpose of this use case, we used different combinations of four datasets: the German Traffic Sign Recognition Database (GTSRD), VIsual PERception benchmark (VIPER), Berkeley Deep Drive 100k (BDD), as well as recordings from the CarVisionLight (CVL) project. The different combinations and distributions of the dataset will be discussed in more detail in Section 4.4.4.

Step 2: Training the CNN

Once a training dataset is selected, it gets processed through the CNN retraining pipeline. Figure 4.4 shows the pipeline for retraining YOLOv3. It starts with data preparation - annotation of the training, test and validation data - followed by defining the classes used for object detection. Next, the values for the anchor box have to be calculated, as previously discussed in Section 2.5.4. Finally, the training sessions were starts.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

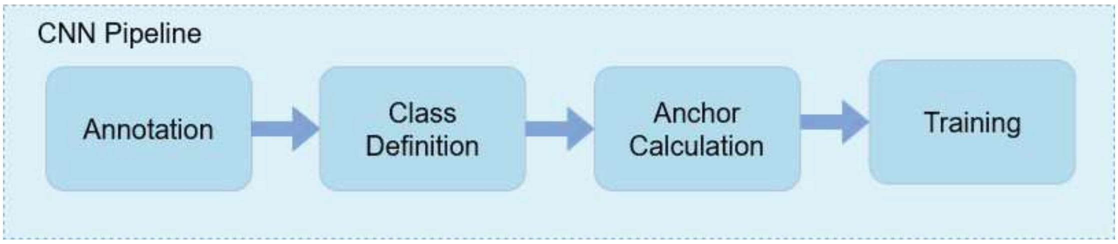


Figure 4.4: A closer look at the pipeline used for retraining the CNN. Here, the annotation of the dataset is followed by the definition of the used classes, anchor calculation and finally the training itself.

Annotation

In order to enable YOLOv3 to train with the provided image data, it first has to be annotated. This is done by creating a training-, validation- and test-file, which points the training program to where the images can be found on the hard drive and where the bounding boxes within each image are located. Therefore, three files named train.txt, val.txt, test.txt file were created beforehand and stored in the ./data/my_data/ directory.

The annotation has to follow a specific format. One line of text corresponds to one image:

```
index image_path image_width image_height box_1 box_2 ... box_n
```

One box consists of the following parameters:

```
class_number, x_min, y_min, x_max, y_max
```

This results in a file of this format (numbers are merely exemplary):

```
File: street_train.txt

0 xxx/xxx/img1.jpg 1280 720 8 126 133 157 211 1 791 383 901 420
1 xxx/xxx/img2.jpg 1280 720 4 433 369 459 383 2 105 247 224 454
```

For each dataset used for training, a script was written, which automatically annotates the data as required by the system. These scripts are named "dataset_annotation.py", with a corresponding name for each dataset (e.g.: "viper_annotation.py"). These scripts read the metadata-file provided with each dataset and parse all relevant information into the format required by YOLOv3.

Since the simulated data of the VIPER dataset can contain boxes of size zero (whereby an object is visible in one pixel of the image), these boxes have to be removed by the "annotation_remove_zero_boxes.py" script. This is done by calculating the dimensions of the bounding box and if they are under a required minimum - which is given as input by the user - they are not added to the file:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
File: annotation_remove_zero_boxes.py

for i in range(box_cnt):
  class_num = meta[i*5+2]
  x_min, y_min, x_max, y_max =
         meta[i*5+3], meta[i*5+4], meta[i*5+5], meta[i*5+6]
  width = int(x_max) - int(x_min)
  height = int(y_max) - int(y_min)

  if(width == user_input): continue
  if(height == user_input): continue

  for res in [class_num, x_min, y_min, x_max, y_max]:
    result.append(str(res))
```

It is possible to adapt the box size that should be removed so that boxes under a specific size are removed and therefore ignored by the training algorithm. In the code example above, this is shown by the variable "user_input". The default value for the use case of the VIPER dataset is 0, but this parameter can be changed when calling the script.

The parameter "index" is the last value which is added to the file. After all datasets have been annotated, the script "enum_images.py" iterates through each line of the final txt-file and adds an incremental value to the front of each row.

These scripts can be executed individually to gain more insight into the annotation system or to change the parameters or datasets which should be annotated. If the user wishes to execute the complete annotation pipeline, it is possible to call "cnn_pipeline.py", which executes the individual annotation scripts and zero box removal script.

Class Definition

The training algorithm requires a file containing all class-names, which has to be named "street-classes.names", with each line representing one class name. If a new class is added, it is not only required to add it in the "street-classes.names"-file, but in the annotations scripts as well.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The following file “street-classes.names” was used for the class definition:

```
File: street-classes.names

person
car
bus
bicycle
truck
motorbike
train
rider
traffic light
traffic sign
```

From the assessment of the first images and videos for the final application, tese classes were able to cover most objects in the recorded test videos - with the exclusion of animals, which can be a subject for future adaptations of the model.

Anchor Calculation

As described in Section 2.6.2, YOLOv3 utilizes anchor boxes as proposed by Richter et al. [RHK17], which allow the faster bounding box calculation of detected objects. Since the anchor boxes are highly dependent on the application and the position of the boxes in the training images, they have to be calculated beforehand. For YOLOv3’s training algorithm, this is done by using k-means clustering.

K-means clustering was first introduced by MacQueen et al. [Mac67]. It describes a process which sorts n data-points into k clusters. In this use case, the different bounding boxes of the input image are sorted into 9 clusters, also called anchor boxes. As discussed in Section 2.5.4, anchor boxes are a set of predefined bounding boxes, which are defined to capture the scale and aspect ratio of objects one wishes to detect. They are calculated based on the object sizes in the dataset. The corresponding anchors of each bounding box predict the offsets from the predetermined set of boxes and are noted as $(width, height)$. These calculated anchor boxes are stored with their initial size, and the one closest in size to the found object will be resized to create a matching bounding box - using the outputs from the neural network.

This is done by Redmon et al. within YOLOv3 [RF18] in the file “yolo_layer.c”:

```
File: darknet/src/yolo_layer.c, Lines 88 to 89

b.w = exp(x[index + 2*stride]) * biases[2*n] / w;
b.h = exp(x[index + 3*stride]) * biases[2*n+1] / h;
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Where $x[...]$ is the output of the neural network, $biases[...]$ are the pre-calculated anchors and $b.w$ and $b.h$ are the resulting width and height of the bounding box that will be shown on the result image. As can be seen, the network does not predict the final size of the object, but only adjusts the size of the nearest anchor to the size of the object. This provides an optimized performance, allowing for the bounding boxes to be calculated and displayed faster than with similar CNNs [yola].

These anchors have to be calculated beforehand by using the $get_k means.py$ script, which returns 9 anchors and an average IoU for all boxes. These values then have to be saved to a .txt file, placed at "./data/yolo_anchors.txt". It is possible to calculate these anchors using other methods, but for reproducibility of the results, it is recommended to use the provided methods.

Training

After the steps have been realised as described above, training can begin. This is done by running the "train.py" script, which uses TensorFlow functions to retrain the YOLOv3 network. The script requires several parameters to be set for each training round, which are described in detail in Table 4.1. This table shows all values which should be changed for a new training session, for the others, the default values are sufficient.

The leftmost column of Table 4.1 lists the parameter names, followed by a parameter description. This description shows how the parameters should be understood and how they influence the training session. This column is followed by "used value", which shows the values that were used for the training setup of this master thesis. Depending on the graphics card used, it is recommended to adapt the batch_size and num_threads size.

Step 3: Storing the Training Results

After the training session has been completed, the session has to be stored. To achieve this in a consistent manner, a folder called "checkpoint_finished_runs" has been prepared, containing one folder for each run (named "run0", "run1" etc.). If a new training session is being started, it is recommended to document the runs accordingly.

These folders contain the trained weights, the "*.names" file, the annotated training and validation files, the anchors and finally the "README.txt". This file contains the different parameter values set for this specific training session. Using this information, it is possible to either use the trained weights for object detection or start a new training session re-using these values.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub

Table 4.1: Parameter description for training sessions

Parameter	*Description*	*Used Value*
train_file	The path to the training txt file	street_train.txt
val_file	The path to the validation txt file	street_val.txt
anchor_path	The path to the anchor txt file	street_anchors.txt
class_name_path	The path to the class names	street_classes.names
batch_size	Size of the batch, or the number of images processed in one epoch	20
img_size	Images will be resized to "img_size" and fed to the network, size format: [width, height]	[416, 416]
letterbox_resize	Whether to use the letterbox resize, i.e., keep the original aspect ratio in the resized image.	True
total_epoches	For how many epochs the program should run. This defines how many times the algorithm works through the entire training dataset	10.000
train_evaluation_step	Evaluate on the training batch after some steps	1.000
val_evaluation_epoch	Evaluate on the whole validation dataset after some epochs. Set to None to evaluate every epoch	5.000
save_epoch	Save the model after the given number of epochs	100
num_threads	Number of threads for image processing used in tf.data pipeline.	20
optimizer_name	Chosen from [sgd, momentum, adam, rmsprop]	adam
learning_rate_init	Learning rate for the first few epochs, should be set higher than ater learning rate	1e-8
lr_type	Chosen from [fixed, exponential, cosine_decay, cosine_decay_restart, piecewise]	"exponential"
lr_decay_epoch	Epochs after which learning rate decays. Int or float. Used when chosen "exponential" and "cosine_decay_restart" lr_type	100
lr_decay_factor	The learning rate decay factor. used when chosen "exponential" lr_type	0.96

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.3.2 Evaluation

After the latest training run has been stored, the qualitative and quantitative evaluation follows, as shown in Figure 4.3. It is important to distinguish between those two evaluation types, since they both test for different demands on the trained network. For this, two different types of scripts exist, which will be presented in more detail in the following section.

Quantitative Evaluation

Quantitative evaluation, in case of this master thesis, consists of numeric analysis. This assessment does not involve human observation, but returns numeric values which grant insight into the networks' performance. These values consist of mean average precision (mAP), recall, precision, total loss, and loss confidence.

As described in Section 2.7, the quantitative evaluation metric used for this master thesis is Intersection over Union (IoU), with a threshold of 0.6. In other words, objects found with an IoU equal or greater than 0.6 result in a True Positive (TP), and everything else in a False Positive (FP).

Using the same IoU methods, recall and precision are calculated. Recall (or "rate of true positives") observes how many of the found positives are true positive predictions. It is calculated by dividing the number of detected objects by the number of ground truths:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{goundtruth}$$

Precision (or "positive prediction value") shows how many of the positive predictions made are accurate. It is calculated by dividing the number of detected objects by the number of predictions made, including the ones labelled as false:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{predictions}$$

Recall and precision are calculated in the script "eval.py", which was written by the user wizyoung on GitHub [git] and adapted to match the requirements for this use case. Therefore, the IoU thresholds were changed and all calculations were done for each class individually. In the original script, precision, recall and mean average precision were all calculated over all classes to show the result of the CNN as a whole. Once recall and precision are calculated, the mean average precision (mAP) is determined by dividing the true positives for all classes by the sum of all detected objects.

Loss is the quantitative measure of difference between the predicted output and the actual output. It results in a measure of mistakes made by the network in detecting objects.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

While loss is calculated for each learning step to provide a measurement on how the network is currently performing and feedback to possible over-fitting, total loss describes the loss over all classes and iterations. The lower the loss, the better a model has been trained. The loss of a CNN is calculated on the validation set. Unlike accuracy, loss is not a percentage, but a summation of the errors made for each evaluation. The main objective in a learning model is to reduce the loss function's value with respect to the model's parameters by adapting the weight vector values through different optimization models. Loss value indicates how well or poorly a certain model behaves after each iteration of optimization. Ideally, one would expect the reduction of loss after each, or several iterations.

The script "eval.py" returns the total loss value, as well as the average loss of confidence - which indicates how confident the network is the detected object is correct. All results were calculated using the validation and test datasets, which used a combination of 4 datasets (GTSRD, CVL, VIPER, and BDD). All possible set permutations were used to create 15 distinctive training rounds. After each round, the evaluation was performed and the results stored for further analysis. These different permutations and therefore resulting training rounds can be seen in a more detailed breakdown in Figure 4.5.

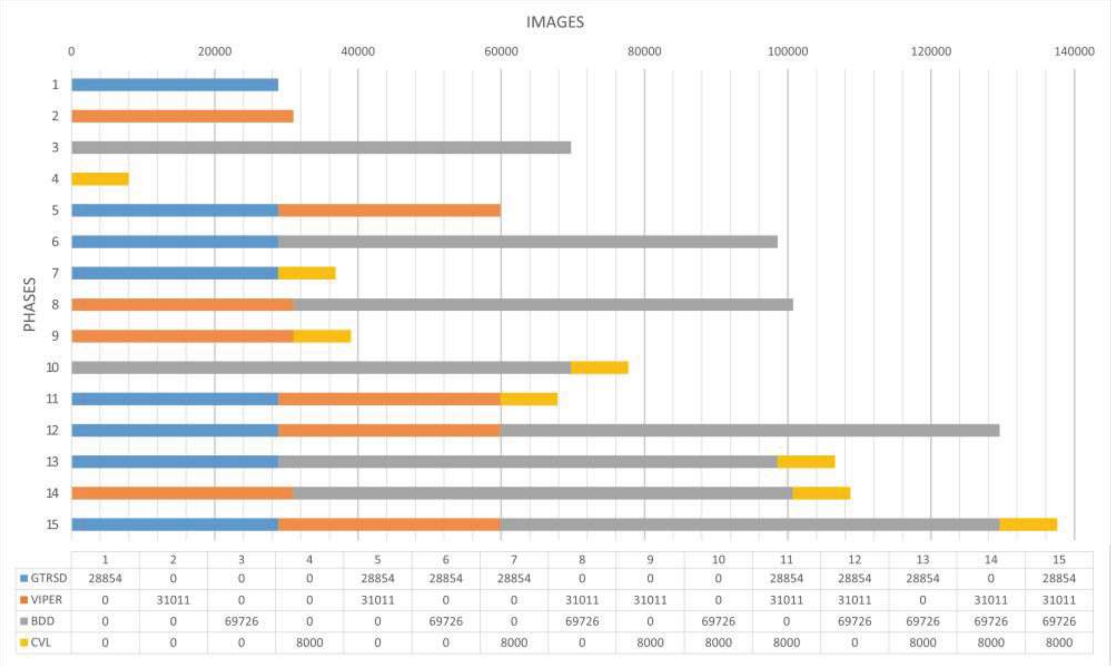
Qualitative Evaluation

In case of this master thesis, qualitative evaluation distinguishes itself by evaluating the results based on visual criteria. This involves sending an image through the trained network and evaluating it based on subjective standards. While this qualitative evaluation involves human observation (along criteria such as: how many objects were detected, were they labelled correctly, etc.), it also includes the knowledge the evaluator gained from previous runs. This allows for a more in-depth comparison to the effectiveness of the training compared to the quantitative evaluation, which only observes the current run. Since qualitative evaluation allows for comparisons between runs, it is possible to see the impact of changing a training set on the final result immediately.

To start qualitative evaluation, either the script "eval_single_image.py" or "eval_ multiple_ images.py" should be executed, depending on the number of images the user wants to evaluate. As a parameter, either the path to the image or the folder containing the evaluation images has to be given, for example: "python eval_single_image.py test_img.png". This then calls the evaluation function, which starts a TensorFlow session, sends the image through the network and returns the bounding box coordinates, scores and labels. These are then displayed in the image and stored in the folder "result".

It is also possible to evaluate a video, by starting the script called "eval_video.py". First, the video gets split into different single images, which then get analyzed one by one. The result is another video, where the images are put back into the stream. In the upper left corner, the processing time for each frame is displayed, as can be seen in Figure 4.6. Please note that since the input video is split into single images, the results may vary from frame to frame. These qualitative results will be discussed in Section 5.2.2.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
■ GTRSD	28854	0	0	0	28854	28854	28854	0	0	0	28854	28854	28854	0	28854
■ VIPER	0	31011	0	0	31011	0	0	31011	31011	0	31011	31011	0	31011	31011
■ BDD	0	0	69726	0	0	69726	0	69726	0	69726	0	69726	69726	69726	69726
■ CVL	0	0	0	8000	0	0	8000	0	8000	8000	8000	0	8000	8000	8000

Figure 4.5: This graph shows the different dataset combinations for each of the 15 training rounds, with each dataset in a different color. The color for each dataset, as well as the exact number of images in each set (not split by day/night), can be seen in the legend below the graph.

4.4 Data Handling

Four datasets were used in this master thesis: The German Traffic Sign Recognition Database (GTSRD), VIsual PERception benchmark (VIPER), Berkeley Deep Drive 100k (BDD), as well as recordings taken within the CarVisionLight project (CVL). While the datasets themselves have been described in Section 3.2, the following sections will take a look at the file structure, explain annotation files and present the annotation scripts created for each dataset.

4.4.1 GTSRD

The training set of the German Traffic Sign Recognition Database contains one directory per class, for over 40 classes. Each directory contains one CSV-file with annotations, named “GT-<ClassID>.csv”. One image can contain several traffic signs, which are annotated with a border of 10% (around 5 pixels), to allow for edge-based object detection processes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

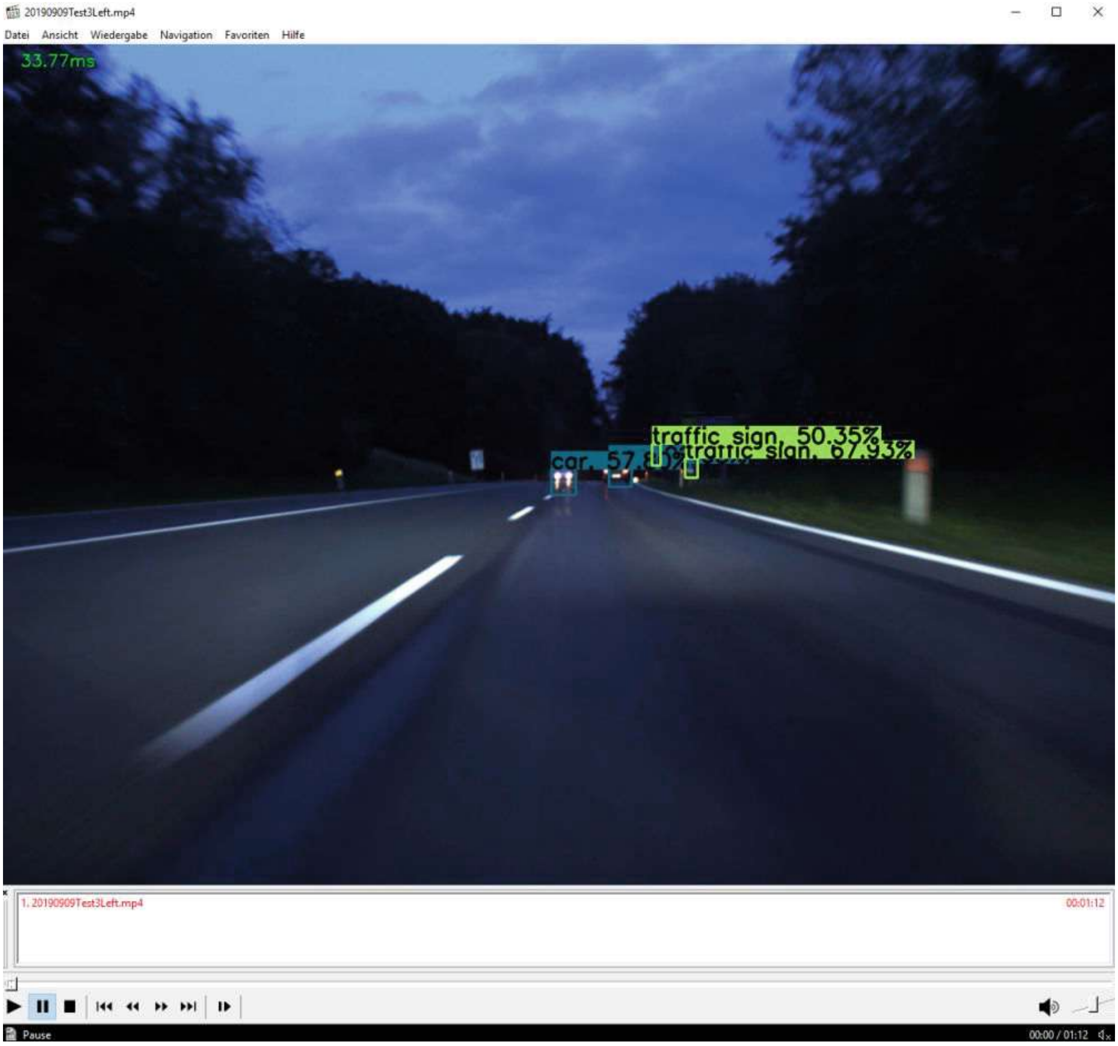


Figure 4.6: Video processing example image. This image shows the result of a qualitative evaluation performed on a video. The upper left corner displays the processing time. The boxes show the detected objects, with a percentage showing the confidence.

For each image in the GTSRD dataset, annotations are provided in CSV-files, containing the following information:

```
Filename,  Width, Height, Roi.X1, Roi.Y1, Roi.X2, Roi.Y2, ClassId
00001.ppm, 1360,  800,     983,     388,     1024,    432,    40
```

The image “00001.ppm”, corresponding to the parameters above, can be found in Figure 4.7.

Within each file, “Width” and “Height” describe the scale of the image, not the objects found within. “Roi” - the Region of Interest pixel locations - represents the corner

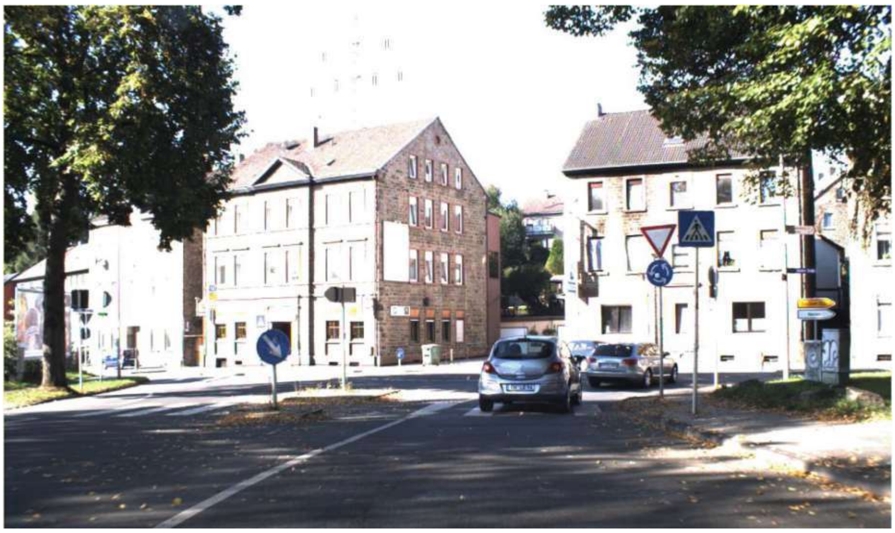


Figure 4.7: Example of the GTSRD dataset, with one car and several street signs in a city environment during day.

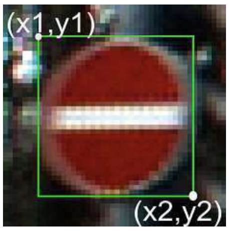


Figure 4.8: Example annotation of the GTSRD dataset, showing a street sign in a green bounding box, with the corners denoted as $(x1, y1)$ and $(x2, y2)$.

coordinates of the bounding box, as can be seen in Figure 4.8. In the annotated file, the value Roi.X1 corresponds to x_min, Roi.Y1 to y_min, Roi.X2 to x_max, and Roi.Y2 to y_max. Finally, "ClassId" describes the class the object belongs to. Since this dataset is only concerned with street signs, each class corresponds to one of the 40 street signs selected for this dataset.

The script "gtsrd_annotation.py", which automatically restructures the provided data into the format required by YOLOv3, ignores the field "ClassId" but assigns them all the same "Street Sign" class. In a first run - discussed in more detail in Section 5.3.1 - these classes were kept individually for training, but this resulted in false positives, an example of which is shown in Figure 4.9.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

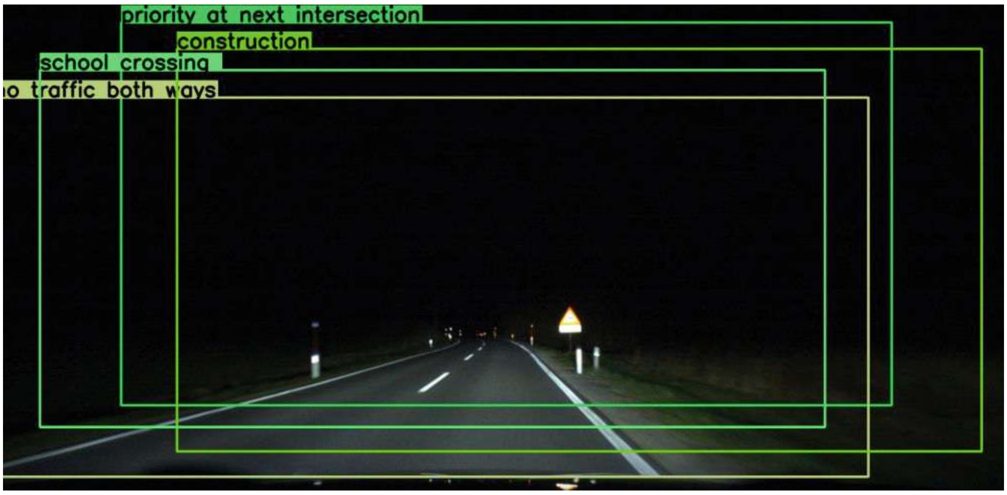


Figure 4.9: Example result of training the network with individual classes for traffic signs, tested on the CVL test dataset. Several false positives were found, with a multitude of street signs being detected.

4.4.2 VIPER

The VIsual PERception benchmark (VIPER) dataset is split into training, validation and test sets. Each subset contains images which were recorded in one of five different environmental conditions (day, sunset, rain, night, snow). For this master thesis, only night scenes were used. Each image was recorded at a resolution of 1920x1080. While the dataset provides camera poses, semantic class segmentation labels and semantic instance segmentation labels, for this use case, only 2D bounding boxe information was used.

The bounding box annotations are stored in a CSV file, where each row of the file corresponds to a single object in the image. The format can be found below, the corresponding example image can be seen in Figure 4.10:

```
instanceID, classID, x_min, y_min, x_max, y_max,
2,          24,      1009,   450,   1053,   478
```

For the 2D bounding box coordinates, no further transformations are needed. InstanceID was not taken into account, since it encodes the recorded instance, as described by Kim et al. [KWLK20], which was not of further interest for the CNN used in this master thesis.

The VIPER dataset contains 31 classes, from which only ten were chosen, since they corresponded to the classes required for detection by the overarching CVL project [cvl]. The original class number is noted in parenthesis next to the class: traffic light (#13), traffic sign (#14), person (#20), bicycle (#22), motorcycle (#23), car (#24), van - labelled as car (#25), bus (#26), truck (#27), train (#29).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

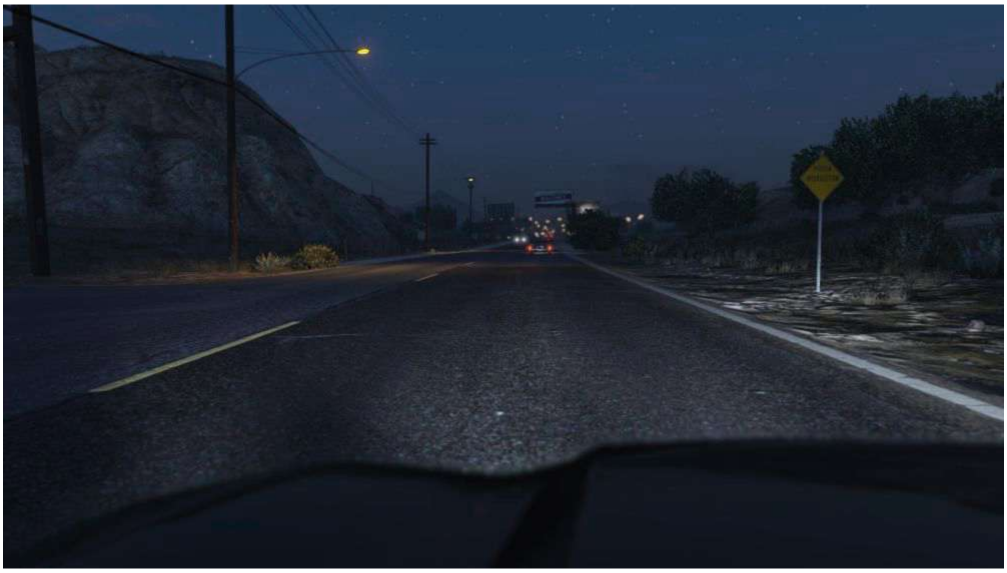


Figure 4.10: Example from the VIPER dataset, showing a night scene with cars and a street sign.

The script “viper_annotation.py” requires the names of the folders which have to be annotated as an additional parameter. This allows for the selection of different folders with distinctive properties, like solely selecting night-scenes. This selection can then be added to the script by stating the folder number next to the starting command, for example: “python viper_annotation.py 008 009 010 011”. The script then takes the corresponding CSV-files, adds the image path to the file - relative from the location of the script, which has to be placed in the same folder as the data - selects the corresponding class and adds the box coordinates as provided.

As mentioned before, after the images have been annotated, some boxes with size zero can occur. Since the underlying data consists of simulated images, the system can label boxes for objects where there should theoretically be an object, but in reality they are too small to be detected. Therefore, the script “remove_zero_boxes.py” was written, which should be executed immediately after “viper_annotation.py”.

4.4.3 BDD

For the Berkeley Deep Drive (BDD) dataset, two JSON-files for training and validation are provided alongside with the images. These contain bounding box annotations of 10 classes: car, traffic sign, traffic light, person, truck, bus, bicycle, rider, motorbike, train.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

As an example, the JSON format for the image “b1c66a42-6f7d68ca.jpg” for one traffic sign is as follows:

```
name: b1c66a42-6f7d68ca.jpg,
attributes: {
   weather: overcast,
      scene: city street,
      timeofday: daytime
   },
timestamp: 10000,
labels: [
   {
      category: traffic sign,
      attributes: {
         occluded: false,
         truncated: false,
        trafficLightColor: none
      },
      manualShape: true,
      manualAttributes: true,
      box2d: {
         x1: 1000.698742,
         y1: 281.992415,
         x2: 1040.626872,
         y2: 326.91156
      },
      id: 0
   }
]
```

The corresponding image for this JSON file example can be seen in Figure 4.11. This JSON file first shows the name of the image, the weather, scene it depicts, time of day, followed by labels for each found object. These labels concern the occlusion and truncation of the object, as well as the colour of the traffic light. These descriptions are then followed by the coordinates of the 2D box values x1, y1, x2, y2, which encode the corner pixels of the bounding box. These values were automatically generated by the creators of the dataset [YXC$^+$18] and contain several decimal numbers, which could not be used any further and were therefore rounded up or down to the next full value, depending whether it is a upper or lower corner.

The script “bdd_annotation.py” reads the JSON files for training and validation and adds the data path, followed by the corresponding class and coordinates for the bounding box (x_1, y_1) and (x_2, y_2).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Figure 4.11: Image from the BerkeleyDeepDrive training dataset, depicting a street in a rural area.

4.4.4 CVL Data

The CarVisionLight (CVL) dataset was recorded at dusk and during night-time in the countryside of rural Austria. The recordings contain over 50 videos, of which over 9k images were annotated. While the recordings were originally videos, single images were extracted, in order to provide the CNN with a training, validation and test dataset. The videos were collected by the project partner ZKW, and annotated by TU Wien with an annotation tool developed by Groh et al. [GSG20].

This annotation tool presented the annotation data in JSON format, an example of which can be found below, with the corresponding Figure 4.12:

```
name: dataset/180409_Test37.avi/0001.png,
url: dataset/180409_Test37.avi/0001.png,
videoName: CVL1_000000,
attributes: null,
timestamp: 10000,
index: 0,
labels: [
  {
    id: 0,
    category: traffic sign,
    attributes: {
      Occluded: false,
      Traffic Light Color: [
        0,
        NA
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
      ],
      Truncated: false
    },
    manualShape: true,
    box2d: {
      x1: 762.1229462321161,
      x2: 784.7505793281514,
      y1: 620.8277516585634,
      y2: 640.1168815109214
    },
    poly2d: null,
    box3d: null
  }
]
```

First, the name of the file to which the annotation belongs to is listed, followed by more attributes belonging to the image like path (called url), time-stamp, and the name of the original video. The next information relevant for the network to be trained concerns the attribute “category”, which describes the bounding box annotations and can contain three possible classes: person, car and traffic sign - the latter corresponds to the class “street sign” in the trained network.

The script “cvl_annotation.py” first distinguishes between the different video names (in the example above “180409_Test37.avi”), since every recording session can have a slightly different resolution - they range from 1936 x 882 to 1936 x 938. Then, the category “labels” is extracted from the JSON file and different values, including category and box2d, are added in the format required for annotation.

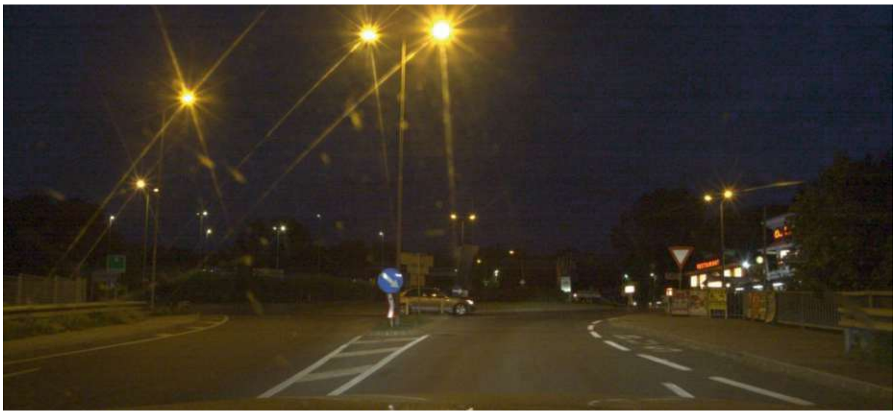
4.4.5 Dataset Distribution

This section discusses the dataset distribution in more detail. The examination on how the different datasets contribute to the final training, validation and test sets allows for a deeper understanding on how the network was trained and ensures the reproducibility of the results.

The distribution of day and night images in the previously discussed datasets (GTSRD, VIPER, BDD, CVL) and their contributions to the final dataset (Final) can be seen in Figure 4.13. Subfigure (a) shows the training dataset, with over 130k images in the final set. The largest contributor of data is BDD, with a greater contribution of day images. The other datasets consist exclusively of day (GTSRD) or night (VIPER, CVL) images. The validation dataset in Subfigure (b) contains over 26k images in the final set. These images were exclusively selected for validation to assess the models performance and distinct from the training dataset. Finally, the test dataset, shown in Subfigure (c), contains over 64k images, neither used for training nor validation. Images from this test set were used for qualitative evaluation as well.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

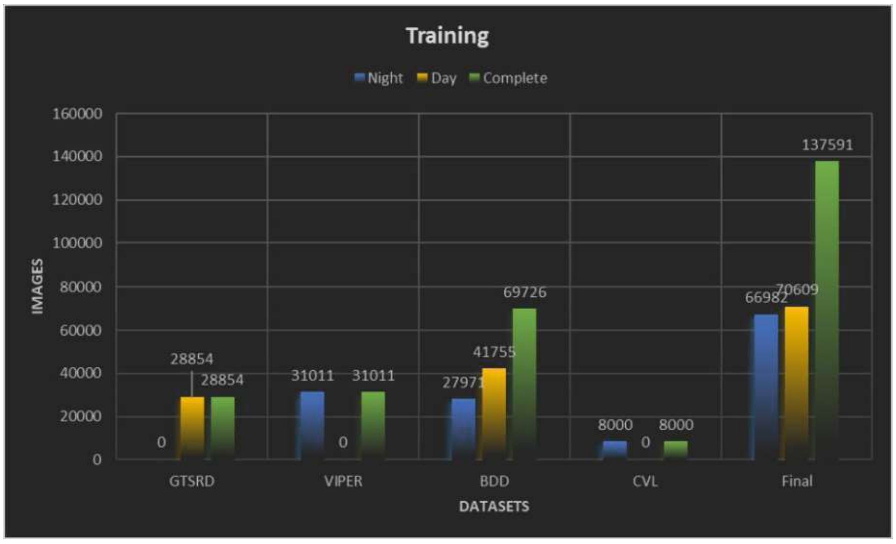
(a) This image shows a street at night on country road in rural Lower Austria. A car and street sign can be seen.



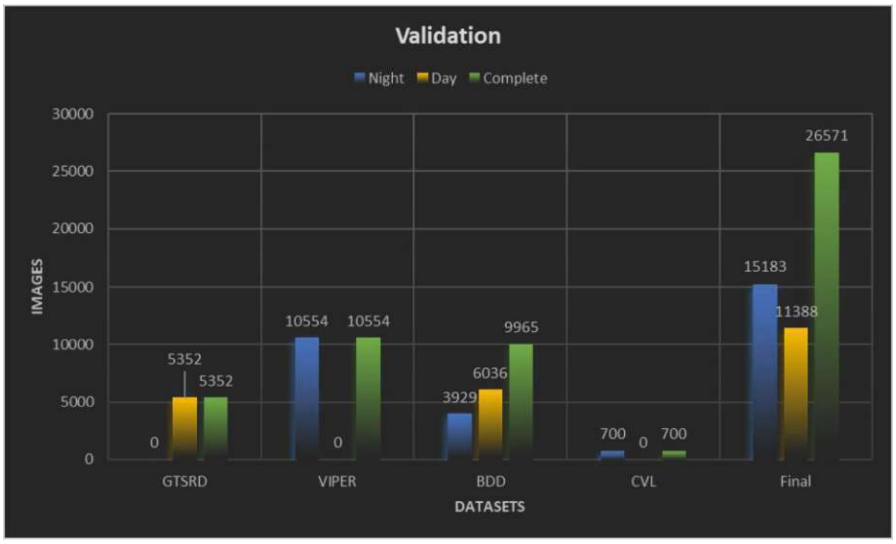
(b) Rural village in Lower Austria, with street lamps being lit, which significantly brightens the scene.

Figure 4.12: Here are two sample images from the CVL dataset. They were captured during the same acquisition drive for the training dataset.

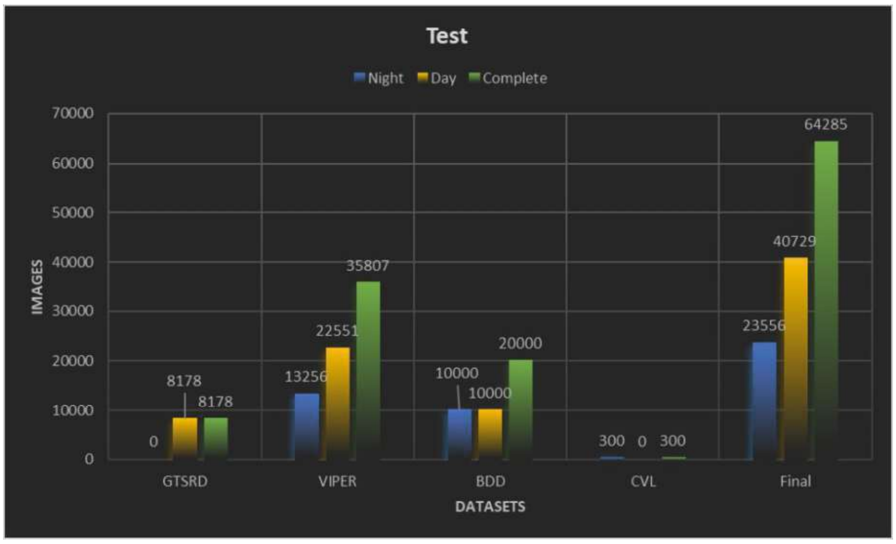
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



(a) Training dataset distribution



(b) Validation dataset distribution



(c) Test dataset distribution

Figure 4.13: Distribution of day and night images in the original datasets (GTSRD, VIPER, BDD, CVL) and contribution to the final dataset (Final).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Evaluation and Results

This chapter presents the results that were achieved by retraining the existing Convolutional Neural Network (CNN) YOLOv3, by using different combinations of publicly availably and self-recorded datasets, to enable object detection in night-scenes. The recognition rates of different object classes, relevant for street-specific contexts, were recorded and evaluated for each training cycle. We provide an in-depth view of the evaluation process and compare the results for each training step. Section 5.1 presents the evaluation setup, which distinguishes between the initial test-run and the fifteen consecutive training sessions. Section 5.2 discusses the results of the evaluation process, reviewing both qualitative and quantitative evaluation results. Finally, Section 5.3 provides the reader with a critical reflection on the work done within this master thesis, as well as present an outlook on possible future works.

5.1 Evaluation Setup

Training and evaluation of the YOLOv3 network consisted of several phases, named “phase 0-15”. The evaluation process started with “phase 0” - named so for being the first evaluation run which would provide a baseline for the following training and evaluation cycles. “Phase 0” evaluated the object detection rate of the unaltered YOLOv3 network, which means the version of YOLOv3 which was pre-trained on the Microsoft COCO dataset, as previously described in Section 3.1. In order to create baseline values against which the results of the “phases 1-15” could be compared to, the unaltered CNN was evaluated using the same evaluation methods and test-dataset used in the following phases. As previously discussed in Section 4.4.5, the test dataset consisted of over 64k images, with 24k depicting night- and 40k showing day-time scenes. As can be seen in Figure 5.1, the test dataset was processed through the YOLOv3 network. Qualitative and quantitative evaluations were done in order to create a baseline for future comparison.

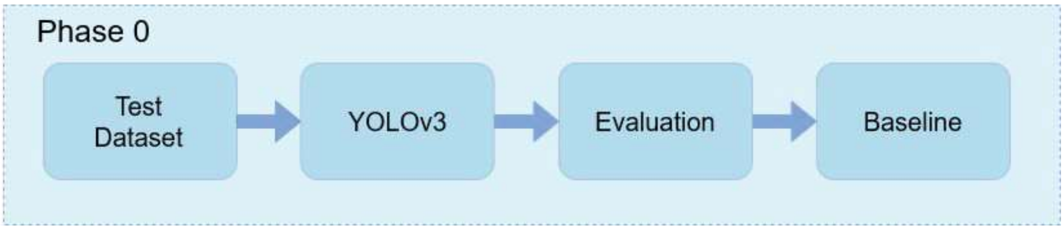


Figure 5.1: Pipeline of “phase 0”, the first evaluation phase. First, the test dataset is loaded into the YOLOv3 network. The results are then evaluated and create a baseline for future comparison.

The following phases - “phase 1-15” - follow a more elaborate training and evaluation structure, which can be seen in Figure 5.2. Each phase follows the same pipeline, split into training and evaluation sessions. During each training session, one of fifteen possible dataset combinations is used to retrain YOLOv3. The resulting network and the parameters used are then stored for further evaluation. This newly trained network is assessed using both qualitative and quantitative methods, and the results are stored as well.

The names “phase 1-15” correspond to the different dataset combinations which were used for the training datasets. They are a composite of four distinct datasets: GTSRD, CVL, VIPER, and BDD. A more comprehensive breakdown of these dataset combinations can be seen in Figure 5.3. In total, fifteen distinct training phases were created, each using a unique combination of these datasets. The initial four rounds used each dataset individually, followed by subsequent rounds that incorporated various combinations of the four datasets, culminating in fifteen distinct training datasets. The results of these different training phases will be discussed in Section 5.2.

An early objective of the project was to evaluate a pre-existing Convolutional Neural Network (CNN) without any additional training, thereby establishing a baseline for further work. The intention was to delve deeper into the intricacies of the existing CNN and assess its applicability to the intended use case, therefore naming this first step “phase 0”. As detailed in Section 4.2, the original YOLOv3 network, trained on the Microsoft COCO dataset, was used for this phase. It was then evaluated using the test dataset, with both qualitative and quantitative methods. Figure 5.4 shows two exemplary result images. Subfigure (a) shows an example, where the untrained YOLOv3 network produces satisfactory results for these daytime scenes, whereas some problems can be seen Subfigure (b). While the traffic light in Subfigure (b) on the left side of the image was detected, the traffic light on the right side, as well as all traffic signs were missed.

In addition to the day-time scenes shown in Figure 5.4, a CNN used for the final application must be able to operate during the night as well, with only street lamps and the cars headlights illuminating the scene. Therefore, the test-dataset consisted of both day and night images - two exemplary results can be seen in Figure 5.5. In this example, the “phase 0” CNN was unable to correctly detect objects under challenging lighting conditions. Both the car and the street signs were overlooked by the object detection

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

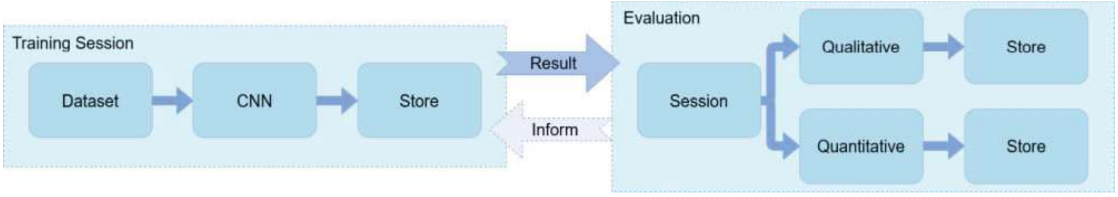
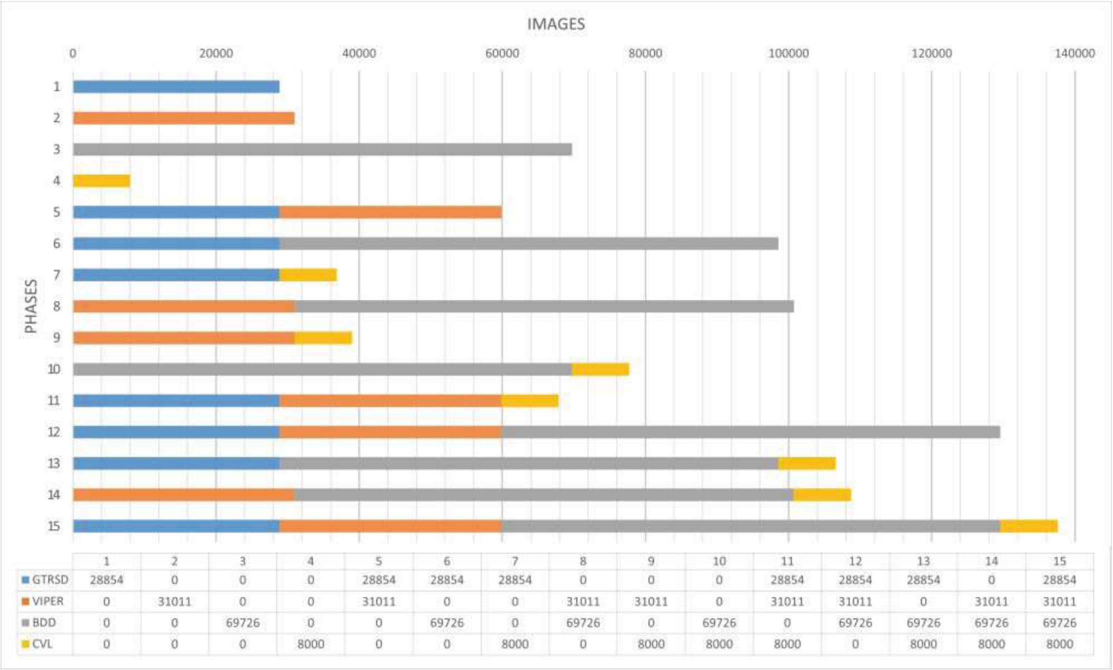


Figure 5.2: Pipeline for CNN development. The left box shows the Training Session, in which the dataset is loaded into the CNN. Then, the CNN is retrained and the result is stored. This result is then sent to the Evaluation (the right side of the image), in which each session is evaluated with qualitative and quantitative methods. The results of each evaluation round are stored for further analysis.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
🟦 GTRSD	28854	0	0	0	28854	28854	28854	0	0	0	28854	28854	28854	0	28854
🟧 VIPER	0	31011	0	0	31011	0	0	31011	31011	0	31011	31011	0	31011	31011
⬜ BDD	0	0	69726	0	0	69726	0	69726	0	69726	0	69726	69726	69726	69726
🟨 CVL	0	0	0	8000	0	0	8000	0	8000	8000	8000	0	8000	8000	8000

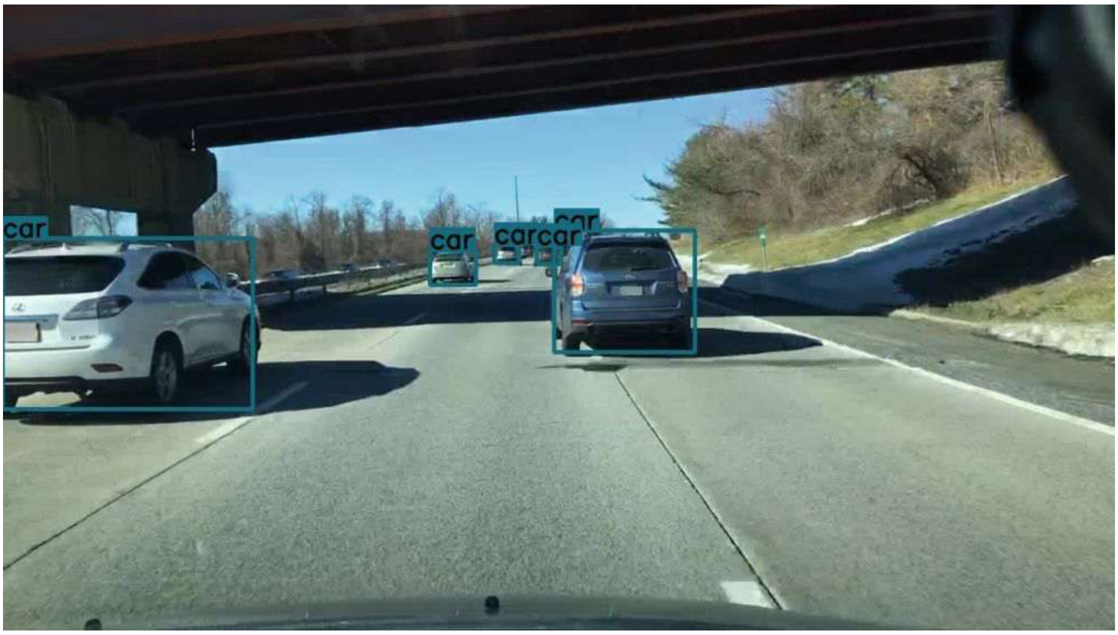
Figure 5.3: This graph shows the different dataset combinations for each of the 15 training rounds, with each dataset in a different color. The color for each dataset, as well as the number of images in each set (not split by day/night), can be seen in the legend below the graph.

algorithm in Subfigure (a), a phenomenon attributable to the characteristics of the training dataset. For example, the absence of the class “street sign” from the dataset may explain its inability to detect it. Subfigure (b) shows an example of false detection, with the back light of the front car being erroneously labelled as a traffic light.

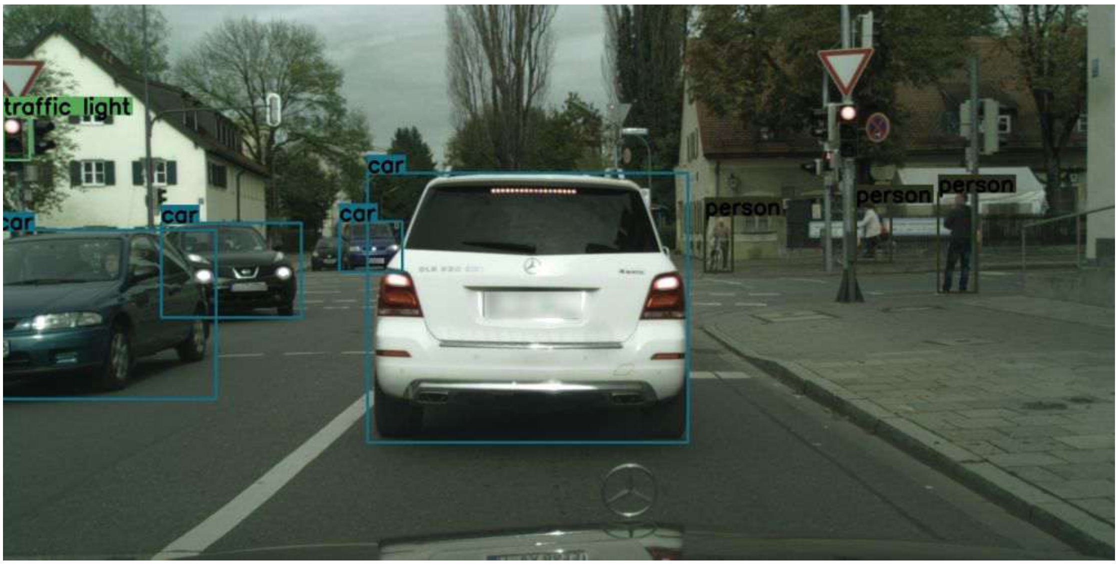
“Phase 0” showed that while the pre-trained YOLOv3 network performed well on day-time and bright city images, during night-time scenes, it struggled with object detection and for example missed the street sign class. The results gained from this phase’s quantitative analysis became the baseline, against which the results of the other phases were compared.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

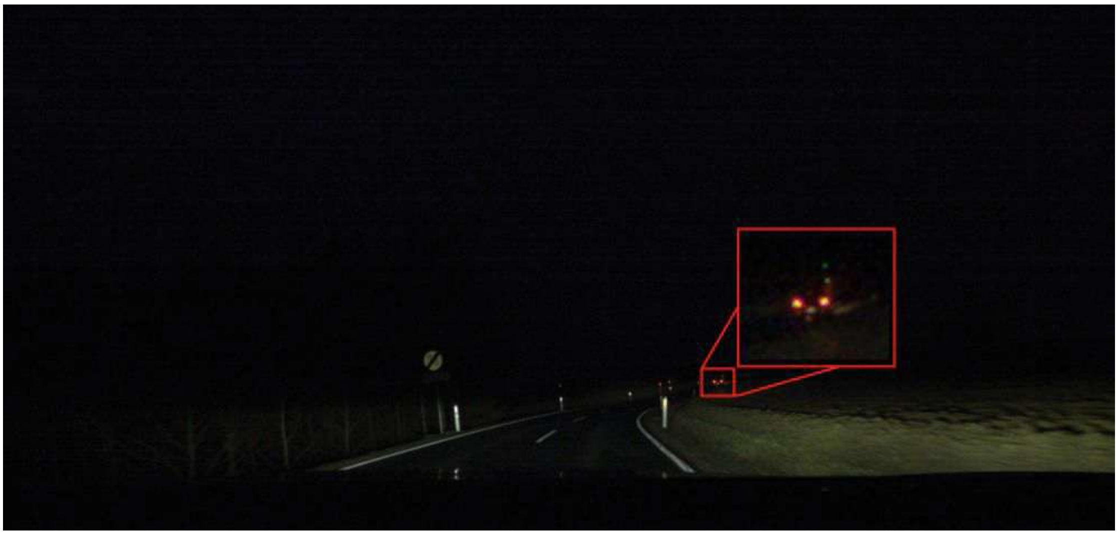


(a) Image of cars driving on a highway in America.

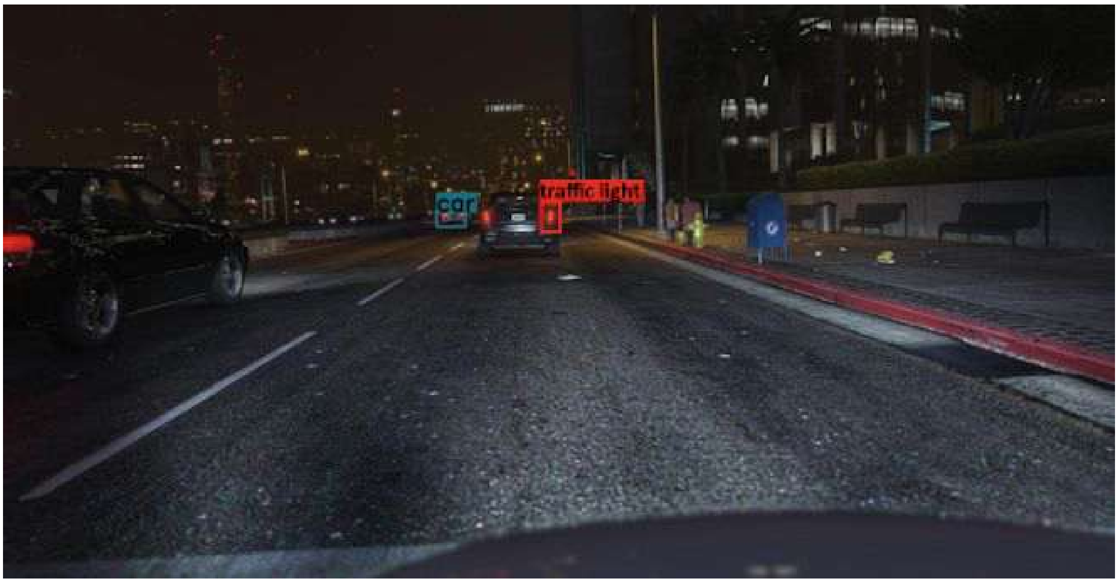


(b) Image of a crossing in a town in Germany, showing cars, traffic lights and traffic signs.

Figure 5.4: Results from applying the pretrained YOLOv3 object detection network on two Cityscapes example images.



(a) Image from the self-recorded dataset (CVL). The car in the distance is highlighted by a red box to improve visibility for the reader.



(b) Image from the VIPER dataset, showing a street during night with several cars.

Figure 5.5: Results from applying the pretrained YOLOv3 object detection network on two night-time example images.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5.2 Results and Discussion

This section delves into the results gained after retraining the YOLOv3 network using the different dataset combinations. As explained in Section 5.1 and shown in Figure 5.3, “phases 1-15” each correspond to a unique dataset combination from the four datasets: GTSRD, CVL, VIPER and BDD. Each phase was subjected to the training and evaluation pipeline, using both quantitative and qualitative methods. While the theory behind these methods has been previously discussed in Section 4.3.2, this section will present and discuss the results. In Section 5.2.1, the quantitative results will be discussed, by showing the improvement that was achieved over all fifteen training rounds. In Section 5.2.2, the qualitative results will be shown, by providing the reader with an insight into the different cases.

5.2.1 Quantitative Evaluation

As discussed previously in Section 4.3.2, the metric used for the assessment of the object detection accuracy in this master thesis is Intersection over Union (IoU), with a threshold value set at 0.6. Any objects identified with an IoU of 0.6 or higher were classified as True Positive (TP) detections, while all other detections were categorized as False Positives (FP). From these results, the mean average precision (mAP) was computed by dividing the number of true positives by the total number of detected objects.

Beginning with “phase 0”, the mAP was calculated for each class individually, in order to gain baseline values. This allows the comparison of the results after each phase. The mAP values were stored for each run in a txt-file, which showed the IoU and mAP for each class. The results were then evaluated manually by us, to see and discuss the various changes. As a selected example, Table 5.1 compares the results of “phase 0” with those obtained by using a combination of all datasets in “phase 15”.

It can be seen that in the first column titled “mAP, phase0”, six classes (Person, Car, Bicycle, Rider, Motorbike and Traffic Light) had a mean Average Precision (mAP) over 50. However, the classes labeled as “Bus”, “Truck”, “Train”, and “Traffic sign” deviated from this trend. Their under-performance in “phase 0” may be attributed to their sparse representation in the COCO dataset, with traffic signs being entirely absent.

The second column of Table 5.1 titled “mAP, phase 15” shows an improvement of the detection rate for individual classes at around 10%, with the two odd cases out being the classes “Rider” and “Traffic Sign”. For the class “Traffic Sign”, the recognition rate was improved by 81%, while “Rider” exhibits a decrease of 10%. We attribute this improvement to the nature of the training datasets used. They contained over 66k images recorded at night-time, which contain fewer images showing riders on motorbikes or bicycles. Stationary motorbikes and bicycles, on the other hand, can still be found during night-time, and were successfully detected and labelled.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Table 5.1: Classification results (mean average precision in %) obtained from “phase 0” and “phase 15”. The results are split for each class, with the final line showing the results for all classes.

Class	*mAP, phase0*	*mAP, phase15*
Person	61	70
Car	68	83
Bus	34	44
Bicycle	59	62
Truck	43	59
Train	31	43
Rider	61	51
Motorbike	52	63
Traffic light	59	79
Traffic sign	0	81
All classes	46.8	63.5

When comparing the improvement of mAP for all classes, an overall enhancement of 16.7% was achieved, starting from a detection rate of 46.8% for all classes in “phase 0”, to a detection rate of 63.5% in “phase 15”. While the evaluations shown in Table 5.1 compare the results of “phase 0” with the final “phase 15”, the individual results for each phase were noted after each evaluation session. These results can be seen in Table 5.2, and are visualized in Figure 5.6. Table 5.2 shows the mAP in % for each individual phase, split for each class. The row “All classes” shows the combined recognition rate per phase over all classes. The bottom row “Datasets” shows which datasets were used for each training phase. Figure 5.6 visualizes the aforementioned table, one line in the graph representing one class.

While a general improvement can be observed for the detection rates of all classes, some cases - for example the class “Train” - fluctuate between different detection rates with new dataset combinations. The decrease in the detection rate for the case “Rider” can be attributed to a lack of training data, which is explicitely stated in the second case description: “Case 2: Class “Rider” ”. Other specific use cases are highlighted in the following case descriptions.

Case 1: Variations in Classes “Train” and “Person”

As visualized in Figure 5.7, the detection rates for the class “Train” vary for each round between 29% and 43%. We attribute this to the nature of the datasets used for each training round. If we take a look at “phase 1”, the detection rate of 0% can be explained by the composition of the GTSRD dataset: this dataset consists of images of street signs, meaning the only detectable class is “Traffic Sign”. This can be observed for

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 5.2: Classification results (mean average precision, mAP) shown for each class and each of the sixteen phases.

Class/Phase	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Person	61	0	46	66	43	46	66	43	64	51	65	49	68	67	70	70
Car	68	0	74	79	81	74	79	81	79	80	81	79	79	81	76	83
Motorbike	52	0	53	61	55	53	61	55	59	56	59	55	60	62	60	63
Bicycle	59	0	47	58	56	47	58	56	53	55	57	56	58	61	61	62
Truck	43	0	49	61	59	49	63	61	58	57	61	47	58	57	57	59
Train	31	0	37	28	27	37	28	27	39	34	29	36	41	29	42	43
Rider	61	0	0	53	37	0	53	37	46	36	49	37	52	51	51	51
Bus	34	0	38	40	39	38	40	39	40	39	40	39	41	40	39	44
Traffic Light	59	0	68	72	75	68	72	75	69	71	73	74	76	74	75	79
Traffic Sign	0	67	59	62	71	73	72	74	62	65	67	75	78	79	71	81
All classes	46.8	6.7	47.1	58	54.3	48.5	59.2	54.8	56.9	54.4	58.1	54.7	61.1	60.1	60.2	63.5
Datasets	COCO	GTSRD	VIPER	BDD	CVL	GTSRD	GTSRD	GTSRD	VIPER	VIPER	BDD	GTSRD	GTSRD	GTSRD	VIPER	GTSRD
						VIPER	BDD	CVL	BDD	CVL	CVL	VIPER	VIPER	BDD	BDD	VIPER
												CVL	BDD	CVL	CVL	BDD
																CVL

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

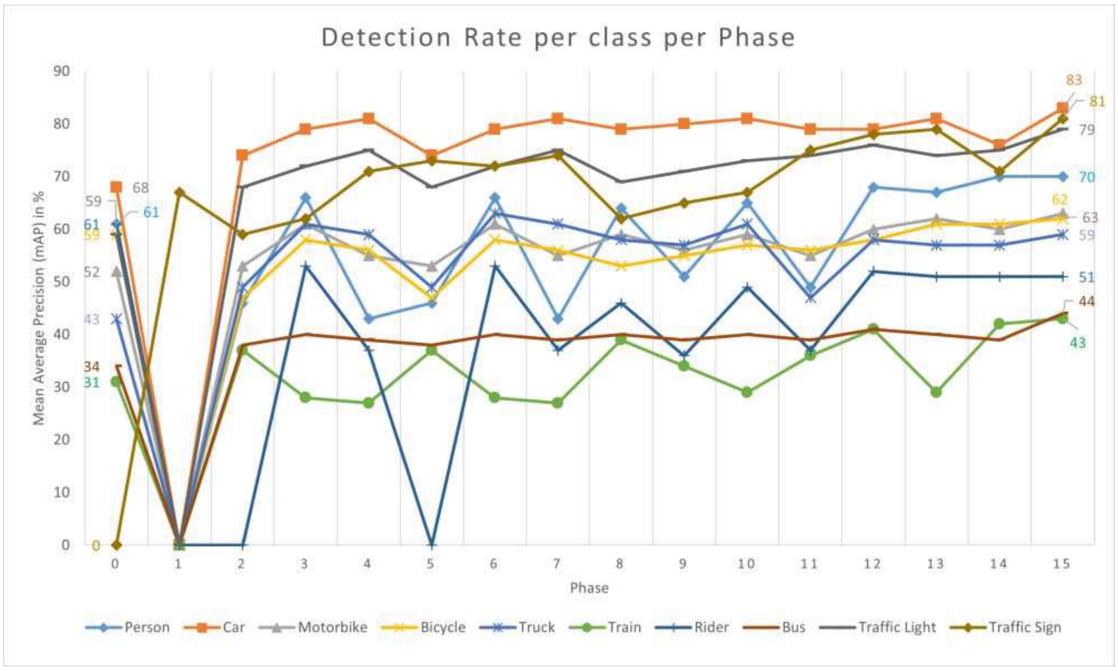


Figure 5.6: Detection results (mAP) in % shown for each class, split per training phase. Each class has a different colour, which can be found in the legend below the graph.

all classes in “phase 1”. Here, all detection rates are 0, with the exception of “Traffic Sign”.

The detection of trains decreases from “phase 2” (37%) to “phase 3” (28%) and “phase 4” (27%). Taking a look at Table 5.2 and Figure 5.3, the datasets that contribute to each phase reveal that “phase 2” used the VIPER training set, which consists of images from the videogame GTA 5. Whereas “phase 3” consists of the BDD dataset - which depicts streets in the San Francisco Bay area. The class “Train” can be found in this dataset, but is less frequent than with the VIPER dataset. This is similar in “phase 4” with the CVL dataset, which faces the same issue with “Train” being one of the classes that is less represented. This issue persists with each phase that uses either the GTSRD, BDD or CVL dataset, resulting in lower results in phases 6, 7, 10 and 13. Adding the VIPER dataset to one of these combinations results in a better detection rate. Phases 10 and 13 have the same detection results, with “phase 10” being a combination of the BDD and CVL datasets, and “phase 13” additionally adding the GTSRD set.

Similar observations can be made for the class “Person”, as visualized in Figure 5.8. The VIPER dataset contains less data on the class “Person”, when compared to the BDD dataset. Similar arguments apply to the CVL dataset, with it being recorded during night-time, with fewer people being present on the street during that time of day. These dataset combinations result in lower detection rates in phases 4, 5, 7, 9 and 11, being combinations of the aforementioned datasets. The inclusion of the BDD dataset improves

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

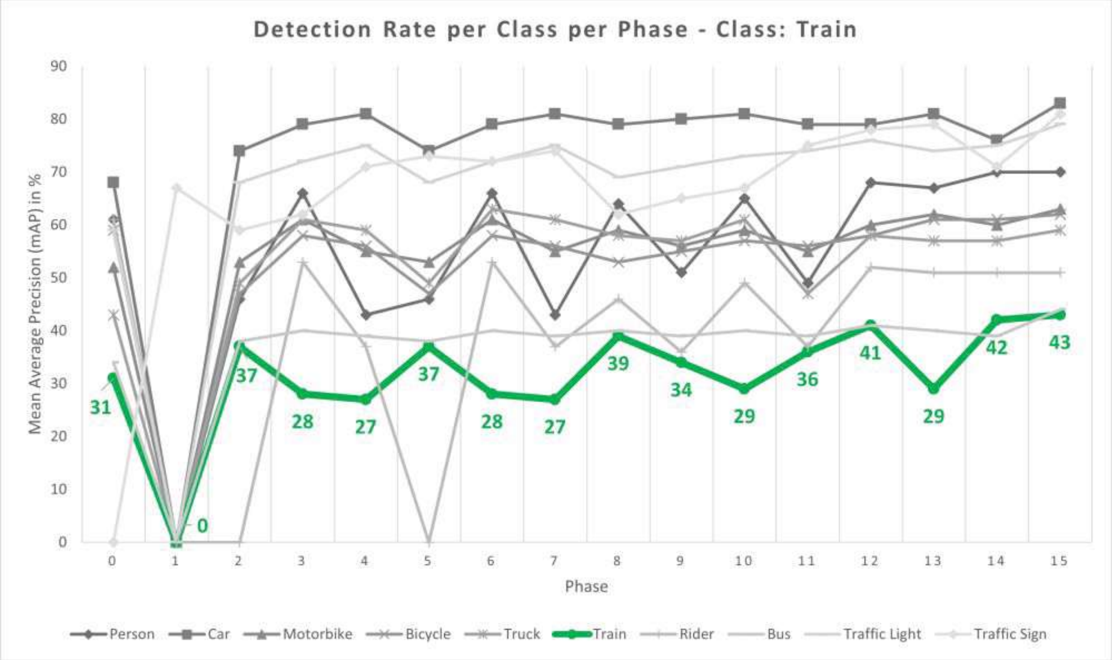


Figure 5.7: Detection results (mAP) in % split by class and phase. The focus of this graph is the class "Train", shown in green, with each mAP shown for each phase.

the detection rates, which can be observed in phases 12, 13 and 14, which all contain the BDD dataset.

Case 2: Class "Rider"

The class "Rider" is the only case where the detection declined during the training process. As can be seen in Figure 5.9, phases 1, 2 and 5 show a detection rate of 0. This is due to the fact that both the GTSRD and the VIPER dataset are missing the class "Rider". The reduction in the detection rate can be explained by the composition of the datasets and the fact that fewer people ride their motorbikes and bicycles at night. With less training data available, the class "Rider" becomes more difficult to detect. Overall, a decrease of 10% of mAP can be observed. For future implementations, it could be considered to combine the classes "Rider" with "Person" - depending on the final use case.

Case 3: GTSRD Dataset

The GTSRD dataset only contains images showing German street signs. Therefore, the only class a network trained with this dataset is able to detect, are different representations of street signs. As Table 5.3 shows, if this dataset is combined with others, for example in "phase 5", the main value that improves is the one where the GTSRD dataset can contribute - "Traffic Sign". For most of the other phases, the values remain unchanged,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

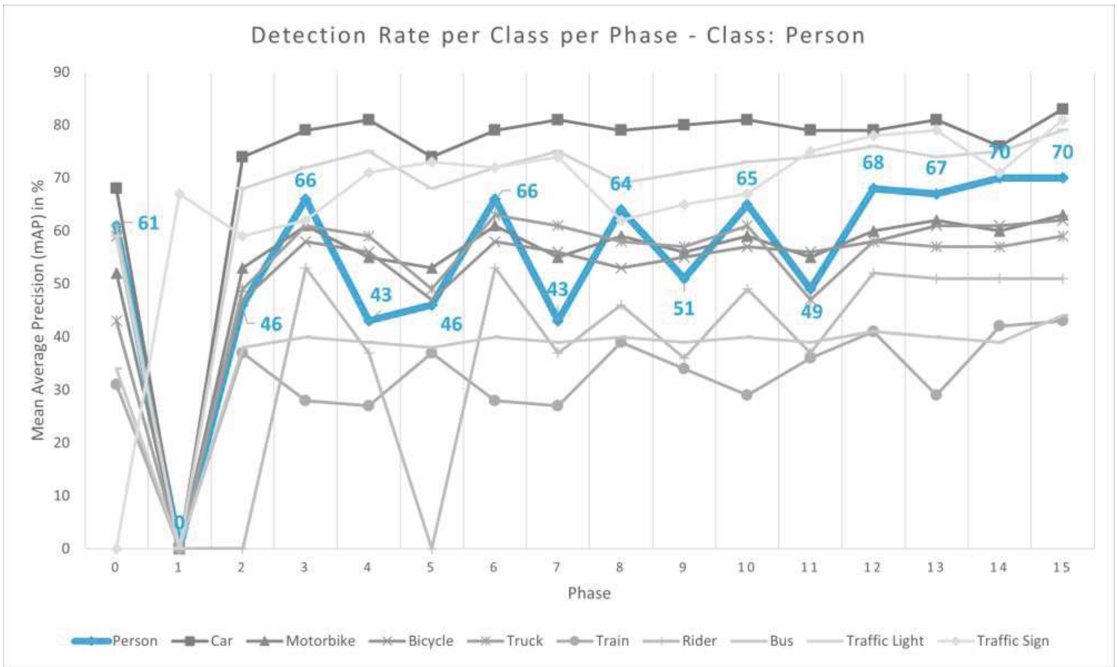


Figure 5.8: Detection results (mAP) in % split by class and phase. The focus of this graph is the class "Person", shown in blue, with each mAP shown for each phase.

with the exception from "phase 14" to "phase 15", which is discussed separately in Case 4.

When comparing phases 11 and 12, it can be observed that the differences in detection rates are caused by the addition of the different third dataset - in this case CVL or BDD. When combining all four datasets, an overall improvement can be seen, with the exception of the class "Rider" as was discussed in "Case 2".

While the addition of GTSRD to the training dataset does not result in a general improvement of the detection rate, the improvement gained for the class "Traffic Sign" is significant. Specifically for the use case of this master thesis, traffic sign recognition was an important factor. The GTSRD dataset was added with this specific implementation in mind, and while an overall improvement in detection rates could not be achieved, there was no aggravation regarding detection rates. This can be seen in Table 5.3 when comparing phase 14 and 15. When the training is done without the GTSRD dataset, the detection rate for the class "Traffic Sign" decreases by 8%. As soon as this is added again in "phase 15", the detection rate of the class "Traffic Sign" improves by 10% to 81%.

This can also be seen in Figure 5.10. Here, the "Traffic Sign" class is emphasized, with a blue highlight where the GTSRD dataset has been added. This graph once again highlights the improvement that can be achieved by adding a dataset that was selected for one specific use case.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

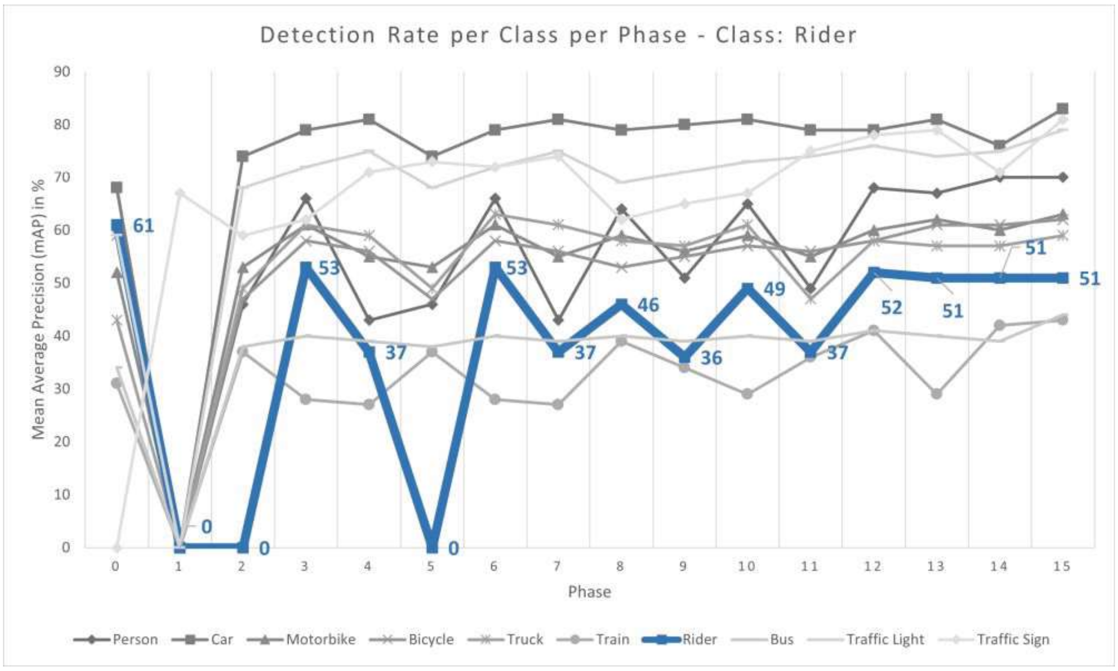


Figure 5.9: Detection results (mAP) in % split by class and phase. The focus of this graph is the class "Rider", shown in dark blue, with each mAP displayed for each phase.

Table 5.3: Classification results (mean average precision in %) obtained from phases 1, 2, 5, 11, 12, 14 and 15.

Class	*phase 2*	*phase 5*	*phase 11*	*phase 12*	*phase 14*	*phase 15*
Person	46	46	49	68	70	70
Car	74	74	79	79	76	83
Motorbike	53	53	55	60	60	63
Bicycle	47	47	56	58	61	62
Truck	49	49	47	58	57	59
Train	37	37	36	41	42	43
Rider	0	0	37	52	51	51
Bus	38	38	39	41	39	44
Traffic Light	68	68	74	76	75	79
Traffic Sign	59	73	75	78	71	81
Datasets	VIPER	GTSRD	GTSRD	GTSRD	VIPER	GTSRD
		VIPER	VIPER	VIPER	BDD	VIPER
			CVL	BDD	CVL	BDD
						CVL

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

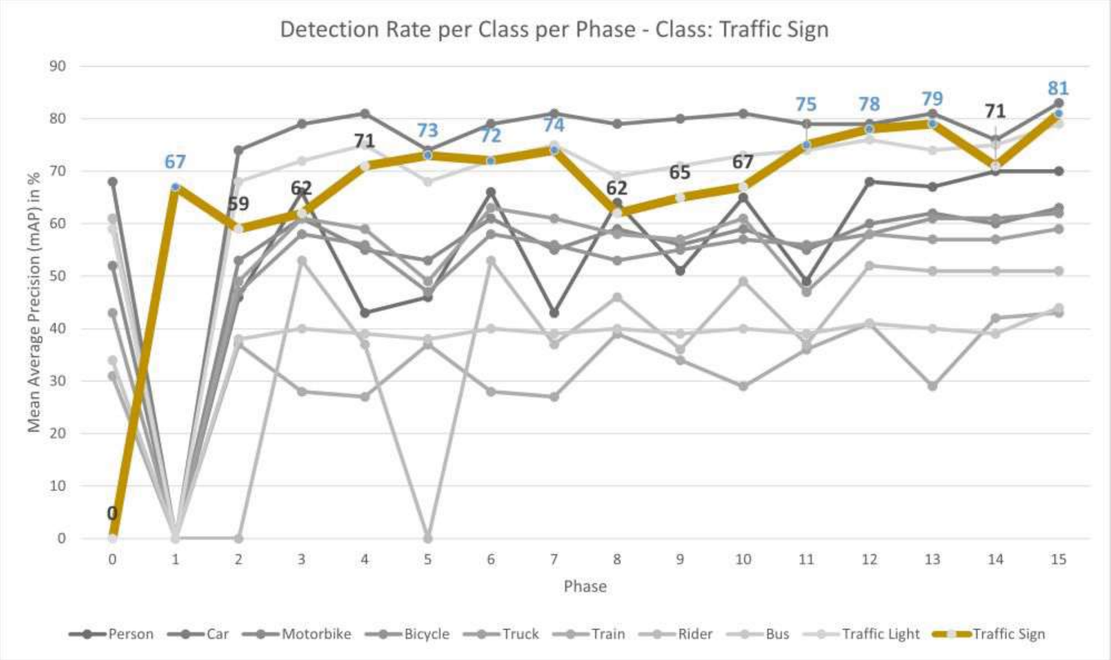


Figure 5.10: Detection results (mAP) in % split by class and phase. The focus of this graph is the class “Traffic Sign”, shown in light brown, with each mAP shown for each phase. The blue highlighted data points shows where the GTSRD dataset was added to the training dataset.

Case 4: Phase 14 to 15

As can be seen in Table 5.3, the recognition rates for some classes improve from “phase 14” to “phase 15”. The expected behaviour would be only the class “Traffic Sign” changing, as the GTSRD dataset is the only one added to “phase 15” and as it only contains images of traffic signs, but no other objects. But it can be observed that the classes “Car”, “Motorbike”, “Bicycle”, “Truck”, “Bus” and “Traffic Light” improve as well.

An example of a related misclassification is shown in Figure 5.11. The traffic sign on the left side of the image shares shape and brightness with the headlights of an oncoming car. In “phase 14”, it therefore falsely labels the headlight of the car as a “Traffic Sign”, missing the actual object “Car”. Once the GTSRD set is added to the training dataset in “phase 15”, the CNN has more information about traffic signs, their possible forms and occurrences. Therefore, it labels the oncoming car correctly as “Car”, as can be seen in Figure 5.12. Figure 5.11 and Figure 5.12 both show the same cars, the car on Figure 5.11 is merely zoomed in for better visibility. This case shows that an improvement in other classes can be achieved as well, even though the GTSRD dataset only contains data for one specific class.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

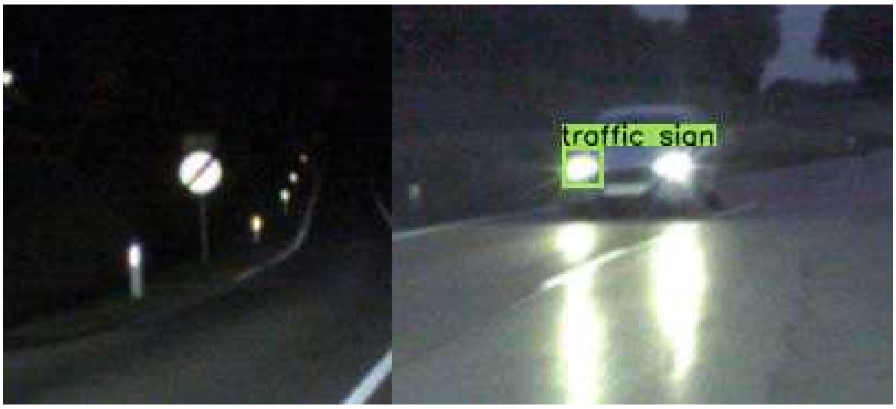


Figure 5.11: The left image shows a traffic sign on the road. The image on the right side shows an oncoming car being falsely labelled as “Traffic Sign”. This image highlights the similarity between a traffic sign at night with the headlights of an oncoming car.

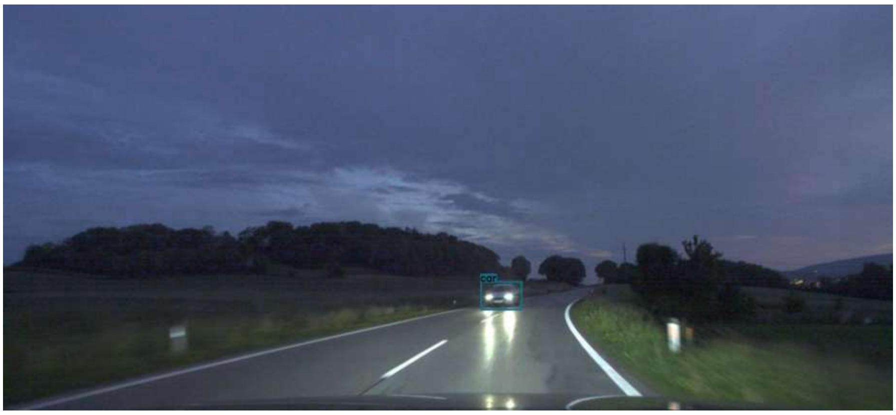


Figure 5.12: Oncoming car on a rural street in Lower Austria being classified as “Car”.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5.2.2 Qualitative Evaluation

The second evaluation method, as described in Section 4.3.2, focuses on the subjective observation of the object detection done by the CNN. For the purposes of this master thesis, qualitative evaluation entails running the object detection algorithm of the Convolutional Neural Network (CNN) on an image that has not been previously used in training, followed by a visual assessment of the resulting image. The identified objects are superimposed on a replica of the original image, marked by bounding boxes, with the detected class annotated above each box. This approach was adopted to provide the user with means to evaluate the work beyond metrics such as mean Average Precision (mAP), offering a visual depiction of the object detection process. Although qualitative and quantitative evaluations are discussed in separate subsections, they should not be considered in isolation. Rather, these two evaluation methodologies were interwoven throughout the process. As outlined in Section 5.1, training and evaluation of the YOLOv3 network consisted of sixteen phases, named "phase 0-15". The exact distribution for each phase was shown in Figure 5.3. Each dataset combination was used to retrain the dataset and afterwards, and the CNN was subject to both quantitative and qualitative evaluations. This section concerns itself with the results given by the qualitative evaluation process.

The test dataset used for qualitative evaluation is composed of a combination of images taken from the BDD, GTSRD, VIPER and CVL datasets. These images were not used for training or validation. The exact image distribution was shown in Figure 4.13. As described in Section 5.2.1, an improvement of 16.7% mAP was achieved, when comparing the result of "phase 15" with the baseline "phase 0". The visual representation of this can be found in Figures 5.13 and 5.14. The first Figure 5.13 shows two images taken from the test dataset, both depicting streets at different times of day. Subfigures 5.13 (a) and (b) show the results of "phase 0", while (c) and (d) show the results of "phase 15". Subfigures 5.13 (a) and (b) show that the CNN trained with the COCO dataset was not able to detect any objects. With the "Street Sign" class completely missing from the COCO dataset, this is an expected outcome. But "phase 0" also missed the car in Subfigure5.13 (a), which was detected in "phase 15", as shown in Subfigure 5.13 (c). This can probably be attributed to the lighting conditions, since the COCO dataset is missing images recorded at night-time and therefore has difficulty detecting the car in the distance.

Similar to the previously discussed figure, Figure 5.14 also compares the results of "phase 0" (in Subfigures 5.14 (a) and (b)) with the results of "phase 15" (Subfigures 5.14 (c) and (d)). Here it can be observed that while a car in Subfigure 5.14 (a) was detected in "phase 0", the other cars, as well as the traffic light were missed. However, "phase 15" falsely classified one car in Subfigure (c) as "Truck". We noticed that the "phase 15" CNN has an issue with falsely classifying cars as trucks. A possible explanation is that the distinction between them becomes more difficult with cars growing in size and the increasing numbers of SUVs on city streets. Subfigure 5.14 (b) shows that the "phase 0" CNN is able to detect cars, traffic lights and people on the side-walk correctly, with only the street sign on the right side missing, which is corrected in Subfigure 5.14

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

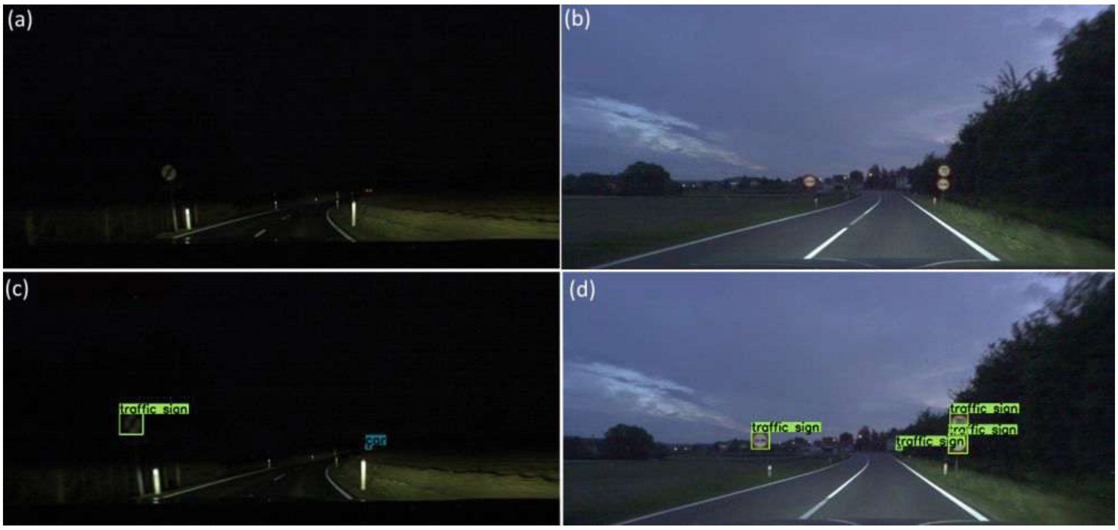


Figure 5.13: Examples of object detection results for different images. Subfigure (a) and (b) were the result of “phase 0”, while Subfigure (c) and (d) were the results of “phase 15”. While no objects are detected in Subfigure (a) and (b), all objects are detected in Subfigures (c) and (d).

(d) (“phase 15”). It should be noted that these largely correct results were achieved for images depicting day-time scenes. As previously mentioned, the COCO dataset only shows object recorded during the day, with street-specific images depicting night-time scenes completely missing from the dataset.

The final “phase 15” CNN was not solely evaluated on images from the test dataset. Given that the intended application of the CNN was to function in a vehicle navigating live traffic and processing a real-time video feed, the network was also assessed using videos. These videos depicted various rural roads in Upper Austria at different times of the day. As an example, Figure 5.15 demonstrates the object detection capabilities of the network when applied to these videos. It highlights the network’s proficiency in identifying objects under varying lighting conditions and also displays the confidence value associated with each detected object. To facilitate the processing of these videos, the network segmented them into individual frames and applied the object detection algorithm to each frame. The time taken to analyze each frame, measured in milliseconds (ms), is displayed in the upper left corner of the frame to ensure real-time applicability. The outcomes of this video processing are detailed as follows: Subfigure 5.15 (a) depicts a country road at dusk where one traffic sign is detected with a confidence of 97.03%. Subfigure 5.15 (b) illustrates a highway at night with two trucks (confidence values of 98.63% and 37.51%) and a car (56.97%) being detected. Subfigure 5.15 (c) presents a village during the day where several cars (confidence values of 96.36%, 99.30%, 85.19%, and 96.64%) and a person (93.59%) are detected. Lastly, Subfigure 5.15 (d) portrays a country road at night with one traffic sign detected (confidence of 96.85%).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

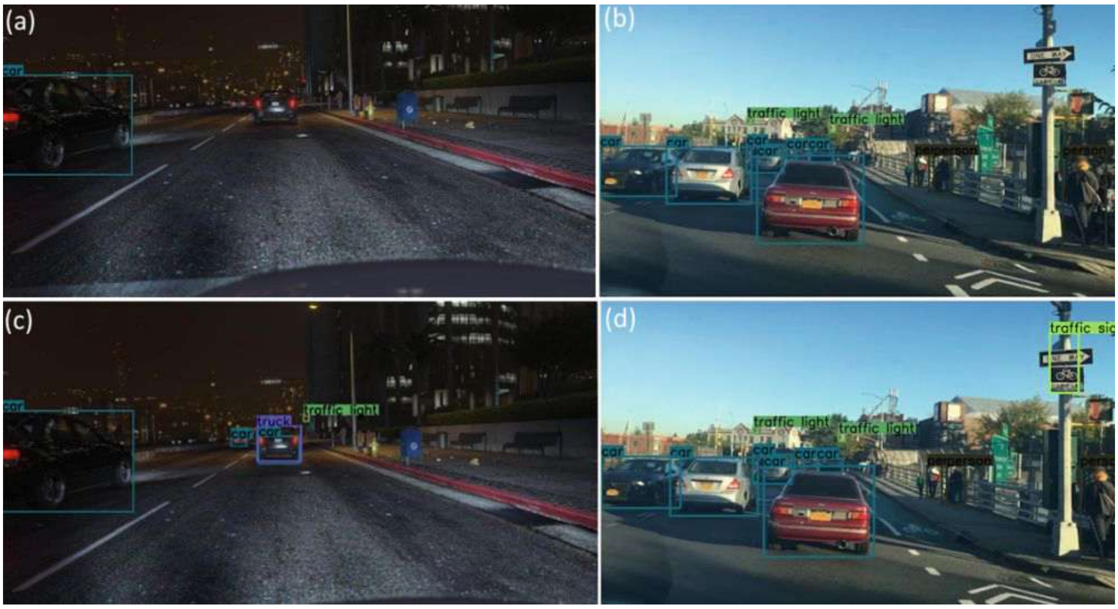


Figure 5.14: Examples of object detection results for different images. This graph compares the results of "phase 0" in Subfigures (a) and (b) with the results of "phase 15" in Subfigures (c) and (d).

While the visual results show an overall improvement, one special case requires distinct discussion. To provide insight into the functionality of the CNN as well as discuss the individual training steps, the following case description highlights this specific case that occurred during training.

Case: Street Sign Recognition

Since "phase 0" was missing the "Traffic Sign" class completely, it was decided to train the network using the GTSRD dataset in "phase 1". The GTSRD dataset differentiates between 40 different street sign classes, which were kept unchanged for the retraining and evaluation "phase 1". As can be seen in Figure 5.16, training the CNN with only the GTSRD dataset resulted in a lot of false positives. It falsely detected the whole image as different street signs, with not being able to detect the actual street signs.

In order to see if this error would persist, the network was retrained by adding the CVL dataset. The issue of false positives persisted, but the addition of the CVL dataset enabled the CNN to detect some traffic signs. Therefore, we decided to merge the over 40 different traffic signs classes from the GTSRD dataset into one singular "Traffic Sign" class and run the evaluation again. This resulted in Figure 5.17. As can be seen, the system was now able to detect various traffic signs (but not the car).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

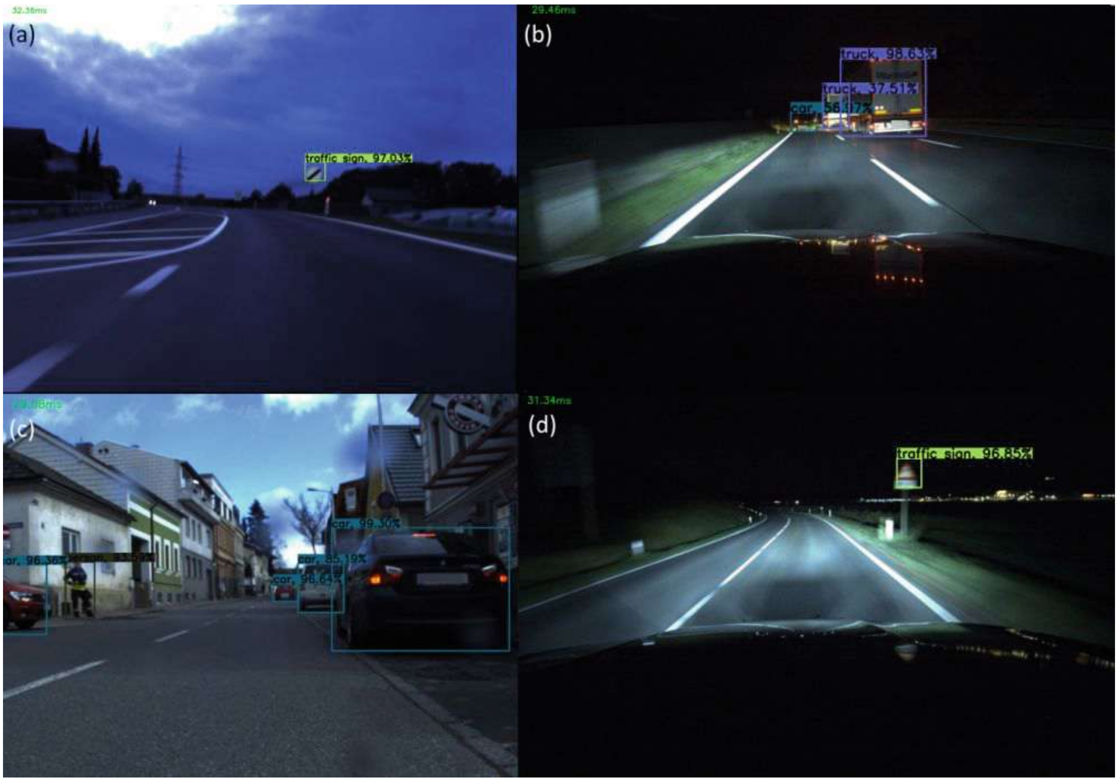


Figure 5.15: Examples of object detection results of the final network under different lighting conditions, with computed confidence values. The images shown were taken from a CVL test video. The confidence values are displayed above each object.

From this on, only one “Traffic Sign” class was available in the CNN. The system was retrained, with “phase 7” being a combination of the CVL and GTSRD dataset, which resulted in Figure 5.18. Now, aside from detecting both traffic signs, the network was able to detect the car as well. Merging different traffic sign classes into one singular “Traffic Sign” class, produced satisfactory results. for the use case of this master thesis.

5.3 Critical Reflection and Future Work

This section is dedicated to a critical reflection of the work done for this master thesis and investigates open questions for future implementations.

5.3.1 Street Conditions

Upon examining the training data, it can be seen that there is a lack of data depicting streets covered in snow. While there are plenty of images captured during night-time and while raining, we advocate for the future inclusion of snow scene images in the training dataset.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

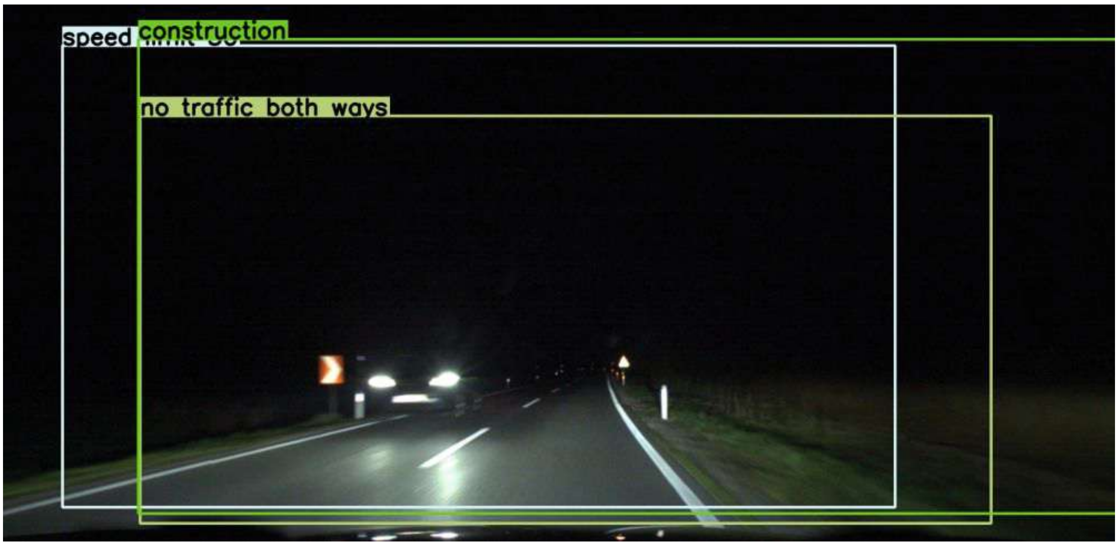


Figure 5.16: Example of false positive object detection for different traffic sign classes. The whole image was labelled as different street signs, while the street signs and car in the image were not found.

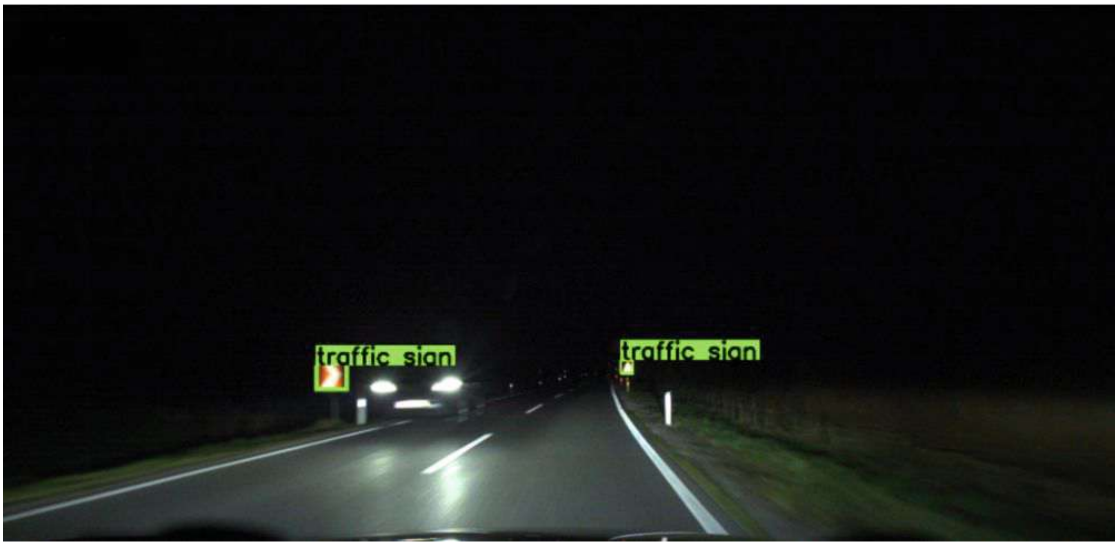


Figure 5.17: Example of object detection from "phase 1", after merging the 40 different traffic sign classes into one class "Traffic Sign".

Additionally, there exists a bias towards American street scenes in the training datasets used, particularly those from the Bay Area in California. Except for the German Traffic Signs Dataset (GTSRD), the two largest datasets, BDD and VIPER, predominantly feature images depicting American streets. We suggest addressing this bias and possibly adjusting the training dataset accordingly in future studies.

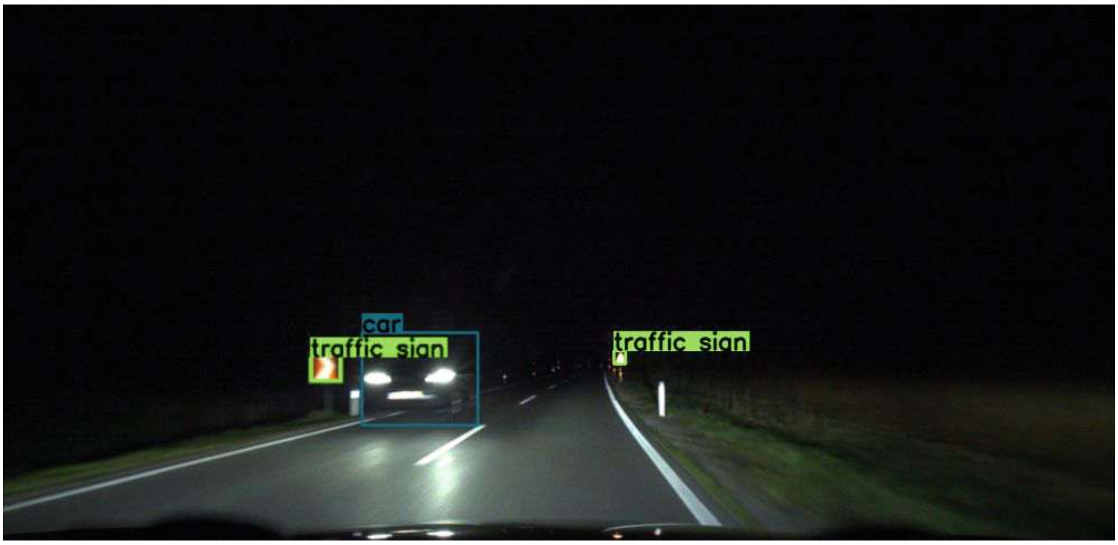


Figure 5.18: Example of a result gained from “phase 7”, which combines the GTSRD and CVL datasets. Both traffic signs and the car are found and labelled correctly.

5.3.2 Rider Detection

As presented in Section 5.2.1, rider detection was the sole category that exhibited a 10% decline in mAP. This could possibly be attributed to the characteristics of the dataset used for training the CNN, which contained night-time images. Consequently, there were fewer riders on motorbikes and bicycles, making them more challenging to detect. In contrast, stationary motorbikes and bicycles (without a rider) were considerably more prevalent and therefore easier for the CNN to detect.

5.3.3 Distinction Car and Truck

The CNN was trained to differentiate between the two classes “Truck” and “Car”. Partly due to the increasing prevalence of SUVs on the roads, the distinction between a truck and a car can sometimes be challenging, particularly when the car is at a distance. An example of this can be seen in Figure 5.19. This could raise a broader discussion about whether this differentiation between a truck and a car is always necessary or beneficial.

5.3.4 Video Detection

When dealing with object detection in videos, one option is to process each frame individually. Consequently, an object that is accurately detected in one frame might be missed in a following frame. This phenomenon can be observed in Figure 5.20. In Subfigure 5.20 (a), the traffic light situated above the bus on the right-hand side is overlooked, while it is correctly detected and labelled in Subfigure 5.20 (b). To mitigate this issue, it would be beneficial to incorporate additional information during processing individual frames. For instance, storing the locations of detected objects and updating

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

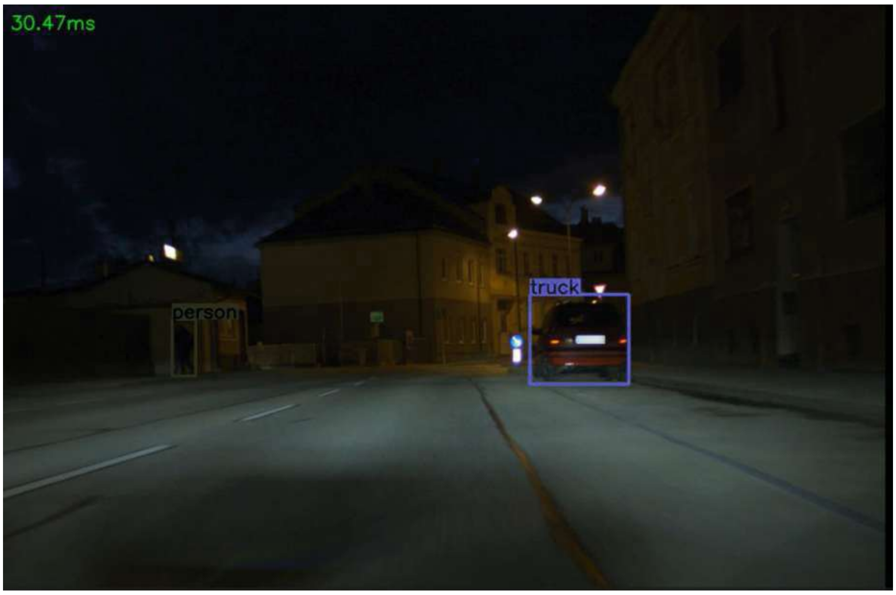


Figure 5.19: Night scene, depicting a parked car on the right hand side. The object detection algorithm labels the object as a “truck”.

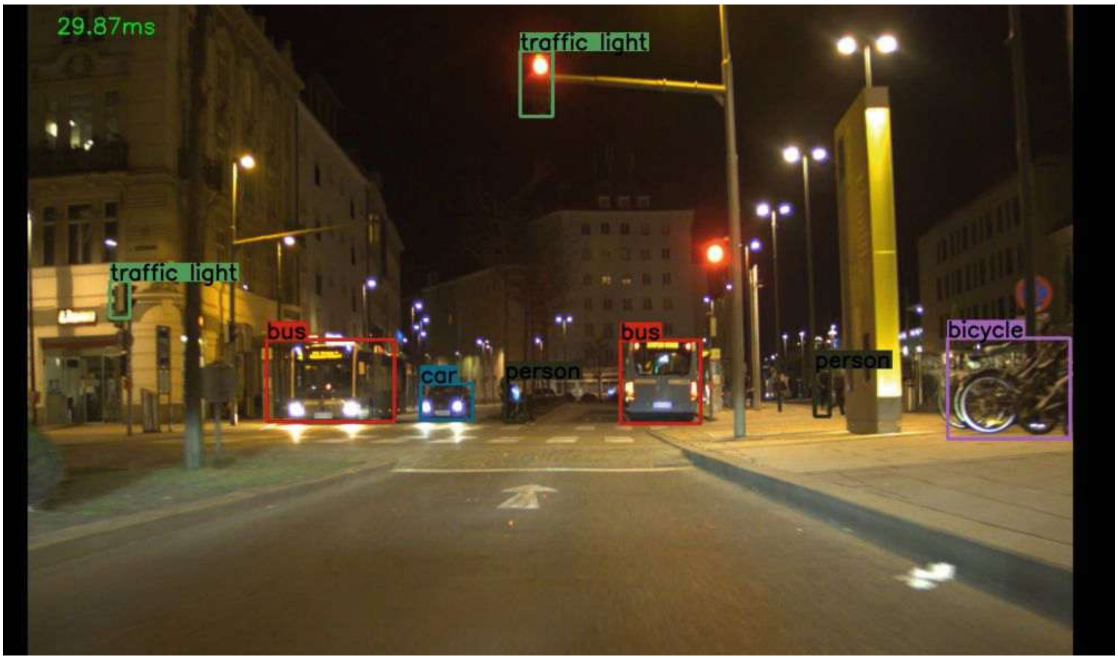
them with a movement vector could prevent them from being lost in the next image. On the other hand, each additional computational and data processing step could potentially slow down the system, thereby limiting the real-time applicability of the CNN. Bearing these considerations in mind, it will be a subject for future research to delve deeper into potential adaptations.

5.3.5 False Positives

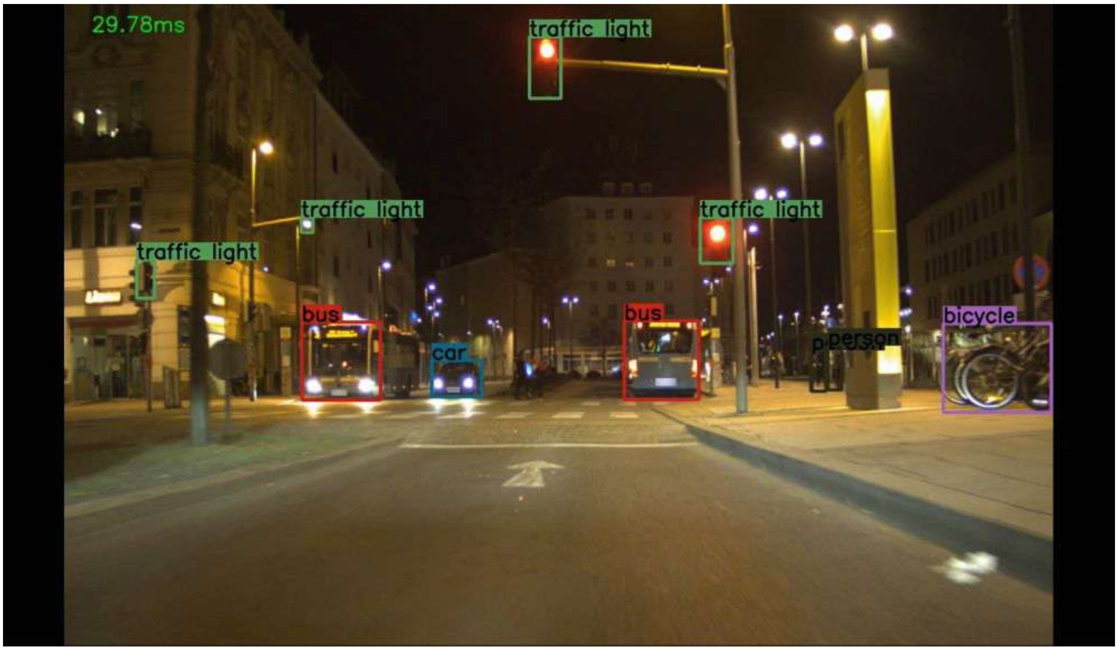
The final topic of discussion concerns the occurrence of false positives, which often arise in object detection systems. However, their analysis can sometimes offer valuable insights into the systems operation and potentially yield solutions to prevalent issues.

During qualitative analysis, we sometimes observed the misidentification of an oncoming vehicle’s headlights as a street or traffic sign. Figure 5.21 provides an example of such a scenario. In Figure 5.21, the vehicle’s left headlight is falsely classified as a traffic sign with a confidence level of 81.44%, indicating a relatively high degree of certainty from the network that the detected object is a street sign. We observed that street signs sometimes bear a striking resemblance with an oncoming car’s headlights. This issue can potentially be mitigated by incorporating contextual information into the network. Additionally, integrating information about the distance from the ground could further enhance the system’s accuracy and prevent such misclassification in the future.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



(a) One frame taken from a test video. It depicts a city street in the evening.



(b) A second frame, taken from the same test video as Subfigure (a). This image was taken a few frames after the one depicted in Subfigure (a), whereby the person next to the bus and one traffic light were not detected.

Figure 5.20: Example images from running the object detection CNN on a test video by CVL. While depicting the same streets a few frames apart from each other, slightly different objects are detected in each frame.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

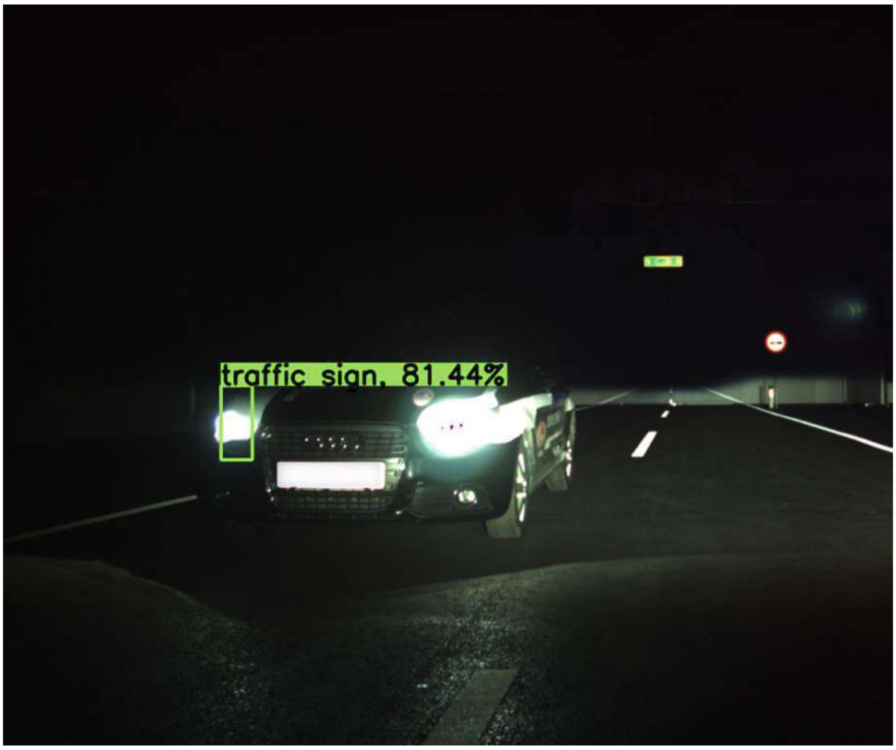


Figure 5.21: Image showing a car with one of the front headlights being detected as a traffic sign with 81.44% accuracy.

Another common error concerns reflections on the wind-shield. The camera's placement, which is behind a glass surface that reflects incoming traffic, results in the creation of unwanted artefacts. These artefacts may subsequently be misclassified, as depicted in Figure 5.22. In this example, the car is also incorrectly classified due to the distortion of the entire object. Such an error can be mitigated by adjusting the camera's placement to eliminate unwanted reflections as far as possible. It may also be necessary to incorporate contextual information, as also indicated in the previous example. However, it is important to consider that the real-time applicability of the system may be compromised as a result.

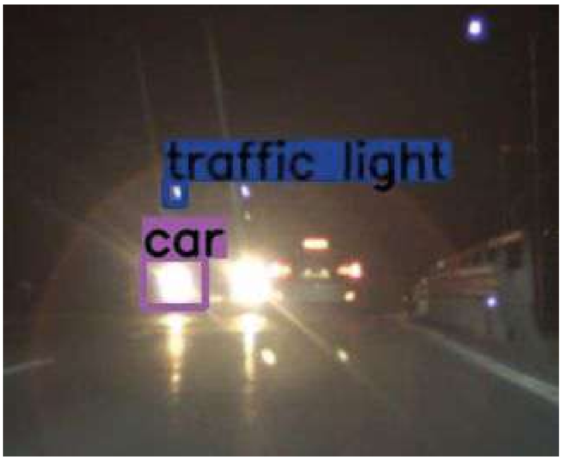


Figure 5.22: Reflections of an approaching car being wrongely classified as traffic lights. The left headlight is misclassified as the complete car.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

Conclusion

The aim of this master thesis was to train the Convolutional Neural Network (CNN) YOLOv3, to enable it to detect objects in night-time traffic scenes. The development process started with the selection of the classes and matching datasets for this task, which were gradually introduced to the training set. This resulted in fifteen distinct training phases, each of which was evaluated individually. While the task of retraining the YOLOv3 network was rather straight-forward, the selection of the datasets presented itself to be more challenging. Since most publicly available datasets have an inherent bias towards day-time scenes, the first step was to find suitable datasets that represent night-time scenes. Aside from introducing virtual data from the videogame GTA5, we also added self-recorded data, which enabled the network to detect objects more reliable. Overall, we observed a 16.7% improvement between the initial and final training outcomes. Notably, the classes “Car”, “Traffic Light”, and “Traffic Sign” achieved detection rates of 80% or higher. Each training session was followed by an evaluation of the current training phase. Initially, we started with visual evaluation, which quickly proved itself to be not sufficient. Therefore, we split the evaluation into “quantitative” - analysing the detection rates, recall and precision - and “qualitative” - evaluating the results based on visual criteria. Each evaluation method provided information for the system: first, the selection of the datasets and after the selection was done, the improvement of the detection rates of each phase. There, we were able to find that the introduction of a dataset only containing traffic signs, improved the detection rate of other classes (“Car’, “Truck”, “Bus” and “Traffic Light”) as well. Introducing a dataset only depicting one type of objects enabled the network to distinguish between - for example - a traffic sign and an oncoming car at night, which due to their shape and brightness can easily be confused.

While this trained network is able to detect objects during day-time as well as night-time scenes, there are still open issues to be tackled in future implementations. First, the training data lacks images depicting snow, which should be introduced in future implementations. The largest issue that still has to be tackled in future implementation is the detection of false positives. The headlights of an oncoming car are often falsely classified as traffic signs. While we were able to improve this error rate with the introduction of a street sign-specific dataset, the issue can also potentially be avoided by taking additional spatial information - like the distance from the bottom of the street to the car's headlight - into account. The final issue was the placement of the camera in the car itself. If the camera is placed behind the windshield or next to the headlights, the glass can distort the oncoming image and create unwanted artefacts. Camera placement should therefore be one of the first issues to consider for future implementations. In conclusion, while the training process was tailored to our specific use case, we were able to gain information on which datasets should be used and how they should be combined to allow for successful object detection in night-scenes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Overview of Generative AI Tools Used

No generative AI tools were used in creation of this work.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

2.1 LabelImg being used for image annotation. The user can draw a rectangle along the object and attaches a class to it - in this case “person” to the football player. Source: [Lab] 6
2.2 This example for object detection shows three detected objects in the image: a dog, a bicycle and a truck. Around each detected object, a bounding box, with the corresponding confidence values is drawn. These confidence values (99.39% for the dog, 99.19% for the bicycle and 94.09% for the truck) indicate the likelihood that this box contains the detected class. A value of 100% represents absolute certainty that the box contains the detected object. Source: [RF18]. 8
2.3 Haar feature applied to the image of a human face. The first feature measures the difference in intensity between eyes and upper cheeks, the second feature shows the intensity in the eye regions compared to the bridge of the nose. Source: [VJ04] 8
2.4 Visualization of SIFT feature points found in an image, after difference of Gaussians has been applied. Source: [sif] 10
2.5 Visualization of HoG feature points. Subfigure (a) shows the image cells in red, in which HoG features are calculated. Subfigure (b) visualizes the HoG descriptor computed for the same image. Source: [FGB13] 10
2.6 Concept of a perceptron by Rosenblatt [Ros57], which takes several inputs and maps them onto a single output. Source: [akc] 12
2.7 Example of a fully-connected multi-layer neural network: Starting with several input values, from x_1 to x_4, which are fed into the input layer (red nodes). This is followed by processing the input in several hidden layers (blue nodes) and finally creating two outputs (green nodes), Class1 and Class2. 12
2.8 Visualization of a step function, which shows a change in the input value from 0 to 1, once a threshold has been met. In this case, the threshold is set at 0. 14
2.9 Visualization of a Sigmoid function, $A = \frac{1}{1-e^{-x}}$. The value change is more gradual when compared to the step function shown in Figure 2.8 14
2.10 Visualization of a tangens hyperbolicus function, $A = \frac{2}{1+e^{-2x}} - 1$. 15
2.11 Visualization of a Rectified Linear Units (ReLUs) function, $A = max(0, x)$. The value of the function is 0 below a predefined threshold and from this threshold forward (in this case 0), the value gets mapped directly. 15

2.12 CNN example structure. An input image is processed by the network, starting with feature extraction through alternating convolution and pooling layers. The objects in the image are then classified by the classification layer, using a fully connected neural network. 17
2.13 AlexNet CNN Architecture. Source: [KSH12] 18
2.14 Example image of how the YOLO object detector [RDGF16] calculates bounding boxes. The image gets segmented into grids, in which each grid predicts N bounding boxes. These are then filtered, resulting in the final classification. Source: [RDGF16] 20
2.15 The YOLOv3 network architecture, as described on the TowardsDataScience webpage [yola], processes an image of a dog by first passing it through a residual block to reduce the image scale. This is followed by several detection layers that attempt to identify the object at the current scale. After this, upsampling layers resize the image to a larger resolution, and the detection process is repeated. Source: [yola] 20
2.16 The YOLOv3 anchor box detection as described by the TowardsDataScience webpage [yolb]. The image gets split into several grid cells (right side of the figure), and each grid cell gets assigned three anchor boxes (left side of the figure). Within these boxes, objects are detected. 22
2.17 The image shows an example of Intersection over Union (IoU) with predicted and ground truth boxes. It shows a dog sitting in front of a bicycle, with two colored boxes: one red and one green. The red box represents the predicted bounding box after the image was processed by the CNN, while the green box represents the ground truth bounding box. The IoU is then calculated by comparing these two boxes. 23
2.18 Area of Overlap and Area of Union, needed for IoU calculation. The left image shows the Are of Overlap (filled gray), calculated from the overlap of the predicted bounding box (red) and ground-truth bounding box (green). The right image shows the Area of Union, calculated as the union of predicted and ground-truth bounding boxes. 24
2.19 Examples for (a) True Positive (TP): the network correctly predicts that an object (in this case, a “Dog”) was found. Subfigure (b) shows a False Positive (FP): The network predicts a positive find for an object that is not actually there (in this case a “Cat”). Finally, (c) shows a False Negative (FN): There are objects in the image that should be detected, but the network fails to find them. 24

3.1 An example from Microsoft’s COCO [LMB$^+$14] dataset, showing various common objects in their natural context. 29
3.2 Example of German Traffic Sign Recognition Database (GTSRD) [SSSI11], showing several street signs in different lighting situations. 29
3.3 An example of the VIPER dataset [RHK17], showing several simulated cars in a simulated city street. 30

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.4 Example of Berkeley Deep Drive dataset [YXC+18], showing several cars and a motorbike in the city of San Francisco, waiting in front of a red traffic light. 30
3.5 Example of the CarVisionLight (CVL) dataset, showing a car and two street signs at night in the rural streets of Lower Austria. 31

4.1 System pipeline. First, the system has to be set up, which is then followed by an initial test-run, to confirm it was set up correctly. After this, training and evaluation follows. 34
4.2 Initial test run of YOLOv3 using the Cityscapes dataset. First, the Cityscapes dataset is processed through the YOLOv3 CNN. This is then followed by an evaluation of the training results. 38
4.3 Pipeline for CNN development. The left box shows the Training Session, in which the dataset is loaded into the CNN. Then, the CNN is retrained and the result gets stored. This result is then sent to the Evaluation (the right side of the image), in which each session is evaluated with qualitative and quantitative methods. The results of each evaluation round are stored for further analysis. 38
4.4 A closer look at the pipeline used for retraining the CNN. Here, the annotation of the dataset is followed by the definition of the used classes, anchor calculation and finally the training itself. 39
4.5 This graph shows the different dataset combinations for each of the 15 training rounds, with each dataset in a different color. The color for each dataset, as well as the exact number of images in each set (not split by day/night), can be seen in the legend below the graph. 46
4.6 Video processing example image. This image shows the result of a qualitative evaluation performed on a video. The upper left corner displays the processing time. The boxes show the detected objects, with a percentage showing the confidence. 47
4.7 Example of the GTSRD dataset, with one car and several street signs in a city environment during day. 48
4.8 Example annotation of the GTSRD dataset, showing a street sign in a green bounding box, with the corners denoted as $(x1, y1)$ and $(x2, y2)$. 48
4.9 Example result of training the network with individual classes for traffic signs, tested on the CVL test dataset. Several false positives were found, with a multitude of street signs being detected. 49
4.10 Example from the VIPER dataset, showing a night scene with cars and a street sign. 50
4.11 Image from the BerkeleyDeepDrive training dataset, depicting a street in a rural area. 52
4.12 Here are two sample images from the CVL dataset. They were captured during the same acquisition drive for the training dataset. 54
4.13 Distribution of day and night images in the original datasets (GTSRD, VIPER, BDD, CVL) and contribution to the final dataset (Final). 55

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5.1 Pipeline of "phase 0", the first evaluation phase. First, the test dataset is loaded into the YOLOv3 network. The results are then evaluated and create a baseline for future comparison. 58

5.2 Pipeline for CNN development. The left box shows the Training Session, in which the dataset is loaded into the CNN. Then, the CNN is retrained and the result is stored. This result is then sent to the Evaluation (the right side of the image), in which each session is evaluated with qualitative and quantitative methods. The results of each evaluation round are stored for further analysis. 59

5.3 This graph shows the different dataset combinations for each of the 15 training rounds, with each dataset in a different color. The color for each dataset, as well as the number of images in each set (not split by day/night), can be seen in the legend below the graph. 59

5.4 Results from applying the pretrained YOLOv3 object detection network on two Cityscapes example images. 60

5.5 Results from applying the pretrained YOLOv3 object detection network on two night-time example images. 61

5.6 Detection results (mAP) in % shown for each class, split per training phase. Each class has a different colour, which can be found in the legend below the graph. 65

5.7 Detection results (mAP) in % split by class and phase. The focus of this graph is the class "Train", shown in green, with each mAP shown for each phase. 66

5.8 Detection results (mAP) in % split by class and phase. The focus of this graph is the class "Person", shown in blue, with each mAP shown for each phase. 67

5.9 Detection results (mAP) in % split by class and phase. The focus of this graph is the class "Rider", shown in dark blue, with each mAP displayed for each phase. 68

5.10 Detection results (mAP) in % split by class and phase. The focus of this graph is the class "Traffic Sign", shown in light brown, with each mAP shown for each phase. The blue highlighted data points shows where the GTSRD dataset was added to the training dataset. 69

5.11 The left image shows a traffic sign on the road. The image on the right side shows an oncoming car being falsely labelled as "Traffic Sign". This image highlights the similarity between a traffic sign at night with the headlights of an oncoming car. 70

5.12 Oncoming car on a rural street in Lower Austria being classified as "Car". 70

5.13 Examples of object detection results for different images. Subfigure (a) and (b) were the result of "phase 0", while Subfigure (c) and (d) were the results of "phase 15". While no objects are detected in Subfigure (a) and (b), all objects are detected in Subfigures (c) and (d). 72

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5.14 Examples of object detection results for different images. This graph compares the results of "phase 0" in Subfigures (a) and (b) with the results of "phase 15" in Subfigures (c) and (d). 73
5.15 Examples of object detection results of the final network under different lighting conditions, with computed confidence values. The images shown were taken from a CVL test video. The confidence values are displayed above each object. 74
5.16 Example of false positive object detection for different traffic sign classes. The whole image was labelled as different street signs, while the street signs and car in the image were not found. 75
5.17 Example of object detection from "phase 1", after merging the 40 different traffic sign classes into one class "Traffic Sign". 75
5.18 Example of a result gained from "phase 7", which combines the GTSRD and CVL datasets. Both traffic signs and the car are found and labelled correctly. 76
5.19 Night scene, depicting a parked car on the right hand side. The object detection algorithm labels the object as a "truck". 77
5.20 Example images from running the object detection CNN on a test video by CVL. While depicting the same streets a few frames apart from each other, slightly different objects are detected in each frame. 78
5.21 Image showing a car with one of the front headlights being detected as a traffic sign with 81.44% accuracy. 79
5.22 Reflections of an approaching car being wrongely classified as traffic lights. The left headlight is misclassified as the complete car. 80

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

4.1 Parameter description for training sessions 43

5.1 Classification results (mean average precision in %) obtained from “phase 0” and “phase 15”. The results are split for each class, with the final line showing the results for all classes. 63
5.2 Classification results (mean average precision, mAP) shown for each class and each of the sixteen phases. 64
5.3 Classification results (mean average precision in %) obtained from phases 1, 2, 5, 11, 12, 14 and 15. 68

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: a system for large-scale machine learning. In *OSDI'16: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pages 265–283. USENIX Association, 2016.

[akc] Roman Akchurin Web Page. `https://romanakchurin.com/neural-networks/introduction/`. Accessed: 2024-08-27.

[AVC15] Akhan Almagambetov, Senem Velipasalar, and Mauricio Casares. Robust and Computationally Lightweight Autonomous Tracking of Vehicle Taillights and Signal Detection by Embedded Smart Cameras. *IEEE Transactions on Industrial Electronics*, 62(6):3732–3741, 2015.

[Bis96] Christopher Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1996.

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer International Publishing, 2006.

[BSB23] Ravit Ben-Shoushan and Anna Brook. Fused Thermal and RGB Imagery for Robust Detection and Classification of Dynamic Objects in Mixed Datasets via Pre-Trained High-Level CNN. *Remote Sensing*, 15(3):723–750, 2023.

[Bur07] Robert Burke. Sir Charles Sherrington's The Integrative Action of the Nervous System: A Centenary Appreciation. *Brain: A Journal of Neurology*, 130:887–894, 2007.

[CFFV16] Zhaowei Cai, Quanfu Fan, Rogerio S. Feris, and Nuno Vasconcelos. A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. In *Computer Vision - ECCV 2016*, pages 354–370. Springer International Publishing, 2016.

[CMGP23] Mario I. Chacon-Murguia and Abimael Guzman-Pando. Moving Object Detection in Video Sequences Based on a Two-Frame Temporal Information CNN. *Neural Processing Letters*, 55(5):5425–5449, 2023.

[COR^{+}16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223. IEEE, 2016.

[cuda] CUDAs Web Page. `https://developer.nvidia.com/cuda-toolkit`. Accessed: 2024-02-03.

[cudb] cuDNN Web Page. `https://developer.nvidia.com/cudnn`. Accessed: 2024-02-03.

[cvl] CarVisionLight Project Page. `https://www.ims.tuwien.ac.at/projects/carvisionlight/`. Accessed: 2019-06-12.

[DT05] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE, 2005.

[e3d] Emotion3D Web Page. `https://www.emotion3d.ai/`. Accessed: 2019-06-12.

[FGB13] Efstathios P. Fotiadis, Mario Garzón, and Antonio Barrientos. Human Detection From a Mobile Robot Using Fusion of Laser and Vision Information. *Sensors*, 13(9):11603–11635, 2013.

[Fuk80] Kunihiko Fukushima. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36:193–202, 1980.

[Gas19] Gilles Gasso. Logistic Regression. *INSA Rouen-ASI Departement Laboratory*, 1(1):1–30, 2019.

[GB07] Enzo Grossi and Massimo Buscema. Introduction to Artificial Neural Networks. *European Journal of Gastroenterology & Hepatology*, 19(12):1046–1054, 2007.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. IEEE, 2014.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[git] YOLOv3 Tensorflow Implementation Wizyoung, Github Page. `https://github.com/wizyoung/YOLOv3{_}TensorFlow`. Online, Accessed: 07-09-2023.

[GSG20] Florian Groh, Dominik Schoerkhuber, and Margrit Gelautz. A Tool for Semi-Automatic Ground Truth Annotation of Traffic Videos. *Electronic Imaging*, 2020(16):200–207, 2020.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[inta] Confusion Matrix for Your Multi-Class Machine Learning Model. `https://tinyurl.com/ys9sjrn7`. Accessed: 2024-01-08.

[intb] Dive Really Deep into YOLOv3: A Beginner's Guide. `https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e`. Accessed: 2024-01-08.

[JC16] Katarzyna Janocha and Wojciech Marian Czarnecki. On Loss Functions for Deep Neural Networks in Classification. *Schedae Informaticae*, 25(1):49–59, 2016.

[JNM17] Morten Jensen, Kamal Nasrollahi, and Thomas Moeslund. Evaluating State-of-the-Art Object Detector on Challenging Traffic Light Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–15. IEEE, 2017.

[Joa98] Thorsten Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142. Springer International Publishing, 1998.

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

[KSK17] So-Hyun Kim, Woo-Jin Song, and Sungho Kim. Infrared Variation Optimized Deep Convolutional Neural Network for Robust Automatic Ground Target Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[KWLK20] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9856–9865. IEEE, 2020.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Lab] LabelImg Git Page. `https://github.com/tzutalin/labelImg`. Accessed: 2020-02-17.

[LBD+89] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten Digit Recognition With a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 2:396–404, 1989.

[LHCB17] Kwangyong Lim, Yongwon Hong, Yeongwoo Choi, and Hyeran Byun. Real-time Traffic Sign Recognition Based on a General Purpose GPU and Deep-Learning. *PLoS One*, 12(3):1–22, 2017.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision, ECCV 2014*, pages 740–755. Springer International Publishing, 2014.

[Low99] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157. IEEE, 1999.

[Mac67] James MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[MN98] Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. CMU School of Computer Science, 1998.

[MP43] Warren S. McCulloch and Walter Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[ncc] NCCL Web Page. `https://developer.nvidia.com/nccl`. Accessed: 2024-02-03.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788. IEEE, 2016.

[RF18] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. In *Computer Vision and Pattern Recognition*, pages 1–6. Springer Berlin/Heidelberg, Germany, 2018.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28:1137–1149, 2015.

[RHK17] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for Benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2213–2222. IEEE, 2017.

[Roj96] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer International Publishing, 1996.

[Ros57] Frank Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[RTG⁺19] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666. IEEE, 2019.

[SBM06] Zehang Sun, George Bebis, and Ronald Miller. On-Road Vehicle Detection: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(05):694–711, 2006.

[sif] SIFT Feature Example Web Page. `labts.troja.mff.cuni.cz/~machl5bm/sift/`. Accessed: 2019-11-18.

[SPL24] Jin-Gyu Song, Jeong-Min Park, and Joon-Woong Lee. CNN Combined with a Prior Knowledge-Based Candidate Search and Diffusion Method for Nighttime Vehicle Detection. *International Journal of Control, Automation and Systems*, 22(3):963–975, 2024.

[SSA17] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation Functions in Neural Networks. *Towards Data Science*, 6(12):310–316, 2017.

[SSSI11] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460. IEEE, 2011.

[ST13] Sayanan Sivaraman and Mohan Manubhai Trivedi. Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1773–1795, 2013.

[sta] Stanford CS Class CS231n: Convolutional Neural Networks for Visual Recognition. `https://cs231n.github.io/`. Accessed: 2024-01-08.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[SW11] Claude Sammut and Geoffrey I. Webb. *Encyclopedia of Machine Learning*. Springer International Publishing, 2011.

[ten] TensorFlow Web Page. `https://www.tensorflow.org/`. Accessed: 2024-02-03.

[TKS$^+$19] Caleb Tung, Matthew R. Kelleher, Ryan J. Schlueter, Binhan Xu, Yung-Hsiang Lu, George K. Thiruvathukal, Yen-Kuang Chen, and Yang Lu. Large-Scale Object Detection of Images from Network Cameras in Variable Ambient Lighting Conditions. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 393–398. IEEE, 2019.

[UGL$^+$18] Andrew Ulmer, Kent W. Gauen, Yung-Hsiang Lu, Zohar R. Kapach, and Daniel P. Merrick. Investigating Dataset Distinctiveness. *The Summer Undergraduate Research Fellowship (SURF) Symposium*, 2018.

[UNC] Convention on Road Traffic, Vienna, 8 November 1968, United Nation Treaty Collection, Chapter XI, Section B, Number 19. `https://treaties.un.org/pages/ViewDetailsIII.aspx?src=TREATY&mtdsg_no=XI-B-19&chapter=11`. Accessed: 2020-06-02.

[VJ04] Paul Viola and Michael Jones. Robust Real-time Face Detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[WHZ$^+$19] Jian Wei, Jianhua He, Yi Zhou, Kai Chen, Zuoyin Tang, and Zhiliang Xiong. Enhanced Object Detection With Deep Convolutional Neural Networks for Advanced Driving Assistance. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1572–1583, 2019.

[WMZT20] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A Comprehensive Survey of Loss Functions in Machine Learning. *Annals of Data Science*, 1(1):1–26, 2020.

[Wu17] Jianxin Wu. Introduction to Convolutional Neural Networks. *National Key Lab for Novel Software Technology*, 5(23):495–526, 2017.

[yola] TowardsDataScience Web Page. `https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e`. Accessed: 2020-05-11.

[yolb] TowardsDataScience Web Page. `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b/`. Accessed: 2020-02-24.

[YXC$^+$18] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. *CoRR*, abs/1805.04687, 2018.