KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Engineering

First cycle, 15 credits

Object Tracking Evaluation: BoT-SORT & ByteTrack with YOLOv8

A Comparison of Accuracy and Computational Efficiency

MAXIMILIAN PETERSSON
NAHOM KIFLE SOLOMON

Stockholm, Sweden, 2024

Object Tracking Evaluation: BoT-SORT & ByteTrack with YOLOv8

A Comparison of Accuracy and Computational Efficiency

MAXIMILIAN PETERSSON

NAHOM KIFLE SOLOMON

Degree Programme in Computer Engineering
Date: June 13, 2024

Supervisor: Afsaneh Mahmoudi Benhangi
Examiner: Ki Won Sung
School of Electrical Engineering and Computer Science
Swedish title: Utvärdering av Objektspårning: BoT-SORT & ByteTrack med YOLOv8
Swedish subtitle: En Jämförelse av Noggrannhet och Beräkningseffektivitet

Abstract

In recent years, rapid advancements in computer vision have led to the integration of innovative use cases into our daily lives spanning from surveillance to autonomous vehicles being out on our streets . Notably, object detection and tracking have witnessed the emergence of powerful models such as You Only Look Once (YOLO), renowned for its real-time efficiency and accuracy. In this project, we conducted a comparative analysis of two tracking algorithms embedded within YOLOv8, BoT-SORT and ByteTrack, aiming to benchmark their performance in terms of Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), and Frames Per Second (FPS). YOLOv8, the latest iteration in the YOLO family developed by Ultralytics, was trained on a diverse dataset encompassing vehicles of varying shapes, sizes, and colors. Our benchmarks reveal that BoT-SORT exhibits greater accuracy and precision, whereas ByteTrack outperforms in terms of speed and computational efficiency. Consequently, the choice between BoT-SORT and ByteTrack hinges on the specific priority of either accuracy or speed.

Keywords

You Only Look Once (YOLO), Machine Learning, Object Detection, Object Tracking, Tracking Algorithms, BoT-SORT, ByteTrack

Sammanfattning

Under de senaste åren har snabba framsteg inom datorseende integrerats inom innovativa användningsområden som påverkar våra vardagliga liv, allt från övervakning till självkörande bilar. Särskilt har objektdetektering och objektspårning bevittnat framväxten av kraftfulla modeller däribland You Only Look Once (YOLO), känd för sin realtidsprestanda och noggrannhet. I detta projekt genomförde vi en jämförande analys av två spårningsalgoritmer inbäddade i YOLOv8, BoT-SORT och ByteTrack, i syfte att mäta deras prestanda med avseende på noggrannhet för spårning av flera objekt (MOTA), precision för spårning av flera objekt (MOTP) och bildrutor per sekund (FPS). YOLOv8, den senaste iterationen i YOLO-familjen utvecklad av Ultralytics, tränades med hjälp av ett mångsidigt dataset som omfattar fordon av varierande former, storlekar och färger. Våra resultat visar att BoT-SORT är ledande inom noggrannhet och precision, medan ByteTrack är ledande vad gäller hastighet och beräkningseffektivitet. Följaktligen beror valet mellan BoT-SORT och ByteTrack på huruvida noggrannhet eller hastighet prioriteras.

Nyckelord

You Only Look Once (YOLO), Maskininlärning, Objekt Detektering, Objekt Spårning, Spårnings Algorithmer, BoT-SORT, ByteTrack

Acknowledgments

We would like to thank our supervisor Afsaneh Mahmoudi Benhangi and examiner Ki Won Sung for their guidance and support during this project. We would also like to thank Supercircuits for their YouTube video of cars driving on a highway. This video was used to construct the test dataset on which our benchmarks were performed.

Stockholm, June 2024
Maximilian Petersson Nahom Kifle Solomon

Contents

1 Introduction **1**
1.1 Problem . . . 1
1.1.1 Original problem and definition . . . 1
1.1.2 Scientific and engineering issues . . . 2
1.2 Purpose . . . 2
1.3 Goals . . . 2
1.4 Research Methodology . . . 3
1.5 Delimitations . . . 3
1.6 Benefits, Ethics and Sustainability . . . 3
1.7 Structure of the thesis . . . 4

2 Background **5**
2.1 Machine Learning . . . 5
2.1.1 Methods of Machine Learning . . . 5
2.1.1.1 Supervised Machine Learning . . . 6
2.1.1.2 Unsupervised Machine Learning . . . 6
2.2 Computer Vision . . . 8
2.2.1 Object Detection . . . 8
2.2.2 Object Tracking . . . 10
2.3 Machine learning Models . . . 11
2.3.1 You Only Look Once (YOLO) . . . 11
2.3.2 Region-based Convolutional Neural Network (R-CNN) 12
2.4 Object Tracking Algorithms . . . 12
2.4.1 BoT-SORT . . . 13
2.4.2 ByteTrack . . . 13
2.5 Related Work Area . . . 14

3 Method **16**
3.1 Constructing the Training Dataset . . . 16

3.1.1 Acquire Data . . . 16
3.1.2 Processing Acquired Data . . . 17
3.1.3 Resulting Dataset . . . 17
3.2 Training the Model . . . 18
3.2.1 Training Environment, Google Colab . . . 18
3.2.2 Key Training Parameters . . . 18
3.2.3 Training Process . . . 19
3.3 Evaluating the Trained Model . . . 21
3.3.1 Bias-Variance Tradeoff . . . 21
3.3.2 Reading Performance Metrics: Finding the Right Balance . . . 22
3.4 Constructing the Test Dataset . . . 25
3.4.1 Gathering and Annotating the Test Dataset . . . 25
3.4.2 Resulting Test Dataset . . . 26
3.5 Benchmarking the Model . . . 26
3.5.1 Benchmark Preparations . . . 27
3.5.2 Frames per Second (FPS) . . . 27
3.5.3 Multiple Object Tracking Accuracy (MOTA) . . . 28
3.5.4 Multiple Object Tracking Precision (MOTP) . . . 30

4 Results and Analysis **32**
4.1 Major results . . . 32
4.1.1 FPS Score . . . 32
4.1.2 MOTA Score . . . 33
4.1.3 MOTP Score . . . 33
4.2 Reliability Analysis . . . 33

5 Discussion **35**
5.1 Choosing a confidence score . . . 35
5.2 GT Accuracy and Its Impact on MOTP . . . 35
5.3 The Difference Between a TP and a Match . . . 36
5.4 State Prediction with Choppy Video Segments . . . 36

6 Conclusions and Future work **38**
6.1 Conclusions . . . 38
6.2 Limitations . . . 39
6.3 Future work . . . 39

References **40**

List of Figures

2.1 Digitisation of images through sampling and quantisation [15]. 9
2.2 The architecture of modern object detectors can be described as the backbone, the neck, and the head. 10
2.3 The architecture of R-CNN [21]. 12

3.1 A high level visualisation of how the project group iteratively trained a new model by adjusting the training parameters until a model performed satisfactory. 20
3.2 The final training results from our model. 22
3.3 Precision score as confidence increase. 23
3.4 Recall score as confidence increase. 24
3.5 F1 score as confidence increase. 25
3.6 This figure contains the images used to construct the cost matrix that Table 3.7 represents. The tracker is BoT-SORT. . . 30

5.1 This figure compares a predicted bounding box to its corresponding GT bounding box for the same object in the same frame. Tracker used In 5.1a: BoT-SORT. 36
5.2 This figure illustrates predicted bounding boxes right before, during, and after a choppy set of frames. Tracker used: BoT-SORT. 37

List of Tables

3.1 Characteristics of the Test dataset. 17
3.2 Characteristics of the CarsByDrone dataset. 17
3.3 Characteristics of the CustomDataset dataset. 18
3.4 Definition of key metrics: TP, FP and FN. 22
3.5 Characteristics of the *CarsOnHighway* dataset. 26
3.6 Definition of key metrics: IDS, GT and IoU. 27
3.7 This table consists of data from the cost matrix used to associate cars between frame 4 and frame 5. Frame 4 and 5 are featured in Figure 3.6. 29

4.1 FPS results for BoT-SORT and ByteTrack on the Nvidia V100 GPU through the Google Colab environment. 33
4.2 Tracking results for both BoT-SORT and ByteTrack on the MOTA benchmark which includes the MOTA score and additional metrics. 33
4.3 Bounding box accuracy results for both BoT-SORT and ByteTrack on the MOTP benchmark which includes the MOTP score and additional metrics. 33

List of acronyms and abbreviations

AI	Artificial Intelligence
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GT	Ground Truth
IDS	Identity Switch
IoU	Intersection over Union
KF	Kalman Filter
MOT	Multi-Object Tracking
MOTA	Multi-Object Tracking Accuracy
MOTP	Multi-Object Tracking Precision
R-CNN	Region-based Convolutional Neural Network
TP	True Positive
YOLO	You Only Look Once
YOLOv8	You Only Look Once version 8

Chapter 1

Introduction

In recent years, the field of computer vision has witnessed remarkable advancements, particularly in object detection and object tracking. These technologies play a pivotal role in various applications, including surveillance, autonomous vehicles, and robotics [1]. The demand for accurate and fast real-time object detection and object tracking has spurred the development of innovative models, such as the You Only Look Once (YOLO) series [2], known for its efficiency and speed. Concurrently, the integration of sophisticated tracking algorithms, exemplified by BoT-SORT [3] and ByteTrack [4], two Multi-Object Tracking (MOT) algorithms, has significantly enhanced the field's capabilities.

1.1 Problem

Object tracking through Artificial Intelligence (AI) finds applications in numerous areas. These use cases span diverse contexts, where a one size fits all tracking method may not be optimal. This study aims to enhance our understanding of when either BoT-SORT or ByteTrack should be chosen over the other. The focus is on highway traffic scenario, characterised by the frequent appearance of new objects, such as vehicles, and occlusion.

1.1.1 Original problem and definition

Given the scene investigated, which tracking algorithm, BoT-SORT or ByteTrack, will ensure the best overall performance in terms of accuracy and computational efficiency?

1.1.2 Scientific and engineering issues

The scientific and engineering issues with the project starts with training a model to perform well on object detection. The second part is creating the test scene and benchmarks to measure performance metrics, the challenge will be to understand object tracking in order to develop sophisticated benchmarks that measure relevant metrics. The third part is to analyse the measured data and compare the trackers' accuracy and computational efficiency. This data is important because we will use it to present a result and draw conclusions.

1.2 Purpose

The purpose of this thesis project is to conduct an evaluation of the object trackers BoT-SORT and ByteTrack in conjunction with You Only Look Once version 8 (YOLOv8). The following aspects will also be addressed:

- Investigate the trade-offs between tracking accuracy and computational efficiency, providing insights into the strengths and limitations of each tracker.
- Assess the real-world applicability of each tracker, considering scenarios such as occlusion and varying object speeds.
- Gain a foot in the field of AI, specifically computer vision.

1.3 Goals

The goal of this project is to acquire quantitative data in order to conclude in which contexts BoT-SORT and ByteTrack perform better than the other. This goal has been divided into the following sub-goals:

- **Subgoal 1:** Develop a vehicle tracking system with YOLOv8 and BoT-SORT/ByteTrack.
- **Subgoal 2:** Analyse the system's performance.
- **Subgoal 3:** Establish selection criteria for BoT-SORT/ByteTrack.

1.4 Research Methodology

The chosen methodology and method for the project is comparative analysis, experimentation, data collection, and performance evaluation metrics. These methods are suitable for the nature of the study and the need for quantitative data and qualitative insights.

The comparative analysis methodology will allow for a systematic and structured examination of the performance of BoT-SORT and ByteTrack with YOLOv8 as the object detection model. This methodology enables a side-by-side comparison. It will also provide a clear framework for evaluating the strengths and weaknesses for each approach, leading to a deeper understanding of their respective performance.

The method of experimentation involves implementing YOLOv8 with the BoT-SORT and ByteTrack algorithm. This method allows for the observation of their behaviour under different scenarios. The benefit here is a collection of firsthand data on the performance of the applications leading to a realistic evaluation of their effectiveness in practical scenarios.

There are other methodologies to look at, such as a qualitative/survey-based approach. However, these methodologies are more suitable for gathering subjective opinions or insights from users or experts. In our case, the quantitative and performance metrics were more important and suitable.

1.5 Delimitations

The project specifically focuses on comparing the performance of YOLOv8 with BoT-SORT and ByteTrack for object tracking. Other algorithms and models beyond what is mentioned above are excluded from this project. The dataset that we used for our project is originally a video that we processed ourselves into a dataset. The findings will be closely intertwined with this dataset, meaning the generalisation of results to all possible datasets is considered beyond the scope for this project. The study will not include an in-depth exploration of the algorithm fine-tuning.

1.6 Benefits, Ethics and Sustainability

Machine learning has potential for enhancing various aspects of society in a wide variety of contexts. Most often, these are scenarios that require automation and the ability to make decisions, such as traffic tracking.

However, as articulated in a paper by Azer Hojlas and August Paulsrud [5], one critical challenge from a sustainability point of view is that machine learning is computationally expensive. Another challenge is mitigating implicit biases that can affect the machine learning model in form of bias data or bias decision prior to its training.

1.7 Structure of the thesis

Chapter 2 presents relevant background information on object detection and object tracking. Followed by background on two state of the art object detection models, YOLO and Region-based Convolutional Neural Network (R-CNN). This chapter also includes a subsection about BoT-SORT and ByteTrack. The chapter is concluded with some work related to our paper. Chapter 3 describes the steps taken in order to analyse the algorithms and presents the methodology and method used to solve the problem. We present the results in Chapter 4. Finally, Chapter 5 includes the discussion and potential future works.

Chapter 2

Background

This chapter provides the necessary theoretical information needed to understand the context of our project. It lifts key areas starting from machine learning as a field to object detection models and tracking algorithms.

2.1 Machine Learning

Machine learning is a subset of AI which can be simply defined as the capability of a machine to imitate intelligent human behaviour [6]. This capability is achieved by developing algorithms and models that enable the computer to make predictions or acquire knowledge from patterns, without being programmed to do so.

The steps to machine learning start with the data [7], such as images or texts. The training data is what the model will be looking at to get educated. The computer model then undergoes a training process, autonomously discerning patterns and formulating predictions. As the model evolves, human programmers can iteratively refine it by adjusting parameters to enhance accuracy. To assess the model's effectiveness when exposed to new data, some data is reserved from the training set and employed as evaluation data. The outcome is a refined model capable of making predictions in diverse scenarios with distinct datasets.

2.1.1 Methods of Machine Learning

There are many machine learning methods and we will mention and explain a couple that affect our project. The machine learning methods

are generally categorised as supervised machine learning and unsupervised machine learning.

2.1.1.1 Supervised Machine Learning

Supervised machine learning is a way of training the model with labelled datasets [8]. This method provides the algorithm with input-output pairs, where the inputs are features or attributes, and the corresponding outputs are the correct labels or target values. The primary goal of supervised learning is for the model to learn the mapping from inputs to outputs, enabling it to make accurate predictions on new, unseen data, so if we have a large data to train on that would make it more robust on unseen data [9].

Supervised learning can be divided into two main types: classification and regression [10]. In classification, the model predicts a category, while in regression, it predicts a continuous value. Commonly used supervised learning algorithms encompass linear regression, logistic regression, decision trees, random forests, and support vector machines. Supervised machine learning has many advantages. One of the main advantages is that it allows for high predictive accuracy when trained on high-quality, representative data. Supervised learning also allows for explicit feedback on the model's predictions, which is valuable for model training and improvement. The disadvantages of Supervised learning are, one, it is very labour and time consuming to manually label the dataset. Second, there is a greater possibility of having a faulty model due to human error when labelling the data set [8].

2.1.1.2 Unsupervised Machine Learning

Unsupervised Machine Learning employs algorithms to analyse and cluster datasets that lack labels [11]. In contrast to supervised learning, unsupervised learning identifies patterns, relationships, or structures within the data independently, without the need for labelled output to guide the process.

Unsupervised learning has three types of tasks: clustering, association rules, and dimensionality reduction. Clustering algorithms group similar data points together based on intrinsic similarities or differences. Clustering algorithms organise similar data points into groups based on inherent similarities or differences. These algorithms process raw, unclassified data objects, forming groups that reveal underlying structures or patterns within the data [12]. Unsupervised learning algorithms that are used for clustering, which include exclusive, overlapping, hierarchical, and probabilistic.

- **Exclusive clustering:** Exclusive clustering is a grouping method that dictates a singular assignment for each data point, meaning that a data point can belong exclusively to one cluster. This concept is often termed ”hard” clustering. The K-means clustering algorithm serves as an illustration of exclusive clustering where data is defined into specified k number of clusters [11].

- **Overlapping clustering:** Data is organised in a manner that allows a single data point to be part of two or more clusters, each with varying degrees of membership. This approach is commonly known as ”soft” clustering.

- **Hierarchical clustering:** involves dividing data into distinct clusters based on their similarities. These clusters are then progressively merged and organized to reflect their hierarchical relationships.

- **Probabilistic clustering:** involves grouping data points based on the probability that they fit within a specific distribution.

Association rule is a type of unsupervised learning that aims to discover interesting relationships between data points in large datasets. It works by using a rule-based approach such as if-then to identify strong rules within a dataset. It is commonly used in retail such as “People who bought this item also bought” [12].

Dimensionality reduction is an unsupervised learning method that decreases the number of features in a dataset. By eliminating unnecessary or random features, it simplifies visualisation and processing.

Although unsupervised learning offers several benefits, it also presents certain challenges when machine learning models function without human oversight. These challenges include[11]:

- Increased computational complexity due to large volumes of training data.

- Prolonged training times.

- Elevated risk of producing inaccurate results.

- Necessity for human intervention to verify output variables.

- Lack of transparency regarding the criteria used for data clustering.

2.2 Computer Vision

Computer vision is a dynamic field within AI, that empowers machines to interpret and comprehend visual information, mirroring the human ability to perceive and comprehend the visual world [13]. Just as we humans have our vision to see and interpret what we see, applications of computer vision leverage input from sensors, artificial intelligence, machine learning, and deep learning to emulate the functionality of the human vision system. Using algorithms trained extensively with visual data stored in the cloud, these applications identify patterns within the data, employing this knowledge to discern and interpret the content of other images. There are many computer-vision-based tasks; we will look at a couple that are relevant for our thesis.

In computer vision, object detection refers to the model's ability to identifying objects in images. Object tracking, on the other hand, is the ability to associate objects between frames.

2.2.1 Object Detection

Object detection is a computer vision task designed to find specific objects within images or videos [14]. Employing machine learning or deep learning, object detection algorithms look to replicate the human ability to swiftly recognize and locate objects of interest when we observe images or video content. The overarching objective is to replicate computers with the capability to emulate this human intelligence in identifying and locating objects.

In computer vision, images are like smooth shapes on a flat surface, as a graph with x and y coordinates (f(x,y)). When we convert these images into computer-friendly versions, we do two main things: sampling and quantisation. These processes turn the smooth shapes into a grid of tiny dots or pixels. This grid helps computers break down images into different sections based on how things look and how close the dots are to each other [15], see Figure 2.1.

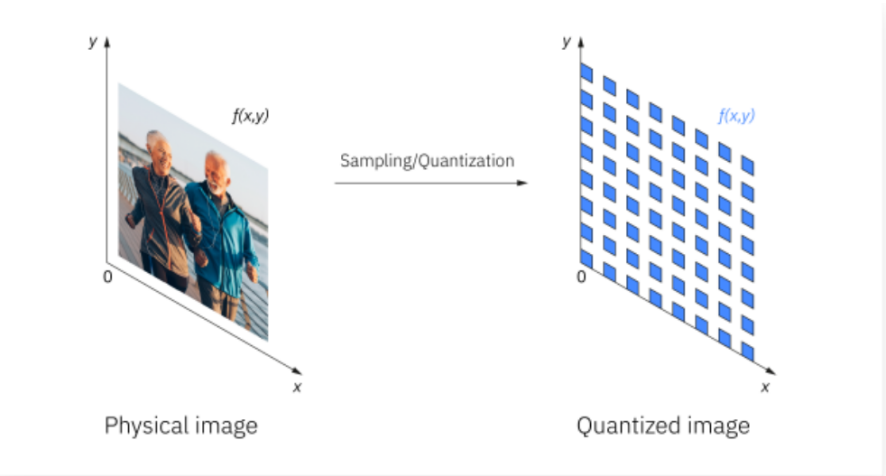


Figure 2.1: Digitisation of images through sampling and quantisation [15].

When users label images with annotations marking regions with certain pixel-level features. This annotation process provides ground-truth information for training the object detection model. So when the model is given new input, it recognises areas of similar features to the training dataset and deems them as same object. Object detection is a form of pattern recognition where the model do not identify objects directly but draws connection from manually annotated training data [15].

Modern object detection models' architectural design has parts known as the body, the neck and the head [15], see Figure 2.2.

- The backbone is responsible for extracting useful features from the input image. The backbone captures hierarchical features at multiple scales: the early layers extract lower-level features like edges and textures, while the deeper layers extract higher-level features such as object parts and semantic information.. The backbone network processes the entire input image and generates a feature map which is then [16].

- The neck is an middle component that links the backbone to the head. It operates on the feature map produced by the backbone, aggregating and refining the features [16].

- The head is the last element of an object detector; responsible for predicting bounding boxes and class probabilities for objects within the input image [16].

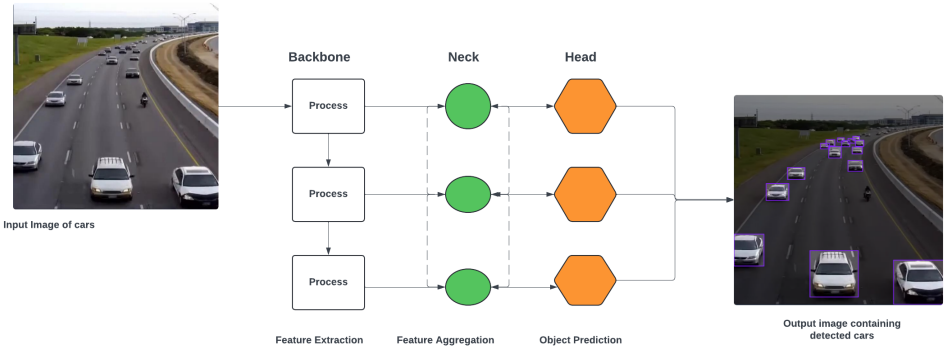


Figure 2.2: The architecture of modern object detectors can be described as the backbone, the neck, and the head.

2.2.2 Object Tracking

Object tracking is a fundamental aspect of computer vision, which focuses on monitoring and tracing the movement of objects across consecutive frames in a video. Differing from object detection, which recognizes objects in a single frame, object tracking ensures continuity by linking identified objects over time. This task involves the initial identification of objects, assigning a unique ID to each detection, and subsequently tracking these objects as they move across frames in a video, preserving the assigned IDs throughout [17]. Object tracking steps according to [18]:

Step 1 Setting Up the Target: First, we determine how many targets we are dealing with and what we're interested in tracking. We mark the object of interest by drawing a box around it. This usually happens in the first image of a sequence or the first frame of a video.

Next, we need to predict where the object will be in the following frames while still keeping an eye on it. This can be done either by humans, manually marking the object's position with boxes or ellipses, or automatically using object detectors.

Step 2 Understanding How It Looks: We need to understand how our target looks visually. Sometimes, the appearance of the object can change in different situations like lighting or angles. Thus, we need to model these changes to keep track accurately.

Step 3 Estimating Movement: Once we know how the shape and location of

the object, we estimate how it will move. We can do this using methods like linear regression, Kalman Filter (KF), or particle filters.

Step 4 Pinpointing the Target: After estimating the motion, we can narrow down the potential location of the object. Then, we use a visual model to find its exact location. This can be done through a search or estimation based on the predicted motion.

2.3 Machine learning Models

In this section we present two object detection models, YOLO and R-CNN.

2.3.1 You Only Look Once (YOLO)

YOLO, short for You Only Look Once, was developed by Joseph Redmon and Ali Farhadi in 2015. It introduced a real-time, end-to-end approach for object detection for the first time. [16]. The YOLO family model is constantly evolving. Several research teams have since released different versions of YOLO, YOLOv8 being the latest iteration by Ultralytics. YOLO variants are underpinned by the principle of high classification and real-time performance, based on limited but efficient computational parameters.[19].

YOLO models stand out because they are single stage object detectors. Previous object detection models used methods that rely on a two-stage process, where selective search generates region proposals, and convolutional neural networks classify and refine these regions[16]. However, YOLO takes a different approach by combining region proposal and object classification into a single step. With this approach, YOLO directly predicts bounding boxes and class probabilities for objects in a single pass through the neural network, without the need for separate region proposal and classification stages [20].

In the YOLO system, the input image is divided into an S×S grid. If the center of an object falls within a grid cell, that cell is tasked with detecting the object. Within each grid cell, the model predicts B bounding boxes along with confidence scores for those boxes. These confidence scores indicate the model's certainty regarding the presence of an object within the box and the accuracy of the box prediction [20]. The YOLO family of object detection models has always aimed to balance speed and accuracy, ensuring real-time performance without sacrificing detection quality. Across its versions, YOLO has consistently optimized this trade-off in different ways. In the original YOLO model, the focus was on high-speed detection achieved by using a

single convolutional neural network (CNN) to predict object locations and classes directly from input images. However, prioritizing speed sometimes resulted in less accurate detections, especially for small or overlapping objects [16].

2.3.2 Region-based Convolutional Neural Network (R-CNN)

Region-based Convolutional Neural Network (R-CNN) is a type of deep learning architecture used for object detection in computer vision tasks. The R-CNN architecture utilises the selective search algorithm to generate approximately 2000 region proposals from the input image. These proposals are then processed by a Convolutional Neural Network (CNN) to extract CNN features. Subsequently, the extracted features are fed into two separate components: a Support Cector Machine (SVM) for object classification and a linear regressor for refining the bounding boxes of detected objects. The SVM classifies the regions to determine the presence of objects in the image, while the linear regressor adjusts the bounding boxes to improve localization accuracy. This multi-step approach allows R-CNN to accurately detect and localize objects within the input image seen in Figure 2.3.

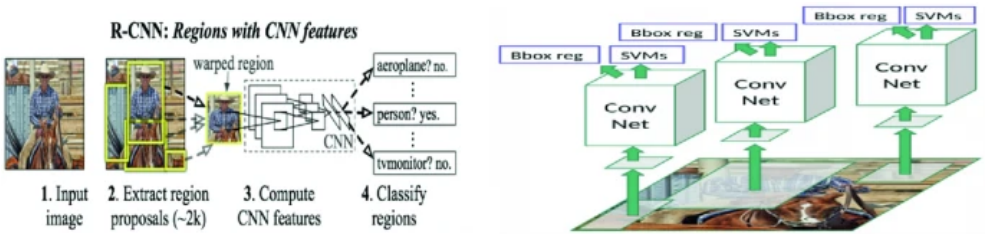


Figure 2.3: The architecture of R-CNN [21].

2.4 Object Tracking Algorithms

Multi-object tracking (MOT) aims to detect and estimate the spatial-temporal trajectories of multiple objects in video streams [3]. In this section, we will introduce two algorithms that implement MOT. One is BoT-SORT and the other is ByteTrack.

2.4.1 BoT-SORT

BoT-SORT (Bidirectional One-Shot Temporal Recurrent Neural Network with Siamese Network for Multiple Object Tracking) represents a significant advancement in the field of Multi-Object Tracking (MOT); MOT is crucial for various applications such as autonomous driving and video surveillance [3].

Over the years, MOT tasks have predominantly followed the tracking-by-detection paradigm. In this approach, object detection precedes object tracking, where an object detector identifies objects in each frame, and these detections are then passed to the tracking method. The tracking method is responsible for associating objects between consecutive frames. Traditionally, many methods have employed the KF as a motion model to predict the position of objects in the current frame [22].

However, with the advent of Deep Learning-based Neural Networks (DNNs), there has been a shift towards more advanced methods in object vision tasks, including tracking. Newer approaches leverage DNNs for object classification, recognition, and tracking tasks. To enhance the object association step in tracking algorithms, CNNs are utilised to extract object appearance features. These features are then used to compute similarity values between the feature maps of two objects detected in consecutive frames. Additionally, CNNs are employed to localize objects in consecutive images, further improving the accuracy of object tracking. However, existing approaches such as SORT and DeepSORT have limitations, particularly in terms of bounding-box accuracy and handling complex scenarios like camera motion [22].

BoT-SORT addresses these limitations by introducing several key innovations. First, it enhances the motion model by incorporating camera motion compensation (CMC) techniques to improve bounding-box localisation. This compensates for inaccuracies caused by camera motion, resulting in more accurate predictions. Additionally, BoT-SORT leverages a fusion of Intersection over Union (IoU) and re-identification (ReID) methods for robust data association, balancing between detection accuracy (MOTA) and identity preservation (IDF1) [3].

2.4.2 ByteTrack

ByteTrack, a tracking algorithm rooted in the Tracking by Detection (TBD) paradigm, operates by leveraging object detection techniques to obtain bounding boxes for targets and facilitates trajectory tracking throughout the process. However, challenges arise in scenarios where targets undergo

temporary occlusion, leading to low-scoring detection boxes for corresponding detections. Traditionally, discarding low-scoring detection boxes in TBD algorithms is common practice, potentially resulting in the loss of tracking trajectories for occluded real targets [4].

To combat this challenge, ByteTrack introduces an innovative data association method called BYTE. The primary aim of BYTE is to mitigate trajectory interruptions caused by low-scoring detection boxes and occlusions. This method prioritises establishing trajectories based on high-scoring detection boxes, effectively filtering out background interference by distinguishing genuine targets from low-scoring detection boxes. In doing so, BYTE ensures the coherence of the trajectory while maximising the preservation of the trajectory information for occluded targets [4].

In essence, ByteTrack utilises both high-scoring and low-scoring detection boxes throughout the tracking process. High-scoring detection boxes are instrumental in initiating target trajectories, while low-scoring ones aid in eliminating background interference and maintaining trajectory coherence. Through the integration of information from both categories of detection boxes, ByteTrack significantly enhances the robustness of the target tracking algorithm, minimises target loss, and effectively manages scenarios involving target occlusion [4].

2.5 Related Work Area

The paper [23], introduces TRASE, a sophisticated traffic surveillance system designed for automated tracking and identification of speeding vehicles using drone video streams. TRASE integrates several key modules, including a detection module employing YOLOv8, a tracking module based on BoT-SORT, a custom speed estimation module, and a license plate recognition module utilising YOLOv8 and Tesseract OCR. These modules collectively ensure accurate and real-time monitoring of speeding vehicles. Notably, to optimise computational resources, all components, except for vehicle detection, operate on separate CPU threads.

To determine vehicle displacement over time, TRASE employs a two-step approach involving detection and tracking. YOLOv8 is utilised for precise vehicle bounding box detection, leveraging contextual information and a decoupled head for improved accuracy. For tracking, TRASE evaluates leading algorithms, including DeepSORT, BoT-SORT, and ByteTrack, ultimately selecting BoT-SORT for its superior ability to maintain target association. Overall, TRASE presents a comprehensive solution for traffic

surveillance, offering high accuracy and real-time monitoring capabilities, vital for effective traffic management.

The study [24], introduces a framework for training YOLOv8, an advanced object detection algorithm, to detect and classify potholes in images. The framework involves training YOLOv8 on a diverse pothole image dataset, annotated with bounding boxes around pothole instances. YOLOv8 is chosen for its accuracy and real-time detection capability. The training process follows a two-step procedure, starting with pre-training on a large-scale COCO dataset and then fine-tuning on the pothole image dataset. Various variants of YOLOv8, YOLOv8n, YOLOv8m, YOLOv8l, YOLOv8x are analysed for their performance on the "Pothole detection Image Dataset." Overall, the study provides a comprehensive framework for effectively detecting and classifying potholes in images using YOLOv8. The authors found out that YOLOv8l and YOLOv8x gave better detection results.

The paper [25], presents a novel approach to tracking football players in real time using a combination of the KF and a modified Hungarian Algorithm. Traditional object tracking methods rely heavily on computationally intensive object detection techniques, but this paper focuses on reducing computational complexity while maintaining tracking accuracy. By implementing the modified Hungarian Algorithm and KF, the system achieves efficient motion prediction and object identification, even in challenging scenarios such as occlusion. The results demonstrate successful implementation with a notable 10% identity maintainability rate across 800 sample frames. Additionally, the analysis shows that the system requires approximately 26 KB of memory per frame, which makes it suitable for real-world applications in football player tracking systems. In general, the study highlights the effectiveness of the proposed approach in balancing computational resources and tracking performance for sports applications.

By integrating the methodologies, optimisation techniques, and performance evaluations from these studies, the thesis can build a robust framework for analysing the performance of YOLOv8 combined with BoT-SORT and ByteTrack in vehicle tracking systems. These references also help in understanding the strengths and weaknesses of each approach, contributing to a more comprehensive analysis and evaluation in the context of the thesis objectives.

Chapter 3

Method

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 describes how the training dataset was created and also introduces its characteristics. Section 3.2 explains in what environment the model was trained, key training parameters and lastly the training process. Section 3.3 focuses on evaluating a trained model by discussing bias variance tradeoff and explaining metrics used to evaluate model training. Section 3.4 describes the process of creating a new dataset which will be used for benchmarking the model. This section involves annotation strategies and characteristics of the resulting test dataset. Section 3.5 introduces the metrics used to evaluate the tracking algorithms. This section also explains the benchmarks by providing insights into the implementation.

3.1 Constructing the Training Dataset

This section presents the methodologies employed by the project group in acquiring data and constructing the training dataset that was utilized to train the model. It involves data characteristics, data processing, and a description of the resulting dataset.

3.1.1 Acquire Data

The goal was to train a model to have a good understanding of what a car looks like. When gathering data, two key requirements were considered: the cars had to be in a traffic scenario and the data had to be diverse, meaning cars of different sizes and colours, shot from different angles. The first requirement

has do to with contextual reasoning. YOLOv8 uses the background to learn the spatial contexts in which objects are often situated in, such as within lane markings on the road or in proximity to other objects [26]. The second requirement has to do with the goal for this model; by diversifying the data, the model gains a better understanding of what a car is.

The project group acquired two labeled datasets, ”Test” [27] and ”CarsByDrone” [28], their characteristics can be seen in Table 3.1 and 3.2 respectively. Combined, they fulfilled the project group's data requirements. The test subsets of both datasets were removed.

Dataset Name	Test
Description	Contains several traffic videos segmented into individual frames.
Number of Images	10000
Image Size	640x640 pixels
Train/Validation/Test Split	70%, 15%, 15%
Reference	[27]

Table 3.1: Characteristics of the Test dataset.

Dataset Name	CarsByDrone
Description	Contains several traffic videos segmented into individual frames.
Number of Images	9840
Image Size	540x540 pixels
Train/Validation/Test Split	80%, 10%, 10%
Reference	[28]

Table 3.2: Characteristics of the CarsByDrone dataset.

3.1.2 Processing Acquired Data

Both acquired datasets contain labels of classes other than cars. Two scripts were developed for preprocessing the data. The first iterates through all the label text files of both datasets and removes unwanted class labels. The second script iterates through one of the dataset's label text files and standardises the format of car IDs to match those of the other dataset.

3.1.3 Resulting Dataset

The train/validation split of the resulting dataset ensures that the model gets trained on a large variety of contexts, with a large enough validation set to ensure enough images exist to validating the model's training progress, and in turn detect overfitting. Characteristics of the resulting dataset can be seen in Table 3.3.

Dataset Name	CustomDataset
Description	Contains several traffic videos segmented into individual frames.
Number of Images	17302
Image Size	640x640 pixels, 540x540 pixels
Train/Validation/Test Split	85%, 15%, 0%

Table 3.3: Characteristics of the CustomDataset dataset.

3.2 Training the Model

This section explains how and in what environment the model was trained. Further, training parameters that are deemed especially important by the project group are explained.

3.2.1 Training Environment, Google Colab

Training a model with a large dataset demands a lot of time, the project group therefore purchased access to powerful graphics processing units (GPUs) through Google Colab.

3.2.2 Key Training Parameters

There are five important key training parameters [29] as follows:

- **Epoch:** The number of times the entire dataset is passed through the model during training.
- **Batch:** A subset of the training set which consists of one or more images. Batches are processed during training, and based on the collective information within a batch, the model updates its weights.
- **Initial Learning Rate:** Determines the step size at which the model's weights are allowed to change during the initial stage of training.
- **Final Learning Rate:** Is a fraction of the initial learning rate. Adjusts the learning rate mid training.
- **Patience:** An early stopping parameter with the purpose to hinder overfitting. The number of epochs to wait before aborting training if the validation losses does not improve.

3.2.3 Training Process

Before training starts, the training data is segmented into batches based on the batch size parameter. A smaller batch size can lead to an improved model but costs more computational power because it is required to perform in more epochs. The project group found the default batch size of 16 to be satisfactory in terms of cost and accuracy. During training, the model updates its weights after processing a batch according to the set learning rate, which was left to be its default value. After an epoch, the model's current state's performance is evaluated against unseen data in the validation set. If the model does not improve on the validation set on a set number of epochs (according to what the patience parameter is set to), the training will halt because of the increased risk of overfitting. through multiple rounds of trying different patience numbers, 20 was chosen as it was most optimal. By analysing the training results from multiple training sessions, the project group concluded good parametric values for the training parameters, see Figure 3.1.

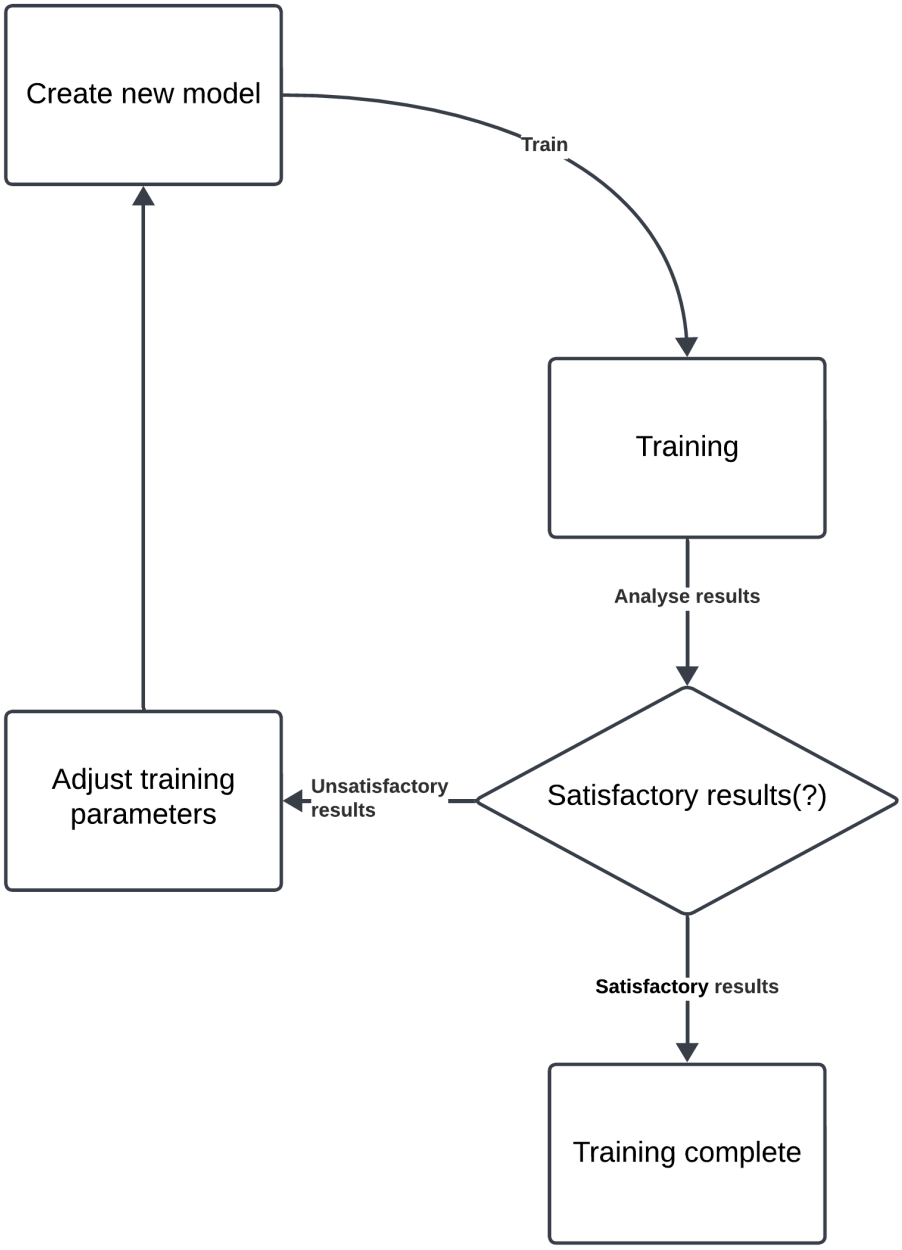


Figure 3.1: A high level visualisation of how the project group iteratively trained a new model by adjusting the training parameters until a model performed satisfactory.

3.3 Evaluating the Trained Model

This section explains training performance results and examines how the model's training is evaluated alongside the training results.

3.3.1 Bias-Variance Tradeoff

Bias variance tradeoff is a delicate balance in training the model to achieve optimal generalization for its specific task. Over training will result in overfitting and cause the model to remember the training data and lose its generalization ability. Under training will result in underfitting and cause the model's generalization ability to be inadequate.

Overfitting occurs when the "model does not generalize well from our training data to unseen data" [30]. It is difficult to detect overfitting by only evaluating the training losses. Because the model at a certain point in training can start to memorize instead of generalize the information in the training set [31]. By including a validation step in the training process, in combination with an early stopping mechanism, the model can halt its training right before it starts to memorize data, thus being in its most generalized form. An overfitted model typically exhibits small training losses, but performs poorly on the validation set resulting in a high validation loss.

Underfitting is the opposite problem to overfitting, and occurs when the model lacks sufficient training to learn the true relationships [32]. An underfitted model exhibits high training and validation losses.

The graphs in Figure 3.2 collectively indicate the model's performance in object detection. These results are good while they suggest a small chance of overfitting. Both the training and validation losses are low while maintaining high scores for precision, recall, mAP50 and mAP50-95. However, both the graphs val/box_loss and val/cls_loss have planed out while the graph metrics/recall(B) have dipped a tiny bit. To determine the model's accuracy a visual test is conducted on unseen data. This unseen data is an unannotated video which we intended to annotate at a later stage and use as a test dataset when benchmarking the model. This visual test proves the model to be accurate and not suffer from overfitting. Previous training with 150 epochs was halted by YOLOv8's built in early stopping mechanism due to our patience parameter being set to 20. This experience in combination with these results we interpret as overfitting warnings, thus preventing us from training the model even more.

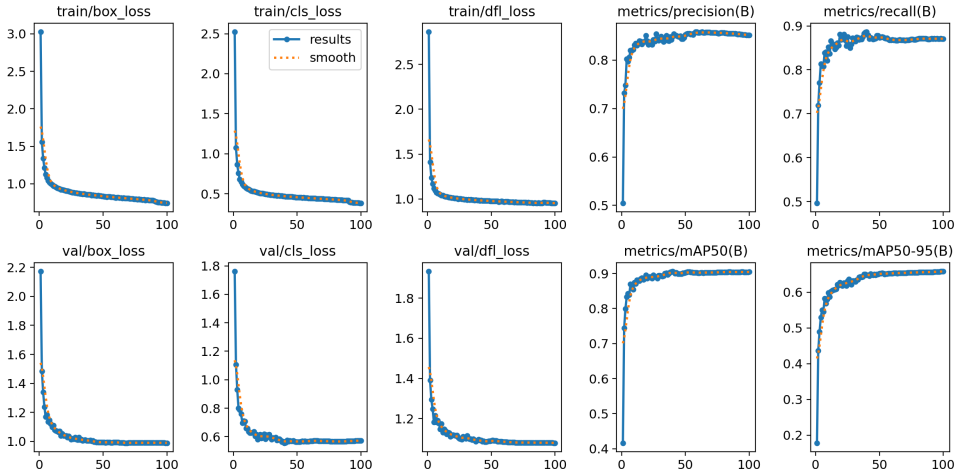


Figure 3.2: The final training results from our model.

3.3.2 Reading Performance Metrics: Finding the Right Balance

The output of a model is composed by one or multiple predictions. Each prediction has been made with a certain confidence, the higher the confidence the more certain is the model that the correct prediction was made. When setting a confidence threshold, the model will only predict objects that it is more confident in than the specified confidence threshold. This subsection will analyse the performance metrics precision and recall and conclude the optimal confidence score for our model. See Table 3.4 for an explanation of the metrics defining this subsections target metrics.

Full Name	Explanation
True Positive (TP)	Predictions correctly made.
False Positive (FP)	Predictions wrongly labelled as positives.
False Negative (FN)	Predictions wrongly labelled as negatives.

Table 3.4: Definition of key metrics: TP, FP and FN.

Precision is a way to measure how accurate the model's predictions are. It answers the question: how many are correct from all the predictions made?

We represent the precision formula as [33]

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.1}$$

Figure 3.3 shows that as the model has to make more confident predictions, the precision score increase. However, the total number of predictions are likely to decrease since uncertain predictions will be left out for predictions made with high confidence rates, explaining the trend upwards. For example, if there are ten cars in an image and the model only detects one of them, the model will receive 100% in precision. Precision on its own does not provide enough information to assess the model.

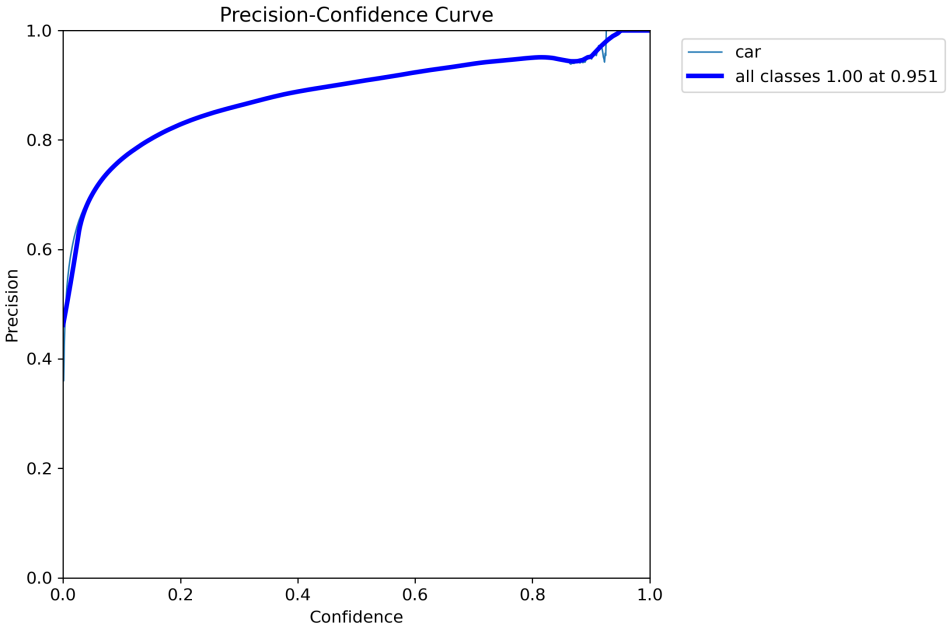


Figure 3.3: Precision score as confidence increase.

Recall is a way to measure how accurately the model finds all positive cases. It answers the question: How many true positive predictions were made out of all existing true positives? We represent the recall formula as [33]

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.2}$$

As the model's confidence rate increases, the model makes fewer predictions; thus, the amount of false negatives increases, resulting in a lower

recall score, as we observe in Figure 3.4. However, recall fails to capture false positive cases which also make this metric insufficient on its own to assess the model.

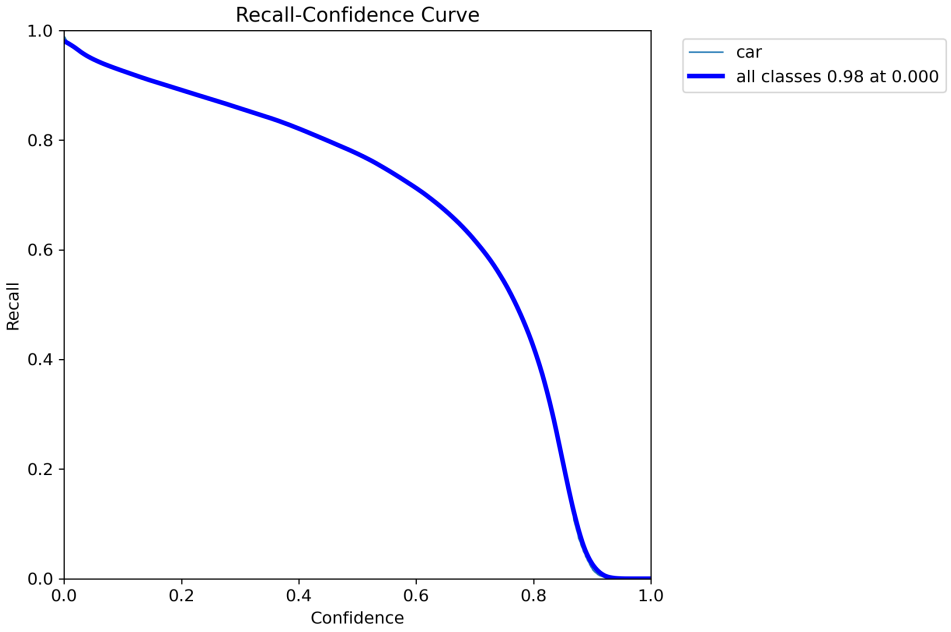


Figure 3.4: Recall score as confidence increase.

We define the F1 score as a combination of precision and recall and aims to find a trade off between the two metrics and ultimately assist in choosing a satisfactory confidence score. However, in some cases, the exact balance between the two metrics might not be desirable. The context in which the model is set to operate within also affects the decision whether the confidence score should prioritise stronger recall or stronger precision. The F1 metric answers the question: How precise and complete are the model's predictions? We represent the F1 formula as

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

In the graph in Figure 3.5, the optimal confidence score is displayed as 0.257. We use this score for benchmarking the model.

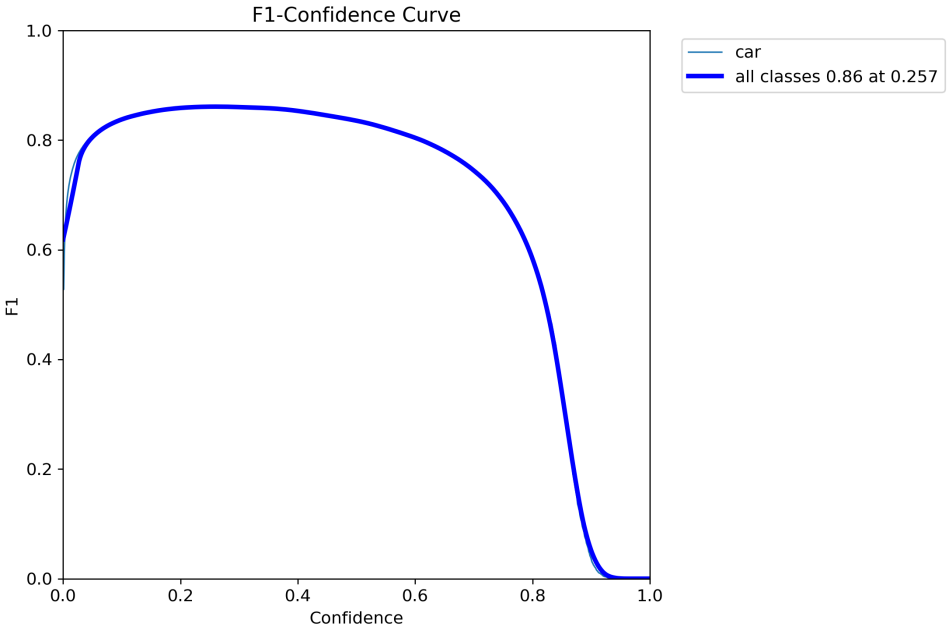


Figure 3.5: F1 score as confidence increase.

3.4 Constructing the Test Dataset

This section addresses the testing of the model on new data. A new dataset is constructed and its features along with the project groups' contribution to the dataset domain are presented.

3.4.1 Gathering and Annotating the Test Dataset

A YouTube video [34] of unidirectional traffic flow on a highway, filmed from a bridge directly above, is cropped to 640x640 pixels, segmented into individual frames, and annotated manually by us, using Roboflow. The same video is used for the visual test conducted under Section 3.3.1.

In order to achieve consistent annotations on our dataset, the first ten images are annotated together. During this time we set up annotation rules cementing our annotation strategy. Once the annotation strategy is setup, the dataset is split equally between us. Before exporting our dataset, we skim through all images to verify similar annotations. Here is a list of our rules:

- Only cars are annotated.

- Cars in the background are ignored.
- The model's effective detection distance is all the way up to the sign furthest away, next to the highway. Only cars within this range are annotated. This range was determined by allowing the model to detect cars prior to annotation. That way we saw how far up the highway the model started to detected cars.
- Cars within the effective detection distance are only annotated if they are easily identifiable by us, meaning a car heavily occluded will not be annotated until it is easily identifiable.
- Once a car has been annotated, it will remain annotated until it completely leaves the frame.

3.4.2 Resulting Test Dataset

Table 3.5 contains properties of our resulting dataset which will be used to benchmark the tracking algorithms.

Dataset Name	CarsOnHighway
Description	Contains one traffic video segmented into individual frames.
Number of Images	767
Image Size	640x640 pixels
Train/Validation/Test Split	0%, 0%, 100%
Reference	[35]

Table 3.5: Characteristics of the *CarsOnHighway* dataset.

3.5 Benchmarking the Model

This section examines the metrics used to benchmark the model. It also provides small implementation insights to highlight key solutions and to help the reader understand the wrong sources that will be discussed in the discussion chapter.

The metrics of Frames Per Second (FPS), Multi-Object Tracking Accuracy (MOTA), and Multi-Object Tracking Precision (MOTP) are used to provide a comprehensive assessment of the model's performance. The metrics in Table 3.4 and 3.6 are used to compute the benchmarks.

Full Name	Explanation
Identity Switch (IDS)	Change in identity assignment for a tracked object.
Ground Truth (GT)	Known correct values.
IoU	Measures the amount of overlap between bounding boxes.

Table 3.6: Definition of key metrics: IDS, GT and IoU.

3.5.1 Benchmark Preparations

We need to verify that we can load images along with their corresponding GT labels, as well as count TP, FP, FN, and IDS. An understanding of the result object returned once the tracking method has been employed is also crucial. Further, debugging tools are needed to confirm that our way of computing all these metrics are correct. Therefore methods that save an image together with its predicted bounding boxes are implemented, alongside other disc management methods in order to be able to store all benchmarks in an organised way. These debugging tools does not affect the benchmarks' performances and will therefore not receive further attention.

3.5.2 Frames per Second (FPS)

This benchmark computes the average minimum FPS, the average FPS, and the average maximum FPS for a given sequence of frames. The result is accomplished by iterating through the sequence of frames eleven times, where the first iteration is a warm up iteration. Each iteration after the warm up, measure the lowest and the highest FPS values obtained for a single frame, and also the average FPS for all frames in the sequence. To compute FPS for a single frame we timed how long it takes for a tracker to process one frame, as

$$\text{FPS} = \frac{1}{\text{end time} - \text{start time}} \tag{3.4}$$

FPS is then evaluated against the current min and max FPS values before being added into an accumulative FPS variable. Once all frames are processed, the accumulative FPS variable is divided by the total number of frames, which computes the average FPS for the sequence of frames in one iteration as

$$\text{Average FPS} = \frac{\sum_i \text{FPS}_i}{N} \tag{3.5}$$

where i is the frame index and N is the total number of frames. Once all this has been done for all iterations, the final average minimum FPS, the

average FPS, and the average maximum FPS are computed for all iterations, as

$$\text{Average Minimum FPS} = \frac{\text{cumulative min fps}}{(\text{iterations} - 1)} \tag{3.6}$$

$$\text{Average FPS} = \frac{\text{cumulative average fps}}{(\text{iterations} - 1)} \tag{3.7}$$

$$\text{Average Maximum FPS} = \frac{\text{cumulative max fps}}{(\text{iterations} - 1)} \tag{3.8}$$

Disclaimer: This benchmark solely focuses on providing an assessment for computational efficiency of a tracker, therefore a tracker's accuracy is neglected in this test.

3.5.3 Multiple Object Tracking Accuracy (MOTA)

The purpose of this MOTA benchmark is to assess our trackers accuracy by considering the factors FP, FN and IDS. Unlike the FPS benchmark, this benchmark is run on our local machine.

The number of FP cases are derived from computing all the TP cases, and subtracting that from all the predictions made. The model only predicts positive cases, therefore all predictions are either TP or FP, as

$$\text{Total Predictions} = \text{TP} + \text{FP} \tag{3.9}$$

which in turn results in this formula for computing FP, as

$$\text{FP} = \text{Total Predictions} - \text{TP} \tag{3.10}$$

This formula assumes that a prediction that is not a TP must be a FP which again is correct since the model only tries to predict true cases.

The number of FN cases are derived from computing all the TP cases, and counting all the GT objects, as

$$\text{FN} = \text{GT} - \text{TP} \tag{3.11}$$

the above formula computes FN with the assumption that GT objects that are not predicted as true cases are falsely missed by the model.

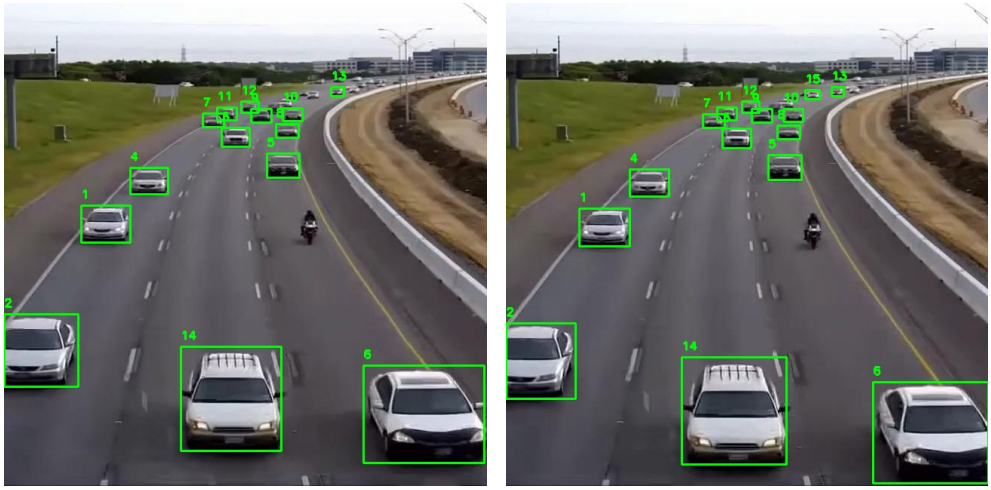
A TP case is detected once a predicted box and its corresponding ground truth box overlaps enough to pass the IoU threshold. In practice we do not know the corresponding ground truth box to any predicted box, therefore, IoU

is computed between a predicted box and all ground truth boxes. The pair with the highest IoU overlaps the most and is tested against the IoU threshold. The IoU threshold is 0.5, meaning that if the boxes overlap by more than 50% they are considered a match. This threshold value is a widely adopted convention in the field of object detection and tracking because it strikes a good balance between precision and recall.

IDS are computed by constructing a cost matrix based on all the IoU values between the predicted boxes from the previous frame and the predicted boxes from the current frame. A cost based on the resulting IoU is computed by subtracting the resulting IoU between boxes with 1. Therefore, a good matching pair of bounding boxes will result in a low cost. Table 3.7 represents a cost matrix, each row is a prediction from the previous frame, and each column is a prediction from the current frame. In this example, the model has detected a car in the current frame that was not detected in the previous frame (fifteenth column), this does not count as an IDS.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0.250	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.000	0.348	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
3	1.000	1.000	0.133	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
4	1.000	1.000	1.000	0.187	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
5	1.000	1.000	1.000	1.000	0.205	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
6	1.000	1.000	1.000	1.000	1.000	0.359	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
7	1.000	1.000	1.000	1.000	1.000	1.000	0.178	1.000	1.000	1.000	0.875	1.000	1.000	1.000	1.000
8	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.154	1.000	1.000	1.000	1.000	1.000	1.000	1.000
9	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.114	1.000	1.000	0.969	1.000	1.000	1.000
10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.161	1.000	1.000	1.000	1.000	1.000
11	1.000	1.000	1.000	1.000	1.000	1.000	0.926	1.000	1.000	1.000	0.143	1.000	1.000	1.000	1.000
12	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.965	1.000	1.000	0.145	1.000	1.000	1.000
13	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.238	1.000	1.000
14	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.277	1.000

Table 3.7: This table consists of data from the cost matrix used to associate cars between frame 4 and frame 5. Frame 4 and 5 are featured in Figure 3.6.



(a) Frame 4 along with its predicted bounding boxes.

(b) Frame 5 along with its predicted bounding boxes.

Figure 3.6: This figure contains the images used to construct the cost matrix that Table 3.7 represents. The tracker is BoT-SORT.

This cost matrix is then traversed using the Hungarian Algorithm [36] in order to match the previous frame boxes with their corresponding current frame boxes. The Hungarian Algorithm is an efficient way of solving the data association problem [37]. In our context, this problem involves finding the optimal assignment between cars from the previous frame, and cars in the current frame, so that there exist at most one assignment between them.

Once the cost matrix has been traversed, the pairs are checked to contain the same object ID, if their ID:s do not match, the benchmark assumes there has been an IDS.

This formula was used for computing MOTA which was acquired from the report ”MOT16: A benchmark for multi-object tracking” [38]:

$$\text{MOTA} = 1 - \left(\frac{\sum_i (\text{FN}_i + \text{FP}_i + \text{IDS}_i)}{\sum_i \text{GT}_i} \right) \tag{3.12}$$

where i is the frame index.

3.5.4 Multiple Object Tracking Precision (MOTP)

The purpose for this benchmark is to assess the average overlap between the predicted bounding boxes and their corresponding ground truth bounding boxes. This assessment considers the IoU between predicted and ground truth

bounding boxes. Additionally, the benchmark takes into account the total number of matches.

Total IoU is a cumulative measure of IoU between all predicted bounding boxes and their corresponding GT bounding boxes. For each prediction we compute IoU with all GT cases and stores the highest resulting IoU into an cumulative IoU variable. If the highest resulting IoU is higher than the IoU threshold (0.5), we increment the total number of matches similar to how we detect a TP case. However, a match and a TP case are similar but not the same, while all TP cases are matches not all matches are TP cases. A match can also be a FP case if the model predicted a car which lacks a GT label. These terms are not to be confused and will be discussed further in the discussion chapter. This formula was used for computing MOTP which was acquired from the report ”MOT16: A benchmark for multi-object tracking” [38].

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{3.13}$$

where c_t is the number of matches in frame t, and $d_{t,i}$ is the IoU of target i with its corresponding GT label.

Chapter 4

Results and Analysis

In this chapter, we present the results and analyse them.

4.1 Major results

BoT-SORT scores higher in MOTA and MOTP compared to ByteTrack which suggest that it is the most accurate and precise tracker, while ByteTrack obtains the highest average FPS score, suggesting it is more efficient in terms of computational cost. Ultimately, these results show that the two trackers have different strengths. BoT-SORT might be better for accurate and precise tracking needs, while ByteTrack might be preferable for applications on less powerful hardware. However, these findings aren't absolute; the specific application context plays a significant role in choosing the right tracker.

4.1.1 FPS Score

Table 4.1 presents our results obtained from our benchmark measuring FPS. The average is derived from eleven runs where the first run is a ”warm up”, meaning no measures are taken the first iteration. The reason for a ”warm up” iteration is because measures from the first iteration deviated from the rest of the measures and they were therefore neglected.

The results clearly indicate that the most efficient tracker is ByteTrack, although its performance fluctuates the most, its average FPS is a lot higher.

Tracker	Average Minimum FPS	Average FPS	Average Maximum FPS	Range of FPS
BotSort	24.8	34.4	41.6	16.8
ByteTrack	40.0	55.3	67.2	27.2

Table 4.1: FPS results for BoT-SORT and ByteTrack on the Nvidia V100 GPU through the Google Colab environment.

4.1.2 MOTA Score

Table 4.2 presents our MOTA score along with other metrics. The data suggest that BoT-SORT is the most accurate tracker with the lowest FP and FN cases.

Tracker	TP	FP	FN	IDS	GT	MOTA
BotSort	8515	359	2470	3	10985	0.7422
ByteTrack	8438	393	2547	2	10985	0.7322

Table 4.2: Tracking results for both BoT-SORT and ByteTrack on the MOTA benchmark which includes the MOTA score and additional metrics.

4.1.3 MOTP Score

Table 4.3 presents our MOTP score along with other metrics. The data suggest similar performance where BoT-SORT obtained the highest score.

Tracker	Total IoU	Total Matches	MOTP
BotSort	7156.4	8515	0.840
ByteTrack	7062.8	8438	0.837

Table 4.3: Bounding box accuracy results for both BoT-SORT and ByteTrack on the MOTP benchmark which includes the MOTP score and additional metrics.

4.2 Reliability Analysis

In order for our benchmark to validate that objects have correct associations between frames, we rely on the Hungarian Algorithm and the IoU between the same object, in consecutive frames. If a car's position drastically changes between frames, so that there's no overlap between its previous-and current bounding boxes, our benchmark would not be able to validate that the frame-to-frame association is correct. Fortunately, we have not observed such a case. However, this implies that our method of associating objects across

consecutive frames is an implementation that might not be applicable across all datasets.

It's hard to measure a consistent frames-per-second (FPS) rate for a tracking algorithm because various background processes on the machine can affect performance, even if we use the same data and hardware for benchmarking. As a solution , we measure the average FPS values and have a *warm up* iteration. The more FPS values that contribute to the average the more robust the result gets, but a limiting factor discussed In 6.2 has an affect on how much data we could base our average FPS on.

Chapter 5

Discussion

5.1 Choosing a confidence score

The confidence score that yields the highest F1 score was chosen so that the model would not have any bias towards precision or recall. The chosen confidence score ensures that the model will perform the best in both of these metrics. There is no reason to choose a bias confidence score in this context, in fact, a bias confidence score towards precision or recall can penalise the model when evaluated against our benchmarks. A bias towards precision risks increasing the number of FN cases in a pursue to minimise FP cases. Similarly, a bias towards recall risks increasing the number of FP cases in a pursue to minimize FN cases. In other contexts, such as in early threat detection applications, a bias towards recall might be beneficial since a false alarm might be preferred over no alarm. But in this context, we want to evaluate the trackers' performance so there is no reason for a bias in the confidence score.

5.2 GT Accuracy and Its Impact on MOTP

The labelling of our test dataset, *CarsOnHighway*, impact the MOTP score due to the low odds of the model's predictions to be exactly like the GT labels. Figure 5.1 shows that the predicted bounding box is tighter than the GT bounding box. The difference in the bounding boxes applies for many predicted bounding boxes. Although we consider both bounding boxes to be good, the difference does affect the MOTP score negatively. The solution would be to re annotate all images, however, this wrong source introduces marginal error and it is therefore not worth the time to re annotate the test

dataset.

(a) Car with a predicted bounding box. (b) Car with the GT bounding box.

Figure 5.1: This figure compares a predicted bounding box to its corresponding GT bounding box for the same object in the same frame. Tracker used In 5.1a: BoT-SORT.

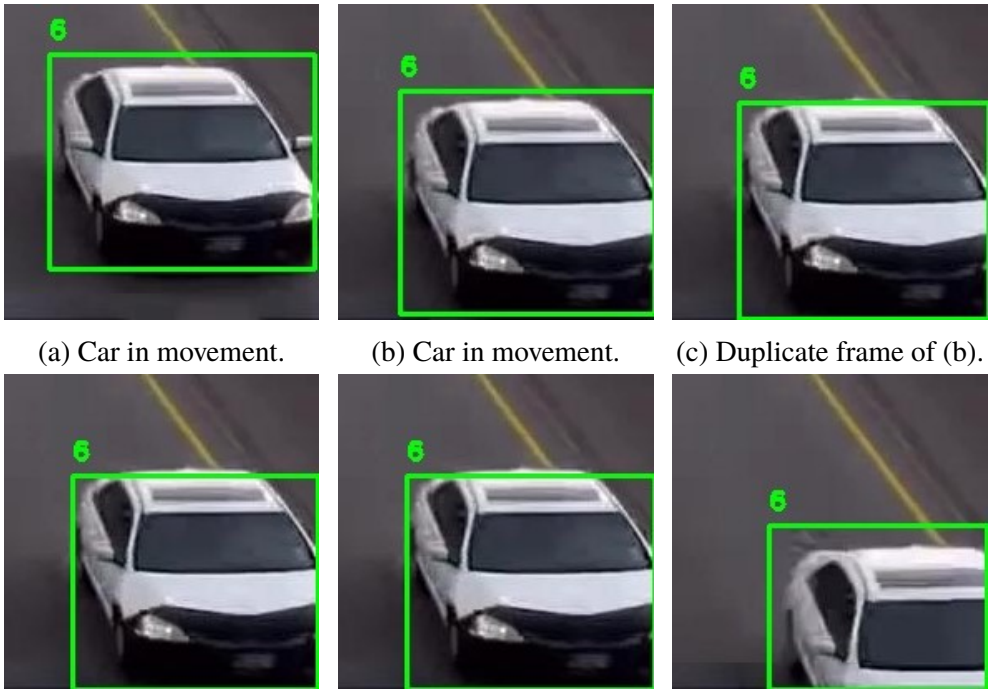
5.3 The Difference Between a TP and a Match

A TP case emphasizes that the detection itself is correct, while a correct match emphasizes that the association itself between a prediction and its corresponding GT is correct. MOTP ”gives the average overlap between all correctly matched hypotheses” [38] where a correctly matched hypotheses is a TP case. Therefore, we calculate matches in the same way we compute TP cases, namely we compare the IoU between predicted-and GT-boxes.

5.4 State Prediction with Choppy Video Segments

The investigated tracking algorithms, BoT-SORT and ByteTrack, both utilise the KF [39], the problem is that object movement is hard to predict due to the source video’s choppy property. The KF predicts object movement based on linear velocities [3], and the source video used in the final benchmark is

choppy, meaning it frequently pauses. The frequent pauses are caused by duplicate frames or frames that are so identical that they are almost duplicates. The choppy property of the video, is inherited by the *CarsOnHighway* dataset and in turn makes it more difficult for the tracking algorithms to predict object movement. This phenomenon can be observed in Figure 5.2. Once the video pauses, the predicted bounding boxes lurk forward.



(a) Car in movement. (b) Car in movement. (c) Duplicate frame of (b).

(d) Duplicate frame of (b). (e) Duplicate frame of (b). (f) Car in movement.

Figure 5.2: This figure illustrates predicted bounding boxes right before, during, and after a choppy set of frames. Tracker used: BoT-SORT.

However, other sources that inherit similar properties to this source video were difficult to find. This fact combined with the marginal error introduced by the choppiness, leads us to the decision to continue using the source video.

Chapter 6

Conclusions and Future work

In this chapter, we present the conclusion of our project, limitation and what future work can be done by building on our project.

6.1 Conclusions

In conclusion, the results indicate that BoT-SORT exhibits the most accuracy, as evidenced by its higher MOTA and MOTP scores, whereas ByteTrack excels in computational efficiency, indicated by its higher FPS. However, a note to be made is that BoT-SORT is more stable in its computational performance, this is implied by its lower range of FPS.

These results concur with the results presented in the related work mentioned above [23]. Even though that related work achieved higher accuracy, the trend is similar, namely, BoT-SORT is more accurate.

Therefore, it is evident that BoT-SORT may be preferred in applications where accuracy is a high priority and hardware efficiency is a lower priority. On the contrary, ByteTrack might be preferred in applications where computational efficiency is a priority.

”Addressing the thesis problem, it is challenging to determine the best overall tracking algorithm for this context due to the subjective nature of prioritizing accuracy versus computational efficiency. In conclusion, the best overall tracking algorithm for this context hinges on whether accuracy or computational efficiency is prioritised.

Hereby, the goals listed under Section 1.3 have been fulfilled, and thereby the goal for this thesis project.

6.2 Limitations

Google Colab was used for training the object detection model and was the environment for the FPS benchmark, since it granted access to powerful hardware. We invested in a subscription plan to gain access to even more powerful hardware on Google Colab. Unfortunately, the plan only granted us access to more powerful hardware when it was available.Additionally, our calculations incurred costs, and our plan included a limited balance, which restricted how many times we could retrain a model and how many iterations the FPS scores could be based on. A low balance risked sudden interruptions during runtime. Our balance did run out once, we had to wait for the next billing to get our balance refilled. This limitation forced us to be more cautious and efficient during our runtime in this environment. We drew the conclusion that with the chosen subscription, there was little to no room for trial and error in the Google Colab environment.

The time limitation of this thesis project narrowed its goals. With more time, tracking algorithms that are not built into YOLOv8 could have been implemented and then benchmarked. Other contexts, for instance datasets which are not characterised by frequently appearing objects, could be tested and new benchmarks measuring relevant metrics for those scenarios would then be needed. These changes would result in a broader comparison.

6.3 Future work

In particular the wrong sources discussed in the Sections 5.2 and 5.4, should be addressed, one suggestion is to choose a different dataset. Also, this thesis could be broadened to include more tracking algorithms, analysed on more datasets, with more benchmarks measuring other relevant metrics for those contexts.

References

[1] "Object tracking with computer vision – types and business use cases," Apr. 2022. [Online]. Available: https://mindtitan.com/resources/blog/object-tracking-with-computer-vision/ [Page 1.]

[2] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023. [Page 1.]

[3] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022. [Pages 1, 12, 13, and 36.]

[4] B. Sheng, Q. Sun, Y. Sun, Z. Hua, Y. Zhang, and J. Tao, "A study on improved yolov7-pose and bytetrack for dual person tracking," in *2023 29th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2023. doi: 10.1109/M2VIP58386.2023.10413427 pp. 1–6. [Pages 1 and 14.]

[5] A. Hojlas and A. Paulsrud, "Predicting future purchases with matrix factorization," 2022, backup Publisher: KTH, School of Electrical Engineering and Computer Science (EECS) Issue: 2022:663 Pages: 46 Series: TRITA-EECS-EX. [Page 4.]

[6] P. Wittek, "2 - machine learning," in *Quantum Machine Learning*, P. Wittek, Ed. Boston: Academic Press, 2014, pp. 11–24. ISBN 978-0-12-800953-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128009536000025 [Page 5.]

[7] "Machine learning, explained," Apr. 2021. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained [Page 5.]

[8] "What is supervised learning?" [Online]. Available: https://www.ibm.com/topics/supervised-learning [Page 6.]

[9] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, "Supervised machine learning," *Department of Information Technology, Uppsala University: Uppsala, Sweden*, p. 112, 2019. [Page 6.]

[10] V. Nasteski, "An overview of the supervised machine learning methods," *Horizons. b*, vol. 4, pp. 51–62, 2017. [Page 6.]

[11] "What is unsupervised learning?" [Online]. Available: https://www.ibm.com/topics/unsupervised-learning [Pages 6 and 7.]

[12] "What is unsupervised learning?" [Online]. Available: https://cloud.google.com/discover/what-is-unsupervised-learning [Pages 6 and 7.]

[13] "What is computer vison?" [Online]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-computer-vision#object-tracking [Page 8.]

[14] "What is object detection?" [Online]. Available: https://www.mathworks.com/help/vision/object-detection.html [Page 8.]

[15] E. K. Jacob Murel Ph.D., "What is object detection?" 2024. [Online]. Available: https://www.ibm.com/topics/object-detection#citation2 [Pages vii, 8, and 9.]

[16] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023. doi: 10.3390/make5040083. [Online]. Available: https://www.mdpi.com/2504-4990/5/4/83 [Pages 9, 11, and 12.]

[17] "Object tracking." [Online]. Available: https://paperswithcode.com/task/object-tracking [Page 10.]

[18] "The complete guide to object tracking." [Online]. Available: https://encord.com/blog/object-tracking-guide/ [Page 10.]

[19] M. Hussain, "Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection,"

Machines, vol. 11, no. 7, 2023. [Online]. Available: https://www.mdpi.com/2075-1702/11/7/677 [Page 11.]

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109/CVPR.2016.91 pp. 779–788. [Page 11.]

[21] P. Bharati and A. Pramanik, "Deep learning techniques—r-cnn to mask r-cnn: A survey," in *Computational Intelligence in Pattern Recognition*, A. K. Das, J. Nayak, B. Naik, S. K. Pati, and D. Pelusi, Eds. Singapore: Springer Singapore, 2020. ISBN 978-981-13-9042-5 pp. 657–668. [Pages vii and 12.]

[22] R. Pereira, G. Carvalho, L. Garrote, and U. J. Nunes, "Sort and deep-sort based multi-object tracking for mobile robotics: Evaluation with new data association metrics," *Applied Sciences*, vol. 12, no. 3, 2022. doi: 10.3390/app12031319. [Online]. Available: https://www.mdpi.com/2076-3417/12/3/1319 [Page 13.]

[23] P. M. Nguyen and K. Hieu Ngo, "Trase: A traffic surveillance system for adaptive speed estimation of vehicles from aerial videos," in *2023 RIVF International Conference on Computing and Communication Technologies (RIVF)*, 2023. doi: 10.1109/RIVF60135.2023.10471858 pp. 533–538. [Pages 14 and 38.]

[24] S. Kumari, A. Gautam, S. Basak, and N. Saxena, "Yolov8 based deep learning method for potholes detection," in *2023 IEEE International Conference on Computer Vision and Machine Intelligence (CVMI)*, 2023. doi: 10.1109/CVMI59935.2023.10465038 pp. 1–6. [Page 15.]

[25] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," in *2016 6th International Conference on System Engineering and Technology (ICSET)*, 2016. doi: 10.1109/ICSEngT.2016.7849633 pp. 109–115. [Page 15.]

[26] L. V. Haar, T. Elvira, L. Newcomb, and O. Ochoa, "Measuring the impact of scene level objects on object detection: Towards quantitative explanations of detection decisions," *arXiv preprint arXiv:2401.10790*, 2024. [Page 17.]

[27] enpo, "Test dataset," https://universe.roboflow.com/enpo-5csjs/test-mxlfe, aug 2023, visited on 2024-03-05. [Online]. Available: https://universe.roboflow.com/enpo-5csjs/test-mxlfe [Page 17.]

[28] Doc, "Carsbydrone dataset," https://universe.roboflow.com/doc/carsbydrone, apr 2023, visited on 2024-03-05. [Online]. Available: https://universe.roboflow.com/doc/carsbydrone [Page 17.]

[29] Ultralytics, "Configuration." [Online]. Available: https://docs.ultralytics.com/usage/cfg [Page 18.]

[30] "Overfitting in machine learning: What it is and how to prevent it," https://elitedatascience.com/overfitting-in-machine-learning, July 2022, accessed: 2024-03-11. [Online]. Available: https://elitedatascience.com/overfitting-in-machine-learning [Page 21.]

[31] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022. [Page 21.]

[32] D. Bashir, G. D. Montañez, S. Sehra, P. S. Segura, and J. Lauw, "An information-theoretic perspective on overfitting and underfitting," in *AI 2020: Advances in Artificial Intelligence: 33rd Australasian Joint Conference, AI 2020, Canberra, ACT, Australia, November 29–30, 2020, Proceedings 33*. Springer, 2020, pp. 347–358. [Page 21.]

[33] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240. [Page 23.]

[34] Supercircuits, "Alibi ALI-IPU3030RV IP Camera Highway Surveillance," Aug. 2014. [Online]. Available: https://www.youtube.com/watch?v=PJ5xXXcfuTc [Page 25.]

[35] carsonhighway, "Carsonhighway dataset," https://universe.roboflow.com/carsonhighway/carsonhighway, mar 2024, visited on 2024-03-08. [Online]. Available: https://universe.roboflow.com/carsonhighway/carsonhighway [Page 26.]

[36] A. Arias, L. H. Martínez, R. A. Hincapie, M. Granada *et al.*, "An ieee xplore database literature review regarding the interaction between electric vehicles and power grids," *2015 IEEE PES Innovative Smart*

Grid Technologies Latin America (ISGT LATAM), pp. 673–678, 2015. [Page 30.]

[37] N. Arun Kumar, R. Laxmanan, S. Ram Kumar, V. Srinidh, and R. Ramanathan, "Performance study of multi-target tracking using kalman filter and hungarian algorithm," in *Security in Computing and Communications: 8th International Symposium, SSCC 2020, Chennai, India, October 14–17, 2020, Revised Selected Papers 8*. Springer, 2021, pp. 213–227. [Page 30.]

[38] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016. [Pages 30, 31, and 36.]

[39] J. Alikhanov and H. Kim, "Online action detection in surveillance scenarios: A comprehensive review and comparative study of state-of-the-art multi-object tracking methods," *IEEE Access*, vol. 11, pp. 68 079–68 092, 2023. doi: 10.1109/ACCESS.2023.3292539. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10173520 [Page 36.]

TRITA-EECS-EX-2024:435