# Real-Time Accuracy and Effectiveness of Computer Vision and Machine Learning in Vehicle Intent Prediction at T-Junctions

**Simon McCool**

A thesis submitted to the University of Hertfordshire in partial fulfilment of the requirement of the degree of

**Doctor of Philosophy**

February 2024

**The research programme was conducted in the School of Computer Science, University of Hertfordshire.**

# Acknowledgements

I am deeply grateful to my esteemed supervisors, Dr. Christoph Salge, Dr. Wei Ji, and Prof. Volker Steuber, for their invaluable guidance, unwavering support, and thoughtful counsel throughout my doctoral research. It was a privilege and an honour to benefit from their remarkable scientific expertise and exceptional personal qualities. Their ability to maintain perspective and their genuine concern for my work has been greatly appreciated.

I am also profoundly thankful for their mentorship, which transcended academic instruction, fostering my professional growth and personal development. Their dedication to excellence and nurturing approach significantly contributed to my journey, offering insights that extended well beyond the confines of my research. Their encouragement during challenging moments and celebration of my achievements have shaped my doctoral experience. It is with heartfelt appreciation that I acknowledge the pivotal role they have played in my academic and personal endeavours.

This work would not have been achievable without the tireless support of my family: Lisa, Hannah, and Sam.

# Abstract

This thesis introduces a systematic approach to devising and evaluating advanced methods for predicting vehicle intent at unsignalised UK T-junctions. The primary focus of this thesis revolves around exploring the application of machine learning and computer vision techniques for the real-time prediction of vehicle intentions, emphasising increasing the prediction distance from the merge line at T-junctions to improve prediction efficacy.

This thesis addresses the sparsity of publicly available data on vehicle behaviour and feature vector data at UK T-junctions by demonstrating methods to create a unique Junction Video Dataset as a foundational contribution to the field. The methodology encompasses collecting, preprocessing, and annotating video data to develop video data input for a pipeline for vehicle intent prediction. The thesis presents a comparative analysis of YOLOv5 and Faster R-CNN models, focusing on their performance in vehicle detection using the curated Junction dataset. It then introduces an innovative fine-tuning process that enhances real-time detection capabilities.

This thesis uses an advanced feature extraction method to extract the stochastic nature of vehicle behaviour exhibited at T-junctions. This approach employs a sophisticated data processing and learning strategy incorporating extracted features. It continuously updates the training dataset with new feature vectors, enabling perpetual learning and the capability to make intent predictions on newly acquired data in real-time.

This thesis evaluates DAISY, a real-time vehicle intent prediction model, by comparing its performance with state-of-the-art systems such as Waymo's ChauffeurNet, Tesla Autopilot, NVIDIA Drive, and Mobileye. Key metrics for comparison include accuracy, latency, robustness, and scalability. DAISY demonstrates competitive accuracy and latency, which are crucial for real-time applications. It also benefits from a modular design that enhances scalability. However, direct comparisons are challenged by systematic differences like proprietary datasets, specialised hardware, and varied algorithm complexities.

This thesis revealed that integrating machine learning and computer vision techniques with high-quality data can accurately predict vehicle intent at T-junctions. Such an approach has the potential to serve as a crucial element within a safety model, functioning as an early warning system or activating driver assistance features.

# Contents


Abstract ..... 3

Glossary of Key Terms ..... 9

Chapter 1: Introduction ..... 11

1.1 Motivation ..... 11

1.2 Research questions ..... 11

1.3 Overview ..... 13

1.3.1 Structure of thesis ..... 14

1.4 Contributions of this thesis ..... 15

1.5 Ethics ..... 17

1.6 Real-Time intent prediction in context ..... 17

Chapter 2: Background and Literature Review ..... 19

2.1 Background information ..... 19

2.1.1 Intent prediction ..... 19

2.1.1.1 Static vs. Dynamic (Vehicle-Mounted) Video Sources ..... 21

2.1.2 Simulated data ..... 25

2.1.3 UK T-junction accident information ..... 26

2.2 Literature review ..... 26

2.2.1 Traffic video datasets ..... 26

2.2.2 Vehicle object detection ..... 28

2.2.3 Vehicle Intent classification at intersections ..... 34

2.2.3 Non-predictive, reactive methods of accident mitigation ..... 39

2.3 Chapter summary ..... 40

Chapter 3: Creating a Target Vehicle Video Dataset ..... 41

3.1 Introduction ..... 41

3.1.1 Organisation of the Chapter ..... 43

3.2 Junction Dataset Considerations ..... 43

3.2.1 Types of unsignalized junction ..... 44

3.3 Selecting Experimental Junctions ..... 46

3.4 Camera position at a T-junction and point of view (POV) ..... 49

3.4.1 Experiments with camera points of view (POV) ..... 49

3.5 Constructing the video for the dataset ..... 52

3.6 Chapter conclusion ..... 53

Chapter 4: Selection of Target Vehicle Detector ..... 54

4.1 Introduction ..... 54

4.1.1 Organisation of the chapter ..... 55

4.2 Anatomy of Faster R-CNN ..... 57
4.2.1 Training Faster R-CNN ..... 59
4.3 Anatomy of YOLOv5 ..... 61
4.3.1 Training YOLOv5 ..... 64
4.4 Selecting a vehicle detection and classification base model ..... 65
4.4.1 Common Objects Dataset ..... 68
4.4.2 Backbone Network Model ..... 68
4.4.2.1 Choice of Faster R-CNN Backbone Model ..... 69
4.4.2.2 Choice of YOLOv5 models ..... 69
4.4.2.3 Limitations of the YOLO model ..... 70
4.4.3 Comparison of YOLOv5 and Faster R-CNN ON COCO 80 ..... 71
4.5 Improving target vehicle detection accuracy with focused target training ..... 74
4.5.1 Target vehicle image dataset creation ..... 75
4.5.2 Image augmentation post-labelling ..... 76
4.5.3 Training target dataset on YOLOv5 models ..... 77
4.6 Chapter conclusion ..... 79
Chapter 5: Optimising target vehicle detection and classification ..... 80
5.1 Introduction ..... 80
5.1.2 Chapter organisation ..... 81
5.2 Feature selection ..... 82
5.2.1 Frame rate ..... 82
5.2.2 Creating video samples of various FPSs and resolutions ..... 83
5.3 Distance and velocity feature vector evaluation metrics ..... 83
5.4 Establishing a detection model ..... 84
5.4.1 Training all YOLOv5 models with transfer learning ..... 85
5.4.2 Selecting dataset and model combination ..... 90
5.5 Experimentation with a variety of FPSs, neural networks, resolutions, and Vo ..... 92
5.5.1 Video samples ..... 92
5.5.2 Class confidence ..... 92
5.5.3 Results from experimentation with FPSs, neural networks, resolutions, and $V_o$ ..... 94
5.5.3.1 Correlation analysis based on results in Table 14. ..... 95
5.6 Selecting the optimal $V_o$ based on resolution, fps, and ground truth data ..... 100
5.7 Chapter conclusion ..... 103
Chapter 6: Creating and extracting feature vectors from target vehicles ..... 104
6.1 Introduction ..... 104
6.2 Chapter organisation ..... 105
6.3 DUKE ..... 105
6.3.1 Bounding box predictions ..... 106

6.3.2 Intersection Over Union (IoU) ..... 108
6.3.3 Anchor boxes and ground truth associations ..... 109
6.3.4 Localisation errors and refinement ..... 111
6.3.5 Confidence ..... 112
6.3.6 Non-max suppression ..... 113
6.3.7 Full bounding box prediction ..... 115
6.3.8 Vehicle tracking ..... 117
6.3.8.1 Overview of DeepSORT ..... 118
6.4 Feature vector extraction ..... 119
6.4.1 Feature vector creation ..... 122
6.4.2 Feature vector constant, variable, and calculated values ..... 123
6.5 Initial analysis of feature capture ..... 126
6.5.1 Discussion of Figure 35 ..... 127
6.5.2 Comparison of DUKE-derived data ground truth values ..... 127
6.6 Chapter conclusion ..... 128
Chapter 7: Intent prediction training dataset DYLE ..... 129
7.1 Introduction ..... 129
7.2 Chapter organisation ..... 130
7.3 Extracting training features from Bo video ..... 130
7.4 Feature training data single junction ..... 131
7.4.1 Prediction Class ..... 133
7.4.2 Subclasses ..... 134
7.4.3 DYLE video training analysis for single junction ..... 136
7.4.4 Empirically and quantitatively determining the reliability of feature vectors ..... 137
7.4.5 Manual classification of training data ..... 141
7.4.6 Complete single junction dataset ..... 144
7.4.7 Accuracy using K-fold cross-validation ..... 144
7.4.8 Pandas profiling report ..... 145
7.5 Dataset creation for other junctions ..... 148
7.5.1 Results from k-fold cross-validation on Aggregated DYLE ..... 152
7.5.2 Results discussion ..... 153
7.7 Chapter conclusion ..... 154
Chapter 8: Intent Prediction using DAISY ..... 156
8.1 Introduction ..... 156
8.2 Chapter organisation ..... 158
8.3 DAISY ..... 158
8.3.1 Intent prediction DAISY ..... 159
8.3.2 Probability density function (PDF) of a Gaussian distribution ..... 160

8.3.3 Intent classification steps ..... 161
8.4 Live prediction by DAISY ..... 161
8.4.1 Intent prediction for a single junction JM377 ..... 164
8.4.2 Initial discussion from results of JM377. ..... 166
8.4.2 Evaluation metrics ..... 167
8.5 Intent prediction for other junctions ..... 169
8.5.1 Junction JM384 ..... 169
8.5.1.2 Initial discussion from results of JM384 ..... 170
8.5.2 Junction JM599 ..... 170
8.5.2.1 Initial discussion from results of JM599 ..... 171
8.5.3 Junction JM454 ..... 172
8.5.3.1 Initial discussion from results of JM454 ..... 172
8.6 DAISY performance with aggregated data for all four junctions ..... 173
8.6.1 K-fold cross-validation of updated DYLE training dataset ..... 173
8.6.2 Analysing ground truth and prediction data ..... 174
8.6.3 Precision-Recall and F1-Score ..... 174
8.6.4 Metric comparison of single and combined junctions ..... 175
8.7 Discussion ..... 177
Chapter 9: Pipeline Autonomy ..... 178
9.1 Introduction ..... 178
9.1.2 Chapter organisation ..... 181
9.2 Ablation study, the effect of sub-classification of feature vectors ..... 181
9.3 Autonomously applying Fpc and Sfpc classification features to feature vectors ..... 183
9.3.1 DUKE: Autonomous Merge line data recording ..... 183
9.4 Interactions with other vehicles ..... 185
9.5 Online data Verification and analysis ..... 187
9.5.1 Single Junction Online Verification JM454 ..... 187
9.5.2 Updaing DYLE with autonomously classified feature vectors from JM454 ..... 190
9.5.3 Verification of autonomous online intent predictions JM454 ..... 191
9.5.4 Accuracy and distance from merge line junction JM454 ..... 191
9.5.5 Comparison of F1 scores and accuracy for given distance ranges JM454 ..... 192
9.6 Online distance accuracy experiments for JM599, JM377 and JM384 ..... 194
9.6.1 JM599 Online verification and distance from merge line accuracy ..... 194
9.6.2 JM384 Online verification and distance from merge line accuracy ..... 196
9.6.3 JM377 Online verification and distance from merge line accuracy ..... 197
9.6.4 Online DYLE class distribution ..... 200
9.6.5 Analysis of the resulting metrics from the online Verification video data ..... 200

9.6.6 Accuracy and distance from merge line ..... 203
9.7 Balancing the Training Data ..... 204
9.7.1 Generating new samples of Hazard class and sub-classes ..... 205
9.7.2 K-Fold cross-validation of SMOTE DYLE ..... 206
9.7.3 Generating new samples of minority $F_{PC}$ classes ..... 207
9.7.4 K-Fold cross-validation of Uniform DYLE ..... 208
9.8 Unseen data online with a new junction ..... 210
9.8.1 UO196 Junction preparation ..... 210
9.8.2 Distribution of classes and K-fold score of Alpha DYLE ..... 210
9.8.3 Accuracy and F1 score based on UO196 video log ground truths ..... 211
9.8.4 Comparison of results from Alpha DYLE + UO196 and other DYLE iterations ..... 214
9.9 Chapter conclusion ..... 215
Chapter 10: Conclusion and Future Work ..... 217
10.1 Thesis summary ..... 217
10.2 Research questions discussed ..... 221
10.3 Comparison with current state of the art real time intent prediction ..... 222
10.4 Discussion and future work ..... 226
10.5 Future work: The Swiss cheese model of safety ..... 227
10.5.1 Contribution to the Swiss Cheese Model for Motorcycle Safety ..... 229
Bibliography ..... 230

# Glossary of Key Terms

| | |
|---|---|
| AP | Average Precision |
| AUC | Area Under the Curve |
| AV | Autonomous Vehicle |
| BO | T-Junction Video Dataset |
| CBAM | Convolutional Block Attention Module |
| CNN | Convolutional Neural Network |
| DAISY | Predictive Model |
| DNN | Deep Neural Network |
| DPM | Deformable Part-Based Model |
| DUKE | Vehicle Detection And Feature Extraction Algorithm |
| DYLE | Feature Vector Dataset |
| EDA | Exploratory Data Analysis |
| Faster RCNN | Faster Region-Based Convolutional Neural Networks |
| FLOPS | Floating Point Operations |
| FN | False Negative |
| FP | False Positive |
| $F_{pc}$ | Final Driver Intent Prediction |
| FPR | False Positive Rate |
| FPS | Frames Per Second |
| ICS | Inevitable Collision State |
| ICW | Intersection Collision Warning |
| IOC | Inverse Optimal Control |
| IoU | Intersection Over Union |
| IPTM | Intersection Prior Trajectories Model |
| LSTM | Long Short-Term Memory Network |
| mAP | Mean Average Precision |
| MAEB | Motorcycle Autonomous Emergency Braking |
| MOMDP | Mixed Observability Markov Decision Process |
| MPR | Market Penetration Rate |
| MSCOCO | Microsoft Common Objects in Contexts (COCO) |
| NMS | Non-Max Suppression |
| PASCAL | Pattern Analysis, Statistical Modelling and Computational Learning |
| PASCALVOC | PASCAL Visual Object Classes |
| PDF | Probability Density Function |

| | |
|---|---|
| QRF | Quantile Random Forest |
| RCNN | Region-Based Convolutional Neural Networks |
| RNN | Recurrent Neural Network |
| ROI | Region of Interest |
| RPN | Region Proposal Network |
| $S_{Fpc}$ | Associated Subclass Predictions |
| SGD | Stochastic Gradient Descent |
| SIMP | Semantic-Based Intention And Motion Prediction |
| SLAM | Simultaneous Localisation and Mapping |
| SSD | Single Shot Detector |
| SVM | Support Vector Machine |
| TN | True Negative |
| TP | True Positive |
| TPR | True Positive Rate |
| TTC | Time-To-Collision |
| VRU | Vulnerable Road Users |
| YOLO | You Look Only Once |

# Chapter 1: Introduction

## 1.1 Motivation

In the UK, vulnerable road users (VRU), such as motorcyclists, represent only 1% of total road traffic yet suffer 18% of road fatalities, as highlighted in Royal Society for the Prevention of Accidents research on common motorcycle crash causes (RoSPA, 2017). Data from the UK police-reported accident data highlighted that 64% of motorcycle accidents occurred at a junction (Senserrick et al., 2017); the majority of these collisions occur at T-junctions when drivers pull out into the path of an oncoming motorcyclist. Considering that many accidents, especially at T-junctions, are attributed to human mistakes, it is crucial to research current methodologies that can help mitigate this type of accident. Despite rapid advances in autonomous vehicle (AV) technology, fully driverless cars are not imminent. Forecasts for widespread adoption of AVs are varied, with more conservative estimates suggesting several decades (Kannan and Lasky, 2020). However, lower levels of AV have been implemented in all types of vehicles. Current collision avoidance technology, such as motorcycle autonomous emergency braking (MAEB), is effective (Savino et al., 2016) but is limited to systems designed to scan the environment to detect possible hazards in navigation paths, not before they enter the navigation path, making a fundamental argument for researching the feasibility of how an accurate prediction of the future intent of merging drivers at a T-junction could further mitigate the severity of a collision. The frequency of this type of accident demonstrates that motorcyclists alone cannot react to threats posed at a T-junction; otherwise, there would be far fewer accidents of this type. Predicting the future state of the driving environment and other road users is a non-trivial task. As with autonomous vehicles, human-controlled vehicles benefit from higher levels of safety in the abstract levels of driving autonomy, and in this thesis, we look at this argument as a computer vision problem. This thesis uses 2D camera video data, machine learning, and deep learning techniques. We aim to identify, track, and evaluate vehicles approaching a T-junction. This allows us to investigate how effectively we can predict the likelihood of vehicles yielding before they enter the path of an oncoming motorcycle.

## 1.2 Research questions

The overarching question is: How effectively can computer vision and machine learning methods be utilised to predict the intentions of vehicles at T-junctions in real-time, and to what degree of accuracy and effectiveness can these predictions be achieved?

This thesis is structured around seven pivotal research questions, each crafted to contribute towards answering the primary overarching question: the viability of using computer vision

techniques for predicting vehicle intentions at T-junctions. It begins with RQ0, which questions the feasability of real video data collection for modeling behaviour. RQ1 examines the impact of a constrained dataset on the training and performance of models. RQ2 investigates the influence of pixel density and frame rate variations on the effectiveness of these models. RQ3 assesses the possibility of capturing accurate pixel-level features from moving vehicles. RQ4 evaluates the consistency of feature vectors under uniform camera setups. RQ5 and RQ6 focus on using 2D video-derived feature vectors to predict vehicular movement at a T-junction and the accuracy and prediction range of these models with new data. Lastly, RQ7 explores the feasibility of incorporating real-time intent predictions into the model without compromising its accuracy or F1 score. These questions, while interrelated, provide distinct insights and benefits tailored to different research interests within the domain of computer vision and machine learning.

**RQ0**: Is it feasible to collect real-world video data from T-junctions that can accurately inform the development of a vehicle intent model for predicting vehicular behaviour?

**RQ1**: How does employing a constrained and focused dataset affect the performance of object detection and Classification?

**RQ2:** Considering the neural network's characteristics in use, how do pixel density and frame rate variations affect real-time object detection and classification models?

**RQ3:** Is obtaining accurate pixel-level features from dynamic vehicles that closely match ground truth data feasible?

**RQ4:** Can our feature vectors' inherent generality be observed per the consistent camera positioning hypothesis? This hypothesis posits that recordings from various junctions maintain a similar perspective due to the standardised factors of camera height, position concerning the merge line, and overall camera placement.

**RQ5**: How accurately can a machine learning model, utilising 2D video-derived feature vectors, predict a vehicle's intention at a T-junction?

**RQ6:** Can a trained machine learning model accurately predict vehicle intent at a T-Junction using new data, and what is its effective prediction range from the junction?

**RQ7:** Can the online model infer and append intent predictions as new inference data in real-

time without negatively affecting the accuracy or F1 score?

## 1.3 Overview

This thesis is structured to follow the sequential logic of our experimental framework. The culmination of our research process is presented in Chapter 9, where readers can finally understand the experimental outcomes in the context of the theoretical groundwork laid out in earlier chapters. Figure 1 illustrates our comprehensive pipeline, beginning with video data input and culminating in vehicle intention prediction. Raw video footage is initially processed and adjusted to align with ground truth benchmarks, such as the distance to merge lines. DUKE then analyzes this optimized video for vehicle detection, classification, and extraction of relevant features. These features are subsequently input into DAISY to predict vehicle intentions, utilising the DYLE dataset for classification. The results generated by DAISY are re-integrated into DYLE, fostering a continuous cycle of data training. The final output is presented either as a warning or as raw data.

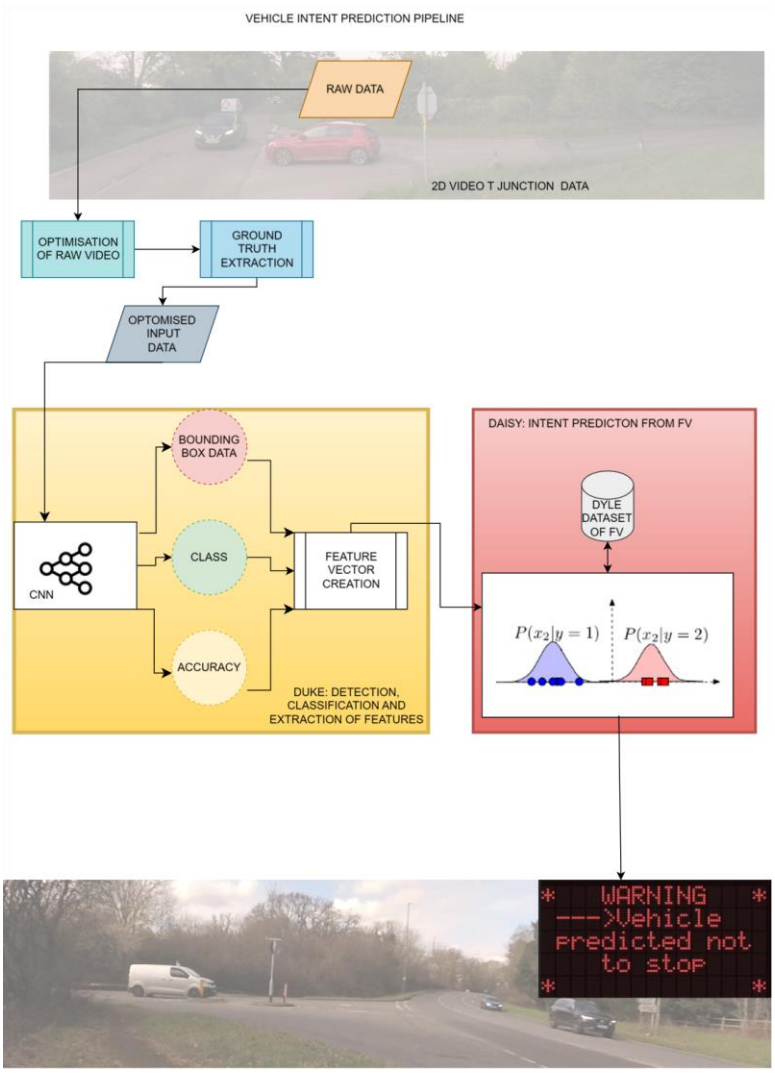


*Figure 1 Overview of our vehicle intent pipeline, with road sign warning as an output.*

### 1.3.1 Structure of thesis

To address the research questions outlined above, our initial step involved significant work in generating our training data, encompassing both video-related and feature vector data. Given the absence of such data in the public domain, this endeavour was essential to our thesis, providing critical insights into the questions posed. Moreover, this effort constitutes a part of our contribution, laying the groundwork for others' future exploration of this topic. This means Chapters 3, 4, and 5 offer technical descriptions of the processes we engaged in to develop the pipeline, ultimately enabling us to experiment with intent prediction at T-junctions.

The thesis is organised as follows:

**Chapter 2**: Background and related work

Chapter 2 gives a background of intent prediction and human behaviour at T-junctions and describes our approach's rationale. We then reviewed the diverse models for intent prediction, including behavioural aspects, object detection, and reinforcement learning available in the literature. The review focused on vehicle intent prediction methods and their components, surmising that the most effective models can infer from real-time video frames. The chapter details methods, techniques, challenges, and creating a custom dataset, evaluating predictive models for intent prediction at intersections and T-junctions.

**Chapter 3**: Junction Video Dataset

This chapter delves into the creation and curation of our Junction Video Dataset. We discuss the methodology employed to collect, preprocess, and annotate the data, providing essential insights into the foundation of our research.

**Chapter 4**: Comparative Analysis of YOLOv5 and Faster R-CNN Models

Building upon the Junction Video Dataset introduced in the previous chapter, and Chapter 4 focuses on a quantitative evaluation of the YOLOv5 and Faster R-CNN models. We rigorously assess their inference times and accuracy in the context of real-time vehicle detection using our specialised video dataset.

**Chapter 5:** Innovative Fine-tuning for Real-time Vehicle Detection

This chapter introduces an innovative approach to fine-tuning real-time vehicle detection and classification models. We highlight the performance improvements achieved through our fine-tuning process, emphasising the practical implications for real-world applications.

**Chapter 6:** Feature Vector Extraction Method
This chapter provides a comprehensive overview of our feature vector extraction methodology. We detail the techniques and algorithms used to extract meaningful feature vectors from vehicle data, which are the foundation for subsequent chapters.

**Chapter 7:** Organising and Classifying Vehicle Feature Vectors
This chapter introduces DYLE, an advanced real-time data handling and learning methodology. DYLE is distinguished by its ability to dynamically manage and refresh training data through feature vectors, allowing training and prediction models to utilise the latest data. It supports ongoing learning and adjustment and is designed for real-time updates with new information. An essential process in DYLE is dynamic enrichment, which involves the accumulation of new feature vectors with each iteration, a function performed by DUKE, as elaborated in Section 6.4. This feature represents a significant advancement in improving machine learning workflows.

**Chapter 8:** Efficient Vehicle Intent Prediction at T-Junctions
Chapter 8 introduces a computationally efficient approach for predicting vehicle intent at T-junctions. We utilise feature vectors derived from video data as training inputs, paving the way for improved decision-making algorithms in critical traffic scenarios.

**Chapter 9:** Quantitative Examination of DAISY's Predictive Abilities
Chapter 9 focuses on a quantitative examination of DAISY's predictive capabilities. We evaluate how accurately DAISY performs when trained on progressively larger datasets, particularly in predicting driver intentions and determining the practical distance from the junction at which predictions remain reliable. This exploration contributes to a deeper understanding of the limits and capabilities of machine learning in the context of driver behaviour prediction at critical road intersections. Additionally, we explore creating and evaluating an online model capable of real-time data inference and integration while maintaining a high base accuracy and F1 score.

## 1.4 Contributions of this thesis

This thesis focuses on the feasibility of using machine learning models trained on existing data to predict driver behaviour at T-junctions when confronted with new, unseen data. Specifically, it involves a quantitative examination of how accurately these models can predict driver intentions and determines the practical distance from the junction at which predictions remain viable. This exploration contributes to understanding the limits and capabilities of machine

learning in the context of driver behaviour prediction at critical road intersections.

Specifically, the contributions include the following:

- **Construction of a target-based vehicle image dataset tailored to our video data**: This involves creating a specific dataset that includes images of vehicles as they would appear in our monocular videos. This dataset is tailored to the unique perspectives, angles, and lighting conditions in the video data collected at T-junctions.

- **Creation of a data-rich video dataset comprising unsignalized UK T-junctions**: This dataset is a collection of video recordings from various T-junctions across the UK that do not have traffic signals. It focuses on capturing various traffic scenarios to ensure that the machine learning models developed can handle different traffic behaviours and conditions. The data-rich dataset allows the accurate extraction of vehicle feature vectors necessary for understanding and predicting vehicle behaviours.

- **Inference time and accuracy quantitative comparison of YOLOv5 and Faster R-CNN models using our bespoke video dataset**: This involves testing and comparing the performance of two popular deep learning models, YOLOv5 and Faster R-CNN, in terms of their inference speed and accuracy in detecting and classifying vehicles using the project's specific video dataset. This comparison helps select the most suitable model for real-time vehicle detection and prediction at T-junctions.

- **Generation of accurate dynamic vehicle feature vectors for utilisation in real-time prediction**: This refers to identifying and extracting dynamic features from the vehicles, such as velocity, direction, and acceleration, which are essential for predicting their future movements. These feature vectors are generated in real-time and are used to analyse vehicle behaviours at the T-junction.

- **Extension of the approach to fine-tuning a real-time vehicle detection and classification model based on performance**: This involves continuously improving and adapting the vehicle detection and classification model based on its performance in real-world scenarios. The model is fine-tuned to enhance its performance in detecting and classifying vehicles in real-time, ensuring it remains effective under various conditions.

- **Development of a method for independently organising and classifying discrete vehicle feature vectors as feature vector arrays and integral components of a comprehensive general dataset**: This process involves developing a systematic approach to organise and classify the extracted vehicle feature vectors into structured arrays. These arrays are then incorporated into a larger dataset, facilitating the comprehensive training and testing of our model.

- **A computationally efficient approach for predicting vehicle intent at a T-junction using video-derived feature vectors as training data**:

- **Implementing a self-learning real-time prediction model** involves developing a prediction model that not only utilises the current data and feature vectors for making predictions but also continually learns and adapts from new data it encounters. This self-learning capability ensures that the model remains accurate and up-to-date with changing traffic patterns and behaviours, improving its reliability and effectiveness in predicting vehicle movements at T-junctions.

## 1.5 Ethics

Before collecting live traffic video data, we secured ethical approval from the University of Hertfordshire's ethics committee.

## 1.6 Real-Time intent prediction in context

In the context of our work, a real-time intent prediction model, "real-time", refers to the capability of the system to process data, make predictions, and deliver actionable insights almost instantaneously or within a concise time frame.

**Definition of Real-Time in Vehicle Intent Prediction**

Real-time processing in vehicle intent prediction implies that the system can continuously analyze incoming data from various sensors, interpret the current driving scenario, predict the potential actions of surrounding vehicles, and communicate these predictions to the vehicle's control systems or the driver with minimal latency, in the context of our work we aim to produce an Immediate Hazard Detection to allow vehicles to take evasive actions immediately, reducing the risk of collisions.

**Technical Challenges:**

**Computational Load:**

- **High Processing Power:** Real-time predictions require substantial computational power, especially for processing data from high-resolution

video.

- **Efficient Algorithms:** Developing efficient algorithms that balance speed and accuracy is challenging.
- **Network Latency:** Ensuring low-latency communication within the vehicle's network architecture is essential for timely predictions.

Real-time intent prediction is a critical feature of our work that ensures the system can process environmental data, make accurate predictions, and instantly communicate necessary actions. Overcoming the technical challenges associated with real-time processing requires continuous advancements in computing power, algorithm efficiency, and data management strategies.

# Chapter 2: Background and Literature Review

## 2.1 Background information

### 2.1.1 Intent prediction

The foundation for vehicle intent prediction involves understanding and forecasting vehicles' future actions or movements in various contexts, such as at intersections or junctions, in traffic, or actions during autonomous vehicle operation, such as lane changes on a highway.

In the reviewed literature, the terms' driver intent' and 'vehicle intent' were often used interchangeably to denote the intentions behind either a human driver's or an autonomous vehicle's (AV) actions. However, given our focus on the tangible dynamics of vehicles, we consistently refer to 'vehicle intent' in this thesis.

Computer vision and machine learning-based vehicle intent prediction integrate various methodologies using sensor data analysis, which utilises tools like 2D and 3D cameras, radar, Lidar and GPS to gauge the vehicle's surroundings and movement. Machine learning models then predict future actions using this data and historical trends. Behavioural modelling considers human factors such as driving habits to refine these predictions. Finally, a contextual understanding of environmental factors like road conditions and nearby traffic is incorporated into the overall framework to generate a complete picture of the problem. Together, these approaches provide a comprehensive system for anticipating vehicle behaviour that enhances road safety and efficiency. Vehicle intent prediction aims to match or surpass human capability in predicting the actions of vehicles on the road.

On the one hand, vehicles equipped with these technologies offer consistency and reliability as they can process extensive data without fatigue, unlike humans, who may be inconsistent due to distractions or emotional factors. They also boast faster reaction times and can be coupled with sensors that provide 360-degree perception, potentially offering an advantage over human sensory capabilities. Additionally, machine learning models continually learn and improve and may surpass human prediction accuracy in the future.

However, there are significant challenges. The complexity of human behaviour presents a considerable hurdle: humans can be unpredictable, and machines may struggle to interpret nuanced behaviours effectively. While machines advance in situational awareness, understanding the full range of human-like contextual clues remains challenging. Lastly, in scenarios where predictions lead to critical decisions, especially in emergencies, humans consider ethical and moral implications, a capacity machines lack.

Our research began by examining the high fatality rates at UK T-junctions through a psychological lens rather than seeking a direct solution, as Crundall et al. (2012) and Crundall,

Howard and Young (2017) do by recommending perceptual training for drivers or Yee Mun Lee, Sheppard and Crundall (2015) regarding the cross-cultural effects of perception of motorcycles, we aimed to understand aspects of human driving behaviour, such as head movement or angle, vehicle position on the road and vehicle kinetics, to identify potential features that could be detected using computer vision technology.

Our initial experiments attempted to capture the head movement of drivers approaching the junction merge line with the same camera used to track vehicles; however, we found that we required a second camera zoomed in to capture a driver's head, which was not feasible, as the data from the vehicle had to relate to the data from the driver and to combine video frame impacted on the computational load and subsequent inference performance.

We also reviewed the work of Peter Chapman from the University of Nottingham, who specialises in the psychological aspects of motorcycle accidents at T-junctions (Robbins et al., 2019) (Robbins, Allen, and Chapman, 2018). One of the findings from Chapman's research, using driving simulators, determined that drivers might still move forward even after noticing an approaching motorcycle, suggesting that using head movement alone is not a dependable sign that a driver has seen an approaching motorcycle. The study explored identifying hesitancy or specific vehicle movement patterns as potential indicators of a driver's intentions at a T-junction, and that was a starting point for further experimentation.

Some studies have approached the problem of high T-junction accident rates via a detailed examination of motorcyclists' behaviours at different T-junctions (Mohd et al., 2022). The present study categorises T-junctions into three types: Type A (conventional T-junction), Type B (unconventional T-junction with a short exit lane for right-turning vehicles on the minor road) and Type C (unconventional T-junction with a short exit lane for through traffic on the major road). The findings demonstrate that Type A junctions have the highest incidence of risky riding behaviour. In terms of the critical gap, which is the time needed for a motorcyclist to safely execute a right turn, Type A junctions require the longest time (9.20 seconds), followed by Type C (7.20 seconds) and Type B (7.00 seconds). These results suggest that Type B junctions are the most efficient and potentially the safest for motorcyclists making right turns from minor roads.

Other relevant studies have discussed driver responses to cyclists at T-junctions (Walker, 2005; Ammar Al-Taie et al., 2023). There has also been extensive work in pedestrian intent prediction, with examples including Moreno et al. (2023) using naturalistic trajectories at unsignalised junctions and Hsu et al.'s (2020) simplified model of the problem of pedestrian AV interactions.

Rather than relying on a single element to predict a vehicle's intent, our approach involves a composite model of various interconnected components structured in a pipeline format that culminates in intent prediction. This thesis explores each segment within the intent prediction

pipeline, scrutinising the contributions at every phase. This study focuses not only on the final prediction outcome or the cause but also on assessing the viability and efficiency of computer vision and machine learning techniques in predicting vehicular intent at a T-junction based on each pipeline segment's performance. In intent and trajectory prediction, common methods involve predicting a vehicle's future path based on its historical positions and often employ recurrent neural networks (RNNs) or long short-term memory (LSTM) networks. Behavioural prediction analyses vehicle dynamics like speed and acceleration to infer intent using machine learning models trained on historical data. Action recognition methods predict intent by identifying specific manoeuvres from video frames, such as turn signals or lane changes. Object detection and tracking involve using algorithms to identify and predict the movements of vehicles in video frames. Semantic segmentation helps understand the scene's context by classifying image parts into categories like roads or vehicles, thereby assisting in intent prediction.

#### *2.1.1.1 Static vs. Dynamic (Vehicle-Mounted) Video Sources*

Computational Challenges:

Static Video Sources:

- Fixed Perspective: Static cameras have a fixed perspective, making background modelling and motion detection relatively straightforward as the background remains constant.
- Limited Field of View: Static cameras cover a limited area, requiring a network of cameras for extensive coverage, leading to data integration and synchronization challenges.
- Stable Imaging Conditions: The stability of static cameras ensures consistent imaging conditions, aiding in more accurate object detection and tracking.

Dynamic (Vehicle-Mounted) Video Sources:

- Changing Perspective: Vehicle-mounted cameras constantly change their perspective, complicating background subtraction and motion detection, requiring more sophisticated algorithms to adapt to varying scenes.
- Wide Field of View: These cameras can cover larger areas as they move, but the constantly changing view necessitates real-time processing to keep up with the dynamic environment.
- Variable Imaging Conditions: Movement introduces variability in lighting, shadows, and weather conditions, making object detection and tracking more challenging.

Practical Difficulties in Intent Prediction:

Static Video Sources:

- Limited Context: Static cameras provide a limited context, often missing out on interactions occurring outside their field of view, leading to incomplete data for intent prediction.
- Predictive Lag: Intent prediction might lag due to the fixed viewpoint, making it harder to anticipate actions that start outside the camera's view and move into it.

Dynamic (Vehicle-Mounted) Video Sources:

- Complex Motion Patterns: Predicting intent from a moving platform involves accounting for both the motion of observed objects and the motion of the camera itself, increasing the complexity of the prediction models.
- Occlusion Handling: As the vehicle moves, objects may be occluded by parts of the vehicle or other objects, making continuous tracking and intent prediction difficult.
- Latency and Real-Time Processing: Real-time intent prediction is crucial for vehicle-mounted systems to make immediate decisions. This requires high computational power and efficient algorithms to minimize latency.

Integration of Multiple Data Sources:

Both static and dynamic video sources can benefit from integration with other data sources (e.g., GPS, LiDAR, radar).

Benefits of Static Video Sources:

Enhanced Object Detection and Tracking:

- LiDAR and Radar: These sensors provide depth and range information, which can be combined with video data to improve object detection and tracking, especially in low-light or poor visibility conditions.
- Increased Accuracy: The fusion of video with depth data from LiDAR helps distinguish between objects and accurately measure their distances and dimensions.

Improved Situational Awareness:

- GPS Integration: Combining video data with GPS information allows for precise geolocation of objects within the camera's field of view, aiding in better situational awareness and context understanding.
- Spatial Context: Static cameras can gain spatial context, enabling better mapping and monitoring areas for security, traffic management, and urban planning.

Data Redundancy and Reliability:

- Multi-Sensor Redundancy: Combining multiple sensor data sources ensures reliability and reduces the risk of single-point failures, leading to more robust surveillance and monitoring systems.

Enhanced Anomaly Detection:

- Cross-Validation: Different sensors can validate each other's data, making detecting anomalies or suspicious activities easier with higher confidence.
- Contextual Analysis: Video data can provide visual context, while LiDAR and radar offer physical context, improving anomaly detection accuracy.

Benefits for Dynamic (Vehicle-Mounted) Video Sources:

Real-Time Navigation and Collision Avoidance:

- LiDAR and Radar: These sensors are crucial for detecting obstacles, other vehicles, and pedestrians in real time, providing depth and speed information that complements video data.
- Enhanced Safety: Integration helps in real-time navigation and collision avoidance decision-making, critical for autonomous driving and advanced driver-assistance systems.

Accurate Localization and Mapping:

- GPS and IMU: Combining video with GPS and inertial measurement unit (IMU) data enables accurate vehicle localization and mapping, which is crucial for autonomous navigation and route planning.
- Simultaneous Localization and Mapping (SLAM): Video data, when fused with LiDAR and GPS, improves SLAM algorithms, providing detailed and accurate maps of the environment.

Improved Object Recognition and Classification:

- Sensor Fusion: Video provides visual details (e.g., colour, texture), while LiDAR and radar offer shape, size, and distance information. Combining these enhances object recognition and classification accuracy.
- All-Weather Capability: Video can be less effective in adverse weather conditions, but radar and LiDAR can compensate, ensuring reliable operation.

Robust Intent Prediction:

- Comprehensive Data: Combining video with other sensor data enables a

more comprehensive analysis of object behaviours and interactions, leading to more accurate intent prediction.

- Temporal and Spatial Consistency: Multi-sensor fusion ensures temporal and spatial consistency in data, which is crucial for real-time intent prediction and proactive decision-making.

General Benefits:

Enhanced Data Quality and Completeness:

- Complementary Data: Different sensors provide complementary data, enhancing the overall quality and completeness of the information used for analysis and decision-making.
- Reduced Ambiguity: Sensor fusion reduces ambiguity and uncertainty, leading to more reliable and accurate interpretations of the environment.

Increased System Robustness:

- Fault Tolerance: Multi-sensor systems are more fault-tolerant. If one sensor fails or is compromised, others can provide the necessary data to maintain system functionality.
- Consistency in Various Conditions: Different sensors perform better in different conditions (e.g., radar in fog, LiDAR at night), ensuring consistent system performance across diverse scenarios.

Scalability and Flexibility:

- Adaptability: Integrated systems can adapt to various applications and environments, from urban traffic management to autonomous vehicles, enhancing their scalability and flexibility.

Integrating video sources with other data sources like GPS, LiDAR, and radar significantly enhances the capabilities of both static and dynamic systems. This integration leads to more accurate and reliable object detection, tracking, and intent prediction, improves situational awareness, and ensures robust performance in diverse conditions. The fusion of complementary data sources provides a comprehensive understanding of the environment, which is crucial for advanced surveillance, navigation, and autonomous systems applications.

However, the challenges differ:

Static Video Sources:

- Data Fusion Complexity: Integrating data from multiple static cameras and sensors requires sophisticated data fusion techniques to create a coherent understanding of the environment.

- Synchronization Issues: Ensuring temporal synchronization across multiple static cameras and sensors is critical for accurate intent prediction.

Dynamic (Vehicle-Mounted) Video Sources:

- Sensor Fusion: Vehicle-mounted systems often integrate data from various sensors (e.g., LiDAR, radar) in real-time, requiring complex sensor fusion algorithms.
- Geospatial Alignment: Aligning data from moving sensors with a dynamic environment map adds another layer of complexity, crucial for accurate intent prediction.

The computational challenges and practical difficulties in intent prediction vary significantly between static and dynamic video sources. Static cameras benefit from stable imaging conditions but are limited in field of view and context. Dynamic, vehicle-mounted cameras offer broader coverage but face challenges due to changing perspectives and variable conditions. Effective intent prediction in dynamic environments requires advanced real-time processing, robust object detection and tracking, and sophisticated data fusion techniques to handle the complex and dynamic nature of the data.

Although there is extensive research on predicting intent, our study concentrated on using 2D camera data exclusively, without incorporating data from other sensors. Our interest lies in methods that offer real-time performance, particularly those related to unsignalized junctions, as reported in the literature.

### 2.1.2 Simulated data

During the COVID-19 pandemic, our preliminary investigations prompted us to conduct experimental trials at a T-junction using traffic simulators. Traffic patterns during both the lockdown and early post-lockdown periods deviated significantly from the norm, rendering the data collected during these times unsuitable for our research objectives. During the initial phase of our study, we opted to use the City Drive simulator (Forward Development, 2023), which is notable for its unique feature of left-hand traffic flow. This distinct characteristic contrasts with most traffic simulators' right-hand traffic perspective. However, our experience with this simulator revealed a significant drawback. The artificial intelligence system responsible for regulating the simulated vehicles' behaviour frequently undermined the scenarios' realism by delaying actions at the junction and allowing vehicles to back up. As a result, the outcomes we observed were often unpredictable and lacked the authenticity we were aiming for. For this work, we required authentic T-junction data, and with no video datasets of UK T-junctions, we had to create our bespoke video data from live locations in the UK.

### 2.1.3 UK T-junction accident information

We selected our test locations for unsignalised junctions based on the history of severe or fatal accidents involving motorcyclists using the Road Safety Foundation EuroRAP data portal (Foundation, 2022) and from the work of Pai and Saleh (2008). We discuss junction choice and our data collection methods in detail in Chapter 3.

## 2.2 Literature review

Our review concentrates on vehicle intent prediction by examining trajectory and intent prediction methodologies, including intention-aware and interaction-aware strategies. An intention-aware system refers to a system designed to recognise, understand, and potentially predict a user's or another system's intentions (Fox et al., 2018).

Interaction-aware systems describe systems or devices that are sensitive to and can adapt based on interactions. These interactions could be between the system and its user, multiple users or even between different systems, and they play a role in understanding T-junction dynamics from various perspectives.

### 2.2.1 Traffic video datasets

As far as we know, no publicly available 2D video datasets are currently specific to UK T-junctions. However, many computer vision-focused traffic video datasets exist and are commonly utilised for benchmarking and forming a foundational part of computer vision research. Most traffic-centric video datasets aimed at intent and trajectory prediction integrate additional sensor data, including Lidar and radar, that is synchronised with AV data and captured from a moving vehicle, overhead drone, or stationary camera. Given the gap in the UK T-junction video data research, we developed our own UK T-junction video dataset inspired by the relevant examples in the existing literature.

Argoverse 2 (Wilson et al., 2021) is a set of datasets designed to enhance self-driving perception and forecasting research. It features a Sensor Dataset with 1,000 multi-modal sequences, a Lidar Dataset with 20,000 sequences for self-supervised learning, and a Motion Forecasting Dataset with 250,000 scenarios focusing on complex interactions. These datasets aim to address diverse and complex machine-learning challenges in autonomous driving.

KITTI-360 (Liao, Xie and Geiger, 2021) is a comprehensive suburban driving dataset that enhances urban scene understanding in both 2D and 3D. It offers richer input modalities, extensive semantic instance annotations, and accurate localisation. This dataset is notable for its geo-registered data of suburban scenes, a WebGL-based annotation tool for 3D space labelling and a method that translates 3D labels into coherent 2D semantic instance annotations. KITTI-360 strives to advance research across computer vision, graphics, and

robotics and establish benchmarks for tasks like semantic scene understanding, novel view synthesis, and simultaneous localisation and mapping (SLAM). SLAM is a method in robotics and autonomous vehicles where the system not only maps the environment and locates itself within it but also understands and labels the environment semantically, recognising and categorising elements in the surroundings such as walls, roads or objects and thus providing a richer, more helpful map.

Leddartech Pixset (Leddartech, 2024) is a novel dataset aimed at autonomous driving research and development. It is notable for its inclusion of full-waveform data from the Leddar Pixell sensor, a solid-state flash LiDAR. This dataset, which includes around 29k frames from 97 sequences recorded in high-density urban areas, is enhanced with 3D bounding boxes for each frame. The dataset's main contributions are the introduction of a new dual LiDAR-type dataset with solid-state and mechanical LiDARs, full-waveform data and the improvement of 3D bounding box annotation accuracy.

The Waymo Open Dataset (Waymo LLC, 2019) is a comprehensive dataset for autonomous driving research. It is known for its high-quality, high-resolution sensor data and labels necessary for various tasks in autonomous driving, such as 3D perception and behaviour prediction. This dataset has significantly contributed to advancing machine learning models by providing data from real-world scenarios and challenges encountered in autonomous vehicle development.

The highD Dataset (Krajewski et al., 2018) is a rich traffic dataset captured from German highways using drones. It focuses on providing a detailed understanding of vehicle behaviour and dynamics in various highway scenarios.

The INTERACTION Dataset (Zhan et al., 2019) is a vehicle trajectory dataset dedicated to understanding interactive driving behaviour in dense traffic environments. It is used to study and model vehicle interactions and to improve traffic flow analyses.

The Oxford Robot Car Dataset (Maddern et al., 2016) offers a large-scale and long-term dataset for developing vehicles capable of sustained autonomy in diverse conditions. It is a valuable resource in the robotics and autonomous vehicle research community, especially for those focusing on long-term vehicle localisation and autonomy in changing environments, and it is UK-road-structure-focused.

Kaggle (Kaggle, 2022) offers diverse open datasets, including traffic and transportation-related sets. Kaggle's platform allows access to datasets and provides a community-driven approach where users can share their datasets; however, as with any community-driven resource, quality and support can vary dramatically.

### 2.2.2 Vehicle object detection

The research explores various vehicle detection methodologies. Our work utilises a version of YOLO (you only look once) for vehicle detection and classification, discussed in detail in subsequent chapters. This section introduces an overview of YOLO to place it in context. YOLO represents a paradigm shift in detecting objects within images and is known for its speed and efficiency in real-time object detection.

The original YOLO (YOLOv1) by Redmon et al. (2016) was groundbreaking in treating object detection as a regression problem, combining region proposal and classification into a single step using a convolutional neural network (CNN). Redmon et al.'s version was faster than previous methods but had limitations in accuracy, particularly with smaller or grouped objects.

YOLO9000, a subsequent version, introduced significant improvements. Developed by Redmon and Farhadi in 2016, it incorporated anchor boxes for better boundary prediction and a new classification model, Darknet-19. This version could detect over 9,000 object categories and improved speed and accuracy.

In 2018, Redmon and Farhadi's YOLOv3 offered incremental improvements, including multi-scale predictions and a more profound architecture with Darknet-53. It enhanced the detection of small objects and struck a balance between speed and accuracy.

YOLOv4, created by Bochkovskiy and Wang (2020), integrated various techniques from the research community to optimise speed and accuracy. It included innovations like Mish activation, cross-stage partial connections and diverse data augmentation methods. It aimed to perform well on standard hardware.

YOLOv5, developed by Jocher in 2021, focused on being lightweight and extremely fast, suitable for speed-critical applications. It introduced scaling and deployment improvements.

Li et al.'s (2022) YOLOv6 emphasised easy deployment across platforms while balancing speed and accuracy, thus improving upon YOLOv5's architecture and training techniques.

YOLO versions 7 and 8, launched in 2023, are awaiting review and comparison with earlier versions. Each iteration of YOLO signifies advancements in object detection technology that balance speed, accuracy and real-world applicability.

Our decision to use YOLO was based on our requirements: a real-time inference speed of vehicle detection and classification, classification accuracy, and its use in the following research.

Numerous algorithms on speed and accuracy have been proposed, such as C3Ghost and Ghost (Dong, Yan and Duan, 2022) modules in YOLO. Dong et al. presented an improved version of the YOLOv5 object detection method, tailored explicitly for vehicle detection. The authors addressed two primary challenges of vehicle detection methods: high computational load and suboptimal detection rates. The key innovations in their approach include the

integration of C3Ghost and Ghost Modules: These modules are incorporated into the YOLOv5's neck network. They aim to reduce the floating-point operations (FLOPs) during the feature channel fusion process, which is a significant step because it helps make the model more efficient by reducing computational demands without sacrificing performance. The Convolutional Block Attention Module (CBAM) was added to the YOLOv5 backbone network. The CBAM plays a crucial role in enhancing the model's focus on relevant information for vehicle detection while suppressing irrelevant data. This selective attention mechanism is geared towards improving the detection accuracy of the algorithm. Dong et al. introduced the CIoU_Loss as the bounding box regression loss function—an essential aspect as this accelerates the bounding box regression rate and improves the algorithm's localisation accuracy, which is vital for precise object detection.

To validate the effectiveness of these improvements, Dong et al. conducted tests using the PASCAL VOC and MS COCO datasets. The results from these case studies are promising and include the following:

- Increased detection precision: The detection precision of the proposed model saw a significant increase of 3.2%.
- Reduced FLOPs: A 15.24% reduction in FLOPs indicates a more efficient computational process. The number of model parameters was reduced by 19.37%, suggesting a leaner, more optimised model structure.

Overall, the authors demonstrate the effectiveness and superiority of the improved YOLOv5 method for vehicle detection through comprehensive case studies and comparisons with the existing YOLOv5 model. The results indicate that the authors have successfully addressed the initial challenges, leading to a more efficient and accurate vehicle detection system.

However, integrating C3Ghost and Ghost modules, along with the Convolutional Block Attention Module (CBAM), into the YOLOv5 architecture increases the model's complexity. Like most deep learning models, the performance of this improved YOLOv5 method heavily depends on the quality and quantity of the training data. If the training datasets (like PASCAL VOC and MS COCO) do not adequately represent the variety of real-world scenarios, the model may not generalise well to different or novel environments. Adding sophisticated modules and mechanisms to improve accuracy might also lead to overfitting, especially if the model is trained on limited or highly specific datasets—overfitting results in an excellent performance on training data but poor generalisation for new, unseen data.

Despite reducing FLOPs and model parameters, the model may still require significant computational resources, particularly during training; this is a limiting factor for our work as we did not have access to high-performance computing facilities. While Dong et al. focus on reducing computational load, the real-time processing capabilities of the model in various scenarios, such as different lighting conditions, weather conditions or high-speed

environments, are not explicitly discussed. These are critical factors in vehicle detection applications, especially for autonomous driving systems. The model is optimised explicitly for vehicle detection. However, this specialisation might limit its adaptability or performance when detecting a broader range of objects in different contexts outside vehicle detection. There is also always a risk of inherent biases in the training data being transferred to the model. If the datasets used have any bias regarding vehicle types, sizes or environmental conditions, the model might inherit these biases, affecting its performance and reliability.

Another approach to the challenging problem of vehicle detection in autonomous driving systems (Chen et al., 2022) focuses on complex traffic scenes and the constraints posed by limited computing resources. Chen et al. propose an improved version of the Single Shot Multibox Detector (SSD) algorithm tailored for fast and accurate vehicle detection.

The authors chose MobileNet v2 as the backbone feature extraction network for the SSD framework. MobileNet v2 is known for its mobile efficiency and embedded vision applications, which likely contribute to the real-time performance improvements of the proposed algorithm. The authors introduced a channel attention mechanism for feature weighting. This approach is designed to enhance the model's ability to focus on more relevant features for vehicle detection, thereby improving its detection accuracy. A deconvolution module further enhances the model's performance. This structure aims to improve the model's ability to recognise vehicles of various sizes and in different parts of the image—a common challenge in complex traffic scenes. The model's effectiveness is demonstrated through its performance on two standard autonomous driving datasets, BDD100K and KITTI. The reported average precision rates of 82.59% for BDD100K and 84.83% for KITTI are impressive, indicating a high level of accuracy in vehicle detection. One of the standout features of the proposed algorithm is its inference speed. With a single inference time of 73 ms, the model is notably faster than the original SSD model. The authors claimed to achieve a balance between improved inference speed and enhanced prediction accuracy. This balance is critical in autonomous driving systems, where fast processing and high accuracy are necessary for safe and efficient operation. The authors thus effectively addressed the dual challenges of speed and accuracy in complex traffic environments by optimising the SSD algorithm with MobileNet v2 and incorporating advanced techniques like channel attention and deconvolution modules. The robust evaluation using standard datasets further underscores the practical applicability and effectiveness of the proposed solution.

While the authors present a promising approach to vehicle detection in autonomous driving systems by improving the Single Shot Multibox Detector (SSD) algorithm, there are potential disadvantages and limitations to consider. Integrating MobileNet v2, channel attention mechanisms, and deconvolution modules increases the complexity of the model, and the inference time of 73 ms is slower than the YOLO-based models.

Accurate real-time vehicle detection using YOLO is discussed by Nafiseh Zarei, Payman Moallem and Shams (2023), who use an innovative algorithm for vehicle position determination explicitly designed for urban traffic monitoring systems. The algorithm operates in two stages: detection and prediction. This separation allows for a balance between accuracy (achieved through detection) and speed (achieved through prediction), thereby offering a flexible solution that can be tailored to specific requirements. Using YOLO as the base for the detection network is a significant choice, reflecting the acceptance of this model in real-time inference problems.

YOLO is known for its speed and efficiency in object detection tasks. The design is tailored to be robust against changes in vehicle scale, which is crucial in diverse traffic scenarios. The detector network generates feature maps that are crucial to increasing detection accuracy. These maps are used for image segmentation into two classes, vehicle and background, utilising differential images and a U-Net-based module. This segmentation is a key step in accurately distinguishing vehicles from their surroundings. Differential images and a U-Net-based module for image segmentation allow for a clear distinction between vehicles and backgrounds. This distinction is critical for accurate vehicle position detection in complex urban environments. The algorithm classifies vehicle manoeuvres to enhance the recursive predictive network's performance and is done by concurrently considering the vehicles' spatial and temporal information.

Such an approach is more effective than methods considering these aspects separately and leads to better prediction accuracy. The simultaneous consideration of spatial and temporal data in classifying vehicle manoeuvres indicates a comprehensive approach to understanding vehicle dynamics and could significantly contribute to the system's predictive accuracy. The algorithm's performance was validated using the Highway and UA-DETRAC datasets, which are standard in urban traffic monitoring research. This validation demonstrates the algorithm's applicability and effectiveness in real-world urban traffic scenarios, where accurate and fast vehicle detection and tracking are essential. The use of advanced techniques like YOLO for detection, U-Net for segmentation, and the consideration of both spatial and temporal data for manoeuvre classification all contribute to the robustness and efficacy of the proposed system. The validation of the algorithm on standard datasets further underscores its practical applicability in urban traffic environments. However, this model is not designed to work on real-time predictions.

In contrast, Zarei et al.'s algorithm and the use of YOLO suggest potential for real-time application. Actual real-time capability would depend on the specific implementation details, the efficiency of the image segmentation process, the balance between detection and prediction, and the available computational resources. Further information or empirical real-time testing would be necessary for a definitive answer.

EnsembleNet (Mittal, Chawla and Tiwari, 2022) uses a hybrid vehicle detection approach and traffic density estimation based on faster R-CNN and YOLO models. Mittal et al. emphasise the growing importance of deep learning technologies, particularly CNNs and in-vehicle identification systems. This choice aligns with the current trend of leveraging advanced AI techniques for complex image recognition tasks. The researchers collected data from several open-source libraries, including MB7500, KITTI and FLIR. This diverse data collection is crucial for building a robust model. Image annotation was performed to classify vehicles into different categories, addressing an essential step in supervised machine learning. Various data augmentation methods were employed to tackle the class imbalance issue and increase the dataset size.

Additionally, the image quality was enhanced through a sharpening process, which is crucial to improving the model's ability to detect and recognise vehicles accurately. The paper introduces a novel hybrid model that combines Faster R-CNN and YOLO, utilising a majority voting classifier. This hybrid approach aims to leverage the strengths of both Faster R-CNN and YOLO, potentially leading to more accurate vehicle detection and counting. The proposed model, EnsembleNet, demonstrated a high detection accuracy of up to 98%, outperforming the base estimators YOLO (95.8%) and Faster R-CNN (97.5%). This comparison is essential to validating the effectiveness of the hybrid model over the individual models. One of the key findings is that the proposed model performs better in estimating traffic density compared to YOLO and Faster R-CNN.

While using a hybrid model of Faster R-CNN and YOLO presents a promising approach, combining Faster R-CNN and YOLO into a hybrid model increases complexity, rendering real-time predictions slower. Advanced deep learning models like Faster R-CNN and YOLO (Kim, Sung and Park, 2020), particularly in a hybrid format, could require significant computational resources for training and inference. While the model has been tested on specific datasets (MB7500, KITTI, and FLIR), its ability to generalise to various real-world traffic conditions, including weather, lighting, and traffic density scenarios, is not explicitly addressed. The reliance on open-source libraries for data collection might introduce biases or limitations inherent in the datasets. Additionally, image annotation is labour-intensive and could introduce human errors or inconsistencies. While data augmentation methods are used to address class imbalance, over-reliance on these techniques can sometimes lead to models that are less effective at generalising from augmented data to real-world scenarios. For traffic management applications, real-time processing is crucial. Kim et al. do not explicitly mention the real-time capabilities of the hybrid model, which is essential for dynamic traffic management systems. The high detection accuracy (up to 98%) reported might raise concerns about overfitting, especially if the model is excessively tuned to the specific datasets used for training and evaluation. While the hybrid model outperforms the base models YOLO and Faster R-CNN in

the tested scenarios, the comparative analysis might not fully account for all possible real-world scenarios and challenges.

YOLO-based models are becoming the standard for real-time detection. Ayush Dodia and Kumar (2023) address the limitations of traditional car object detection algorithms, particularly in generalisation capacity and recognition rate. Their research focuses on vehicle detection to manage vital traffic data, such as vehicle count and movement. It compares contemporary object detectors and traffic situation estimations, specifically examining different YOLO algorithm versions for optimal vehicle detection. The YOLO algorithm process involves clustering analysis for dataset optimisation and network structure enhancement to improve vehicle prediction. Ultimately, the study proposes an improved vehicle identification technique based on the YOLO algorithm and evaluates three versions for effective vehicle detection. Ayush Dodia and Kumar (2023) examine different versions of the algorithm—YOLO-v3, YOLO-v5 and YOLO-v7—for vehicle detection and counting in adaptive traffic light systems, assessing the accuracy of these versions, concluding that YOLO-v7 is the most effective, with 95.74% accuracy and a low frame rate of 3.5 ms per frame, facilitating faster object detection. The paper also notes the utility of YOLO-v5 and YOLO-v7 in detecting other objects, such as pedestrians and traffic signs for autonomous vehicles. It suggests that future work might explore YOLO v8, a promising next-generation facial recognition software with higher accuracy, speed, and broader identification capabilities, potentially setting a new standard for segmentation and object detection. However, YOLO v7 was designed to improve the efficiency of YOLO v5 in terms of generalisation ability, speed of inference and overall accuracy, yet YOLO v7 requires a significant increase in computational resources for both training and implementation.

Small object detection, such as distant vehicles, is addressed by Wen et al. (2023). They proposed LSD-YOLO, a variant of the YOLOv5 algorithm designed for higher accuracy and efficiency in detecting small objects. LSD-YOLO enhances small-scale feature maps to improve detection capabilities and introduces a new structure, FasterC3, to decrease network latency and parameter volume. LSD-YOLO incorporates coordinated attention to identify attention regions in complex driving scenarios and employs LeakySPPF, a spatial pyramid pooling method, to speed up computation by up to 15%. Additionally, Wen et al. (2023) presented a medium-sized dataset, Cone4k, to address category limitations in the VisDrone dataset. Extensive testing on the VisDrone2021 dataset demonstrated that LSD-YOLO achieves significant improvements in mean Average Precision (mAP) and F1 score, along with a reduction in parameter volume, making it a promising solution for enhancing small object detection in autonomous driving. The algorithm, primarily based on YOLOv5s and extended to other versions like YOLOv5m and YOLOv5l, utilises small-scale feature maps to enhance the detection of small objects. It integrates the FasterC3 module into the network for improved

accuracy and speed and adopts Focal-EIoU to address computational issues in CIoU. The research indicates that LSD-YOLO outperforms the original network, with a 4.6% improvement in mean Average Precision (mAP) and a 3.6% increase in F1 score, while reducing the parameter count by 7.5%. However, there is a compromise between increased accuracy and reduced inference speed.

### 2.2.3 Vehicle Intent classification at intersections

Methods for classifying vehicle intent vary with the specific application, and there is a large corpus of research on classifying vehicle intent from moving vehicles, such as autonomous vehicles or semi-autonomous driving aids. Amini, Omidvar and Elefteriadou (2021) and Afifah, Guo, and Abdel-Aty's (2023) research also emphasised the focused development of interconnected vehicles, which can, for example, pass current and intended trajectories between vehicles in the immediate vicinity and static infrastructure placed at intersections.

An early study into AV navigation of unsignalised junctions was carried out by Sezer et al. (2015) and has inspired our work. Sezer et al. (2015) addressed the interaction between autonomous driverless vehicles and human-driven vehicles, mainly focusing on merging scenarios at intersections. Their study uses Mixed Observability Markov Decision Process (MOMDP) tools to approach this interaction as an intention-aware motion planning problem. The key innovation here is the application of intention-aware planning frameworks to predict and adapt to human drivers' actions, specifically at T-junction intersections. A driver behaviour model was developed to calculate probabilistic state transition functions for the MOMDP model, thereby allowing the autonomous vehicle to make more informed decisions. The results suggest that this intention-aware approach significantly improves performance over current methods, notably reducing accident probabilities and intersection navigation times.

The results presented here demonstrate the application of a MOMDP in managing interactions between an autonomous vehicle (robot) and human-driven vehicles at an intersection. The critical aspect of this experiment is how the robot adapts its behaviour based on its perception of the human driver's intentions. In the first scenario, the MOMDP is used to interpret an aggressive driver's behaviour. The robot initially perceives the driver as aggressive and stops. However, as the driver's behaviour changes (slowing down and yielding), the robot updates its beliefs and eventually merges onto the road.

In the second scenario, the MOMDP method demonstrates its ability to handle a persistently aggressive driver. The robot initially stops, recognising the driver's aggressive approach. However, it then decides to proceed before the driver has passed the intersection based on its calculation of a low probability of conflict. This action is driven by the model's inclusion of a time penalty, encouraging more efficient navigation.

These results showcase the efficacy of the MOMDP approach in navigating complex and dynamic scenarios involving human drivers. By continuously updating its beliefs about the drivers' intentions, the robot can make safer and more efficient decisions, demonstrating a significant advancement in autonomous vehicle navigation in mixed-traffic environments. This approach enhances safety and improves traffic flow by reducing unnecessary stops or delays caused by uncertainty in interpreting human drivers' intentions.

The MOMDP framework, while innovative and valuable for the interaction of autonomous vehicles with human-driven vehicles, does have several disadvantages. Firstly, MOMDPs are inherently complex because they must account for many variables and potential states in dynamic environments like road traffic. This complexity can lead to increased computational requirements, making real-time decision-making challenging. Secondly, MOMDP effectiveness relies heavily on accurate sensing and prediction of human driver behaviour. The model might be trained and tested under specific conditions (like certain types of intersections, traffic densities or driver behaviours). Its performance could vary significantly under different conditions, which limits its generalisation to all real-world driving scenarios. As traffic density increases or scenarios become more complex (e.g., multiple human-driven vehicles with varying behaviours), the decision-making process's computational load and complexity increase, impacting the system's scalability in dense urban environments. While MOMDP aims to predict human behaviour, human drivers can be unpredictable and may not always behave according to the model's parameters. This unpredictability poses a significant challenge, although current machine and deep learning methods we explore in subsequent chapters promise to alleviate some of these issues.

Our research is focused on T-junction or intersection-specific models using static sensors; however, some of the approaches in the literature focused on AV intent and trajectory prediction using sensors such as radar and Lidar, and these have assisted in developing a framework to apply to our work.

Zhang et al.'s (2021) proposed method using an LSTM-based framework for intersection traffic focuses on predicting vehicle intentions and trajectories at intersections. Assuming that vehicle motion trajectories at intersections align with historical data once driving intentions are determined, the study establishes an Intersection Prior Trajectories Model (IPTM) by clustering and analysing extensive historical traffic flow trajectories. This model helps approximate the distribution of predicted trajectories and serves as a benchmark for credibility evaluation. The paper employs another LSTM model for intention prediction, a critical early-stage trajectory forecasting element which links with the IPTM and enhances the framework's predictability. Validated on the NGSIM and INTERACTION datasets, this framework significantly improves trajectory prediction accuracy by approximately 20% over other methods. However, LSTM models are inherently complex and require substantial

computational resources, making them expensive and challenging to deploy in real-time traffic management systems. In real-time prediction, any latency in processing and decision-making can be critical. The LSTM-based framework might face challenges in ensuring the minimal latency necessary to be effective in real-time applications.

Another approach to intent predictions at unsignalised intersections is a method called Multi-Modal Trajectory Prediction with Uncertainty (Zyner, Worrall and Nebot, 2018). The approach combines RNNs with a mixture density network (MDN) output layer designed to deal with the unpredictable nature of vehicle movements at intersections. Introducing a clustering algorithm to analyse the output probability distribution is innovative, helping to identify and rank possible paths. Using a real-world dataset with over 23,000 vehicles across five intersections adds significant strength to the study. It ensures that the findings are based on diverse, real-world scenarios. A Lidar-based tracking system for data collection is state-of-the-art, providing high-quality and reliable data. Using various metrics to assess the model's performance against several baselines demonstrates a comprehensive evaluation approach.

Testing the model on multiple intersections with a large dataset suggests good generalisability; however, the model's effectiveness heavily depends on the quality and diversity of the training data. If the data is not sufficiently varied or representative of all possible scenarios, the model's ability to generalise to new, unseen intersections might be limited. While practical, using RNNs and MDNs is computationally intensive, and valuable real-time predictions are impossible as inference time is in seconds, not the milliseconds required for our work.

A similar, improved method by Jeong et al. (Jeong, Kim and Yi, 2020) applies combined LSTM and RNN networks to predict turning behaviour at an intersection. Incorporating the subject vehicle's future states as an input feature to the LSTM-RNN is an innovative approach. It allows the model to consider the interaction between the subject and surrounding vehicles. The application results indicate improved recognition timing of the preceding vehicle, and the similarity of longitudinal acceleration with human drivers suggests practical benefits, implying that the system can make decisions more aligned with human driving patterns. More specifically,  (67.36%) of the acceleration error falls within ±0.5 m/s², and a significant majority (91.97%) is within the ±1.0 m/s² range, suggesting that the algorithm's performance is roughly aligned with human driving behaviours.

In contrast, the acceleration error distributions for the other models, like CV/Path and CTRV, were broader and less accurate in predicting a target vehicle's acceleration. These models struggled to accurately forecast the target's motion, resulting in a wider spread of error values. However, the Vflow/Path model showed a bias in the negative plane, demonstrating that the model—which assumes the vehicle follows the traffic flow—has limitations in responding to varying in-lane behaviours. This bias occurs because deceleration is usually more pronounced than acceleration in normal intersection driving conditions.

Wang and Shi (2020) used deep reinforcement learning to predict how a driver turns at an unsignalised intersection. Wang and Shi focused on the impact of intersection collision warning (ICW) systems on vehicle safety at non-signalised intersections. The authors developed a Matlab-based simulation platform to test their hypothesis and analysed various safety indices such as collision probability, conflict index, and collision rate. Their results reveal that vehicle safety at non-signalised intersections improves as the ICW system market penetration rate (MPR) increases. Significant safety benefits were observed even at a relatively low MPR of 20%.

With a 20% MPR and vehicles connected by vehicle-to-everything technology, there were reductions in collision probability (20%), conflict index (20%) and collision rate (35%).

The simulation method allowed for the establishment of a relationship between the ICW system MPRs and vehicle safety indices at non-signalised intersections.

The study acknowledges that the diversity and unpredictability of human driving behaviour could impact the effectiveness of ICW systems. The experiment was conducted in a simulated environment, as real-world field testing is difficult and risky. The paper concludes that the ICW system, when widely adopted, could significantly reduce collision metrics at non-signalised intersections; however, the variability in human behaviour and the need for real-world testing are areas that require further investigation. The simulation method presented in this paper offers us a framework for analysing the impact of intelligent vehicle technologies on road safety.

Hu et al. (Hu, Zhan and Tomizuka, 2018) proposed a semantic-based intention and motion prediction (SIMP) method to predict vehicle intention simultaneously. The authors highlighted that existing research often limits the scope of driving intentions to specific scenarios, failing to account for the diversity in real-world driving environments, and emphasised the need for an intention prediction method capable of adapting to various traffic scenarios given the multitude of possible driving manoeuvres. The method adopted improves prediction performance by integrating motion information with classified intentions and obtaining temporal information about predicted destinations, thus generating optimal trajectories for predicted and autonomous vehicles.

The SIMP method uses semantically defined vehicle behaviours to adapt to various driving scenarios. This method employs a deep neural network within a probabilistic framework to estimate intentions, final locations, and timing for surrounding vehicles.

Support vector machine (SVM) is primarily used for classification tasks but can also be adapted for regression. It works by finding the hyperplane that best separates data classes in a high-dimensional space. For non-linearly separable data, SVM uses kernel tricks to transform data into a higher dimension where it is separable.

Quantile random forest (QRF) is an extension of the random forest algorithm, an ensemble

learning method. While a standard random forest generates predictions by averaging the results of multiple decision trees, a quantile random forest provides a distribution of possible outcomes. This is beneficial for estimating the uncertainty and variability in the predictions.
SIMP was compared to baseline SVM, and QRF models in two representative driving cases. The SIMP method reportedly outperformed these models in prediction accuracy and confidence intervals. A key conclusion is the efficacy of combining different prediction tasks using semantics in a single framework, which allows for generalisation to various traffic scenarios and competitive performance against traditional methods.
DESIRE (Deep Stochastic IOC RNN Encoder-Decoder; Lee et al., 2017) is a framework for predicting the future positions of multiple interacting agents in dynamic scenes. It is an end-to-end trainable neural network model incorporating a deep IOC (inverse optimal control) framework. The framework accounts for the multi-modal nature of future predictions, acknowledging that the same current context can lead to different future outcomes. DESIRE can foresee potential future outcomes and make strategic predictions based on accumulated future rewards, akin to IOC frameworks. DESIRE considers past motion histories, semantic scene contexts, and interactions among multiple agents, thus providing a holistic view of the scenario. Despite its complexity, DESIRE maintains computational efficiency and iteratively refines predictions to boost accuracy using a conditional variational autoencoder component to generate a diverse set of hypothetical future prediction samples.
The model was evaluated using two publicly available datasets: KITTI (focused on vehicle interactions) and the Stanford Drone Dataset. The model's prediction accuracy was benchmarked against other methods and demonstrated significant improvements, particularly in scenarios with rich interactions. While DESIRE's top 1% sample may show higher error than direct regression baselines, multiple samples (e.g., top 10%) yield much better prediction accuracy. DESIRE offers accurate long-term predictions in complex scenes, integrating static and dynamic contexts.
In our work, we simplified the prediction algorithm regarding the number of agents and built on the work of Wang et al. (2020), Hu et al. (2018) and Lee et al. (2017) to generate our feature vectors.
To classify intent based on historical feature vector data, most models in the literature have the same goal: to represent a conditional probability distribution P (o | f ) outcome at a junction given a vector of features f. However, these models differ in how they extract features. The literature describes challenges in developing pragmatic intersection intention prediction models. These challenges include developing models that can generalise across unseen intersections and drivers, predict over longer prediction horizons, and do not require driver eye-tracking or similar inputs. All models are probability distributions rather than mere classifiers, which allows them to capture the inherent stochasticity in human driving behaviour.

Benchmarking is common across the literature, giving a certain degree of generalisation.
Our tailored video dataset uniquely addresses the specific needs of our study as existing benchmarks do not adequately cover all potential vehicular scenarios. This approach mirrors Ristani et al. (2016), who developed customised benchmarks for various video tracking and analysis situations. They introduced innovative precision-recall performance measures that uniformly tackle different errors while emphasising accurate identification. Their significant contribution includes compiling the most extensive, fully annotated, calibrated dataset for multi-target, multi-camera tracking. This dataset encompasses over two million high-definition video frames, recorded using eight cameras and tracking more than 2,700 distinct identities.

### 2.2.3 Non-predictive, reactive methods of accident mitigation

Existing accident mitigation technologies, such as automatic emergency braking (AEB), are reactive measures that activate when there is an imminent collision between a vehicle and an object directly in their path. This issue is explored in the context of motorcycles by Savino et al. (2016), who made a significant advancement in motorcycle safety by extending the implementations of the inevitable collision state (ICS) theory to motorcycles. Savino et al. (2016) successfully adapted ICS theory, which has predominantly been applied to cars for use with motorcycles to consider the unique avoidance capabilities of motorcycles, allowing for more accurate predictions in motorcycle-to-car crash scenarios. The findings enable the development of more sophisticated and effective motorcycle safety systems, such as AEB and airbags. By integrating this advanced understanding of collision inevitability, these systems can be designed to activate only when a collision is physically unavoidable. Improving time-to-collision (TTC) analysis allows for better analysis of TTC values when assessing the effectiveness of safety systems before a crash occurs.
Savino et al. presented simulation results for 10 motorcycle crash cases. They traced the relationship between TTC and actual impact speed, as well as impact speed reduction versus actual impact speed. Their results were based on motorcycles travelling in a straight line and not performing lateral avoidance manoeuvres. It also considers only a single opponent car, although the approach remains conservative in scenarios with multiple obstacles. Their study demonstrated the practical feasibility of applying ICS to real-world motorcycle crashes and suggested a lookup table approach for implementing such systems; the simulations also provided quantitative estimates of potential impact speed reductions.
Motorcycle autonomous emergency braking (MAEB) systems were reviewed by Haufe et al. (2021). MAEB technology is similar to autonomous car braking that applies braking force automatically to reduce impact speed in emergencies, and it is designed to reduce the severity of motorcycle crashes by applying autonomous braking during emergencies. Haufe et al.

(2021) evaluated how acceptable and controllable these automatic braking events are to typical riders in realistic riding scenarios. Fifty-five riders participated in field tests on three different motorcycle types. An investigator triggered The MAEB system remotely during four specific riding manoeuvres at speeds between 35–50 km/h.

This research was pivotal as it evaluated the impact of MAEB systems on safety enhancement as well as their feasibility and acceptance among regular riders. It advanced comprehension of the importance of integrating cutting-edge safety technologies into motorcycles—a mode of transport often linked with increased hazards. However, the study's relatively large participant pool may not fully encapsulate the entire spectrum of motorcycle users. The effectiveness of the AB interventions was assessed under controlled conditions, which may not accurately mirror real-life situations. Additionally, some participants expressed a preference for testing anticipated AB interventions before unanticipated ones, suggesting a possible experimental bias due to anticipation. These constraints, especially the limited diversity of the sample group and the artificial test environment, underscore the necessity for additional research in more varied and realistic settings to corroborate the results.

Our investigation into predicting vehicle intent at T-junctions parallels the studies of Savino et al. and Haufe et al. In this context, by forecasting a vehicle's intention at a junction, MAEB can be activated before a motorcycle reaches an unavoidable collision state, thereby enhancing TTC. Further research would help to answer this question: Could augmenting a motorcyclist's reaction time through a predictive intent factor, alongside MAEB, prevent a collision or substantially lessen its severity?

## 2.3 Chapter summary

Our review of related work explored various models and approaches to build a complete pipeline for intent prediction at T-junctions. We studied behavioural aspects, object detection, action recognition, semantic segmentation, hybrid approaches, reinforcement learning and end-to-end learning approaches. We also evaluated existing models and algorithms in the context of vehicle detection and intent prediction. This chapter describes various methods for predicting vehicle intentions and motions at intersections and gives a comprehensive overview of the techniques and methodologies used in vehicle intent prediction, focusing on applying YOLO algorithms for vehicle detection and classification. We also highlighted the challenges and limitations of current technologies, created a custom dataset, and evaluated various predictive models in the context of road safety and traffic management.

# Chapter 3: Creating a Target Vehicle Video Dataset

## 3.1 Introduction

The comprehensive literature review in the preceding chapter highlighted a crucial gap in the available video datasets relating to unsignalized junctions in the United Kingdom. Several video datasets, such as those by Hadi Ghahremannezhad, Shi and Liu (2023), feature intersections, roundabouts, and crossroads captured by stationary cameras, whereas others, like Oxford's Robocar dataset (2020) and KITTI datasets (Geiger, 2023), are collected from mobile platforms; we analysed these datasets to devise our methodology. However, relying on an off-the-shelf dataset that did not specifically focus on UK T-junctions would unduly restrict our research by introducing a generalisation that would dilute our focus. Therefore, we created a bespoke dataset and video specifications tailored to our research objectives. To our knowledge, no existing datasets fulfil our specific UK T-junction requirements, which are target vehicles from a left-hand drive perspective approaching a major road from a minor road where the target vehicle must yield to traffic on the major road. Hence, creating a unique video dataset was imperative to serve both as the benchmark for evaluating detection and classification models and as the primary input for our analytical pipeline.

The decision to collect T-junction video data stemmed from recognising a critical gap in existing datasets related to unsignalised T-junctions in the United Kingdom. This gap, identified through a comprehensive literature review, underscores the need for original data collection to support our specific research goals and hypotheses. Rather than relying on off-the-shelf datasets or simulators, we opted to create a bespoke video dataset tailored to the unique requirements of the research objectives. This decision reflects a commitment to methodological innovation and ensures that the collected data are specifically designed to address the research questions. The T-junction video data collection process involved extensive fieldwork, including visits to multiple T-junctions across the southern UK. This approach was characterised by methodological precision, systematic data collection protocols, and adherence to ethical guidelines regarding data privacy and consent. Rigorous quality assurance measures were implemented throughout the data collection process to ensure the data's integrity and reliability. This included meticulous documentation of ground truth annotations, verification of data accuracy, and ongoing validation of data quality against established benchmarks. The real data collected from T-junctions provides a rich and diverse dataset that captures the complexities and nuances of real-world vehicular behaviour. Unlike simulated data or existing datasets, the T-junction video data offers insights into traffic

patterns, driver behaviours, and environmental factors that influence vehicle interactions at T-junctions. The collected T-junction video data is the foundation for subsequent analysis, modelling, and experimentation. The collection of T-junction video data represents a significant scientific step involving methodological innovation, rigorous data collection protocols, and a commitment to advancing knowledge in the field.

Opting to construct our unique video dataset rather than relying on a pre-existing dataset or a simulator introduced a substantial workload but offered several notable advantages to our research. Firstly, our video dataset is explicitly tailored to our domain of interest. Existing video datasets and driving simulators struggle to encompass the precise scenarios and conditions essential for our research. The real-world data we collect includes a broader spectrum of diversity and variability compared to simulations. Simulators may not necessarily fail due to inherent limitations but because they lack the necessary data to model real-world scenarios' complex and unpredictable dynamics accurately. When researching vehicle behaviours before a T-junction, distinguishing between those that will yield and those that will not pose a challenge, especially in the absence of existing literature on the subject. Without concrete data or understanding of these behaviours, simulators cannot replicate these effects accurately, reducing their output to educated guesses rather than precise simulations. This limitation does not stem from the simulators' inability to capture real-world intricacies but from the current gap in our knowledge about specific vehicle behaviours. Collecting our data captures the subtleties, unforeseen events, and variations necessary for building robust models.

Furthermore, published datasets sometimes suffer from constraints such as limited or outdated ground truth data. Developing our dataset ensures that our ground truth annotations remain accurate and current, which is vital for training and further evaluation within our pipeline. Published datasets or simulators also may not fully address privacy and ethical concerns linked to the use of real-world data. The decision to create a unique video dataset stems from a gap identified in existing literature regarding the availability of datasets tailored to UK T-junctions. By acknowledging this gap and taking proactive steps to fill it, the research demonstrates a commitment to addressing fundamental challenges in the field. Constructing our dataset empowers us to collect data responsibly and ethically, respecting privacy and consent requirements. Another notable advantage is that we possess complete ownership and authority over the data, which offers the option to commercialise our research or leverage the data for strategic purposes. Despite our extensive search, we could not identify a combination of existing datasets or driving simulators that would adequately suit the unique demands of our research.

It is essential to emphasise that creating our video dataset is a foundational step within this undertaking rather than its ultimate objective. Our overarching aim was to extract pixel-level

feature vectors from vehicles, culminating in a comprehensive feature vector dataset that will be instrumental in training a predictive model. This dataset will continuously update with new feature vector data to enable real-time predictions, facilitating our research objectives.

Benchmarking is a well-established practice in the literature and often contributes to the generalisability of research outcomes. Ergys Ristani et al. (2016) developed tailored benchmarks encompassing a broad spectrum of video tracking and analysis scenarios. These benchmarks were deployed in response to the varying contextual conditions of assessments carried out by subsequent researchers. With this work, it is imperative to construct a vehicle-specific video dataset of unsignalized T-junctions in the UK to create benchmarks and evaluate our model on relevant data. This chapter describes our methods for acquiring high-quality T-junction-based video data. Extensive fieldwork involving visits to multiple junctions across the southern UK yielded raw video material that encompasses the topography of junctions, thereby providing a holistic view of vehicle behaviour at and near these junctions. Our video dataset forms the foundational stage of our pipeline, as all stages of the pipeline rely on the quality of our video data for baseline reference.

**The research question this chapter addresses is**

**RQ0**: Is it feasible to collect real-world video data from T-junctions that can accurately inform the development of a vehicle intent model for predicting vehicular behaviour?

The novel contribution covered in this chapter is

- The creation of a data-rich video dataset on unsignalised UK T-junctions. This has formed the content of a research paper that is due to be published shortly.

### 3.1.1 Organisation of the Chapter

Firstly, we introduce the rationale underpinning the creation of the video dataset comprising unsignalized T-junctions in the United Kingdom. Secondly, we describe the unique topographical characteristics of these junctions and illuminate the challenges encountered when positioning cameras to capture the data. Lastly, we provide details on the formatting and storage of the video data used in Chapter 4.

## 3.2 Junction Dataset Considerations

Finding patterns in traffic is stochastic; therefore, our video dataset needed to contain sufficient data to encompass the full spectrum of traffic over a range of locations and times. Drawing upon the insights presented by Krajewski et al. (2018), the measurement methods employed must not influence users to ensure the preservation of realistic behaviour; this entails avoiding

the use of visible sensors that resemble traditional traffic surveillance cameras because such devices could alter the behaviour of those being observed. In our case, our empirical observations at the experimental junction sites indicated that most drivers were unaware of our presence; as the majority of drivers drove past, we saw no acknowledgement of the camera and our researcher was positioned out of view and nearly invisible to road users as they passed. Additionally, the dataset must be scalable, allowing data to be added and diversified. Diversity should also extend to the recording sites and times of data collection. Measurements should be taken from multiple recording sites at different times to systematically cover various road layouts, traffic rules, and traffic densities; we cover several junctions at various times to diversify our data.

Furthermore, it is crucial that these measurements occur predominantly on public roads rather than private property to ensure the practical relevance of the dataset, particularly in the context of accidents on public roads, which are very hard to replicate realistically. The dataset must encompass all types of road users without limiting itself to specific categories such as pedestrians or cars. A comprehensive approach involves tracking every road user, given that their interactions and influence on each other are vital components of real-world traffic scenarios. Lastly, to recognise the significant impact of road layout and local traffic rules on road user behaviour, it is essential to include detailed infrastructure records in the dataset. By incorporating precise information about road layout and traffic regulations, the dataset can better reflect the real-world conditions that shape road user behaviour.

### 3.2.1 Types of unsignalized junction

Unsignalized junctions come in various forms and are devoid of strict traffic control measures like traffic lights or stop signs. Examples include roundabouts, crossroads, staggered intersections, and T-junctions. This research centres on the T-junction layout, a ubiquitous unsignalized junction in the United Kingdom. A T-junction is where a minor road intersects with a major road. as depicted in Figure 2. Traffic approaching on the minor road must yield to traffic on the major road from both directions.

Of all the junctions on UK roads, T-junctions have the highest incidence of accidents, according to multiple sources in 2019 Statista data (Statista, 2023.).

This thesis's primary objective is to predict a vehicle's intention to stop or merge, when approaching the major road from the adjoining minor road based on data on merging from the minor road onto the major road at T-junctions.

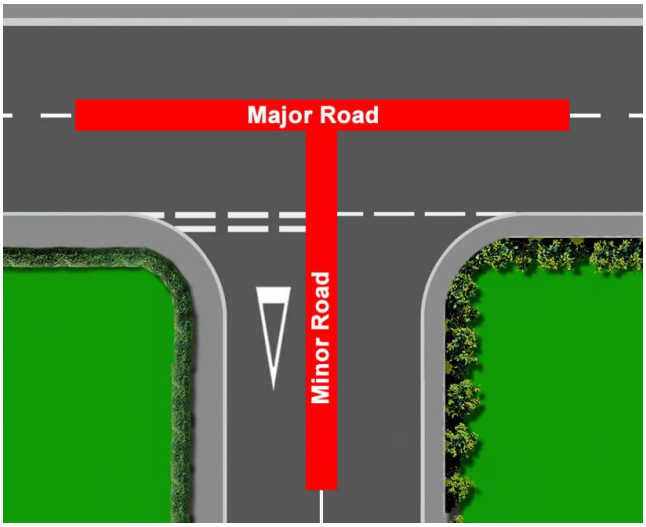



*Figure 2: T-junction configuration in the UK.*

Traffic coming from the minor road onto the major road can take two distinct paths: 1) a left turn, merging seamlessly with the flow of traffic without crossing any lanes, as illustrated in Figure 3; or 2) a right turn, necessitating a crossing of oncoming traffic from the right side, as depicted in Figure 4. Our video dataset encompasses both scenarios, capturing vehicles on the minor road as they approach the T-junction and vehicles already in motion on the major road. Our junction video dataset is used to gather and process feature vector data from vehicles at the T-junction. This processed data is then forwarded to the next stage in our data analysis pipeline. Our primary aim is to closely examine the conduct of vehicles as they approach the T-junction from the minor road and, when a vehicle is identified as posing a potential hazard to the safe passage of drivers on the major road, issue timely warnings.

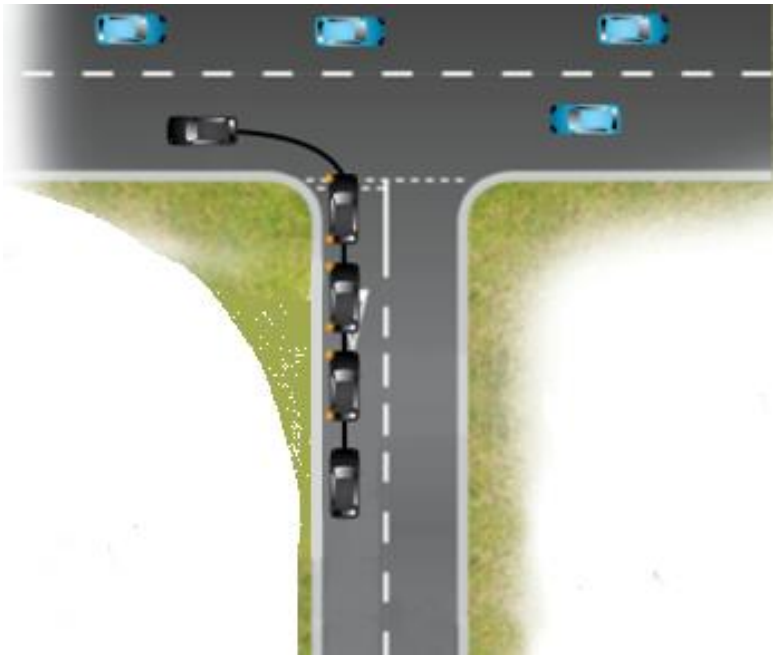


*Figure 3: Left turn from minor to major road.*

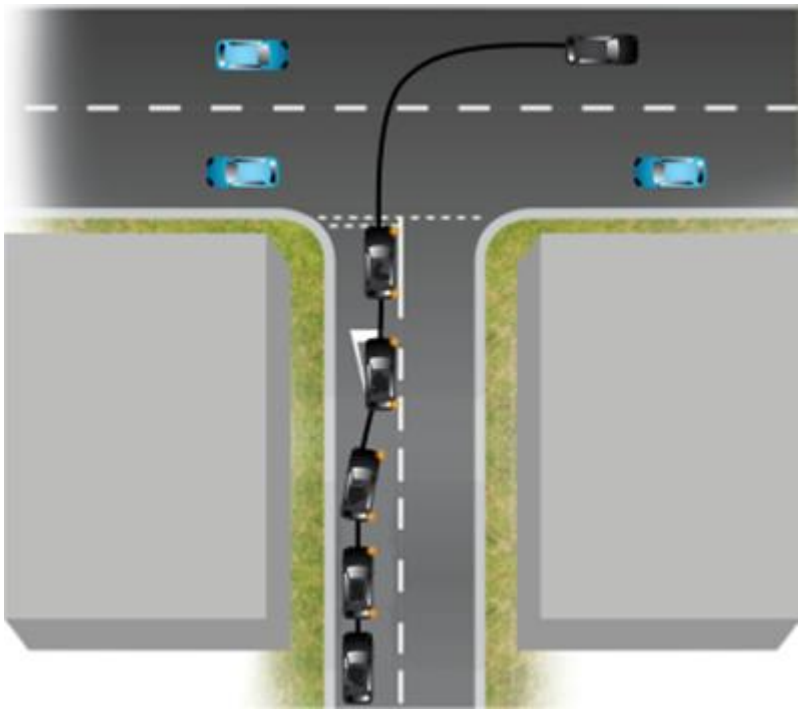


*Figure 4: Right turn across traffic from minor to major road.*

## 3.3 Selecting Experimental Junctions

Our live experimental junctions were chosen based on several critical criteria. Firstly, traffic density played a crucial role in our selection process. In the initial phases of our investigation, certain junctions were ruled out due to traffic intensity. Some were deemed too quiet for meaningful traffic interaction, while others were so congested that traffic movement was minimal and a smooth flow was virtually non-existent. While adjusting the time of day at these junctions could partially alleviate this issue, our research required a more generalised approach. Consequently, we identified more suitable junctions that offered a better balance and mixed traffic flow.

One of our key objectives is to implement a system that issues warnings to traffic on a major road when a vehicle from a minor road fails to stop while entering the T-junction. This specific behaviour is a significant contributor to the majority of accidents that occur at T-junctions. As a result, our research focuses on accurately modelling and capturing the stochastic nature of traffic conditions at these high-risk junctions. High-risk T-junctions, known for posing

challenges to road users, are characterised by a notable incidence of accidents; we sourced relevant data in this regard from the Road Safety Foundation (RSF) EuroRAP website (rsfmaps.agilysis.co.uk, 2023). The accident data from this source covers two distinct periods, spanning from 2015 to 2017 and 2018 to 2020 (inclusive). Post-2020 data yields a reduced accident rate, likely due to changing driving habits during the pandemic. An illustrative example of this data is available in Figure 5, where visual representations showcase the locations of fatal and serious accidents on roads in the UK.

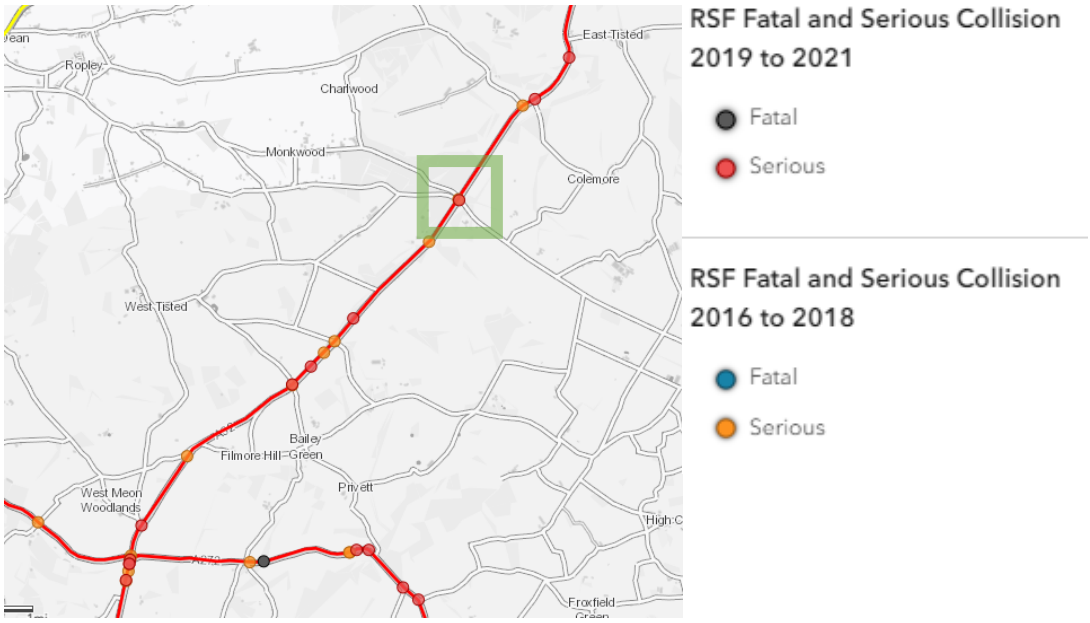



*Figure 5: One of our experimental junctions, JM599, inside the green box, as depicted by RSF EuroRap. Accident sites are marked as circles.*

Our video data collection process involves recording raw video footage of traffic at the designated junction, capturing the junction infrastructure, and providing a comprehensive view of all traffic within and surrounding the area. Our initial survey involved remote analysis before the junction was selected as a suitable location—an image of the junction JM599 from a satellite view is in Figure 6, and the camera position is in Figure 7.

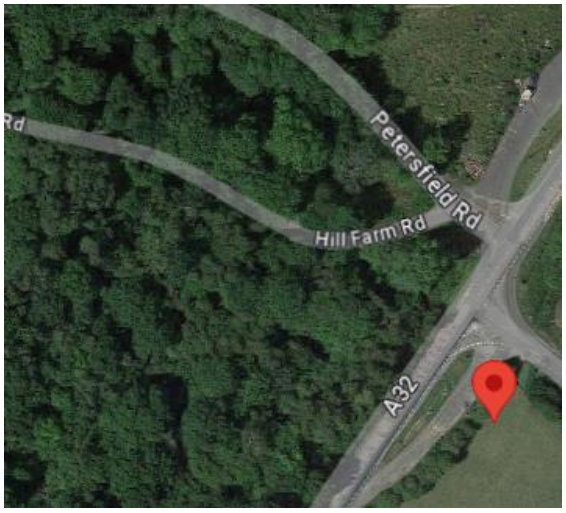



*Figure 6: Satellite image of one of our selected junctions depicted in Figure 4.*

*Figure 7. Our camera is discretely positioned to capture maximum data from a height of 3 m, as seen in the left image; a view from the camera can be seen in the right-hand image. Raw video ($V_r$) is recorded and optimised as a manual process generating an optimised video ($V_o$). The capture of $V_r$ involves stationing a camera at specific test locations and recording at a frame rate of 50 frames per second with a resolution of 1920x1080. The camera is directed towards the approach to the junction's stop line.*

Our initial survey identified four locations based on traffic volume, composition, and a relatively high accident record (as indicated in Table 1). A safety plan for video recording was established for each of these locations, which also involved obtaining ethical approval for recording on public highways. Additionally, one more location was selected based on data indicating a relatively low accident rate, with no accidents recorded between 2018 and 2020. This location holds significant value in our pipeline's intent prediction training stage because predictions made using data from this junction are expected to be particularly insightful, as

discussed in Chapter 5.

| EuroRAP Route | Fatal and Serious Accidents 2015–2017 | Fatal and Serious Accidents 2018–2020 | Junction Location | Fatal and Serious Accidents at a T-junction 2018–2020 | FSC Risk Rate /250 |
|---|---|---|---|---|---|
| JM377 | 3 | 9 | Oxshot Road | 2 | 106.9 |
| JM384 | 26 | 25 | A248 | 3 | 88.9 |
| JM559 | 33 | 34 | Petersfield Road | 1 | 121.5 |
| JM454 | 10 | 15 | Rowhook Road | 2 | 115.9 |
| UO196 | 5 | 0 | Jacobs Well Road | 0 | 0 |

*Table 1: Fatal or Serious Collision (FSC) Risk Rate (the number of FSC per billion kilometres travelled by vehicles along the route).*

## 3.4 Camera position at a T-junction and point of view (POV)

All our video recordings were produced with a Garmin Virb 360 Camera, which can capture 360˚ footage of 1920x1080 pixels using two lenses simultaneously. This high-definition resolution ensures clear and detailed visuals. Furthermore, the videos were recorded at a frame rate of 50 fps, allowing for a reduction in fps during video processing if necessary for tasks like maintaining optimal inference times later in the processing pipeline. This combination of high resolution and high frame rate contributes to the overall quality and versatility of the recorded footage. In alignment with Krajewski et al. (2018), our experiments encompass the entire junction space, taking into account all aspects of traffic and the junction's geography. We conducted trials with various camera points of view and subsequently assessed the raw video footage. Given the need to capture the complete topography of the junction, an ideal view would be 360˚. However, achieving a 360˚ view necessitates the combination of two 180˚ views, which unavoidably introduces some distortion at the stitch areas and, thus results in occlusions in the detection zones.

A secondary concern arises when attempting to detect and classify vehicles in a busy environment from a stitched 360-degree point of view. The increased visible traffic from an additional lens leads to a significant increase in the average inference time per frame, jumping from 20 ms to a maximum of 70 ms due to the higher number of detections required in each frame. The subsequent images in Figures 7–10 relate to our camera point-of-view trials.

### 3.4.1 Experiments with camera points of view (POV)

To establish the optimum camera position for all junctions in this study, we conducted a series

of experiments from different points of view (POVs) using junction JM377, as shown in Table 2.

The primary target vehicles in Figures 8, 9, 10 and 11 are those approaching from direction (a), and they are primarily responsible for most accidents when they enter the major road and collide with traffic coming from the right (b) or left (c). Traffic approaching from (b) is at risk from the target traffic (a) if they are not visible while making a right or left turn onto the major road. Similarly, traffic approaching (c) is at risk from target traffic (a) if they are not seen and are making a right turn onto the major road. Our objective is to predict the behaviour of target vehicles approaching from direction (a) before they cross the give-way line. One of the significant challenges we face is the limited time available to capture, analyse, and predict the behaviour of vehicles from direction (a). At the most hazardous junctions, target traffic from direction (a) has a restricted view until they are very close to the give-way line, providing only a few seconds for the entire process. Our video dataset needs to cover the approach to the junction from direction (a) as much as possible and include comprehensive topographical and other traffic details in each frame.

| Label | Definition |
|---|---|
| POV | Camera point of view. The direction the camera is facing. |
| a | Target vehicle approaching from a minor road. |
| b | Traffic approaching the T-junction from the right on the major road. |
| c | Traffic approaching the T-junction from the left on the major road. |

*Table 2: Key to Labels in Figures 7, 8, 9, and 10.*

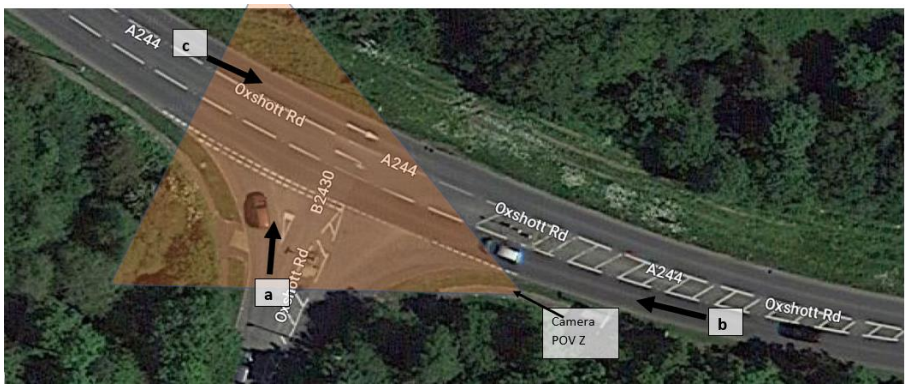



*Figure 8 Camera POV z*

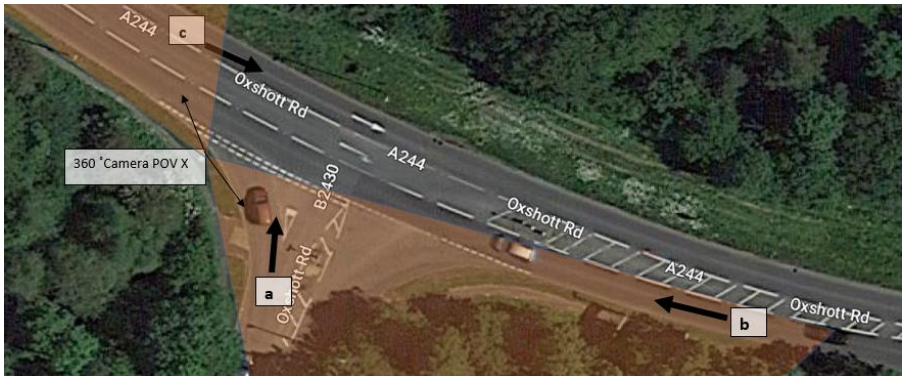



*Figure 9 Camera POV 360˚x*

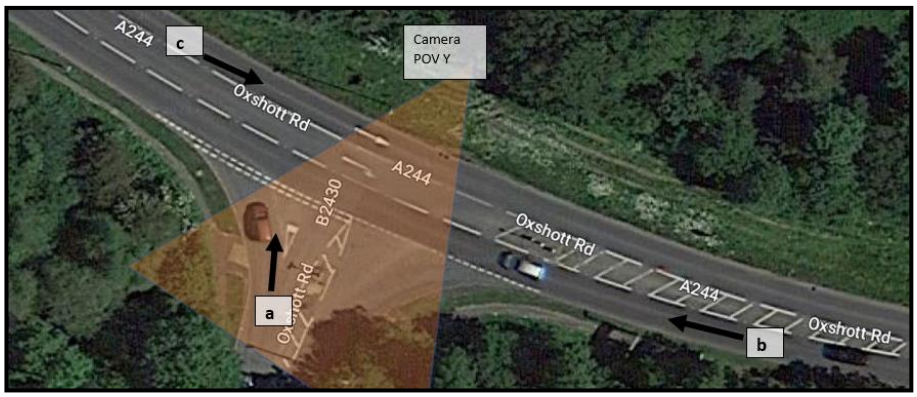



*Figure10 Camera POV y*

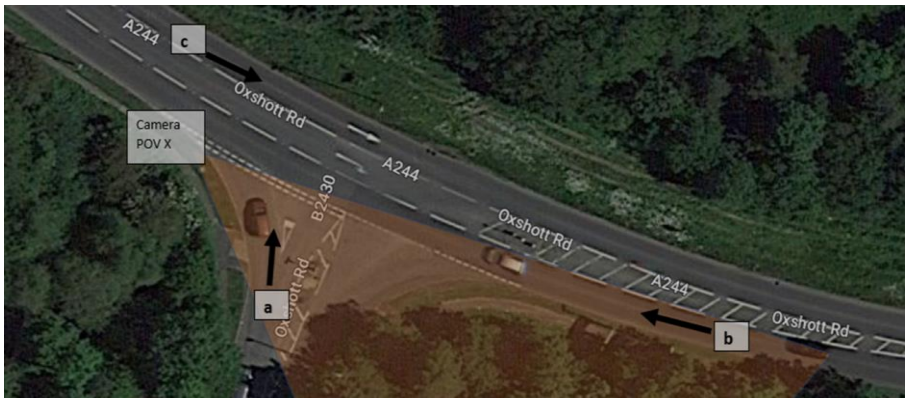



*Figure 11 Camera POV x*

An overview summary of our POV experiments is presented in Table 3. These experiments combined subjective assessments with empirical data. The selection of the POV was influenced by what a human observer would naturally see from the given perspective and how well the POV video performed in beta testing, particularly in terms of inference time. After examining the recordings listed in Table 3 and evaluating their appropriateness for video data collection, we selected Point of View (POV) X. By choosing POV X, we aim to maximize inference times for our target vehicles by minimizing the potential for occlusions. Each occlusion could increase the computational burden by necessitating separate, unnecessary inferences. From the perspective of directions y and z, passing traffic creates obstructions, and it has been observed that when traffic decreases in speed and occasionally comes to a complete halt, all target vehicles approaching from the direction a are occluded.

| Camera POV | Experimental summary from POV study | | | |
|---|---|---|---|---|
| | Direction a | Direction b | Direction c | Comment |
| x | Right | Frontal | Passing | There is a good balance between a clear view of a whole junction and minimising target vehicle occlusions. |
| y | Frontal | None | None | A clear view of target vehicles. However, a is frequently occluded by traffic from b c. |
| z | Left | Passing | Frontal | Good view of c and of b passing or turning into junction. b occludes the camera from target vehicles if traffic turns left into the junction. |
| 360 x | Right | Frontal | Frontal | Data-rich view. Overwhelming when attempting to isolate the target vehicle. Occlusions in the stitch sector of the joint lenses. |

*Table 3. Camera POV summary*

We recorded all the videos for this dataset from POV x, which was chosen to balance data quality, minimal occlusions, and an optimal view of traffic and junction topography. All video data utilised in this dataset was collected from POV x at each junction.

## 3.5 Constructing the video for the dataset

After the junctions were selected, the camera position was established, and a safety plan was drafted, we conducted multiple visits to each junction listed in Table 4. These visits occurred at different times of the day, with varying traffic densities and light conditions. The objective was to create individual junction datasets, each named after the EuroRAP route on which they are situated. Additionally, we compiled a combined dataset named 'Bo', which included all the junctions.

We placed our tripod in a similar position to ensure consistency and to maintain the same vantage point for each visit to every junction. We achieved this by marking the tripod legs' positions on the ground using long-lasting biodegradable chalk, thus allowing for accurate camera position calibration.

| EuroRAP route | Junction location | Combined hours of data collected over multiple visits |
|---|---|---|
| JM377 | Oxshot Road | 4.5 |
| JM384 | A248 | 2 |
| JM559 | Petersfield Road | 4 |
| JM454 | Rowhook Road | 3.5 |
| UO196 | Jacobs Well Road | 2 |
| Total hours of data in Bo | | 16 |

*Table 4. The storage structure of our video dataset in the raw form was prepared for the next stage in our pipeline.*

All the video data was collected and preserved in the MP4 format. The videos were organised into junction-specific collections and consolidated into a comprehensive video repository named Bo.

## 3.6 Chapter conclusion

**Research question revisited: RQ0**: *Is it feasible to collect real-world video data from T-junctions that can accurately inform the development of a vehicle intent model for predicting vehicular behaviour?*

The process described in selecting the Point of View (POV) for our experiments directly relates to the feasibility of collecting real-world video data from T-junctions to develop a vehicle intent model for predicting vehicular behaviour. By meticulously considering factors such as visibility, occlusion potential, and computational efficiency, we aimed to ensure that the collected video data would accurately capture the dynamics of vehicular behaviour at T-junctions. Our approach underscores our prioritisation of optimizing the data collection process to maximize the effectiveness of the collected data in informing the development of a vehicle intent model. Therefore, through careful selection of the POV and thorough evaluation of its suitability, we aimed to demonstrate the feasibility of collecting real-world video data from T-junctions that can accurately inform the development of a vehicle intent model for predicting vehicular behaviour.

Our contribution revisited:

- The creation of a data-rich video dataset on unsignalized UK T-junctions.

This dataset is designed to enable the extraction of accurate feature vectors from vehicles and will be made available to the research community in conjunction with our published paper.

This chapter elucidated the methodologies employed to craft a tailored video dataset aligned with our specific requirements. In Chapter 4, we delve into the object detection and classification techniques applied to the consolidated video repository (referred to as 'Bo') to extract precise vehicle feature vectors. Chapter 5 focuses on extracting feature vectors from pixel-level data within the individual video files. These feature vectors will collectively form a dataset that serves as the basis for predicting driver intent.

# Chapter 4: Selection of Target Vehicle Detector

## 4.1 Introduction

A vehicle object detector and classifier is a complex system that uses machine learning and deep learning computer vision algorithms to identify and categorize vehicles in images or video feeds. It works through two main steps: object detection and classification. Object detection uses algorithms like CNNs or R-CNNs to find vehicles and mark them with bounding boxes based on features such as edges and colours. Classification then assigns these detected vehicles to specific categories (e.g., car, truck) using a trained model on a labelled dataset. Accuracy depends on the quality of training data, algorithm sophistication, and computational power. However, challenges like vehicle appearance variability, environmental factors, and real-time processing demands exist.

While the main aim of this thesis is not to enhance the performance of detection and classification models, conducting a series of experiments was essential to identify the most effective vehicle detection model for our pipeline. 'Effective' in this context means that we can detect vehicles in real-time, locate them in the video frame and classify the type of vehicle. As discussed in the previous chapter, no viable unsignalized T-junction datasets were available for our work, and we had to develop our own original dataset; therefore, no benchmark data is present that we can build. In this chapter, we discuss how using our video dataset Bo and various detection models, we iteratively singled out the most effective vehicle detection models for our work based on our data and requirements.

Given the wide range of vehicle detection and classification models analysed in the literature

review in Chapter 2, this thesis demands prioritising rapid inference while upholding acceptable accuracy levels. Given our focus on detecting vehicles, we have prioritised models designed for swift inference, as detecting smaller objects necessitates redundant additional computational processing. Two of the most prominent detection and classification models in this context are Faster R-CNN and YOLOv5. In real-time vehicle detection and classification using 2D video, most literature suggests using YOLOv5; this aligns with our objectives because YOLOv5—coded in Python, a language we are familiar with—offers a variety of pre-trained models tuneable to our requirements. Our version of Faster R-CNN is also written in Python and has a range of pre-trained models tunable to our requirements. While Faster R-CNN has been shown to be much more accurate in detecting smaller objects than YOLOv5, as detailed in Chapter 2, it does not perform as well in inference time.

Our initial evaluation of our faster R-CNN demonstrated promising results using our data in inference speed and vehicle detection accuracy. In contrast, our initial assessment of YOLOv5 revealed certain areas needing improvement when incorporated into our workflow. Instead of hastily selecting the theoretically superior model, we conducted a brief study utilising our dataset to perform a precise real-time performance analysis. Per many of the studies in Chapter 2, faster R-CNN and YOLOv5 can detect and classify vehicles in real-time. However, an additional neural network is required to track vehicles processed from faster R-CNN or YOLOv5 detections. This further computational load significantly impacts our work's real-time element as we introduce multiple subsequent steps into the target of real-time driver predictions. Our overall goal is to strike a balance between detection accuracy and inference speed. In this section, we clarify the architectural complexities of Faster R-CNN and YOLOv5 and undertake a comparative analysis of their performance using our video dataset, Bo.

### 4.1.1 Organisation of the chapter

This chapter is organised as follows. Firstly, we introduce our test models, Faster R-CNN and YOLOv5. Secondly, to select the optimal detection classification model, we quantitatively compare the models' inference time and accuracy based on our test models using our own Bo video dataset. Then, we create a bespoke image data set from our Bo data and open sources, focused on images of our target vehicles at UK junctions. We use this to train our chosen model to improve accuracy by enriching the classification training data to ultimately define our vehicle detection model specification.

**The research question this chapter addresses is**

Research Question 1 **(RQ1):** *How does employing a constrained and focused dataset affect the training process and subsequent real-time performance of object detection and*

*classification?*

Contributions described in this chapter include the following:

- Inference time and accuracy quantitative comparison of YOLOv5 and Faster R-CNN models using our video dataset.

- The construction of a target-based vehicle image dataset tailored to our video data.

## 4.2 Anatomy of Faster R-CNN

As discussed in Chapter 2, the evolution of region proposal methods was decisive in object detection algorithms. Firstly, it operates as a distinct standalone component in Fast R-CNN and R-CNN, such as the Selective Search algorithm (Ren et al., 2017). The Region Proposal Network (RPN) constitutes a pivotal component within the Faster R-CNN framework, which is fundamental in generating prospective regions of interest or region proposals in images potentially containing objects. The RPN uses the concept of attention mechanisms inherent in neural networks, effectively instructing the subsequent Fast R-CNN detector on where to focus when identifying objects within the image.

Ren et al. describe the fundamental constituents of the RPN as encompassing the following elements: anchors are employed as predefined boxes, each characterised by distinct scales and aspect ratios. Anchor boxes are positioned at various locations across feature maps. An anchor box primarily comprises two essential parameters, namely scale and aspect ratio. The RPN executes as a sliding window mechanism traversing the feature map derived from the CNN backbone. Within this process, a compact convolutional network, typically embodied as a 3×3 convolutional layer, operates to process features within the receptive field of the sliding window (Figure 12).

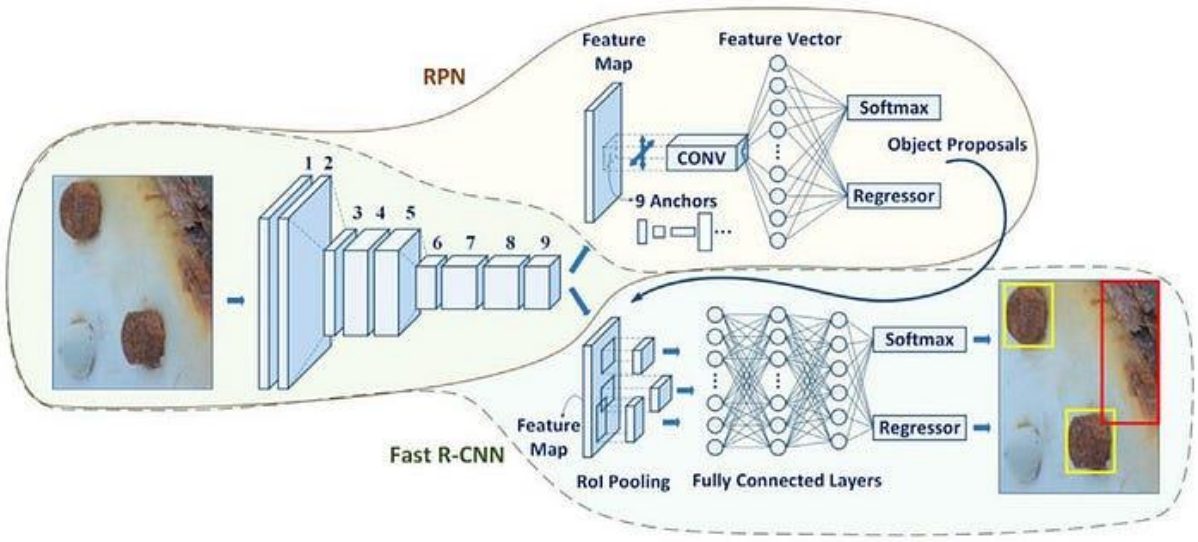



*Figure 12. Faster R-CNN model illustration showing input to make predictions (Ren et al., 2017).*

This convolutional operation produces scores that indicate the likelihood of the presence of an object and regression values that allow adjustments to the anchor boxes. The objectness score indicates the probability that a given anchor box encloses an object of interest rather

than just representing background elements. Within the faster R-CNN architecture, the RPN computes this score for each anchor. The objectness score expresses confidence in the anchor's association with a significant object region. During training, this score assists in classifying anchors as either positive, indicating an object, or negative, indicating background. In situations where a significant number of region proposals are generated, overlap and redundancy among them may be common, often corresponding to the same object instance. This issue is addressed by utilising the Non-Maximum Suppression (NMS) technique. NMS ranks anchor boxes based on their objectness probabilities and selects the top-N anchor boxes with the highest scores. Using NMS ensures the final proposal selection is precise and free from overlapping instances.

Consequently, these selected anchor boxes are considered viable regional proposals. The classification process unfolds by combining features derived from the region proposal with shared weights originating from the CNN backbone. An integral component of Faster R-CNN is bounding box regression, which aims to refine the position and size of the bounding box associated with each region proposal. This refinement process improves the accuracy and alignment of the bounding boxes with the actual object boundaries in the image.

The first layer of the bounding box regression consists of a softmax layer with N+1 output parameters, where N represents the number of class labels, including an additional class for the background; the prime purpose of this layer is to predict an object within the region proposal and, if present, to classify it into one of the N classes or as background.

The second layer is the bounding box regression layer, comprising 4*N output parameters. Each set of four parameters corresponds to a specific class label. These parameters are responsible for adjusting the bounding box's location and size associated with the object in the image. The softmax layer predicts the object's presence or absence within the region proposal by providing probabilities for each class, including background. The bounding box regression layer predicts adjustments to the region proposal's bounding box, including its x-y coordinates, width, and height. These adjustments are specific to each class, each with four regression parameters.

Intersection over Union (IoU) is a metric for the overlap between two bounding boxes. A more comprehensive relational explanation of this concept is provided in Chapter 5. IoU computes the ratio of the overlap area between the two boxes to the area of their combined region, expressed as

$$IoU = \frac{Area\ of\ Intersection\ box\ 1\ \&\ box\ 2}{Area\ of\ Union\ of\ box\ 1\ \&\ box\ 2} \quad (1)$$

### 4.2.1 Training Faster R-CNN

During training, the most valuable metric evaluation is loss; the model aims to minimise this composite loss during training to improve object detection and localisation performance (Figure 13).

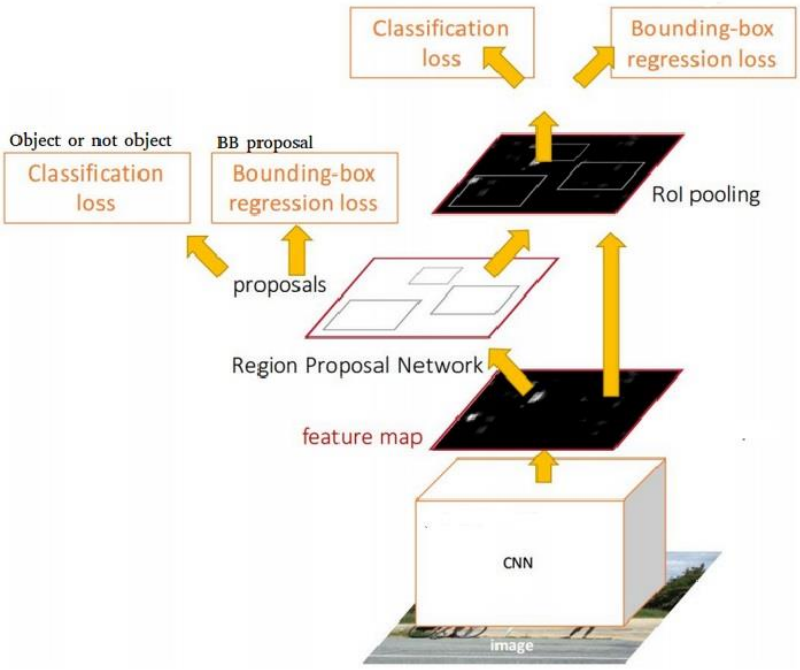



*Figure 13: A visual representation of a two-stage Faster R-CNN object detection system, as introduced by Ren et al. (2017). This system employs a Region Proposal Network (RPN) to extract regions of interest directly from the feature map. Subsequently, it utilises a fully convolutional object classifier. It is important to note that the network architecture is shared between the RPN and the object detector phase.*

All through training, the model learns to predict bounding box adjustments and class probabilities by minimising a combined loss function considering classification and regression loss. The Faster R-CNN multi-task loss function combines the classification and regression losses, refining bounding box predictions.

In the methodology Ren et al. introduced in their 2017 paper, each anchor is assigned a binary class label that determines whether the anchor is indicative of an object's presence. A positive label is designated for an anchor if it fulfils either of the following conditions: (i) it exhibits the highest IoU overlap with a ground-truth bounding box, or (ii) it possesses an IoU overlap greater than 0.7 with any of the ground-truth boxes. Importantly, a single ground-truth box can produce positive labels for multiple anchors.

Anchors that do not meet these criteria for positive labelling, including those with an IoU ratio below 0.3 for all ground-truth boxes, are instead assigned a negative label. Anchors that do

not conform to positive or negative categories are not employed in the training process.

The loss function used for the RPN in this context is further defined as follows:

$$L(\{p_i\},\{ti\}) = \frac{1}{N_{cls}} \sum_i L_{cls}\ (p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^*\ L_{reg}\ (t_i, t_i^*) \quad (2)$$

In the context of this description, 'i' represents the sequential identifier for an anchor box within a batch, and 'pi*' signifies the predicted probability of anchor box 'i' being classified as an object. The ground-truth label 'p' assumes a value of 1 when the anchor box is deemed positive and 0 when it is considered negative. The variable 'ti' is a vector that encodes the parameters for the predicted bounding box, with the ground-truth box for a positive anchor box being represented by 'ti*'.

The parameters' N_cls' and 'N_reg' correspond to the mini-batch size and the total number of anchor boxes. 'λ' serves as a hyperparameter, and it plays a crucial role in balancing the impact of the classification and regression losses within the comprehensive loss function. During training, one can emphasise achieving accurate classification and precise localisation of bounding boxes. Higher values of 'λ' prioritise the refinement of bounding box regression, while lower values give more importance to achieving classification accuracy.

The classification loss function, denoted as 'L_cls' in Ren et al. (2017), is characterised as a logarithmic loss applied over two categories: object and non-object.

The definition of $L_{cls}$ is as follows:

$$L_{cls}\ (p_i, p_i^*) = \begin{cases} -\log p_i & if\ p_i^* = 1 \\ -\log(1 - p_i) & if\ p_i^* = 0 \end{cases} \quad (3)$$

In addition, the regression loss function $L_{reg}$ is presented as follows:

$$L_{reg}(t_i, t_i^*) = R(t_i, -t_i^*) \quad (4)$$

The robust loss function (smooth L1), denoted as $R$ and defined in Fast R-CNN (Girshick et al., 2017), is employed. Following the network's prediction of class probabilities and bounding box adjustments, a post-processing phase involving NMS is carried out. This step is vital in refining the final detection results by reducing redundant detections while retaining the most confident and non-overlapping detections.

RPN training is achieved using an end-to-end approach facilitated through backpropagation and stochastic gradient descent (SGD). This means the entire network, including the newly introduced RPN and the shared convolutional layers, is jointly optimised. The main goal of this

optimisation process is to minimise the loss function, thereby improving the overall performance of the object detection system.

According to Cheng et al. (2018), the Optimisation Faster R-CNN series' architecture involves a feature extractor as the backbone and two specialised branches for region classification and localisation tasks. However, optimising in this manner may lead to convergence towards a suboptimal solution that compromises the performance of both tasks due to the simultaneous consideration of the sum of two losses rather than treating each loss separately. CNNs encounter a significant challenge concerning their fixed receptive fields. In the context of image classification, inputs are typically cropped and resized to standardised dimensions. The network architecture is intentionally designed with a receptive field that is slightly larger than the input region. However, this approach has limitations because it results in the loss of contextual information during the cropping process and necessitates the resizing of objects of various sizes. To achieve accurate object understanding and recognition, it is essential for the 'effective receptive field' to encompass the entire object. In the case of Faster R-CNN, the introduction of Region of Interest (ROI) pooling is utilised to extract objects from 2-D convolutional feature maps and transform them into a 1-D fixed-size representation for subsequent classification tasks. This process establishes a fixed receptive field, meaning the network concentrates on a fixed-size window within the input image.

Nevertheless, as objects in images can significantly differ in size, this fixed receptive field can result in varying amounts of contextual information. The context may be pervasive for smaller objects, hindering the network's effective focus on the object itself. Conversely, the receptive field may be too limited for larger objects, causing the network to examine only a portion of the object. While some approaches have attempted to address this issue by aggregating features with different receptive fields to introduce multiscale features, the effectiveness of such methods can vary, and achieving a balance between capturing context and object details remains a challenge in deep convolutional neural networks (Waldner and Diakogiannis, 2020), which had a direct impact on our work.

## 4.3 Anatomy of YOLOv5

In contrast with Faster R-CNN in the previous section, YOLOv5 employs regression, where The whole image's classes and bounding boxes are predicted in one algorithm run. YOLOv5 demonstrates exceptional performance coupled with swift object detection capabilities, effectively meeting the demands of real-time applications. This singular neural network architecture executes all necessary steps in object detection. In traffic research, where detection speed and accuracy are paramount, YOLOv5 achieves real-time processing speeds (Zhou, Zhao and Nie, 2021).

The YOLOv5 architecture comprises three essential components, as illustrated in Figure 9. The first component is the cross-stage partial network CSPNet (Cheng et al., 2018), designed to address gradient-related challenges effectively. This component optimises the algorithm by reducing the parameter count and the number of floating-point operations per second (FLOPS). As a result, it boosts both inference speed and accuracy, all within a compact architectural footprint (S. P. Lakshmi Priya et al., 2023).

The backbone of the architecture consists of multiple convolutional layers, four CSP bottlenecks, three convolutional layers, and a spatial pyramid pooling component (SPPF). Primarily, the CNN serves as a feature extractor, capturing feature maps of different scales from input images.

The neck component plays a pivotal role in feature fusion. It aggregates and transmits features from deeper layers to the detection head, facilitating the extraction of valuable feature information and generating output feature maps in three distinct sizes.

The head section encompasses multiple convolutional layers, four CSP bottlenecks, three convolutional layers, and upsampling and concatenate layers. Its principal function involves predicting visual characteristics, delineating bounding boxes around target objects, and determining object classes.

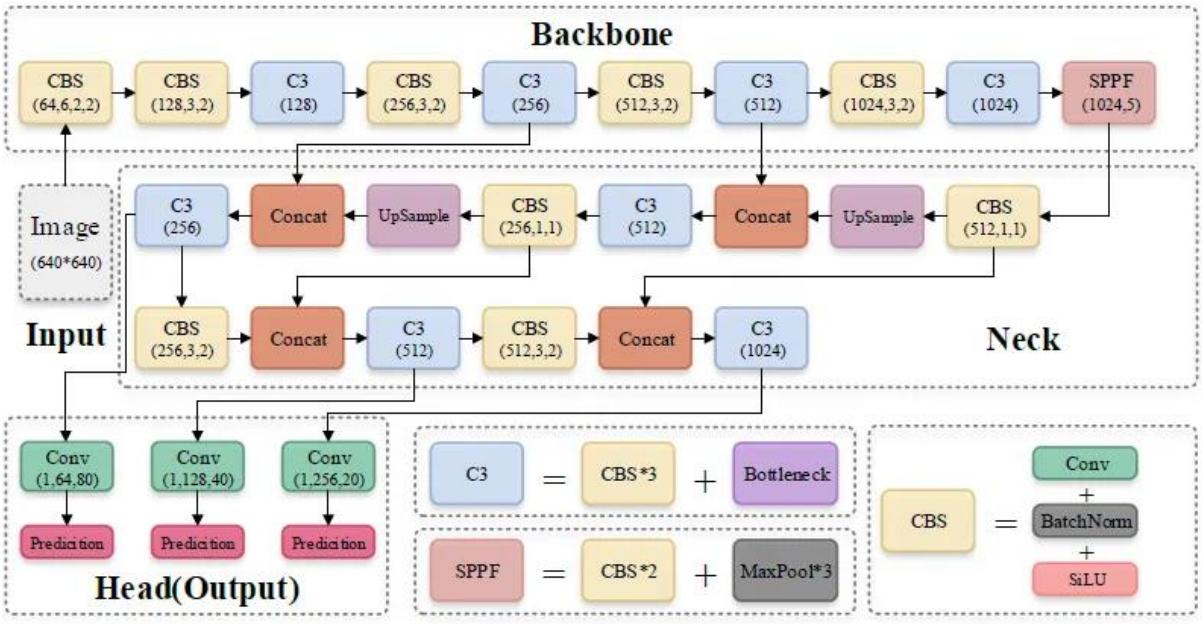



Figure 14: YOLOv5 Architecture adapted from Liu et al. (2022).

The initial 20 convolution layers of the model are pretrained using an image dataset, with a temporary average pooling and a fully connected layer added. The last fully connected layer predicts both class probabilities and bounding box coordinates.

Figure 15 illustrates the architecture of the CNN model that acts as the backbone for YOLO.

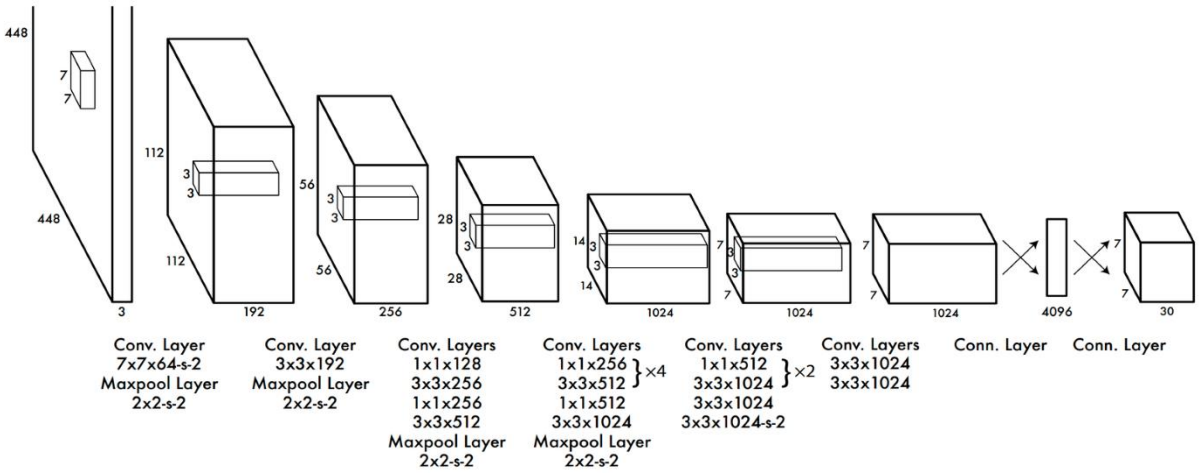



*Figure 15. YOLO (You Only Look Once) algorithm architecture (S. P. Lakshmi Priya et al., 2023).*

YOLO functions by dividing an input image into a grid, with each grid cell responsible for detecting any object whose centre is within that cell. In each grid cell, YOLO predicts 'n' bounding boxes and associated confidence scores for these boxes. These confidence scores serve a dual purpose: they convey the model's confidence in the existence of an object within the box and indicate the accuracy of the box's location prediction.

During YOLO's training, the pivotal strategy involves predicting multiple bounding boxes within each grid cell. The primary aim is to allocate one bounding box predictor to each object and determine which has the highest IoU with the actual object location. This specialised approach enables each predictor to become proficient at predicting specific object sizes, aspect ratios, or classes. This specialisation improves the recall score, enhancing YOLO's object detection capabilities.

After making multiple predictions, YOLO employs non-maximum suppression (NMS) as a post-processing step to bolster the accuracy and efficiency of object detection. Since generating multiple bounding boxes for a single object in an image is common and some boxes may overlap or be off-centre, NMS plays a crucial role in identifying and removing redundant or erroneous bounding boxes. Its purpose is to ensure that only a single accurate bounding box is retained for each object in the image. The accuracy of the IoU score relies heavily on the precise localisation of the bounding box around the object, such as a vehicle. The precise positioning of the bounding box around an object is given as follows:

$$\sigma\frac{g}{r} = O_{r,g} * \mho\frac{p}{t} \quad (5)$$

Where r represents the bounding box within the g grid and $\sigma\frac{g}{r}$ denotes the confidence score of this bonding box. '$O_{r,g}$ indicates the vehicle's presence within the g box. The $\mho_{i,j'}$ value equals 0 if the vehicle is within the g box; otherwise, it is 1. $\mho_{i,j'}$ corresponds to the IoU score.

### 4.3.1 Training YOLOv5

Training YOLOv5 involves optimising the model using a loss function that quantifies the discrepancies between predicted bounding boxes, class probabilities, and ground truth annotations. The primary loss components include objectness or confidence loss, which gauges the model's confidence in predicting the existence of an object within a bounding box, and classification loss, which assesses the accuracy of predicted class probabilities. Additionally, localisation loss evaluates the precision of predicted bounding box coordinates compared to ground truth boxes. These loss components are combined to minimise the overall loss function during training. The model is optimised using methods like stochastic gradient descent (SGD), which aims to iteratively adjust its parameters to excel in object detection by improving localisation accuracy and class prediction while minimising loss. The specific formulation and weighting of these loss components can vary based on the YOLOv5 variant and dataset. Training is an iterative process that evaluates the model's performance by monitoring its ability to reduce loss over multiple epochs or iterations.

The loss function $(LF)$ in YOLOv5 is the aggregate of three key components: regression loss for bounding boxes, confidence loss, and classification loss. It is computed as follows:

$$L\,F = l_{bx} + l_s + l_j \quad (6)$$

where $l_{bx}$ is the regression function for the bounding box, $l_j$ is the loss function for confidence and $l_s$ is the loss function for the classification.

$l_{bx}$ is calculated using

$$l_{bx} = \lambda_{cd}\sum_{i=0}^{s^2}\sum_{m=0}^{b} I_{i,m}^{j}\, bj(2 - W_i * h_i\left[\begin{matrix}(x_i - x'^{m}_{i})^2 + (y_i - y'^{m}_{i})^2 + (w_i - w'^{m}_{i})^2 \\ +(h_i - h'^{m}_{i})^2\end{matrix}\right] \quad (7)$$

where *h* and w are the target's height and width, respectively and $y_i$ , $x_i$ are the correct coordinates of the target. $\lambda_{cd}$ is the indicator function of whether the cell *i* contains

an object.

The $l_s$ is calculated as follows:

$$l_s = \lambda_s \sum_{i=0}^{s^2} \sum_{m=0}^{b} I_{i,m}^{j} \sum_{C \in cl} V_i(c) \log(VV_i(c)) \quad (8)$$

where $V_i$ represents a vector of predicted probabilities for class c at spatial location i.

The $l_j$ is calculated as follows:

$$l_j = \lambda_{noj} \sum_{i=0}^{s^2} \sum_{m=0}^{b} I_{i,m}^{noj} (c_i - c_l)^2 + \lambda_j \sum_{i=0}^{s^2} \sum_{m=0}^{b} I_{i,m}^{j} (c_i - cc_l)^2 \quad (9)$$

where $\lambda_{noj}$ is the category loss coefficient, $\lambda_s$ is the classification loss function, *cl* is the confidence score, and *cc* is the class.

YOLOv5 aims to minimise this loss during training to improve its accuracy in object detection, bounding box regression, classification, and confidence score prediction, which are all crucial to this thesis, where the precise detection and classification of target vehicles are a critical component in the pipeline. Adjusting the hyperparameters can help fine-tune the training process based on a given task's specific requirements, which are detailed in Chapter 5.

## 4.4 Selecting a vehicle detection and classification base model

The literature contains many examples of comparisons between YOLOv5 and Faster R-CNN (Mahendrakar et al., 2022; Jabir, Falih and Rahmani, 2021). However, we required a comparison based on our data as our requirements are very focused and not to be based on a generalised object dataset. Our requirements dictate that we must detect and classify target vehicles with a reasonable degree of accuracy and in real-time. Other studies (He et al., 2023) have demonstrated that it is possible to refine the performance of models based on the desired objective. Many of the refinements of models in the literature focus on detection accuracy and small objects; our refinements were implemented to increase vehicle detection accuracy (larger objects) and decrease the inference time. To compare YOLOv5 and Faster R-CNN for our data, we must first define the accuracy metrics used to benchmark most object detection and classification models in computer vision.

Precision (P) and recall (R) represent vital assessment metrics for object detection models, serving to evaluate the models' proficiency in accurately identifying and categorising objects

within images. Precision quantifies how accurately a model identifies positive cases by measuring the ratio of true positives (correctly predicted positives) to the sum of true positives and false positives. Precision quantifies the model's ability to make correct identifications—an essential factor in our work, where false positives can have significant repercussions. Precision is calculated as

$$P = \frac{TP}{TP+FP} = \frac{TP}{total\ predictions} \tag{10}$$

Precision offers an understanding of the model's positive predictions' reliability. High precision implies that the model's object predictions are likely accurate.

Recall, also called sensitivity or true positive rate, evaluates the model's capacity to detect all relevant objects in the dataset correctly. It is computed as the ratio of true positives (TPs) to the total of TPs and false negatives (FNs).
Recall is calculated as

$$R = \frac{TP}{TP+FN} = \frac{TP}{allgroundtruths} \tag{11}$$

Recall is crucial because it quantifies the model's ability to find all the objects of interest in the image, avoiding false negatives.

These metrics are frequently employed to offer a well-rounded evaluation of an object detection model. They exhibit an inverse relationship: as precision increases, recall may decrease, and conversely, when recall increases, precision may decrease. Striking the appropriate balance between precision and recall hinges on the unique use case and the implications of false positives and false negatives.

To visualise this trade-off, plotting a precision-recall curve (PR curve) or calculating the F1-score, which represents the harmonic mean of precision and recall, is common. This allows for a more comprehensive assessment of the model's performance and its suitability for different applications and is calculated as

$$F1 = 2 * (P * R) / (P + R) \quad (12)$$

The F1 score is functional when combining precision and recall into a single metric, especially when the balance between false positives and false negatives is essential.

Mean average precision (mAP) is one of the primary metrics for evaluating the accuracy of object detection models, especially when dealing with multiple object classes. It provides a comprehensive assessment of the model's performance across all classes by calculating the average precision for each class and then taking the mean of these values.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} Pi \quad (13)$$

Average precision (AP) is a key metric, where $N$ represents the total number of classes. Generally, AP is computed as the average precision values ($APi$) across different IoU thresholds ($R$) within the range [0, 1]. Each $APi$ corresponds to the average precision for the $i$-th class over a range of IoU thresholds. Typically, these IoU thresholds are chosen in increments of 0.05, ranging from 0.5 to 0.95.

Specifically, two important metrics, $mAP50$ and $mAP95$, are used to assess model accuracy:

mAP50 calculates the mean average precision when considering the IoU threshold of 0.5. It evaluates how well the model performs when detections are considered accurate if their IoU with the ground truth bounding box exceeds 0.5. mAP95 calculates the mean average precision but at a stricter IoU threshold of 0.95. This metric assesses the model's accuracy when a high degree of overlap between the detected bounding box and the ground truth bounding box is required for a positive classification.
Furthermore, mAP50:95 calculates a range of IoU thresholds between 0.5 and 0.95 to obtain a mean across the range.
These metrics allow for a nuanced evaluation of the model's performance, considering both relatively lenient and stringent criteria for object detection accuracy. It is expected to report both $mAP50$ and $mAP95$ to provide a comprehensive understanding of how well the model performs under different IoU thresholds (Hamzenejadi and Mohseni, 2023).
To evaluate the real-time performance of object detection models, we must consider accuracy, inference speed, and computational complexity. Complex models may achieve high accuracy but often demand significant computational resources, hindering real-time operation. Two fundamental metrics for assessing model performance in real-time object detection are frames

per second (FPS) and inference time. FPS measures how many frames or images the model can process per second. Higher FPS values indicate faster inference speeds and are desirable for real-time applications. Inference time quantifies the time it takes for the model to process a single frame or image in milliseconds (ms). Lower inference times are preferred for real-time applications, contributing to higher FPS.

Model size, both in terms of memory and storage requirements, is an essential consideration. Smaller models are typically more memory-efficient and load faster, which is advantageous for real-time inference. Another critical metric is the number of FLOPs (floating-point operations) needed for inference. Lower FLOPs indicate lower computational complexity and faster inference.

The total number of parameters in a model can serve as an indicator of its computational complexity. Smaller models tend to have fewer parameters, which can lead to faster inference. A low inference time is vital for our work, where we aim to balance accuracy with computational efficiency, particularly for detecting relatively large objects like vehicles at a T-junction.

In our case, the vehicles under investigation demand swift detection to facilitate data transmission through the pipeline for real-time feature vector generation. This strategy aids in reducing the computational workload by filtering out smaller objects, with the strategy's primary focus on identifying target vehicles. To evaluate and compare the faster R-CNN and YOLOv5 models using our video data as a reference point, we gauge their performance based on these real-time metrics.

### 4.4.1 Common Objects Dataset

This section discusses the dataset utilised for our comparison experiments, the Common Objects Dataset. The Microsoft Common Objects in Context (COCO) dataset (Lin et al., 2014) is comprehensive for various computer vision tasks, including object detection, image segmentation, key-point detection, and captioning. This dataset comprises 328,000 images and annotations, including bounding boxes and per-instance segmentation masks for 80 object categories. For our purpose, COCO contains five vehicle classes: car, truck, bus, motorcycle, and bicycle.

### 4.4.2 Backbone Network Model

Officially supported YOLOv5 and Faster R-CNN models integrate pre-trained network models trained on the COCO dataset. However, they exhibit differences in network depth, layer count, and layer size, leading to their respective model performance variations. Selecting a model for a specific task is a practical and empirical process. To make an informed decision, we conducted a comparative analysis by assessing two pre-trained COCO Faster R-CNN models

alongside two pre-trained COCO YOLOv5 models. This analysis strives to identify the optimal configuration that can serve as the initial stage in our pipeline. This empirical evaluation equips us with the knowledge to select the model that best aligns with our objectives and requirements.

#### *4.4.2.1 Choice of Faster R-CNN Backbone Model*

A Faster R-CNN architecture requires selecting a network for feature extraction, commonly known as the backbone. Numerous studies, such as those by Muhammad Jehanzaib Yousuf et al. (2022) and Elharrouss et al. (2022), have explored various feature extraction networks for deep learning, including well-known architectures like VGG, ResNet, AlexNet, GoogleNet, Inception, Xception, DenseNet, and SqueezeNet, among others. For most computer vision tasks involving the COCO dataset, ResNet and MobileNet are commonly chosen as backbone networks.

Two official, faster R-CNN models come pre-trained on the COCO dataset, each employing distinct backbone architectures. Studies that feature ResNet50, such as Renjun et al. (2022), have indicated that faster R-CNN with a ResNet50 backbone offers high accuracy but operates more slowly. This model utilises a ResNet-50-FPN backbone and can achieve the highest mean average precision (mAP) when fine-tuned on a new dataset with slower inference times.

Conversely, faster R-CNN with a MobileNet v3 backbone is faster but slightly less accurate. (Yusuf Gladiensyah Bihanda, Chastine Fatichah and Anny Yuniarti, 2023). This model provides high-resolution feature extraction, delivering over twice the speed of the ResNet50 variant on equivalent hardware (GPU). However, a trade-off exists as its mAP performance is slightly reduced due to the higher frames per second (FPS). The choice between these models depends on the task's requirements and the balance between speed and accuracy that is most suitable for the application.

#### *4.4.2.2 Choice of YOLOv5 models*

The YOLOv5 series comprises several model variants denoted by letters, including YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x (see Table 5). These variants differ in model sizes, featuring varying numbers of layers and parameters. The design of these models focuses on achieving a balance between speed and accuracy.

| Model | size (pixels) | mAP$^{val}$ 0.5:0.95 | mAP$^{val}$ 0.5 | Speed CPU b1 (ms) | Speed V100 b1 (ms) | Speed V100 b32 (ms) | params (M) | FLOPs @640 (B) |
|---|---|---|---|---|---|---|---|---|
| YOLOv5n | 640 | 28.0 | 45.7 | 45 | 6.3 | 0.6 | 1.9 | 4.5 |
| YOLOv5s | 640 | 37.4 | 56.8 | 98 | 6.4 | 0.9 | 7.2 | 16.5 |
| YOLOv5m | 640 | 45.4 | 64.1 | 224 | 8.2 | 1.7 | 21.2 | 49.0 |
| YOLOv5l | 640 | 49.0 | 67.3 | 430 | 10.1 | 2.7 | 46.5 | 109.1 |
| YOLOv5x | 640 | 50.7 | 68.9 | 766 | 12.1 | 4.8 | 86.7 | 205.7 |

*Table 5. Comparison of YOLOv5 models (Ultralytics, 2023)*

The YOLOv5 series encompasses a range of models tailored to different object detection requirements. YOLOv5n, a recent addition, is specifically designed for edge devices and serves as a comparative benchmark in Chapter 5. YOLOv5s prioritises speed, making it well-suited for real-time applications with limited computational resources, albeit with a slight sacrifice in accuracy. YOLOv5m balances speed and accuracy, making it versatile for general object detection tasks. YOLOv5l, a larger model, excels in accuracy but slightly reduces inference speed, making it suitable for tasks where precision is paramount and ample computational resources are available. Lastly, YOLOv5x is the series' largest and most accurate model, but it comes at the cost of slower inference speed.

When comparing YOLOv5 with Faster R-CNN in benchmark tests using the COCO dataset, it becomes evident that the model's size impacts accuracy. However, the choice of model depends on the specific task and desired outcome. In scenarios like vehicle detection, achieving high real-time accuracy is crucial. It is worth noting that no single model perfectly balances accuracy and speed, often necessitating a trade-off between the two depending on the specific requirements of the task.

#### *4.4.2.3 Limitations of the YOLO model*

YOLO presents spatial constraints on bounding box predictions, with each grid cell responsible for predicting only two boxes and one class, which can limit the precision of object localisation and classification. This limitation is particularly noticeable for objects with complex shapes or unconventional positions. Additionally, detecting small objects, especially when they appear in groups or clusters, can be challenging as the model may not allocate sufficient resources for accurate identification. YOLO's ability to generalise to objects with new or unusual aspect ratios can also be problematic as the model learns from the training data and may struggle with objects with significantly different aspect ratios than those encountered during training.

Efforts to mitigate this challenge include the creation of a focused image training dataset in Section (4.5). However, despite these limitations, YOLO remains a popular choice for object detection due to its real-time capabilities and overall effectiveness in a wide range of applications.

### 4.4.3 Comparison of YOLOv5 and Faster R-CNN ON COCO 80

In the following series of experiments, we conduct a comparative analysis of similarly performing backbone models for faster R-CNN and YOLOv5. As illustrated in Table 6, these comparative models exhibit various similarities. This observation hints at a discernible relationship between the model's size and accuracy when benchmarked against the COCO dataset, with the larger models proving to be more accurate, albeit slower. However, it is crucial to acknowledge that benchmarking results can be highly task-specific and dependent on the desired outcomes. Laboratory-based testing, for example, differs substantially from real-world situations, particularly when addressing traffic-related challenges like vehicle detection. In this context, the primary goal is to create a precise detection and classification model that can be performed in real-time. Such a model would facilitate further data processing without imposing an undue computational load.

Our research findings suggest that no model achieves the ideal balance between accuracy and speed for vehicle detection and classification for our purpose (see Table 6). Hence, in practical terms, some degree of compromise between accuracy and speed is inevitable.

| *Model and Backbone COCO dataset* | *mAP* | *Size MB* | *Parameters M* |
|---|---|---|---|
| *Faster R-CNN ResNet50* | *0.43* | *98* | *23* |
| *YOLOv5 Medium* | *0.45* | *90* | *21.2* |
| *Faster R-CNN MobileNet v3* | *0.32* | *15.3* | *4.97* |
| *YOLOv5 Small* | *0.37* | *14* | *7.2* |

*Table 6: Model and backbone specification and benchmark (COCO) accuracy adapted from Zhou et al. (2021).*

We evaluated Faster R-CNN ResNet50, Faster R-CNN MobileNet v3, YOLOv5 Medium, and YOLOv5 Medium without any modifications using the same pre-trained COCO weights. The assessment involved our custom Bo video dataset and sought to assess each model's FPS rate, accuracy, and inference speed on a CPU and a GPU. Our testing process involved manually processing, for ground truth verification, a series of 15-minute video clips from each

junction-based video clip in our dataset. Each video clip was passed through every model sequentially. The videos were maintained in their original form, with a frame rate of 50 FPS and a resolution of 1920x1080 pixels, and were fed into each model for analysis. We recorded key metrics for each video test segment and model, including detection confidence scores, inference time, and frame rate. Our primary focus was detecting cars, which represent the highest number in the vehicle class and our study's most common vehicle type. To ensure the reliability of our results and minimise any junction-specific anomalies influenced by factors such as object occlusions, camera distance, lighting, and traffic density, we averaged the scores across all video clips.

The detailed results of our evaluation can be found in Table 7. These findings provide valuable insights into each model's performance under real-world conditions and inform the selection of the most suitable model for our specific application. Our primary metric of interest is the confidence level in detecting our target vehicle rather than a mAP of the training data.

| *Model and Backbone COCO weights and Bo video data* | *FPS* | *Mean Class Confidence Score for car** | *Inference time ms CPU* | *Inference time ms GPU (RTX 3070)* |
|---|---|---|---|---|
| *Faster R-CNN ResNet50* | *22* | *0.96* | *112* | *63* |
| *YOLOv5 Medium* | *49* | *0.91* | *98* | *21* |
| *Faster R-CNN MobileNet v3* | *110* | *0.7* | *24* | *6* |
| *YOLOv5 Small* | *74* | *0.81* | *81* | *17* |
| ** Calculated as confidence in the object being of class n* | | | | |

*Table 7: Model comparative results using COCO weights and Bo T-junction video data.*

***Initial results discussion***

**Faster R-CNN ResNet50**

FPS: 22 - This model processes 22 frames per second, which is relatively low compared to the other models listed, indicating it is slower at making predictions.

Mean Class Confidence Score for Car: 0.96 - It has a very high confidence score, suggesting it is very accurate at detecting cars when it does make a prediction.

Inference Time on CPU: 112 ms - It takes 112 milliseconds to process a single frame on a CPU, which is the slowest among the models listed.

Inference Time on GPU (RTX 3070): 63 ms - Processing time improves significantly on a GPU but is still the slowest among the compared models.

**YOLOv5 Medium**

FPS: 49 - This model processes frames more than twice as fast as the Faster R-CNN ResNet50 model, making it suitable for applications requiring quicker object detection.

Mean Class Confidence Score for Car: 0.91 - Although slightly less accurate than the Faster R-CNN ResNet50, it still maintains a high confidence score.

Inference Time on CPU: 98 ms - It's quicker than the Faster R-CNN ResNet50 on a CPU but slower compared to the other models listed.

Inference Time on GPU (RTX 3070): 21 ms - Shows significant improvement on a GPU, indicating good optimization for GPU-based inference.

**Faster R-CNN MobileNet v3**

FPS: 110 - This model shows the highest frames per second, indicating it is the fastest model for processing frames, suitable for real-time detection tasks.

Mean Class Confidence Score for Car: 0.7 - The confidence score is the lowest among the models, which might indicate a higher rate of false negatives or less certainty in its predictions.

Inference Time on CPU: 24 ms - Shows very fast processing on the CPU, the fastest among those listed.

Inference Time on GPU (RTX 3070): 6 ms - Extremely fast on the GPU, making it highly efficient for applications that can leverage GPU acceleration.

**YOLOv5 Small**

FPS: 74 - This model offers a good balance between speed and accuracy, with a high frame rate indicating it can process video data quickly.

Mean Class Confidence Score for Car: 0.81 - This score is lower than the YOLOv5 Medium and Faster R-CNN ResNet50 but higher than Faster R-CNN MobileNet v3, suggesting moderate accuracy.

Inference Time on CPU: 81 ms - Faster than the Faster R-CNN ResNet50 but slower than the other models.

Inference Time on GPU (RTX 3070): 17 ms - Shows good performance on a GPU, though not as fast as the MobileNet v3.

**Results Summary**

For high accuracy: Faster R-CNN ResNet50 is the best choice, with the highest confidence score, but at the cost of speed.

For real-time processing: Faster R-CNN MobileNet v3 stands out with the highest FPS and lowest inference times, though it sacrifices some accuracy.

For a balance between speed and accuracy, YOLOv5 models, especially the Medium variant, offer a good compromise, with decent FPS and confidence scores and much better speed on GPU compared to Faster R-CNN ResNet50.

Based on the results presented in Table 7, our findings indicate that the most accurate combination for vehicle detection in terms of model and backbone is Faster R-CNN ResNet 50. On the other hand, for GPU-based applications, Faster R-CNN MobileNet v3 offers the fastest performance. However, it is worth noting that the YOLOv5 medium model has proven to strike a good balance between accuracy and inference speed on our video dataset, Bo. Given these results, we have decided that all future work related to the detection and classification stage of our thesis's pipeline will be done using one of the YOLOv5 models, including the variants s, m, l, or xl. We plan to enhance the YOLOv 5 model performance through a series of iterative experiments in later chapters, with an initial focus on improving accuracy.

## 4.5 Improving target vehicle detection accuracy with focused target training

While there are numerous large-scale datasets available for object detection tasks, such as the work by Charles-Éric Noël Laflamme, Pomerleau, and Philippe Giguère (2019), the scarcity of high-quality datasets continues to be a substantial obstacle, particularly in the context of vehicle classification. Many existing vehicle detection datasets are general and lack the specificity required to tackle the unique classification challenges presented at T-junctions. In response to this limitation, transfer learning has emerged as a valuable approach for addressing specific class imbalances that result from this generalisation. Transfer learning allows models to adapt and specialise in the context of specific tasks, leveraging pre-trained knowledge from broader datasets to enhance performance in junction-specific classification challenges.

Transfer learning is a machine learning technique where a model developed for a specific task is repurposed as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models improve computational efficiency and model performance, especially when data for the second task is scarce or when training a model from scratch is computationally expensive—transferring the weights that a model has learned from one task to another leverage previously learned patterns, reducing the time and resources required for training on the new task. This approach is efficient in natural language processing and computer vision, where models pre-trained on large datasets can significantly boost performance on related tasks with minimal additional training.

Using pre-trained models for transfer learning (Sowmya and Radha, 2021), a technique increasingly deployed in situations characterised by limited training data or the imperative for rapid, enhanced performance, comprises a multi-step process. This process encompasses identifying a source model akin to the target domain, adapting the source model to align with the target model's requirements, and training the source model to attain the desired target

model characteristics. The model closest to our domain is the COCO dataset, with many labelled vehicle instances (Table 8) as a starting point for training our vehicle image dataset.

| Class | Number of instances | |
|---|---|---|
| | Training | Validation |
| Car | 43,533 | 1,918 |
| Motorcycle | 8,645 | 367 |
| Bus | 6,061 | 283 |
| Truck | 9,970 | 414 |
| Bicycle | 7,056 | 314 |

*Table 8: Number of instances per traffic class extracted from the COCO dataset adapted from Panero Martinez et al. (2021).*

We aim to improve the detection accuracy of YOLOv5 models on our data by creating a focused vehicle dataset to enrich the instances found in COCO, Table 8, and training all five YOLOv5 models using transfer learning using the full COCO dataset as a foundation and adding our weights.

### 4.5.1 Target vehicle image dataset creation

Our ultimate objective is to predict driver behaviour in real-time, which is imperative to do quickly and accurately. However, the accuracy of our predictions is profoundly affected by the detection and classification stage, and missing vehicles due to misclassification can impact all subsequent stages of our pipeline.

Acquiring training data resembling our target vehicles is paramount to enhancing detection accuracy. Our first step in compiling a customised vehicle dataset involved collecting screenshot frame images from our video dataset by pausing junction video where a vehicle was in a target frame, taking a screenshot, and saving it as a single image. This was repeated for the other junction videos. These images serve as the foundation for our image dataset, from which we further infer and generate feature vectors in the pipeline. We collected images from all the routes and created 158 images of cars, trucks, and buses from each junction. We sourced an additional 981 publicly available images from the Internet. These images were selected to encompass a wide range of representations of various vehicle types, representing vehicles captured from the POV from camera x in section (3.4.1) viewing traffic (a). A sample of the images can be seen in Figure 16. Our target vehicles were detected from the right side or the front at all our test T-junctions.

The subsequent step involved aggregating all the collected images, categorising them into

relevant classes (cars, trucks, and buses), and manually labelling them using bounding box annotations. Since we were utilising transfer learning and COCO already had labels for cars, trucks, and buses, we labelled our images as 'cars_1,' 'trucks_1,' and 'buses_1' to be added to the class label array. This process forms the basis for our dataset, facilitating the training and improvement of our vehicle detection and classification models.

*Figure 16. Image data showing type and perspective pre-data labelling.*

### 4.5.2 Image augmentation post-labelling

After the manual labelling, the original 1,139 images underwent an augmentation procedure, effectively expanding the dataset to 5,381 images. This augmentation was executed by systematically applying several pre-processing techniques to each image. The augmentation process comprised the following steps: all images were uniformly resized, ensuring dimension consistency across the dataset. A grayscale filter was applied to introduce variations in colour representation, thereby augmenting the dataset's diversity. Each image was subjected to a random rotation operation, with angular adjustments from -15 to +15 degrees. This introduced variations in orientation, enhancing the dataset's robustness. Horizontal and vertical shearing operations were applied randomly at -15° to +15°. These shear transformations introduced distortion effects, contributing to a more comprehensive training dataset. Regarding noise incorporation, noise was introduced to 10% of the images to diversify the dataset further. This noise included up to 10 pixels of blurring, alterations in brightness ranging from -25% to +25%, and the application of bounding box noise to 5% of the dataset. The specifics of these noise operations are represented visually in Figure 17. This augmentation strategy substantially increased the dataset size and introduced variability and realism into the images, enhancing the model's generalisation ability.

*Figure 17: The augmentation process applied to the dataset, explicitly highlighting the incorporation of additional noise into 10% of the images.*

Our noise augmentation procedure involved a variety of transformations intending to diversify

the dataset, thereby enhancing its ability to capture real-world variations and complexities. These augmentation techniques encompass blurring, brightness adjustments within a range of -25 % to +25 %, and the introduction of bounding box noise. Collectively, these measures create a more robust and representative dataset for subsequent analysis and model training.
Overfitting is a phenomenon in which a neural network learns a function with extremely high variance, essentially memorising the training data perfectly, as explained by Shorten and Khoshgoftaar (2019). The advantages of employing augmentation methods are quite significant. Data augmentation is a valuable strategy for mitigating overfitting and giving the model a more comprehensive and diverse dataset; this, in turn, enhances the model's capacity to generalise effectively to novel data while bolstering its resilience against data noise and variations. Furthermore, data augmentation expedites the training process, effectively reducing the time needed to train a model on extensive datasets.
However, data augmentation comes with its own set of challenges. It can be time-consuming, particularly if performed manually, and it can incur substantial computational costs when applied to large datasets. Additionally, if the augmentation transformations are not thoughtfully selected, they can introduce bias into the dataset, adversely impacting the model's performance. Therefore, it is crucial to exercise care and diligence in considering and validating augmentation techniques to ensure their efficacy in enhancing both model generalisation and performance.

### 4.5.3 Training target dataset on YOLOv5 models

Given the results in section (4.4.4), where the YOLOv5 medium model proved to strike a good balance between accuracy and inference time, initial training was undertaken using YOLOv5m pre-trained with COCO weights. The standard COCO dataset partitioning for weight acquisition was executed as follows: 83% of the data was designated for training, 8% for validation, and 9% for testing. Our new image dataset was partitioned similarly and added to the training model. Transfer learning training was carried out throughout 300 epochs. The training was conducted utilising a single GPU, specifically the RTX 3070 with 8 GB of memory. The training model was standardised to the official YOLOv5 parameters for this model. These operations were carried out using Python 3.10 and the PyTorch framework.
The results of the training in Table 9 were very close as the benchmark mAP for both metrics on training data showed little significant change. There is a slight increase in confidence score using unseen video data and evaluating cars, similar to the experiment in section (4.4.4).

| Model YOLOv5 m | Benchmark mAP | With our dataset mAP |
|---|---|---|
| mAP 0.5 | 0.45 | 0.45 |
| mAP 0.5  0.95 | 0.64 | 0.65 |
| Class Confidence cars (tab 3) using Bo video. | 0.91 | 0.93 |

*Table 9. Post-training evaluation metrics for transfer learning using YOLOv5 m.*

Initial results discussion

**mAP at 0.5:** This metric evaluates the model's precision (i.e., its ability to correctly identify objects) at an IoU threshold of 0.5. An IoU of 0.5 means that the overlap between the predicted bounding box and the ground truth bounding box is at least 50%. Both the benchmark and your dataset show a mAP of 0.45, indicating that the model performs equally well on both datasets at this level of IoU threshold, correctly identifying objects with at least 50% overlap with ground truths 45% of the time.

**mAP 0.5:0.95:** This metric averages the mAP calculated at different IoU thresholds, from 0.5 to 0.95 (in steps of 0.05). This provides a more comprehensive view of the model's performance across various levels of precision and recall. The benchmark shows a mAP of 0.64, while our dataset shows a slightly better performance with a mAP of 0.65. This suggests that our dataset is either more representative of the model's application context or contains less challenging examples, leading to a slightly higher average precision across different IoU thresholds.

**Class Confidence for Cars using Bo video:** This metric specifically measures the model's confidence in identifying cars within the dataset provided by Bo video. A confidence score of 0.91 in the benchmark and 0.93 in our dataset indicates high reliability in detecting cars, with our dataset yielding slightly higher confidence. This could be due to various factors, such as the quality of the images, the representation of cars in the dataset, or the model's tuning parameters being better suited for the characteristics of our dataset.

In summary, the YOLOv5 model performs comparably on both the benchmark and our dataset for object detection at an IoU threshold of 0.5. It exhibits a slight improvement in average precision across a range of IoU thresholds from 0.5 to 0.95 when tested on your dataset. Additionally, the model shows high and slightly improved confidence in detecting cars in our dataset compared to the benchmark. These results suggest that our dataset is either well-suited for the model or contains characteristics that allow for slightly improved detection capabilities, especially for cars.

## 4.6 Chapter conclusion

In this chapter, we probed Research Question 1 **(RQ1):** how does employing a constrained and focused dataset affect the real-time performance of object detection and classification? By developing a specialised dataset centred on target vehicles, we can leverage transfer learning techniques to fine-tune YOLOv5m, thereby enhancing its ability to identify our specific vehicles. This specialised training will be conducted without compromising the model's existing accuracy for vehicle recognition on novel data, as demonstrated by its performance on the unseen Bo video dataset during real-time detection tasks.

Contributions described in this chapter include the following:

- Inference time and accuracy quantitative comparison of YOLOv5 and Faster R-CNN models using our video dataset.

- The construction of a target-based vehicle image dataset tailored to our video data.

The YOLO family of models is well-recognised for its challenges in detecting small objects. Our comparative analysis of Faster R-CNN and YOLOv5 revealed that YOLOv5m outperforms Faster R-CNN ResNet50 in inference speed and is only marginally behind in detection and classification accuracy. This observation negates the significance of the limitation related to small object classification, as our primary focus centres on detecting larger objects.

Initial attempts to enhance the accuracy of YOLOv5m by creating a specialised dataset yielded marginal success. To thoroughly evaluate the impact of such a dataset, we plan to conduct experiments across all YOLOv5 models, a topic we discuss in Chapter 5.

Transfer learning plays a pivotal role in our methodology. In this context, we harnessed the widely acclaimed COCO dataset to expedite the training of our models. The COCO dataset encompasses various vehicle classes, offering the dual advantage of reducing training time and granting access to an extensive collection of tens of thousands of pre-labelled images that seamlessly complement our customised dataset.

Chapter 5 addresses additional experiments aimed at assessing model performance. These experiments involved modifications to the raw video input data and adjustments to neural network dimensions. Throughout these experiments, we maintained transfer learning as a critical technique for evaluating various YOLOv5 models and determining the most effective parameters for extracting feature vectors from moving vehicles using 2D video.

# Chapter 5: Optimising target vehicle detection and classification

## 5.1 Introduction

This chapter provides an in-depth look at our steps to refine a detection and classification model that allowed us to extract feature vectors from target vehicles and pass them to the next stage of our pipeline. We establish the link between the research carried out in Chapters 3 and 4, related to the Bo video dataset, and the subsequent phase in the pipeline. This subsequent phase involves the extraction of feature vectors from target vehicles identified in the recorded video footage. In this chapter, we are investigating research question 2 (RQ2) Considering the neural network's characteristics, how do pixel density and frame rate variations affect real-time object detection and classification models?

The primary goal of this thesis is to utilise driver's historical behaviour at a T-junction to investigate how to make precise predictions about their future actions. The first feature we study is the vehicle's velocity. We carefully establish an evaluation velocity vector through a manual calibration process. This helps us to tune the accuracy of the captured velocity feature vector when using various combinations of video specifications.

We use the dynamic pixel-level feature vectors derived through DUKE for vehicles. These feature vectors cannot be obtained from readily available datasets or driving simulators but result from the substantial work detailed in Chapters 3 and 4. This deliberate approach has been adopted to ensure the utilisation of authentic data obtained from our video input methodology. The aim of this chapter is to test various inputs, frames per second, resolution, and neural network dimensions to determine the optimal settings for maximum performance and precision in producing feature vectors from two-dimensional video for our specific problem.

This chapter explains the fine-tuning process applied to the Bo video data, as outlined in Section 3.1. This process involves the computation of ground truth values from junction images. We aim to configure the input video to meet the specifications for ensuring real-time capabilities for detection, classification, tracking, and the subsequent extraction of feature vectors. Following this optimisation, the enhanced video feed seamlessly integrates into the DUKE framework, facilitating the capture of pixel-level feature vectors. These feature vectors are consecutively consolidated within a comprehensive dataset, which undergoes thorough examination in Chapter 7. Our methodologies have been designed to examine the interaction of various frame rates and resolution combinations that impact target vehicles' detection performance and accuracy. This analysis seamlessly integrates with our broader investigation of Research Question 2.

**The following contribution is covered in this chapter:**

We demonstrate an innovative approach to fine-tuning a real-time vehicle detection and classification model based on performance.

### 5.1.2 Chapter organisation

This chapter evaluates and optimises the real-time vehicle detection system's performance, including factors such as frame rate and resolution and selecting suitable YOLOv5 models and datasets. This paper demonstrates the experimental results using various frame rates and image resolutions to establish the optimum configuration for DUKE input based on accuracy, inference time, and ground truth realism.

Firstly, we lay the groundwork for evaluating the real-time vehicle detection model's performance. Our approach involves an extensive examination of evaluation metrics and the procedures employed for this assessment. This encompasses a discussion of various evaluation metrics, with particular attention given to frame rate and resolution samples (5.3.1 and 5.3.2) and the creation of video samples featuring diverse frame rates and resolutions (5.3.3). Moving to Section 5.4, we delve into the distance and velocity feature vector evaluation metrics assessment. Subsequently, in Section 5.5, we introduce the concept of a performance-based detection model. This entails training all YOLOv5 models through transfer learning (5.5.1) and revealing our methodology for selecting datasets and model combinations (5.5.2). Section 5.6 is dedicated to the experimental phase, wherein we explore the impact of varying parameters. We systematically investigate different combinations of frame rates, network architectures, and resolutions to discern the most optimal input video. This section also discusses selecting the most suitable input video (5.6.1). The concluding section (5.7) summarises the pivotal findings and discusses the selected $V_o$ and detection model/dataset combination.

## 5.2 Feature selection

Feature selection is the initial step in optimizing our video input. We have decided to focus on two primary features, distance and velocity, and setting various FPS rates and a range of video input resolutions.

### 5.2.1 Frame rate

The frame rate, measured in frames per second (FPS), indicates the speed at which individual video frames are captured or shown. For the Bo video dataset, the FPS rate directly influences DUKE's operational efficiency, as it determines the volume of data processed every second. A higher FPS means more data moves through our pipeline each second, potentially slowing down our processing speed but providing greater detail in each frame for analysis. Conversely, a lower FPS rate speeds up processing but provides less detail per frame for inference.
Reducing the FPS involves using video editing software to remove frames symmetrically by deleting alternate frames or using an asymmetric trimming technique, as demonstrated in Table 10, to achieve a specific frame rate reduction, allowing us to refine $V_o$ without restricting data from $V_r$.
Bo video data, $V_r$, is recorded at 50 fps. We have experimented with various frame rates; for example, to get to a target frame rate of 30 fps, we used Asymmetrical frame rate reduction, which removes 20 frames from each second of the original 50-frame video. Because 50 fps and 30 fps are evenly divisible by five frames, we divided each second of the video into five distinct blocks. This frame-trimming process transforms a ten-frame block from the source video clip into a six-frame block in the destination video clip, achieving the desired frame rate reduction. Please refer to Table 10 for more details.

| Step | Action | New Frame(s) in Destination Block |
|---|---|---|
| 1 | Frame 1 is deleted | |
| 2 | Frames 2 and 3 are played successively | 1, 2 |
| 3 | Frame 4 is deleted | 1, 2 |
| 4 | Frame 5 is displayed | 1, 2, 3 |
| 5 | Frame 6 is deleted | 1, 2, 3 |
| 6 | Frame 7 is exhibited | 1, 2, 3, 4 |
| 7 | Frame 8 is deleted | 1, 2, 3, 4 |
| 8 | Frames 9 and 10 are presented | 1, 2, 3, 4, 5, 6 |

*Table 10 Asymmetrical frame rate reduction*

The procedure outlined in Table 10 is applied iteratively to the entire video clip to achieve the

desired frame rate reduction. It is important to note that the use of asymmetric trimming, as described, can negatively impact the dimensional data, primarily due to the perturbation it introduces to the optical flow of the original video, which can further affect the tracking of target vehicles in our pipeline.

Using Bo $V_r$, we created samples of 50 fps, 30 fps, 25 fps, and 10 fps to analyse extremes and fine-tune the balance between the quantity of data available per second and inference speed.

### 5.2.2 Creating video samples of various FPSs and resolutions

The area in pixels establishes the resolution of a video. The input video can be passed to DUKE in various resolutions; however, the object detection and classification steps resize the images based on a hyperparameter setting. YOLOv5 is trained on resized images set to 640x640 and uses padding and cropping to establish the correct parameters. Cropping would involve cutting off parts of the original image to fit the new size and thus potentially losing important content. Padding would add blank space around the image to maintain its dimensions. We created samples of Bo $V_r$ at 1920x1080, 1280x720, 640x640, and 320x320. Our experimental video samples were created from the same clip of Bo $V_r$ converted from the original 1920x1280 at 50 fps to the combined samples presented in Table 11.

| Sample Resolution in pixels | FPS | | | |
|---|---|---|---|---|
| 1920x1280 | 10 | 25 | 30 | 50 (Bo $V_r$) |
| 1280x720 | 10 | 25 | 30 | 50 |
| 640x640 | 10 | 25 | 30 | 50 |
| 320x320 | 10 | 25 | 30 | 50 |

*Table 11 Input video samples created from a single Bo $V_r$ clip.*

## 5.3 Distance and velocity feature vector evaluation metrics

To obtain real-time dynamic data from video input, validating and establishing calibration references accurately reflecting real-world scenarios is essential. These references are used to verify feature vector data and establish the optimum parameters of input video, network, and model size. Specifically, we need the capability to compute metrics such as junction dimensions, vehicle size, and the velocity of vehicles. We demonstrated our methodology for calculating distance and velocity metrics using a single video frame, as depicted in Figure 18. This procedure was carried out for each of our tesT-junctions to establish specific baseline metrics tailored to each junction. The process includes manual calculations using the image manipulation software GIMP (GNU Image Manipulation Program 2023). We manually assessed the following samples: 1920x1080, 1280x720, 640x640, and 320x320, as detailed

in Sub-section 5.3.2.

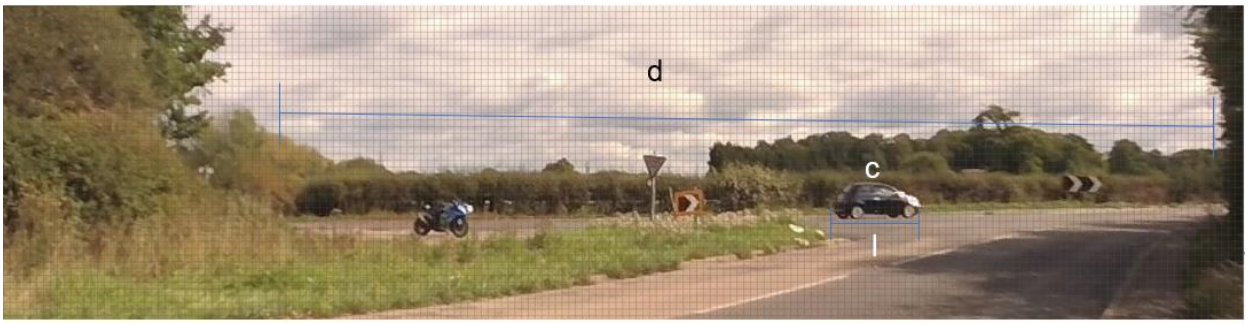



*Figure 18, Junction image with a grid overlay not to scale (GNU), Shows how we calibrate for a junction based on a single image.*

The image presented in Figure 14 corresponds to a single frame extracted from Bo Vr, recorded at a resolution of 1920x1080 pixels and a frame rate of 50 fps.
In the video frame in Figure 14, the vehicle identified as a Fiat 500 (labelled as 'c') possesses an actual physical length of 3,571 mm, as reported by Parkers (2021). This vehicle's pixel length in the image is 63 pixels, labelled as l and resolved as 1 pixel, corresponding to 56.68 mm (1 px = 56.68 mm). The distance covered by the Fiat 500 c in this clip is 639 pixels, relative to a physical distance of 36,220 mm (or 36.22 meters), denoted as d. The time to traverse this distance is computed as 5.10 seconds, t, extrapolated from Vr data across all frames. The estimated mean velocity for the Fiat 500 in Figure 14 is 7.10 m/s or 7.10 mm/ms. This velocity is further resolved into a mean relative velocity, $v_c$, of 0.125 px/ms for the sample 1920x1080 at 50 fps. As the vehicle moves horizontally, the relative size does not change significantly; however, we took length measurements of the car in pixels over the entire frame length and averaged the target vehicle length to 63 px. $V_c$ for all tested resolutions was within 5% of 0.125 px/ms due to the relative pixel size and actual physical measurements. A higher fps can represent life-like vehicle motion more accurately because it captures more incremental positions of the object as it moves through space; this makes the vehicle appear to move smoothly and at its actual speed. When the fps is lower, vehicle objects appear to jump between frames, making them appear slower than they are.

## 5.4 Establishing a detection model

After defining the scope of input video specifications and evaluation metrics for our DUKE experiments to determine the optimal input (Bo $V_o$), our next step is fine-tuning the model's base to achieve the desired inference time and accuracy combination to exploit real-time feature vector capture.

### 5.4.1 Training all YOLOv5 models with transfer learning

Expanding on the details provided in Chapter 4, which outlined our approach to transfer learning using our specific vehicle image dataset, we then perform transfer learning on all five YOLOv5 models, denoted as 'n,' 's,' 'm,' 'l,' and 'x'. The employed training datasets are as follows:

1. **COCO 80**: This extensive dataset encompasses all the original classes from the COCO dataset, providing a comprehensive context for object detection. It is worth noting that all YOLOv5 models are pre-trained using this dataset.
2. **Bo (80)**: Our specialised vehicle weights have been incorporated into the COCO 80 dataset during training, enriching the dataset with our target vehicle instances.
3. **COCO 5**: This dataset has been refined to include classes directly pertinent to vehicle detection tasks exclusively. It eliminates all other classes, retaining only the categories of bicycle, motorcycle, car, truck, and bus from the original COCO dataset.
4. **Bo (5)**: Building upon the foundation of COCO 5, this dataset integrates our proprietary vehicle classes, further enhancing the dataset to create a more finely tuned and specialised vehicle classification model.

The YOLOv5 models underwent transfer learning training following the procedures outlined in Sub-section 4.5.3. The resulting outcomes are presented in Table 12, featuring performance metrics such as mean average precision (mAP) at various intersection over union (IoU) thresholds, inference speeds on CPU and RTX 3070 GPU, and frames per second (FPS). Throughout the training process, our primary focus was to minimise the overall loss. This objective was achieved through stochastic gradient descent optimisation (SGD), as detailed by Li et al. (2022). During training epochs, the model's weights were systematically adjusted to reduce the loss by employing a combination of loss functions to quantify the disparities between predicted bounding boxes and ground-truth bounding boxes, as well as objectness and class predictions. Further insights into the primary loss components are available in Sub-section 4.3.1.

| Dataset | Neural Network | mAP 0.5:0.95 | mAP 0.5 | Mean Inference (ms) CPU | Mean Inference (ms) RTX 3070 | FPS |
|---|---|---|---|---|---|---|
| Bo (5) | YOLOv5n | 28.9 | 46.7 | 76.9 | 10.9 | 94 |
| COCO 5 | YOLOv5n | 28.1 | 45.8 | 77.2 | 11.9 | 93 |
| Bo (80) | YOLOv5n | 28.2 | 46.4 | 73.4 | 12.8 | 81 |
| COCO 80 | YOLOv5n | 27.9 | 45.7 | 75.4 | 13.1 | 85 |
| Bo (5) | YOLOv5s | 37.5 | 56.8 | 88.3 | 13.1 | 61 |
| COCO 5 | YOLOv5s | 37.1 | 56.7 | 90.9 | 14.6 | 61 |
| `Bo (80) | YOLOv5s | 37.3 | 57.7 | 115.7 | 16.4 | 69 |
| COCO 80 | YOLOv5s | 37 | 56.7 | 116.1 | 17.2 | 74 |
| Bo (5) | YOLOv5m | 46.4 | 65.3 | 105.1 | 18.2 | 53 |
| COCO 5 | YOLOv5m | 45.8 | 64.7 | 107.2 | 19.8 | 51 |
| Bo (80) | YOLOv5m | 45.9 | 65.0 | 132.2 | 20.5 | 38 |
| COCO 80 | YOLOv5m | 45.3 | 64.1 | 134.5 | 21.9 | 48 |
| Bo (5) | YOLOv5l | 50.1 | 68.3 | 117.2 | 23.2 | 49 |
| COCO 5 | YOLOv5l | 49.3 | 67.9 | 119.0 | 24.7 | 48 |
| Bo (80) | YOLOv5l | 49.5 | 67.7 | 144.2 | 30.4 | 40 |
| COCO 80 | YOLOv5l | 49 | 67.3 | 146.9 | 31.6 | 44 |
| Bo (5) | YOLOv5x | 51.6 | 69.0 | 129.1 | 32.6 | 28 |
| COCO 5 | YOLOv5x | 50.6 | 68.9 | 131.5 | 33.8 | 25 |
| Bo (80) | YOLOv5x | 50.9 | 69.7 | 157.0 | 42.9 | 38 |
| COCO 80 | YOLOv5x | 50.5 | 68.9 | 158.4 | 43.6 | 22 |

*Table 12: Comparative Analysis of Neural Network Configurations and Performance Metrics for YOLOv5 Architecture in Target Vehicle Detection*

In Table 12, we present a comparative analysis of the performance of neural networks utilising the YOLOv5 architecture. The datasets underused include the refined datasets Bo (5) and COCO 5 and the entire object datasets Bo (80) and COCO 80.

**Key to Table 12:**

**Dataset**: Bo (5), COCO 5, Bo (80), and COCO 80, described above.

**Neural Network**: lists the specific neural network architecture used for the evaluation. In all cases, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x are trained with Bo.

**mAP 0.5:0.95**: is the primary benchmark used in the literature for object detection mAP is averaged over all categories in the dataset, using a range of IoU 0.5 to 0.95

**mAP 0.5**: provides a singular evaluation of detection accuracy compared to a range of IoU thresholds.

**Mean Inference (ms) CPU**: This column shows the mean inference time in milliseconds when the model is run on a CPU.

**Mean Inference (ms) RTX 3070**: This column displays the mean inference time in milliseconds when the model is run on an NVIDIA RTX 3070 GPU.

**FPS**: This measures how quickly the model can process images, with higher values indicating faster processing.

Initial results discussion based on Table 12:

- As the model size increases from n to x, a clear trend of increasing mAP (0.5:0.95 and 0.5) indicates higher accuracy but at the cost of longer inference times and lower FPS.
- The inference time on the CPU is significantly higher than on the RTX 3070 GPU across all models, highlighting the advantage of using powerful GPUs for deep learning inference tasks.
- Generally, there is a trade-off between accuracy (mAP) and speed (FPS, Mean Inference Time). Larger models are more accurate but slower, making them suitable for high-accuracy requirements where inference time is less critical. Conversely, smaller models are faster but less accurate, suitable for real-time or low-latency applications.

For Maximum Accuracy: The YOLOv5x model achieves the highest mAP scores across both datasets, indicating their superior ability to detect objects accurately. However, this comes at the cost of lower FPS and longer inference times, especially on GPU.

Balance Between Accuracy and Speed: Mid-range models like YOLOv5m or YOLOv5l are better choices. They offer a middle ground in terms of both accuracy and processing speed.

Upon concluding the training phase, an analysis was conducted on the data outlined in Table 12. Next, we generated accuracy and inference plots. This methodology enabled us to pinpoint models demonstrating either low accuracy or excessive latency, as shown in Figure 19. This figure presents a comprehensive view of the dataset, model, inference, and accuracy, showing that the more accurate and robust models have the highest latency. Furthermore, in Figure 16, we conducted a comparative analysis of the processing speed of each model in terms of FPS to establish insights into the efficiency of each model in image processing tasks for an informed assessment of their suitability in real-world scenarios.

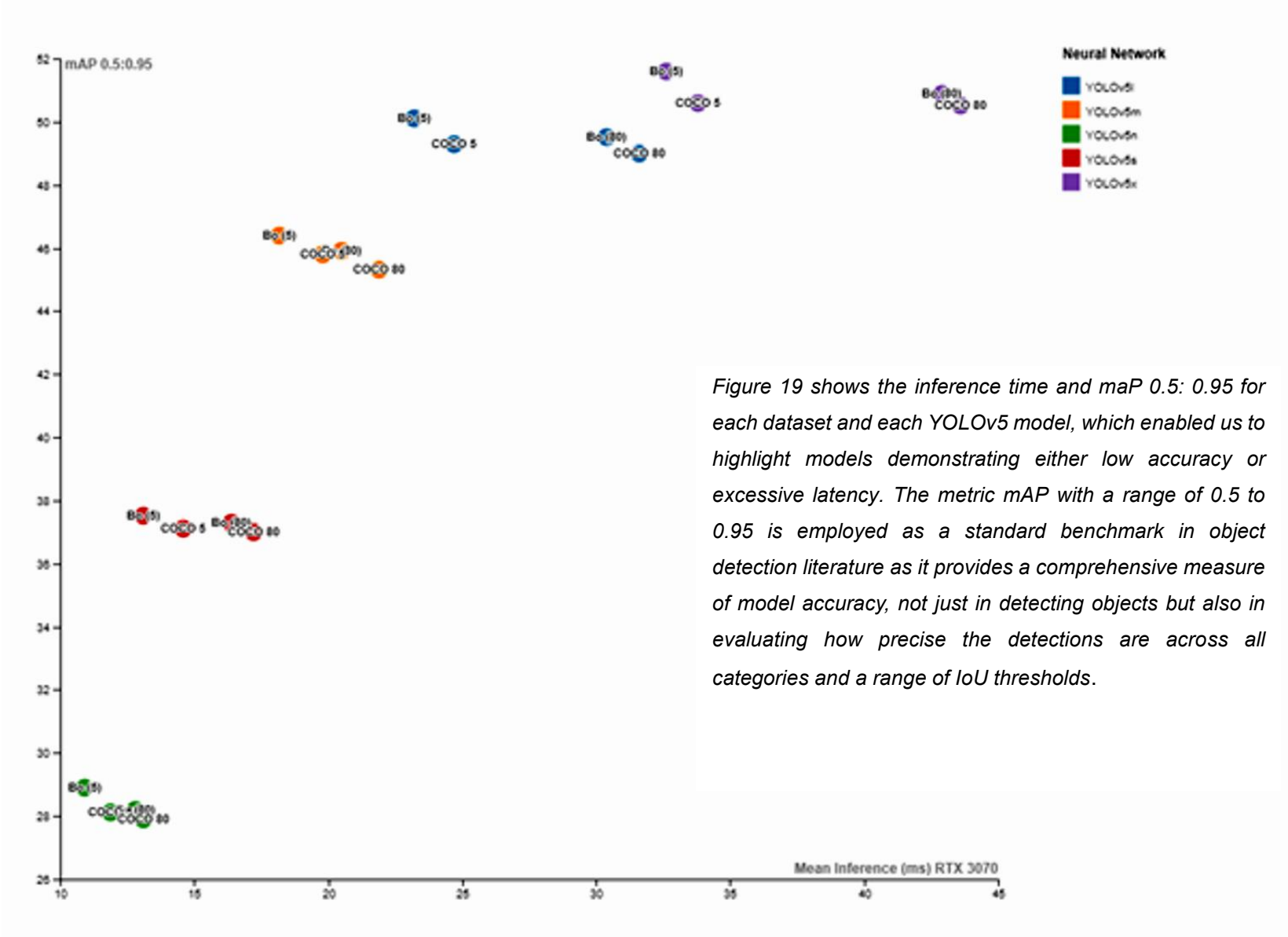



*Figure 19 shows the inference time and maP 0.5: 0.95 for each dataset and each YOLOv5 model, which enabled us to highlight models demonstrating either low accuracy or excessive latency. The metric mAP with a range of 0.5 to 0.95 is employed as a standard benchmark in object detection literature as it provides a comprehensive measure of model accuracy, not just in detecting objects but also in evaluating how precise the detections are across all categories and a range of IoU thresholds*.

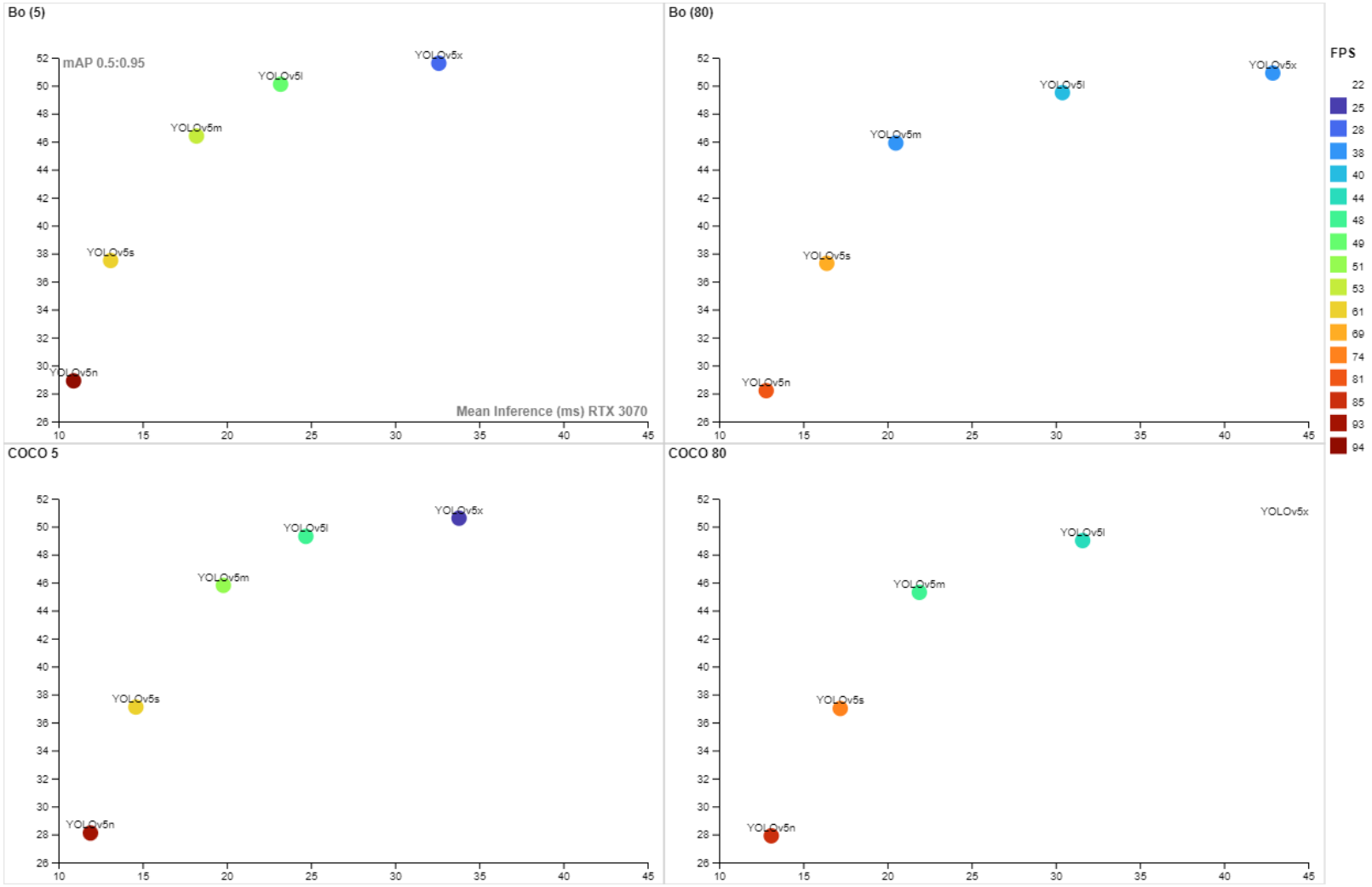



*Figure 20 Dataset performance based on YOLOv5 model with FPS.*

### 5.4.2 Selecting dataset and model combination

The findings from Sub-section 5.4.1 indicate that models with the quickest image processing capabilities, specifically the 'nano' and 'small' models, display reduced accuracy on both training and testing datasets for both COCO and Bo combinations. This trend is attributed to these faster models' simpler neural network architecture. In contrast, the filtered datasets COCO 5 and Bo (5) have demonstrated the best performance in terms of accuracy and inference, particularly with the medium, large, and xlarge models ( m, l, and x, respectively). We chose to discard less accurate and slower models to streamline our approach and focus on models that balance speed and accuracy. This decision led us to retain the following models for the next stage in creating a fast and accurate detection and classification process to feed into our pipeline: Bo (5) and COCO (5) medium, large, and xlarge models, as seen in Table 13 and Figure 17.

| Model | mAP 0.95 | mAP 0.5 | FPS | Inference ms |
|---|---|---|---|---|
| COCO 5 x | 50.6 | 68.9 | 28 | 33.8 |
| Bo (5) x | 51.6 | 69.0 | 25 | 32.6 |
| COCO 5 l | 49.3 | 67.9 | 48 | 24.7 |
| Bo (5)l | 50.1 | 68.3 | 49 | 23.2 |
| COCO 5 m | 45.8 | 64.7 | 51 | 19.8 |
| Bo (5) m | 46.4 | 65.3 | 53 | 18.2 |

*Table 13 Summary of models displaying a balanced performance and accuracy*

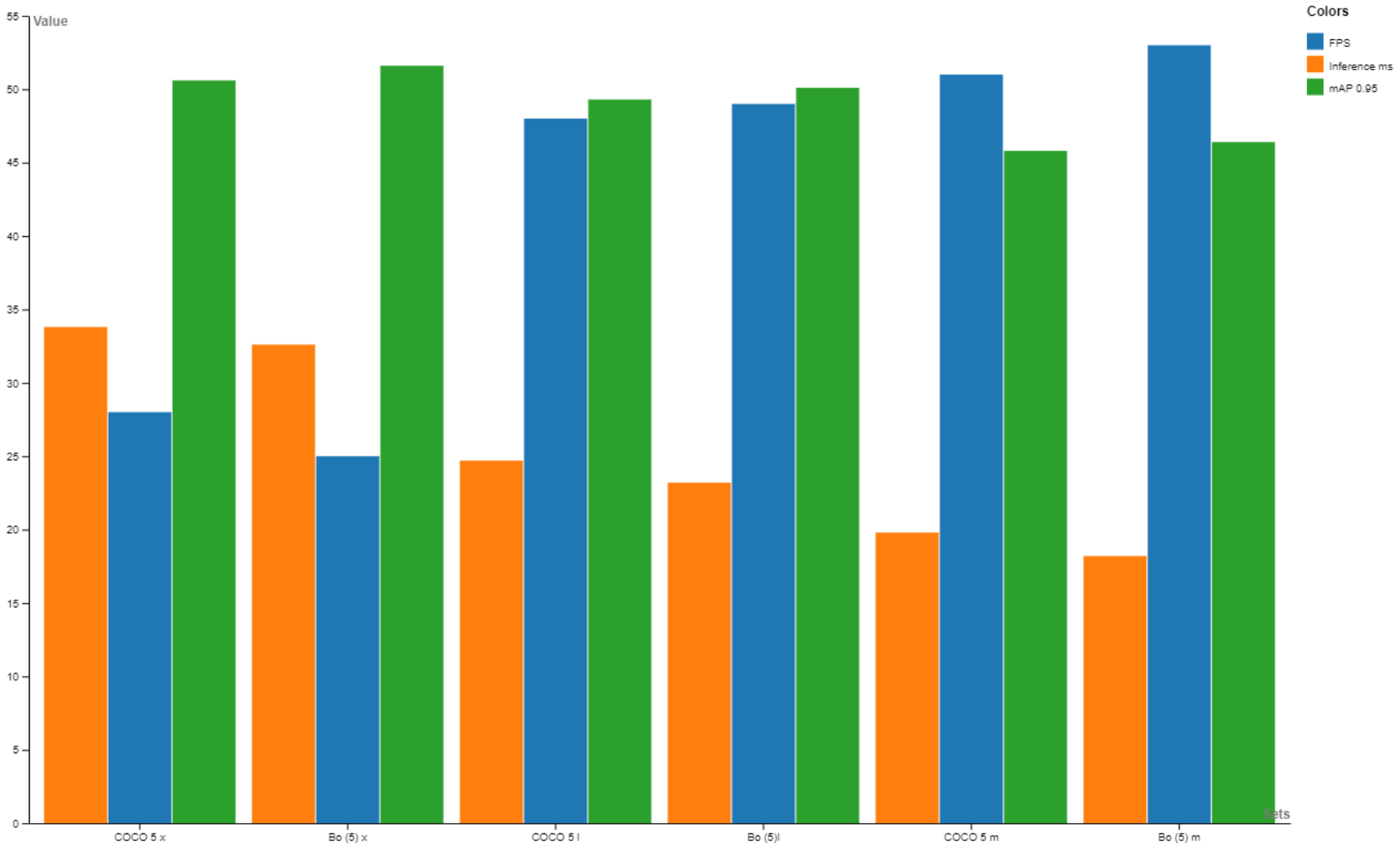



*Figure 21 compares datasets and models regarding FPS, inference time, and mAP (mean average precision) at a 0.5:0.95 IoU threshold.*

Further experimentation is required to establish the most appropriate model for our work. We next pass variations of $V_o$ through each model and analyse the real-time behaviour. The challenge lies in developing a model that simultaneously achieves high accuracy, high FPS, and a low inference time.

### 5.5 Experimentation with a variety of FPSs, neural networks, resolutions, and Vo

Our next step was to establish the effect of variations in pixel density and frame rate in $V_o$ on the performance of our object detection and classification methods and how the dimensions and architecture of the YOLOv5 models affect this **(RQ2).**

Each experiment recorded as a row in Table 14 represents a unique combination of the abovementioned variables. The system's performance is evaluated under each condition to understand how different factors affect its accuracy (car class confidence), speed (average velocity and inference time), and overall efficiency.

The system processes video or image inputs at the specified resolution and frame rate, using the indicated image dataset and model for object detection and classification.

Performance metrics are recorded for each setup, including the confidence level of the classifications made, the average velocity of detected vehicles, and the time the system takes to infer results from the input data.

#### 5.5.1 Video samples

In the next batch of experiments, we used the samples of $V_o$ from Sub-section 5.3.3 (Table 11) at varying resolutions and FPSs and used the selected models from Sub-section 5.5.2 Bo (5) and COCO 5, m,l,x. to test real-time detection confidence and to compare the estimated velocity with our benchmark velocity from Section (5.4) of 0.125 px/ms. Figure 18 represents the viable data yielded from our experiments after removing outliers, such as the 10 fps samples, due to very low confidence scores and distance from benchmark velocity.

#### 5.5.2 Class confidence

The class confidence metric in YOLOv5, and other object detection models like it, quantify the model's certainty in its predictions regarding the presence and class of objects within an image. In YOLOv5, this confidence score is between 0 and 1; higher values indicate greater confidence in the prediction. This metric is crucial for filtering out detections with low confidence, thus reducing false positives and improving the overall precision of the model.

Objectness Score: Each bounding box predicted by YOLOv5 has an associated objectness score that indicates the model's confidence that the box contains an object versus the background. This score helps filter out bounding boxes that likely do not contain any object.

Class Confidence Score: For each bounding box, the model also predicts a class confidence score for each class. This score reflects the model's confidence that the object in the bounding box belongs to a specific class.

Combined Confidence Score: The final confidence score for a prediction is typically the product of the objectness score and the class confidence score. This combined score represents the model's overall confidence that a specific class object is present in the predicted location.

Non-Maximum Suppression (NMS): YOLOv5 applies Non-Maximum Suppression to eliminate redundant bounding boxes after predictions are made. NMS uses confidence scores to retain the best bounding box when multiple boxes overlap and detect the same object. The box with the highest confidence score is kept, while others are discarded.

Thresholding: Users can set a confidence threshold to filter out detections. Detections with confidence scores below the threshold are discarded. This threshold can be adjusted based on the application's requirements to balance between precision (high confidence threshold) and recall (low confidence threshold).

Tuning the confidence threshold parameter is critical to optimizing YOLOv5 for specific tasks, as it allows users to balance between detecting as many objects as possible (recall) and ensuring the detections are accurate (precision).

Class confidence is typically outputted by the last layer of the object detection model, which often involves a softmax or sigmoid function providing a probability distribution over all possible classes. The most common metric used to evaluate class confidence and localisation in object detection is IoU combined with precision and recall metrics. Since mAP represents the mean accuracy across all dataset classes, the confidence level of the vehicle class in ground-truth data is crucial.

### 5.5.3 Results from experimentation with FPSs, neural networks, resolutions, and $V_o$

| # | Vr Resolution pixels | FPS of $V_r$ | Image Dataset & Model | Car Class Confidence % | Ave Velocity px/ms | Inference ms |
|---|---|---|---|---|---|---|
| 1 | 320 | 50 | Bo (5)m | 59 | 0.1 | 22 |
| 2 | 320 | 50 | COCO 5m | 60 | 0.11 | 22 |
| 3 | 320 | 30 | Bo (5)l | 66 | 0.13 | 24 |
| 4 | 320 | 30 | COCO 5l | 65 | 0.135 | 24 |
| 5 | 320 | 50 | COCO 5l | 61 | 0.14 | 24 |
| 6 | 320 | 50 | Bo (5)l | 61 | 0.142 | 24 |
| 7 | 320 | 50 | COCO 5x | 62 | 0.104 | 30 |
| 8 | 320 | 50 | Bo (5)x | 63 | 0.108 | 30 |
| 9 | 320 | 30 | Bo (5)x | 68 | 0.125 | 30 |
| 10 | 320 | 30 | COCO 5x | 70 | 0.126 | 30 |
| 11 | 640 | 50 | Bo (5)m | 69 | 0.139 | 20 |
| 12 | 640 | 50 | COCO 5m | 67 | 0.14 | 22 |
| 13 | 640 | 50 | Bo (5)l | 76 | 0.106 | 23 |
| 14 | 640 | 50 | COCO 5l | 74 | 0.105 | 26 |
| 15 | 640 | 30 | Bo (5)l | 79 | 0.15 | 26 |
| 16 | 640 | 30 | COCO 5l | 79 | 0.16 | 26 |
| 17 | 640 | 30 | Bo (5)x | 80 | 0.13 | 32 |
| 18 | 640 | 30 | COCO 5x | 80 | 0.13 | 32 |
| 19 | 1280 | 30 | Bo (5)m | 78 | 0.158 | 22 |
| 20 | 1280 | 30 | COCO 5m | 77 | 0.16 | 22 |
| 21 | 1280 | 50 | Bo (5)m | 79 | 0.14 | 24 |
| 22 | 1280 | 25 | Bo (5)l | 83 | 0.17 | 24 |
| 23 | 1280 | 25 | COCO 5l | 83 | 0.173 | 24 |
| 24 | 1280 | 50 | COCO 5m | 77 | 0.141 | 25 |
| 25 | 1280 | 30 | Bo (5)l | 82 | 0.11 | 26 |
| 26 | 1280 | 30 | COCO 5l | 82 | 0.12 | 26 |
| 27 | 1280 | 50 | Bo (5)l | 80 | 0.11 | 29 |
| 28 | 1280 | 50 | COCO 5l | 81 | 0.1 | 30 |
| 29 | 1920 | 30 | Bo (5)m | 77 | 0.098 | 22 |
| 30 | 1920 | 30 | COCO 5m | 75 | 0.099 | 22 |
| 31 | 1920 | 50 | COCO 5m | 78 | 0.128 | 23 |
| 32 | 1920 | 50 | Bo (5)m | 79 | 0.129 | 23 |
| 33 | 1920 | 25 | Bo (5)l | 81 | 0.16 | 35 |
| 34 | 1920 | 25 | COCO 5l | 80 | 0.16 | 35 |
| 35 | 1920 | 25 | Bo (5)x | 84 | 0.1 | 38 |
| 36 | 1920 | 25 | COCO 5x | 83 | 0.11 | 39 |

*Table 14 summarises input data for $V_o$ and outputs based on the dataset and model. It comprises the parameters that define each model's real-time performance and capabilities, including resolution, processing speed, confidence levels, car velocities, and the time taken for inference.*

**Table 14 Key**

- Resolution (pixels): Input pixel resolution ranges from 320 to 1920 pixels.
- Vr Video data FPS(set when the video is edited): Range from 25 to 50 frames per second.
- Dataset and Model Combinations: Yolov5 model and dataset combination.
- Car Class Confidence (%): Confidence levels for car classification range from approximately 59% to 84%.
- Average Velocity (px/ms): Average car velocity in pixels per millisecond varies from 0.1 to 0.173.
- Inference Time (ms): The time taken for classification ranges from 20 to 39 milliseconds.

#### *5.5.3.1 Correlation analysis based on results in Table 14.*

$V_r$ Resolution and Car Class Confidence: A strong positive correlation (0.74) between $V_r$ resolution and car class confidence suggests that higher resolutions result in higher confidence levels.

FPS and Car Class Confidence: There's a moderate negative correlation (-0.50) between FPS and car class confidence, indicating that higher FPS rates might negatively impact confidence levels, potentially due to the increased computational demand.

$V_r$ Resolution and FPS: A negative correlation (-0.32) suggests that higher resolutions often come with lower FPS due to the increased processing required for higher resolutions.

Average Velocity and Car Class Confidence: A positive correlation (0.22) indicates that higher average velocities are slightly associated with higher confidence levels.

Inference Time: A positive correlation (0.35) between car class confidence and inference time suggests that higher confidence might come at the cost of slightly longer processing times. Additionally, inference time shows a slight positive correlation (0.23) with $V_r$ resolution, indicating that higher resolutions may lead to longer inference times.

**Interpretation:**

Resolution's Impact: The increase in $V_r$ resolution is positively associated with higher car class confidence, indicating that higher resolution may provide more detailed images for more accurate classification despite potentially lower FPS and slightly longer inference times.

FPS's Role: The negative correlation between FPS and car class confidence suggests that while a higher FPS is desirable for smooth motion, it might compromise classification confidence due to the reduced time available for processing each frame.

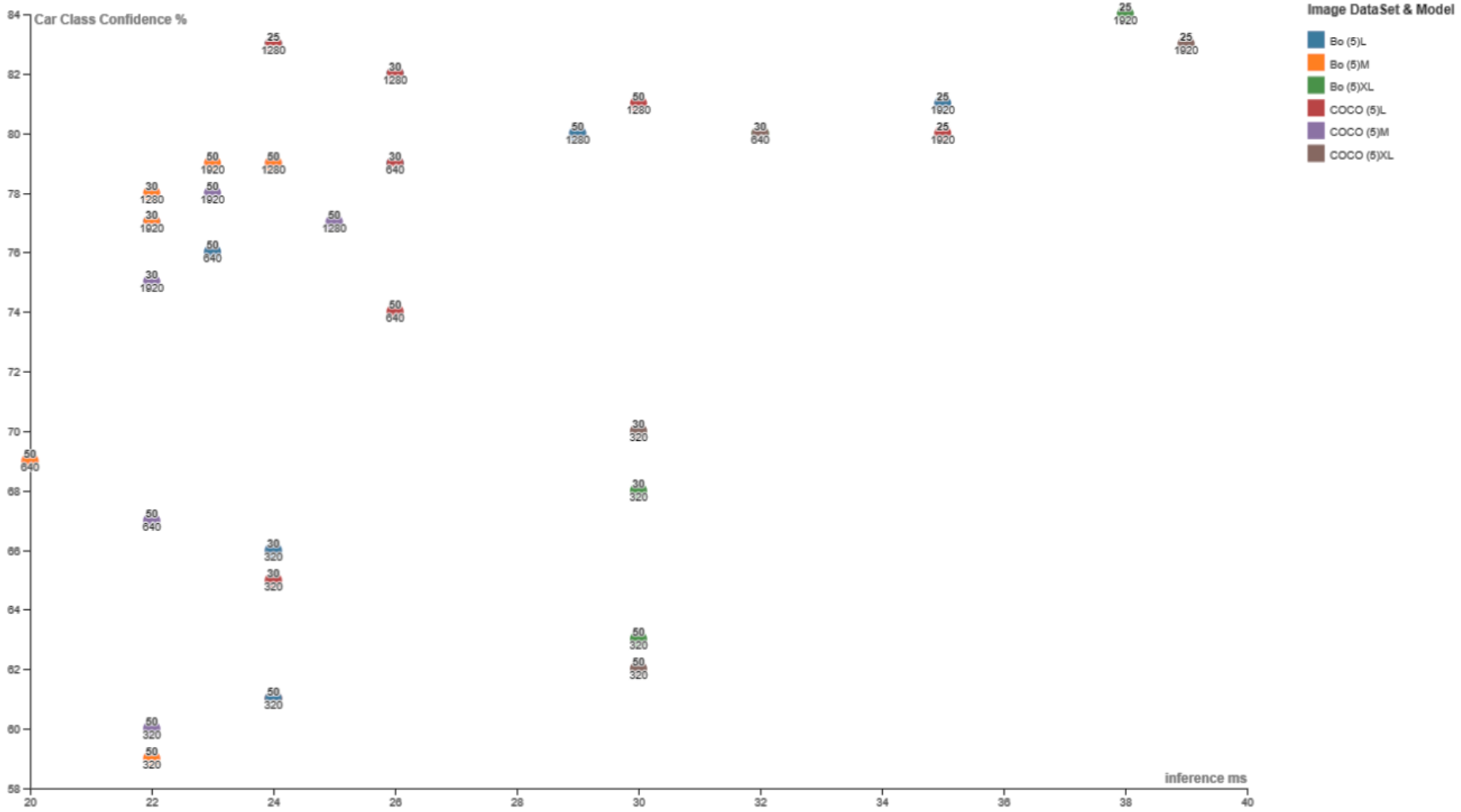



*Figure 22 Relationship between real-time classification confidence inference time, model, and $V_o$ input.*

These experiments help us address Research Question 2 (RQ2), examining the impact of pixel density and frame rate changes on real-time object detection and classification models. Figure 18 illustrates the complex relationship between input data quantity, resolution, FPS, neural network model size, inference time, and class confidence. To define our model's parameters, we compared the results from Table 14 by utilising our benchmark velocity of 0.125 px/ms, as outlined in Section 5.3, to gauge the accuracy of various model combinations against our real-time ground truth data. We excluded outlier data where velocity measurements fell outside the 0.12 to 0.13 px/ms range. The remaining data, adhering to our 0.01 px/ms tolerance, is summarised in Table 15.

| Vr Resolution pixels | FPS of Vr | Image Dataset & Model | Car Class Confidence % | Ave Velocity px/ms | Inference ms |
|---|---|---|---|---|---|
| 1280 | 30 | COCO 5l | 82 | 0.12 | 26 |
| 320 | 30 | Bo (5)x | 68 | 0.125 | 30 |
| 320 | 30 | COCO 5x | 70 | 0.126 | 30 |
| 1920 | 50 | COCO 5m | 78 | 0.128 | 23 |
| 1920 | 50 | Bo (5)m | 79 | 0.129 | 23 |
| 320 | 30 | Bo (5)l | 66 | 0.13 | 24 |
| 640 | 30 | Bo (5)x | 80 | 0.13 | 32 |
| 640 | 30 | COCO 5x | 80 | 0.13 | 32 |

*Table 15 Filtered results from Table 14 based on those closest to benchmark velocity of 0.125 px/ms in the realistic tolerance range of 0.12 to 0.13 px/ms*

The Bo models have Bo (5)l with the lowest confidence at 66%, Bo (5)m at 79%, and Bo (5)x at an average confidence of 74%. The COCO models show varying confidence levels, with COCO 5l at 82% (highest), COCO 5m at 78%, and COCO 5x at 75%.

The data suggests that a balance between resolution, FPS, and the choice of image dataset and model is crucial for optimizing car class confidence, inference time, and maintaining a relative ground truth of average velocity, as displayed in Figure 19.

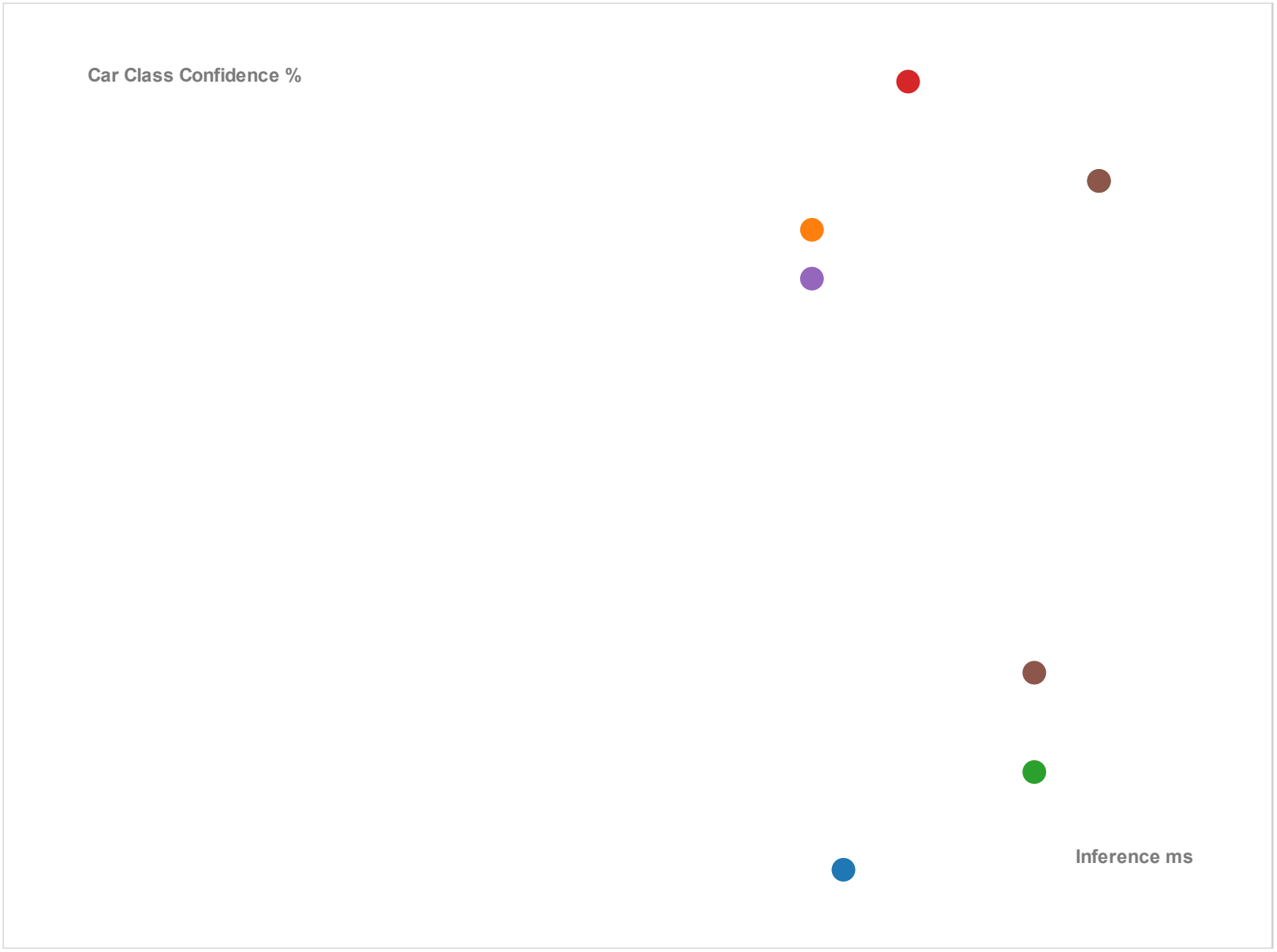



*Figure 23 Comparison of data from Table 15 above, showing dataset/ model class confidence and inference time.*

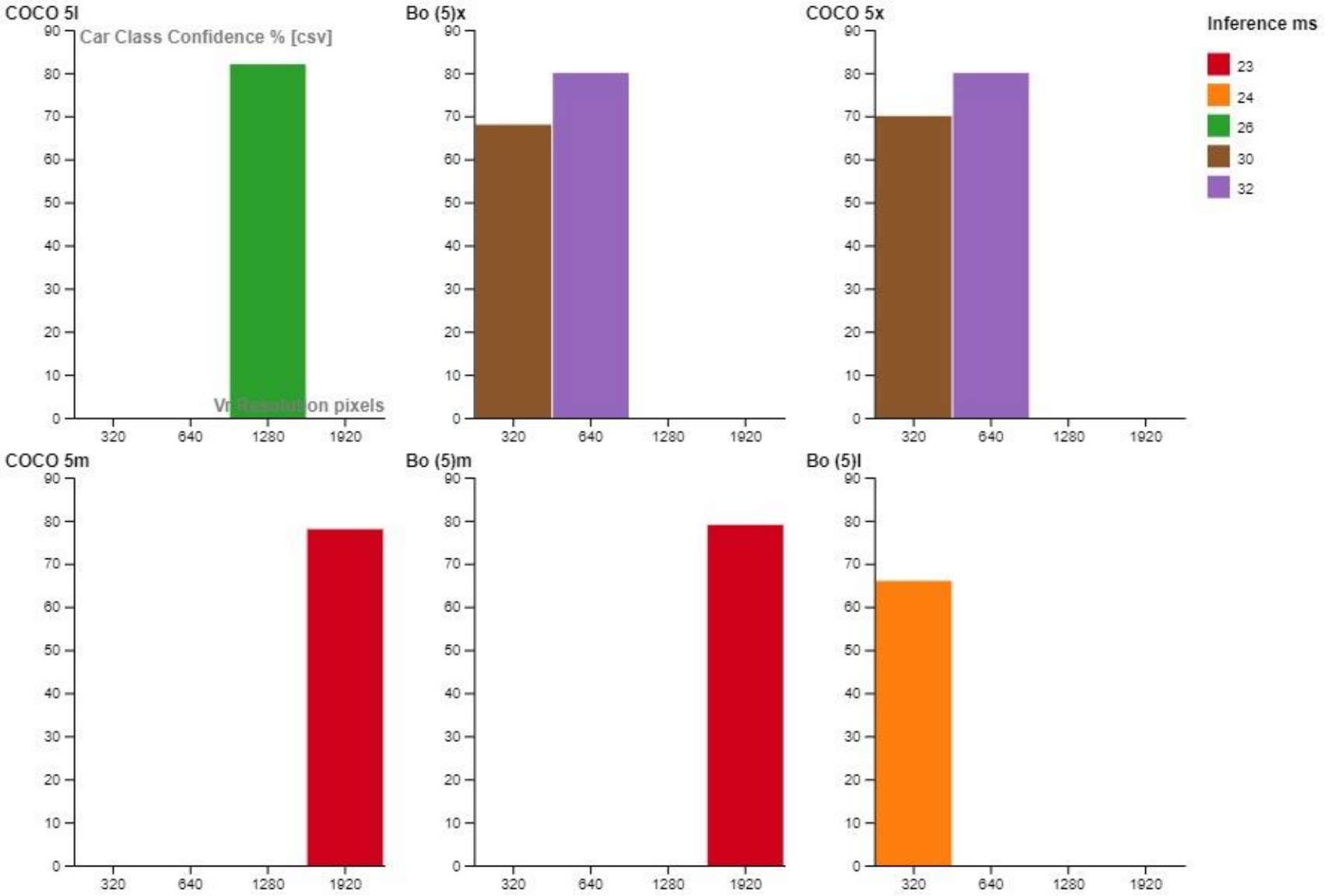



*Figure 24 A comparison of neural network/dataset combination, resolution and performance regarding inference time and car class confidence.*

*Figure 24 illustrates the connection between the dataset/model combination, resolution, car class confidence and inference time for values within our ground truth velocity threshold.*

The highest car class confidence is achieved by the COCO 5l model at a resolution of 1280 pixels and 30 FPS, with a confidence of 82%, shown in Figure 20. The Bo (5)m and COCO 5m models achieve the fastest inference times of 23 milliseconds at the highest resolution of 1920 pixels and 50 FPS, with car class confidence percentages of 79% and 78%, respectively. The lowest car class confidence is for the Bo (5)l model at 320 pixels resolution and 30 FPS, with a confidence of 66%. The inference time does not correlate directly with the resolution or confidence percentage. Both Bo (5)m and COCO 5m models show reasonable high confidence and low inference time balance, suggesting efficient performance. The COCO 5l model shows the highest confidence but at a lower resolution, which indicates a trade-off between resolution and accuracy.

## 5.6 Selecting the optimal $V_o$ based on resolution, fps, and ground truth data

A lower pixel density, such as 320, diminished the input images' data load, resulting in quicker inference times as smaller images generally entail less computational demand. However, this reduction in pixel density may compromise accuracy because it can lead to the loss of object details in scenarios involving occlusions or low lighting. Detecting and classifying objects, especially those at a distance, can become more challenging under such circumstances; our data illustrates this phenomenon. Regarding resolutions set at 320 (Table 14 # 1–8), we observe significantly reduced confidence in identifying the same vehicle compared to tests with higher resolutions despite the substantially quicker processing. Higher pixel density demands more significant computational resources, resulting in slower inference times. However, it frequently translates to heightened accuracy as the model benefits from additional image details, enabling more precise predictions. This trend is illustrated in our data (Table 14 # 35, 36), where data with rich pixel information at a resolution of 1920 exhibits the highest confidence in classifying target vehicles but incurs the slowest inference time.

Our experiments reveal that lower frame rates result in quicker inference times. Nonetheless, this approach may lead to inaccuracies in detecting moving objects, particularly those with high speeds, making the data from all 10 fps samples and the 25 fps data from the medium models unusable, as indicated in Table 14. We found that lower frame rates diminish object detection accuracy by reducing the available instances for analysis, and if $V_o$ is processed too quickly, our models fail to classify vehicles accurately. Overall, while higher FPS can be associated with better performance in some instances (as seen with COCO 5m and Bo (5)m), it is not the sole determining factor for car class confidence, which seems to be more dependent on the specific model and dataset rather than the FPS alone. Our empirical

observations demonstrate that increased frame rates require additional computational resources for all models, leading to slower inference times.

Nonetheless, a higher frame rate facilitates the capture of more comprehensive data per frame, which is especially advantageous for tracking and classifying rapidly moving objects, ultimately resulting in enhanced accuracy. The choice of neural network architecture plays a pivotal role in object detection. Smaller YOLOv5 models, such as nano and small, are designed for computational efficiency, whereas the large and xlarge models prioritise accuracy.

Our research also highlights the crucial trade-off between resolution and frame rate, a concept well-documented in the work of Huang et al. (2016). The higher the resolution, the greater the computational power required, potentially leading to a reduced processing frame rate during the detection phase and vice versa. It is paramount to find the right balance between resolution and frame rate, and this balance must be tailored to the specific requirements of our application, as emphasised in Chai et al. (2021).

Given our specific requirements, we conducted extensive experiments with various configurations, including pixel density, frame rate, and neural network models, to determine the optimal trade-off between inference speed and accuracy for our pipeline. It is worth noting that while we have not yet ventured into experimenting with hardware acceleration, model quantisation, or software optimisations, these approaches hold the potential to enhance the delicate equilibrium between speed and accuracy. However, these aspects fall beyond the scope of our current research in its present form. The models we have selected and presented in Table 15 and Figure 24 demonstrate that the choice of network model significantly impacts both inference time and accuracy, particularly when considering different combinations of $V_o$. Notably, the models in Figure 25 were initially selected from a larger batch of models, as outlined in Table 14. The subsequent filtering process focused on achieving a benchmark velocity within a narrow range of +/- 0.05 px/ms.

Our primary challenge, based on RQ2, is to identify the optimal combination of a model and $V_o$ for integration into our pipeline while minimising any adverse effects on the performance of our real-time driver predictions. In this context, the first prerequisite is inference time, and the best results were obtained using COCO 5m and Bo (5)m, operating at a $V_o$ of 50fps and a resolution (1920 pixels) equivalent to $V_r$.

The next crucial consideration revolves around accuracy, and the tests indicate that Bo (5) m marginally outperforms COCO 5m; the performance difference in car class confidence between COCO 5m and Bo (5)m is approximately 1.27%. Since both models have the same inference time, no performance difference exists. Therefore, the decision hinges on a balance between inference speed and accuracy. For our specific use case, choosing Bo (5) m with a $V_o$ of 50 fps at a resolution of 1920 pixels appears to be the most promising configuration.

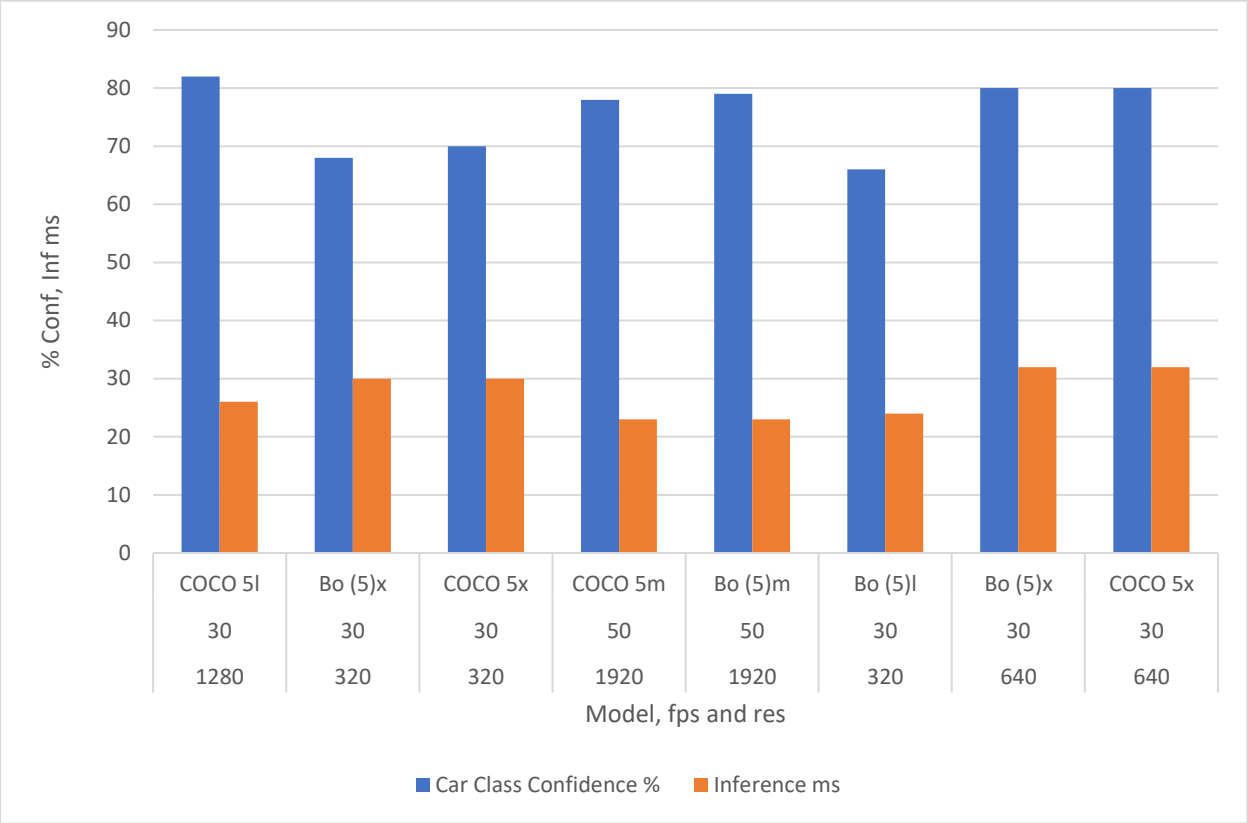



*Figure 25 Filtered models based on data from Table 15*

## 5.7 Chapter conclusion

In this chapter, we explored Research Question 2 (RQ2): Considering the neural network's characteristics, how do pixel density and frame rate variations affect real-time object detection and classification models?

We observed a well-defined relationship between the quantity of data and the computational resource requirements for our vehicle detection and classification model. Our work enhanced this relationship during the transfer learning stage, where we created a vehicle-specific image dataset that supported higher confidence in car class prediction over the base dataset COCO. It became evident that an extensive neural network, such as YOLOxl, delivers precise vehicle classifications when provided with high-resolution images at a high frame rate. In contrast, a simple neural network, such as YOLOn, fed low-resolution images at a low frame rate either exhibits erratic predictions or fails to detect vehicles. Our model requirements lie between these two extremes. Through iterative experimentation, we managed to identify a model and input specification that allowed us to capture the necessary level of detail from vehicles, enabling us to generate feature vectors as swiftly as possible, considering the limitations of our available models. This groundwork is instrumental in our pursuit of building a prediction model capable of detecting, classifying, tracking, extracting feature vectors, storing those vectors, and making predictions in real-time. With these findings, we are prepared to employ our chosen model of Bo (5) m, with a frame rate of 50 fps and a resolution of 1920x1080, as the video input for the next stage of our pipeline, elaborated on in the following chapter.

In Chapter 6, we introduce an innovative approach to extracting feature vectors from our video dataset Bo and storing these for training our predictive model.

# Chapter 6: Creating and extracting feature vectors from target vehicles

## 6.1 Introduction

In the preceding chapter, we determined the ideal model for this thesis, emphasising the delicate equilibrium between accuracy and inference speed. Our selected model and specifications are tailored to our target vehicles and geared towards the real-time detection and classification of objects from live video streams. Through a careful blend of high-resolution 2D video data and a high frame rate, we achieved the accuracy and detail necessary while keeping computational demands in check by employing a moderately sized neural network.

This chapter focuses on the end-to-end process, encompassing the various stages of target vehicle detection, classification, tracking, and the generation and storage of feature vectors. Our research uses historical vehicle behaviour to construct a learning model capable of anticipating a driver's intentions at a T-junction, ideally with as much lead time as possible, while ensuring real-time processing. This chapter investigates Research Question 3 (RQ3): Is obtaining accurate pixel-level features from dynamic vehicles that closely match ground truth data feasible?

Chapter 2 discusses various techniques for detecting and classifying vehicles within two-dimensional video streams. This chapter introduces 'DUKE', our combined approach for detecting, classifying, tracking, and extracting feature vectors from video data containing vehicles. DUKE builds upon the foundational work of YOLOv5, detailed in Chapter 5, for detection and classification tasks. For vehicle tracking, we implement an enhanced version of the DeepSORT algorithm (Wojke, Bewley, and Paulus, 2017) to establish vehicle trajectories. Additionally, we integrate our proprietary techniques for extracting feature vector data from the tracked vehicles.

DUKE is fine-tuned to recognise specific target vehicles. To achieve this, we utilise thresholds that align with the size characteristics of the target vehicles, and we configure anchor box sizes, intersection over union (IoU), and non-maximum suppression (NMS) thresholds accordingly. These threshold values are thoughtfully chosen to optimise the detection performance. The anchor box sizes are computed in pixel units and are independent of the input image's dimensions. Our transfer learning training approach aimed to adjust the ratio between input dimensions and anchor sizes, which are defined relative to a grid size—a fixed number of pixels that can be customised to enhance the detection process.

The final step in object detection, NMS, is pivotal in selecting the most suitable bounding box for an object based on a single value. Hence, choosing the NMS threshold is critical to determining the model's overall performance.

## 6.2 Chapter organisation

This chapter introduces DUKE for object detection, tracking, and feature vector extraction. Section 6.2 covers bounding box predictions, IoU, anchor boxes and ground truth associations, localisation errors and refinement, confidence scores, NMS, full bounding box prediction, and vehicle tracking. Section 6.3 explores feature vector extraction, addressing feature vector creation using constant, variable, and calculated values; recording features as vectors; and initial analysis of feature capture. We then compare DUKE-derived data with ground truth values to assess accuracy and consistency. The chapter concludes with Section 6.5 to summarise the key points and findings.

The contribution discussed in this chapter is our innovative approach to generating accurate dynamic vehicle feature vectors for utilisation in real-time prediction.

## 6.3 DUKE

We developed a robust feature vector creation and capture model, DUKE, by refining YOLOv5 models, fine-tuning parameters such as video frame rate, resolution, anchor boxes, and IoU metrics, and incorporating a tailored image dataset utilising transfer learning. An overview of the DUKE algorithm can be found in Figure 26.

The first step in the process is the acquisition of raw video data, denoted as Vr, which is manually optimised to ensure superior quality. Subsequently, the optimised video, Vo, undergoes analysis utilising the DUKE algorithm for object detection, classification, and tracking tasks. Vo is subjected to further computational processing to extract pixel-level feature vectors, facilitating detailed analysis and interpretation within an academic context.

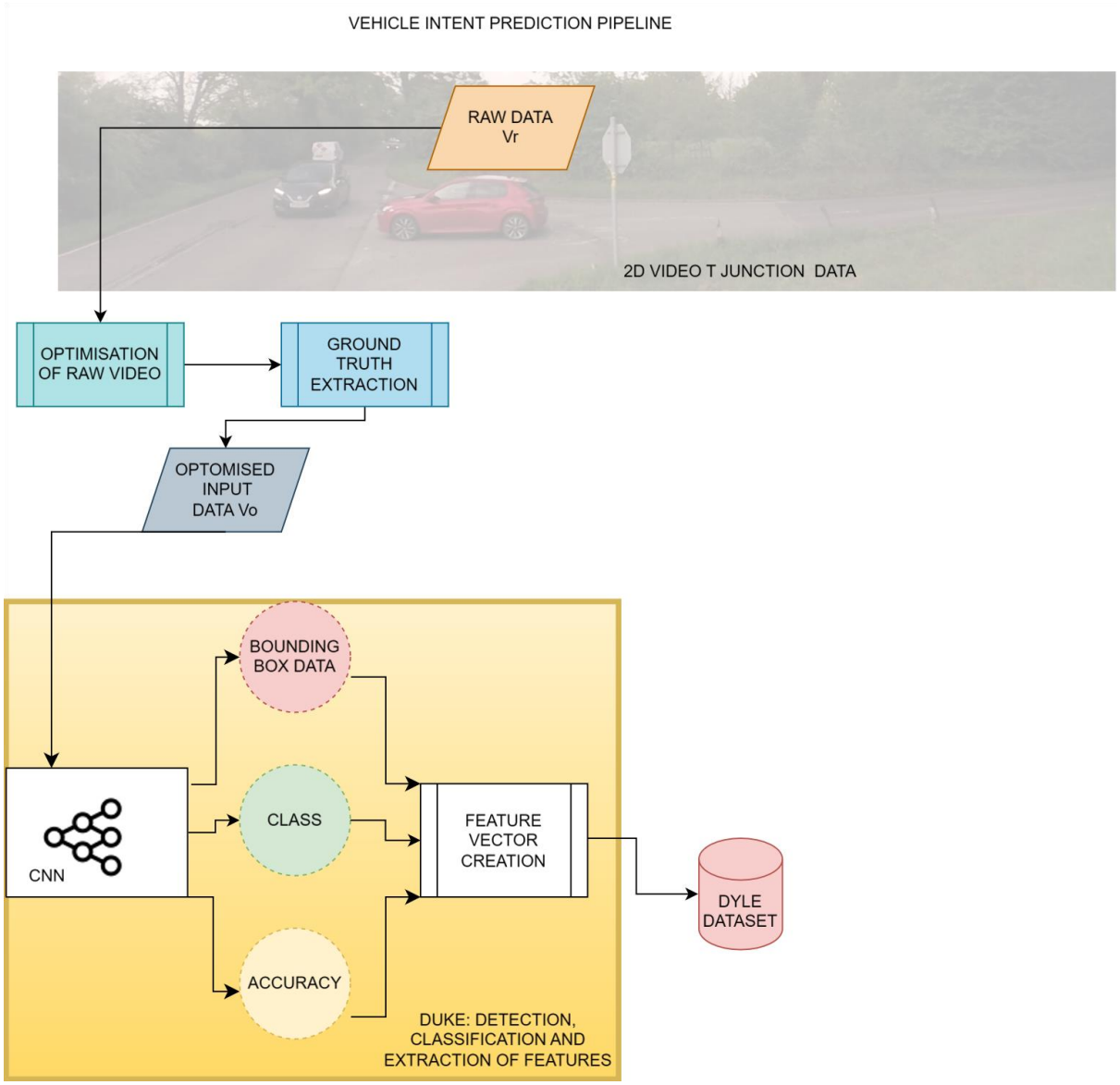



*Figure 26 The DUKE pipeline flow diagram depicts the key stages within our prediction framework leading up to the creation and storage of feature vectors. Initially, $V_r$ is obtained using a stationary camera at specific test locations, a process explained in Chapter 3. This process is optimised, as discussed in Chapters 4 and 5. Subsequently, $V_o$ is transmitted to DUKE, where feature vectors are extracted from identified target vehicles in DUKE. These feature vectors are stored in an online dataset, as detailed in this chapter.*

### 6.3.1 Bounding box predictions

The first step in extracting feature vectors from target vehicles is to locate the vehicle in the video frame and classify it. We accomplished this by using a bounding box prediction.

The predictions generated by YOLOv5 consist of several components. These predictions provide the (x, y) coordinates of the bounding box's centre, width, and height, all relative to

the image dimensions. Two other predictions are made; the first prediction assigns class scores to each predicted bounding box, representing the confidence that the object belongs to a specific pre-defined object class and a numerical objectness score is also assigned to each box, indicating the model's confidence in detecting a valid object within the box rather than background noise. YOLOv5 can predict multiple bounding boxes for each grid cell in the input image. Following prediction, a confidence threshold is applied to filter out less confident predictions, and non-maximum suppression is used to eliminate redundant or overlapping boxes. The final output includes bounding boxes with associated class labels and confidence scores, as seen in Figure 27.

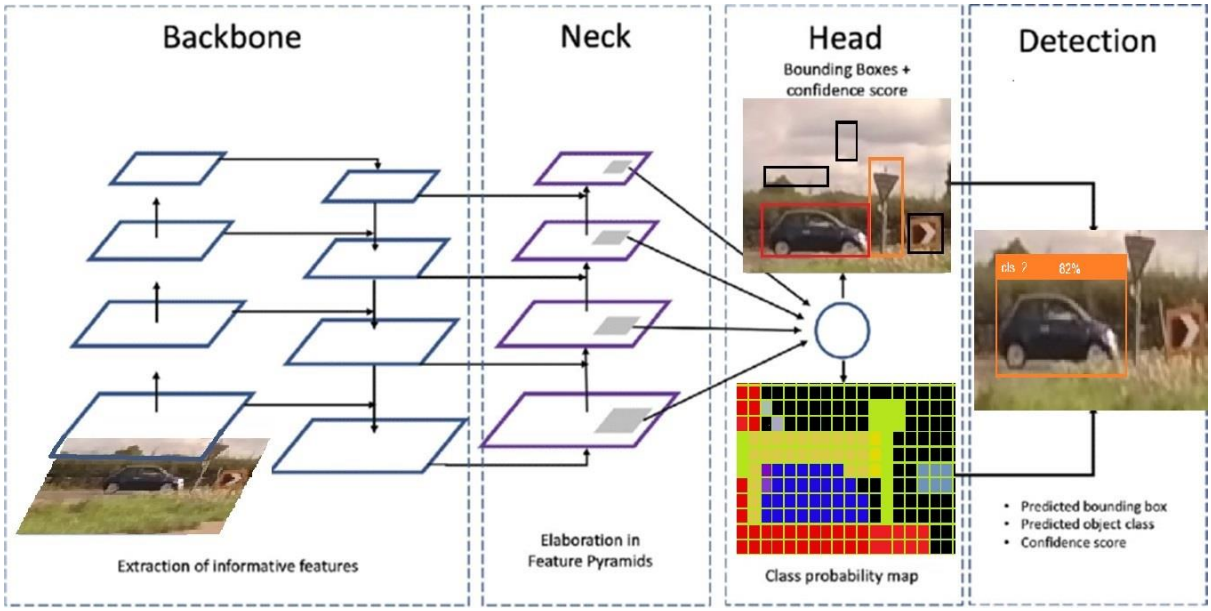



*Figure 27 DUKE object detection and classification summary of a single iteration adapted from Figure 14 (Section 4.3) to depict ground truth detection as an image*

YOLOv5m is the section of DUKE's system responsible for generating bounding box values and providing contextual information for objects in each video frame. When video frames are processed through DUKE, object predictions are generated based on a set of pre-defined anchor boxes, as depicted in the Head section of Figure 22. These anchor boxes were established using our training data and are tailored to vehicle sizes. In YOLOv5, creating and refining anchor boxes involves initially setting their sizes and aspect ratios based on dataset characteristics, training the model to predict bounding boxes using these anchors, analyzing model performance to identify deficiencies in anchor placement, refining anchor parameters iteratively, and evaluating the updated model until satisfactory results are achieved. This process ensures that the anchor boxes effectively capture object variations within the dataset,

leading to improved object detection accuracy.

Large anchor boxes are designed to detect larger vehicles, such as buses and trucks, or closer vehicles, while small anchor boxes are employed for smaller or more distant vehicles. As illustrated in Figure 28, these predictions result in a final bounding box characterised by its centre (bx, by), height (bh), and width (bw).

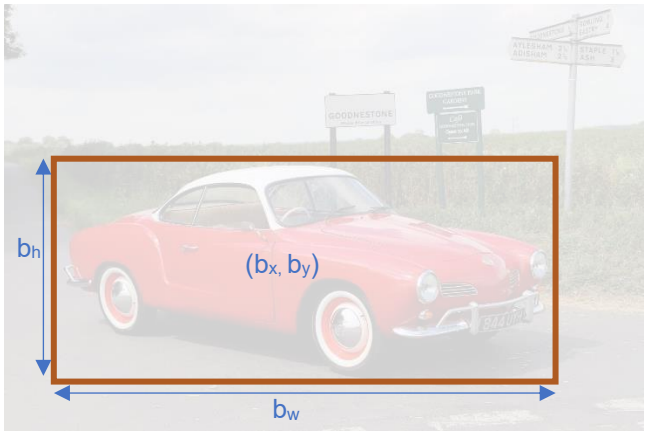



*Figure 28 Final bounding box prediction with dimensions of h, w, and centre*

### 6.3.2 Intersection Over Union (IoU)

To ensure the accuracy of the predicted bounding boxes, we employ a validation process using ground truth data. This validation is carried out using the IoU function, which quantifies the accuracy of the predicted bounding box. $B_P$ aligns with the actual bounding box $B_a$, as demonstrated in Figure 29. The actual bounding box, $B_a$, is crafted during the pre-model training phase. Each instance of an object within a specific class is manually labelled by drawing a bounding box around the object's perimeter. This forms the foundation of object classes and serves as the ground truth for our system.

IoU is a metric used to evaluate the overlap between two bounding boxes in an image and is calculated using the following formula:

$$IoU(B_{P,}B_a) = \frac{B_p \cap B_a}{B_p \cup B_a} \qquad (14)$$

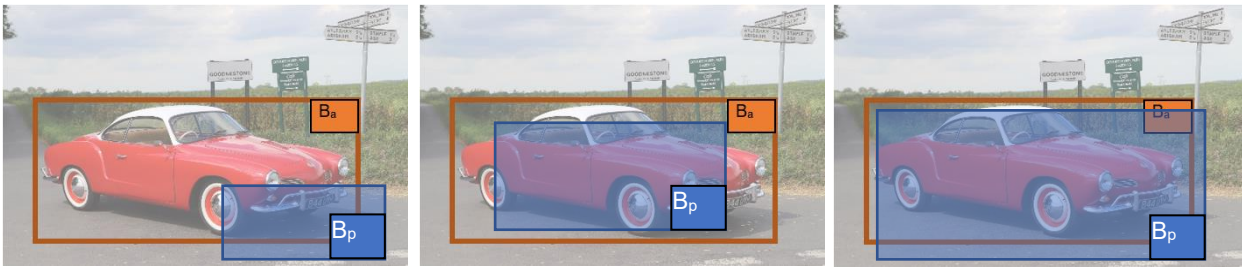
IoU = (Area of Intersection) / (Area of Union)

- **Area of Intersection**: The region where the two bounding boxes overlap. In the case of two rectangles, the area is common to both.
- **Area of Union**: This is the total region both bounding boxes cover. It includes the areas of both bounding boxes but subtracts the area of their intersection to avoid double-

counting.

The IoU value ranges from 0 to 1:

- IoU = 0: There is no overlap between the two bounding boxes; they are entirely separate.
- 0 < IoU < 1: There is some degree of overlap or intersection between the two bounding boxes.
- IoU = 1: The two bounding boxes are identical or completely overlap.



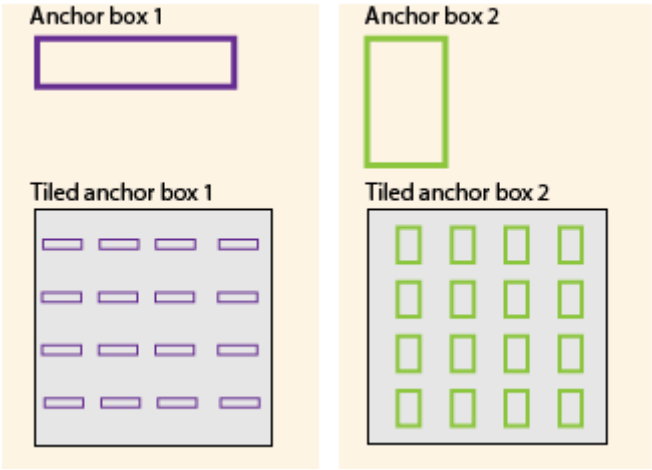

$IoU(B_p,B_a)=0.1$ $IoU(B_p,B_a)=0.5$ $IoU(B_p,B_a)=0.9$

*In Figure 29, the Intersection over Union (IoU) ranges from low to high, and an IoU greater than or equal to 0.5 is considered the detection threshold. Anything below this threshold is not considered.*

### 6.3.3 Anchor boxes and ground truth associations

In YOLOv5, creating and refining anchor boxes involves initially setting their sizes and aspect ratios based on dataset characteristics, training the model to predict bounding boxes using these anchors, analyzing model performance to identify deficiencies in anchor placement, refining anchor parameters iteratively, and evaluating the updated model until satisfactory results are achieved. This process ensures that the anchor boxes effectively capture object variations within the dataset, leading to improved object detection accuracy.

Anchor boxes comprise preconfigured bounding boxes of specified height and width, as illustrated in Figure 30. These boxes are tailored to match specific object classes' scale and aspect ratio attributes, and their precise definitions are established through our extensive training procedures.




*In Figure 30, the pre-defined anchor boxes are tiled across the image during detection.*

The network does not directly predict bounding boxes but instead predicts the required probabilities and refinements corresponding to the tiled anchor boxes. DUKE returns a unique set of predictions for every anchor box defined. The final feature map represents object detections for each class. Using anchor boxes enables DUKE to detect multiple objects, objects of different scales, and overlapping objects. The position of an anchor box is determined by mapping the location of the network output back to the input image. The process is replicated for every network output. The result produces a set of tiled anchor boxes across the entire image. Each anchor box represents a specific prediction of a class. For example, in Figure 31, there are two anchor boxes to make two predictions per location.

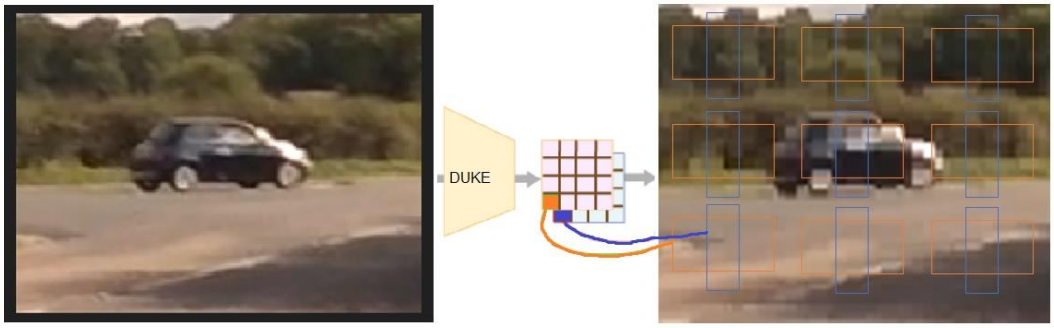



*Figure 31 DUKE output from the network (i,j) maps to the image (i,j) to determine if there is an object in that grid. Each anchor box is tiled across the image. The number of network outputs equals the number of tiled anchor boxes. The network produces predictions for all outputs.*

### 6.3.4 Localisation errors and refinement

In YOLOv5, the distance or stride between adjacent anchor boxes is determined by the level of downsampling performed within the model's CNN component. Downsampling is achieved through strided convolutions and plays a crucial role in expanding the network's receptive field and capturing features across various scales, as discussed by Akhtar and Ragavendran (2019). Different downsampling factors are employed within the specific YOLOv5m context, resulting in anchor boxes that are distributed with a coarser grid. The choice of these downsampling factors directly influences the spacing between these anchor boxes. For example, if downsampling factors range from 4 to 16, the spatial resolution of the feature maps is reduced by 4 to 16 compared to the input image, as visualised in Figure 32.

The coarsely tiled anchor boxes refer to anchor boxes that cover larger areas of the image, and as a result they may lead to localisation errors relating to determining the position and size of objects within the image. When anchor boxes are spaced too far apart due to excessive downsampling, it can be challenging for the model to accurately localise and predict the positions of objects, especially small or closely packed objects.

To mitigate these localisation errors, we carefully choose the downsampling factors and anchor box sizes appropriate for our dataset and object sizes. Additionally, we introduce anchor boxes of various aspect ratios and scales to improve object localisation, allowing the model to adapt to different object sizes and shapes.

Downsampling is followed by convolutional layers that capture more abstract and higher-level features based on the lower-resolution but more informative feature maps obtained after downsampling (Hesse, Schaub-Meyer and Roth, 2023).

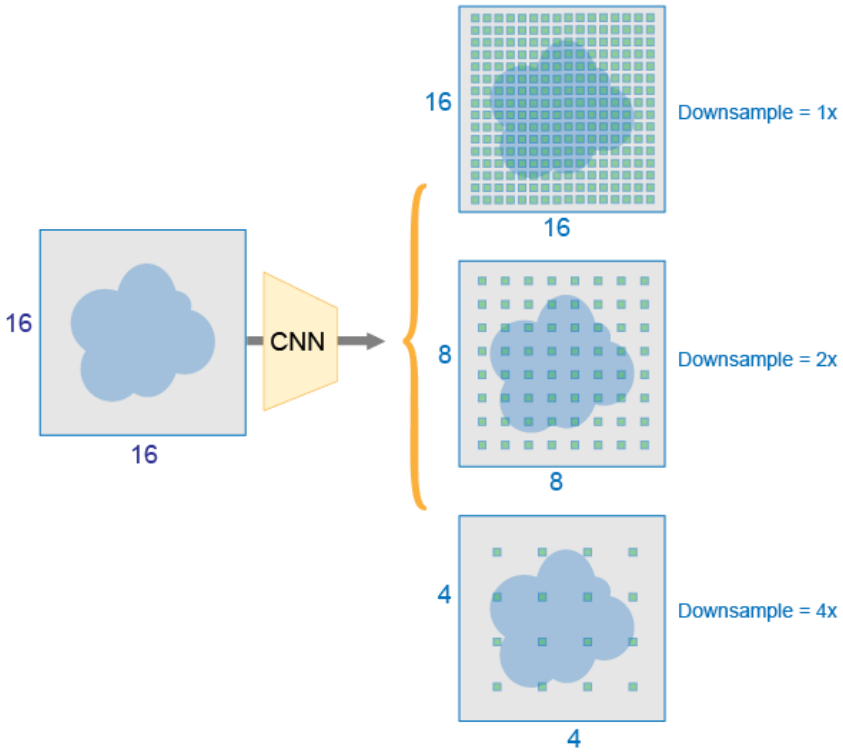



*Figure 32 DUKE learns offset values to apply to each tiled anchor box to fix localisation errors, refining the anchor box position and size.*

Through exhaustive experimentation on our dataset with YOLOv5m, we discovered that optimising inference performance is achieved by minimising the stride property of the max-pooling layers. Notably, we observed that reducing downsampling did not compromise the confidence of our predictions, which can be attributed to our focus on detecting larger objects.

### 6.3.5 Confidence

The process of selecting pre-defined anchors is designed to closely match the ground truth boxes' characteristics and is determined through a K-means clustering approach (Zhong et al., 2020). This method follows a specific sequence.

In the first step, all ground truth bounding boxes are repositioned to have their centres at the coordinate (0,0), assuming the objects are located at the origin. The clustering algorithm commences by initialising five centroids via a random selection from the ground-truth bounding boxes. The clustering algorithm consists of two alternating steps: each ground truth box is assigned to one of the centroids based on the IoU as a distance measure. This step results in the creation of five distinct clusters or groups of ground-truth bounding boxes. New centroids are computed for each cluster by selecting the bounding box that minimises the mean IoU with all the other boxes in the same cluster. Following these steps, the final values are passed

through a logistic activation function (sigmoid) for all the output parameters except for the relative width and height. This processing ensures that the output values are constrained from 0 to 1.

In YOLOv5, the confidence score, often called 'confidence', is a crucial output parameter associated with each detected bounding box. It represents the model's confidence or certainty that a given bounding box contains an object of interest (i.e., an object that the YOLO model is trained to recognise). The confidence score is a probability value ranging from 0 to 1, where a higher score indicates a higher level of confidence that the bounding box accurately encloses an object.

### 6.3.6 Non-max suppression

Non-maximum suppression (NMS) is the concluding step in the vehicle detection process within DUKE, playing a fundamental role in selecting the most suitable bounding box for an object. We have explored several methods to fine-tune NMS for vehicle detection within this context, focusing on greedy NMS and soft NMS.

NMS is the technique that eliminates redundant, overlapping bounding boxes, retaining only the most representative ones. It operates as one of the final layers within the network, as depicted in Figure 33.

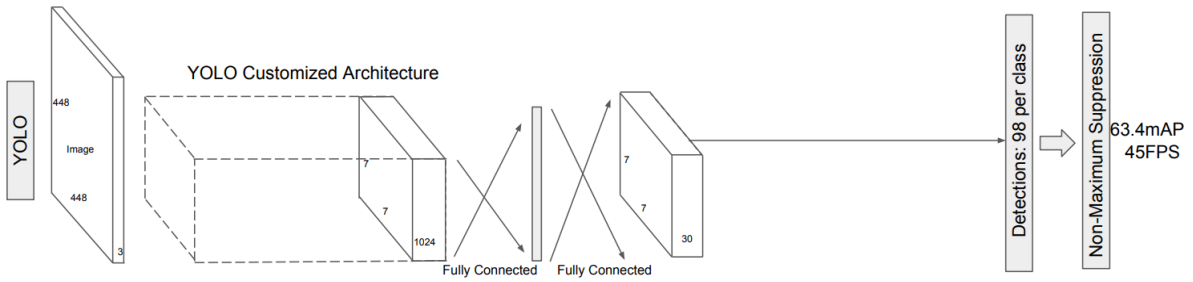



*Figure 33 NMS is an integral component of the CNN, acting as the final layer.*

Greedy NMS and soft NMS are utilised in object detection to reduce the multitude of bounding boxes generated by detection algorithms. However, they differ in their approach to handling overlapping boxes and how they influence the confidence scores of these boxes. Greedy NMS promptly removes boxes that fail to meet a defined IoU threshold compared to the box with the highest confidence score. Boxes that overlap with the top-scoring box beyond the IoU threshold are entirely disregarded, and their associated confidence scores are discarded. Greedy NMS makes a binary decision: boxes are retained or discarded based on their IoU with the highest confidence box. Unfortunately, this binary approach can lead to a loss of

information regarding the removed boxes' relative importance or confidence levels. Our empirical observations revealed that this binary decision is insufficient for detecting all vehicles in a crowded T-junction. The greedy NMS equation is represented as follows:

$$S_i = \begin{cases} S_i, & IoU\big((M, b_i)\big) < N_t \\ 0, & IoU\big((M, b_i)\big) \geq N_t \end{cases}, \quad (15)$$

$S_i$: score of probability i

$b_i$: box corresponding to probability i

$M$: box corresponding to maximum confidence

$N_t$: IOU threshold

Soft NMS (F. Frank Chen et al., 2023) presents a straightforward adjustment to the greedy NMS algorithm and is designed to tackle the challenge of overlapping bounding boxes in object detection, rather than immediately discarding boxes that do not meet an IoU threshold, soft NMS advocates for a more nuanced approach. It proposes reducing the confidence scores of these overlapping boxes, considering the extent of their overlap as measured by the IoU; this means that boxes with substantial overlap (high IoU) receive a more substantial reduction in their confidence scores, while boxes with lower IoU values—indicating less overlap—are penalised to a lesser degree. Soft NMS thus introduces a 'softening' mechanism to mitigate the issues associated with harshly eliminating potentially relevant detections in crowded scenes, as shown in the following equation:

$$S_i = \begin{cases} S_i, & IoU\big((M, b_i)\big) < N_t \\ S_i\big(1 - IoU(M, b_i)\big), & IoU\big((M, b_i)\big) \geq N_t \end{cases}, \quad (16)$$

We determined that the critical distinction between greedy NMS and soft NMS is their treatment of overlapping boxes and the associated confidence scores. Greedy NMS eliminates boxes that do not meet a threshold, leading to a binary decision and a loss of information. In contrast, soft NMS softens the confidence scores of overlapping boxes based on their IoU, allowing for a more continuous representation of box quality while retaining all boxes. According to the research by F. Frank Chen, soft NMS is frequently used to alleviate the severe suppression of bounding boxes in densely populated scenarios, like our T-Junction video data. Our findings align with this, indicating that employing soft NMS reduces data loss when tracking multiple vehicles.

### 6.3.7 Full bounding box prediction

A bounding box prediction is a frame-by-frame process completed in one pass.
DUKE follows these steps to detect and classify a target vehicle, generate bounding box parameters within an image, and produce the output $y$:

1. Grid-Based Image Division: The input frame is partitioned into a grid of dimensions G×G.
2. Grid Cell Object Detection: For each grid cell, a backbone CNN is employed to predict the values of $y$ with the following structure:

Detecting and classifying a target vehicle as an output $y$.

$$y = \left[p_c, b_{x,} b_y, b_{h,}\, b_w, c_1, c_{2,\ldots}, c_{p,\ldots}\right]^T \in \mathbb{R}^{GxGxkx(5+p)} \qquad (17)$$

Where $p_c$ is the probability of detecting an object,˚-
$b_{x,} b_y, b_{h,}$and $b_w$ are the properties of the detected bounding box,
$c_1, c_{2,\ldots}, c_{p,\ldots}$ is a time stamp representation of which $p$ classes were detected,
and $k$ is the number of anchor boxes.
$\in RGxGxkx(5+p)$: This denotes the space to which the vector $y$ belongs.
$\mathbb{R}$: Represents the real number space.
$GxG$: Typically, this would indicate the grid size over which the image is divided. In the YOLO framework, an image is divided into a $G$x$G$ grid.
$k$: The number of anchor boxes that each grid cell can predict.
(5+$p$): This signifies the length of the prediction vector for each bounding box. The '5' accounts for the objectness score and the four bounding box coordinates. The '$p$' accounts for the class probabilities.

The vector $y$ is a prediction from the model for a specific grid cell and anchor box. It includes the objectness score, bounding box coordinates, and class probabilities. The vector belongs to a space that is determined by the grid size, the number of anchor boxes per grid cell, and the number of classes plus five additional parameters (objectness score and bounding box coordinates).

If $p_c$ = 0, DUKE does not detect any object and the corresponding predictions are disregarded.

3. NMS (Eq 16) is implemented to eliminate potential duplicate and overlapping bounding boxes. This procedure includes the following sub-steps:

The DUKE code header in Figure 34 contains hard-coded parameter values, such as Confidence_Threshold and IoU_Threshold. These values can be adjusted at the beginning of each run.

```
qv0="videos\hazmeona32_crop_x2.mp4" #fixed as of 6-4-23
conf=0.5 #fixed as of 6-4-23
nms=0.7 #fixed as of 6-4-23
resolution=1920 #fixed as of 6-4-23
model='yolov5m.pt' #fixed as of 6-4-23
```

Figure 34 shows the header of the DUKE code, where qv0 represents the input video, Vo stands for the confidence threshold, nms corresponds to the IoU threshold, and the resolution pertains to the input video. Additionally, this header specifies the model currently in use for the DUKE.

In this process, the bounding boxes undergo a series of steps for effective object detection and classification. First, they are organised in descending order based on their confidence scores. Boxes with confidence scores below the designated Confidence_Threshold are then eliminated from consideration. Next, an iterative assessment begins with the box with the highest confidence score. During this assessment, the IoU is computed for the current box with respect to every other remaining box of the same class. If the IoU between the current box and another exceeds the specified IoU threshold, the box with the lower confidence score is removed. This procedure helps refine and filter the bounding boxes, retaining those representing objects of interest with high confidence and minimising false positives in object detection. The process is repeated until all boxes in the frame have been evaluated, as shown in Figure 35.

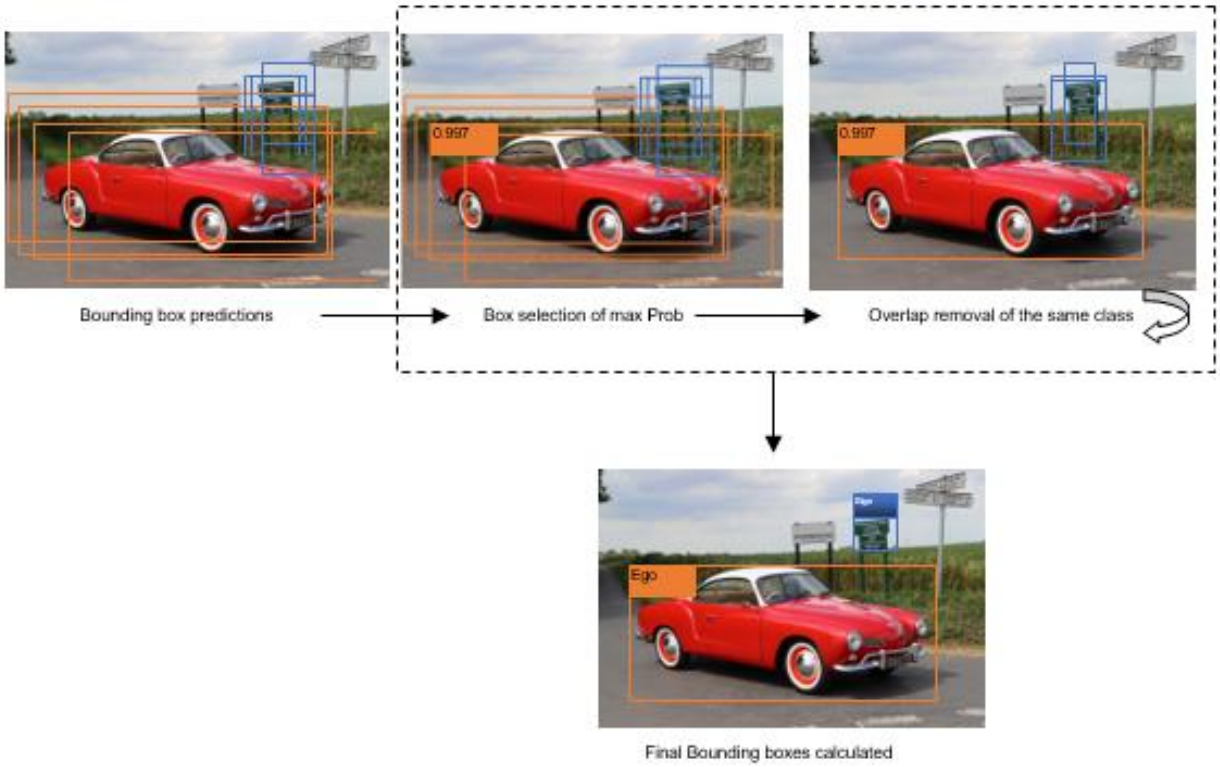



*Figure 35 NMS iterates through the bounding box predictions until the prediction with the highest probability remains as the classified target vehicle (Ego)*

DUKE aims to effectively identify and categorise target vehicles within video frames while preventing false positives and duplicate detections through NMS. The configuration of the Confidence_Threshold and IoU_Threshold parameters holds significant importance in achieving the desired balance between recall and precision during the object detection process.

After obtaining output $y$ for each target vehicle in the frame in the input video, the next step is to extract the bounding box parameters from these results. These parameters are then utilised for tracking purposes and for generating feature vectors. This process involves using the bounding box information to track target vehicles across frames, ensuring continuity in object tracking. Simultaneously, the extracted parameters are employed to create feature vectors.

### 6.3.8 Vehicle tracking

Tracking is a crucial step in our process after the detection and classification stages outlined in Section 6.2. After these initial stages, we pass the bounding box prediction data to the tracking model. This model assigns a unique identifier to each target vehicle detected in the frame. These identifiers are integral to our feature vectors, enabling us to distinguish behaviour between vehicles and vehicle types.

DUKE utilised Simple Online and Real-time Tracking (DeepSORT) by Wojke, Bewley, and Paulus (2017) to facilitate the tracking process. DeepSORT is an object-tracking algorithm that seamlessly integrates with the object detection component, effectively tracking associated objects across consecutive video frames.

#### *6.3.8.1 Overview of DeepSORT*

Once objects are detected using YOLOv5, Deep SORT extracts deep feature embeddings for each detected object. These embeddings represent the object's appearance and are crucial for associating the same object across different frames despite changes in its position or appearance due to factors like occlusion or viewpoint changes. YOLOv5 provides bounding box coordinates that can extract the corresponding regions from the frame, which are then passed through a deep feature extraction network to obtain embeddings. The obtained feature embeddings are used to associate the detected objects across different frames. Deep SORT utilises techniques such as the Kalman filter (Kalman, 1960) and Hungarian algorithm (Hamuda et al., 2018) to predict object trajectories and match detections to existing tracks based on their similarity in appearance features and predicted motion. As new detections are made in subsequent frames, Deep SORT updates existing object tracks by associating them with the most likely detection based on their predicted trajectories and appearance similarities. Tracks are maintained for each object over time, allowing for the continuous tracking of objects as they move throughout the scene. Deep SORT also includes mechanisms for track management, such as handling occlusions, track termination, and track re-identification. It employs strategies to handle situations where objects temporarily disappear from view or multiple objects merge or split in the scene.

It is a popular choice for real-time applications such as surveillance, autonomous vehicles, and sports analysis, as Ciaparrone et al. (2020) demonstrated.

DUKE incorporates Wojke et al.'s method for tracking objects, which has been adopted for multiple object-tracking benchmarks. During the detection and classification stages of DUKE, bounding box parameters are created; DUKE passes to the DeepSORT model, which defines an eight-dimensional state space $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ that contains the bounding box centre position (u, v), aspect ratio γ, height h, and their respective velocities in image coordinates. A standard Kalman filter is used with constant velocity motion and a linear observation model, where we take the bounding coordinates (u, v, γ, h) as direct observations of the object state. A tracking ID is applied to the target vehicle, and DUKE generates continuous feature vector data for each vehicle in a frame.

Once objects are detected and passed to Deep SORT, features are extracted to help distinguish between them. These features are based on the object's appearance and include

information like the object's colour, shape, texture, and more.

Deep SORT uses a tracking algorithm to associate objects between frames. The primary challenge in tracking is maintaining the identity of objects as they move through the video frames. Deep SORT combines a Kalman filter and the Hungarian algorithm. The Kalman filter estimates each object's state, position, and velocity, while the Hungarian algorithm assigns detected objects to existing tracks, ensuring that each object maintains a consistent id across frames. The Kalman filter performs a prediction step to estimate the likely state of an object in the next frame based on its previous motion. When a new detection occurs in the next frame, the Kalman filter then corrects its prediction based on the observed data, helping to reduce errors in tracking. Deep SORT also includes a thresholding mechanism to determine when a new detection should be associated with an existing track or if it should be treated as a new track. This helps in handling temporary occlusions and false detections. However, we found that this threshold requires fine-tuning to prevent the tracking ID of a vehicle that just left the frame from being applied as a new detection, assuming it is the same vehicle.

The tracker maintains a list of active tracks and their corresponding IDs. It can create and delete tracks as objects appear or disappear from the video feed, including post-processing steps to refine the tracking results, such as eliminating short-lived tracks or smoothing object trajectories.

## 6.4 Feature vector extraction

The selection and design of feature vectors represent a scientific work step in this research due to several key aspects. Selecting and designing feature vectors begins with formulating hypotheses about which vehicle attributes are most relevant for predicting behaviour within a T-Junction scenario. These hypotheses are grounded in existing literature, domain knowledge, and empirical observations. The choice of feature vectors involves a systematic experimental design process. We carefully consider which vehicle attributes to include in the feature vector, how to quantify and represent these attributes numerically, and how to ensure that the selected features adequately capture the dynamic nature of vehicle motion. Once the feature vectors are defined, we collect data to populate these vectors. This data collection process involves gathering information about various vehicle attributes, such as distance from the merge line, velocity, acceleration, spatial coordinates, and relative size. We then analyse the collected data to determine how well the chosen feature vectors capture the desired aspects of vehicle behaviour. This analysis involves statistical techniques, visualisation methods, and comparison against ground truth data. The feature vectors serve as input to the predictive model for vehicular behaviour. The selection and design of feature vectors involved an iterative refinement process. We experimented with different combinations of features, adjusted feature

representation methods, or incorporated feedback from initial model evaluations to improve predictive performance. Overall, the selection and design of feature vectors represent a critical scientific work step in this research, serving as the foundation for developing and evaluating a predictive model for vehicular behaviour within a T-Junction scenario.

In this thesis context, employing feature vectors is imperative for encapsulating a target vehicle's stochastic motion patterns. These feature vectors are foundational in constructing a robust dataset, which is essential for training a predictive model. This model is designed to forecast vehicle behaviour with high accuracy. The necessity for such predictive capabilities arises from the objective to understand and anticipate the dynamics of vehicular motion, particularly within the confines of a T-Junction.

Feature vectors are multidimensional entities with numerical representations of an object's attributes. Our feature vectors incorporate various dynamic aspects of a vehicle's motion, including but not limited to its distance from the junction's merge line, velocity, acceleration, positional coordinates within a given frame (denoted by the x and y coordinates of the vehicle's bounding box), and the vehicle's relative size in pixels. Such attributes vary dynamically, meaning they change over time as the vehicle moves and interacts with its environment.

The dynamic nature of these feature vectors is critical for modelling and predicting vehicular behaviour in real-time scenarios. By integrating velocity, acceleration, and spatial positioning measurements, our predictive model can infer patterns in how vehicles approach and navigate through the T-Junction. Including data about other vehicles, especially those traversing the major road from a specific direction (referred to as direction b), adds another layer of complexity. This acknowledges the interactive and interdependent nature of vehicular motion, where the movements and presence of others influence the behaviour of one vehicle.

We phrase this as dynamic feature vectors as the dynamic component of the feature vectors lies in their capacity to represent the temporal and spatial fluctuations of vehicular movements. These vectors capture a snapshot of various parameters at a given time, providing a detailed and quantitative overview of the vehicle's state. As such, they are instrumental in training our predictive model to recognise and predict patterns in vehicular behaviour, which is contingent upon the vehicle's actions and the surrounding traffic conditions. In Figure 36, we illustrate the initial testing of the detection and classification components, demonstrating the passing of data to DeepSORT for ID acquisition.


*Figure 36 Junction JM559 (Section 3.3). The target vehicle was detected with the following feature vector data: classified as a car (2), confidence 93% (0.93) with bounding box drawn and tracking ID assigned (6).*

We can visualise our feature vector data on the bounding box for analysis and testing, allowing us to assess the accuracy of our data against ground truth for each frame and across multiple frames. In Figure 37, we introduce bounding box annotations for attributes such as acceleration, distance from the junction merge line, and the bounding box's area around the target vehicle.

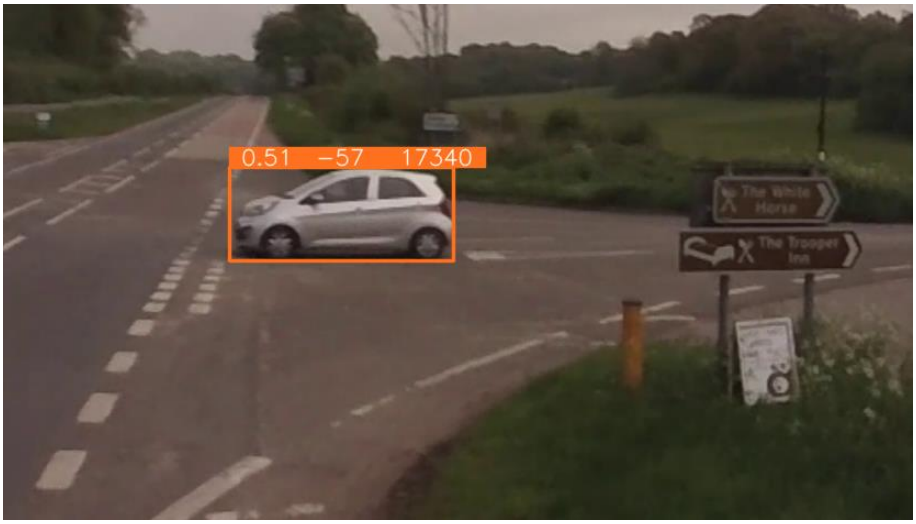



*Figure 37 Junction JM559. Target vehicle showing feature vectors for acceleration (0.51 px/ms$^2$) distance from the junction (57 px) from the bottom left corner of the box and the area of the bounding box (17,340 px$^2$).*

We enhanced inference speed by reducing false positives by fine-tuning DUKE to detect and

process vehicles approaching from the front and left sides (as discussed in Sub-section 4.5.1). This allows us to concentrate our detections on the target vehicle and vehicles potentially endangered by reckless drivers entering the main road. Figure 33 shows a target vehicle that has stopped (acceleration = 0) approximately 110 pixels from the junction merge line. There's also a second detection from direction b (as detailed in Sub-section 3.4.1). This is another tracked detection, and the bounding box data includes the distance from the junction's starting point relative to the vehicle (50 px) and the vehicle's class (2). These features, Approach_b distance and Approach_b class, are essential data that can significantly impact a driver's behaviour. Our goal is to capture and classify this behaviour for training purposes, which is fully discussed in Chapter 7.


*Figure 38 Junction JM559. Target vehicle showing feature vectors for acceleration (0.0 px/ms$^2$) distance from the junction (110 px) from the bottom left corner of the box and the bounding box area (14,742 px$^2$). Approach_b distance is (50 px) to the entrance to the junction, and Approach_b class is (2).*

### 6.4.1 Feature vector creation

During each iteration of DUKE, features are generated as each video frame is analysed, as explained in Sub-section 6.2.7. Equation 17 (section 6.3.6 ) illustrates the output $y$ and associated metrics, including $b_x, b_y, b_h,$ and $b_w$. When a target vehicle is detected and classified, the bounding box data is passed to DeepSORT to be assigned a tracking ID. This tracking ID serves as our first feature. The second feature is the vehicle class, categorising the vehicle as a car, truck, bus, motorcycle, or bicycle. This classification aids in predicting behaviour and speed based on known vehicle characteristics. Next, we have features such as $Px$ and $Py$ coordinates of the centre of the drawn bounding box and the bounding box's height and width prediction in pixels. With these initial features, we can derive calculated attributes

like velocity, acceleration, area of the bounding box, distance from the junction, and time. List of features collected as a single feature vector;

- Junction ID
- Tracking ID
- Class (type*)
- Distance
- Velocity
- Accelerati
- Area (dim box)
- Approach
- Approach b class *(type*)*
- Intent prediction *($F_{pc}/S_{Fpc}$)***

*type=class of vehicle

**See 7.4.2 Subclasses for definition of (Fpc/SFpc)

In each iteration optimised for performance, a set of features is generated directly from the detection and classification function, from tracking, or through in-iteration calculations. All these features are stored in a single list, creating a feature vector per iteration for each vehicle, primarily keyed to the tracking ID. In continuous tracking of a target vehicle, we can generate a set of features for every frame. However, fps is altered during processing, which can result in reduced fps in heavily congested traffic situations and thus a reduced number of feature vectors generated.

### 6.4.2 Feature vector constant, variable, and calculated values

Creating an instant historical record of each frame is essential to computing features from real-time data. This record allows us to compare variables and derive features such as velocity (v), acceleration (a), and distance (d).

Calculated variables are determined for each frame.

**Calculating Velocities:**

We use the positional information extracted from the bounding boxes in successive frames to determine the velocities of the tracked objects by analysing the change in position relative to the elapsed time between two consecutive frames. This measurement indicates the target vehicle's dynamic motion as it approaches the merge line over time and is expressed in units of pixels per millisecond.

$$v = \left(\frac{\Delta Px}{\Delta t}, \frac{\Delta Py}{\Delta t}\right) \quad (18)$$

Where $Px$ and $Py$ are the coordinates of the estimated bounding box, and $\Delta t$ is the inference time difference between the two frames.

We can further deconstruct the velocity vector into its $Px$ and $Py$ components by isolating and assessing changes in position. This approach allows us to accommodate various traffic scenarios, including traffic approaching from different directions, enabling us to capture the entire junction scene comprehensively.

**Acceleration:** The rate of change in the vehicle's velocity toward the junction over time.

$$a = \frac{v-U}{t} \quad (19)$$

Where $U$ is the initial velocity taken at first inference.

**Distance from the merge line:** The distance between the vehicle's prior position and the stop line, aiding in velocity and direction estimation, can be calculated using the bounding box bottom left corner coordinates $(Px, Py)$ and $(Px_1, Py_1)$ with the Euclidean distance formula:

$$d = \sqrt{((Px1 - Px)^2 + (Py1 - Py)^2)} \quad (20)$$

**Approach_b Distance:** This metric indicates whether an approaching vehicle has been activated from direction b; if so, equation (20) is applied to the central coordinates of the bounding box, and a distance measurement is calculated from the junction's starting point and the distance of the approaching vehicle.

**Area:** The vehicle's dimensions inside the estimated bounding, including its width and height, are used to distinguish between different vehicle types. These dimensions are directly calculated from the bounding box variables' bh' (height) and 'bw' (width).

Variables are assigned for each frame, and these values include:

**ID:** A tracking ID assigned by the DeepSORT network at the initial detection of the object.

**Class:** The object's class label, determined at the end of the detection phase based on training data.

**Px_x and Px_y** refer to the coordinates denoting the bounding box's estimated position. These coordinates can originate from any point within or around the bounding box.
Constant values are established for each junction. As described in Section 5.4, we demonstrated the process of extracting ground truth values from video frames; this is done to guarantee the precision and accuracy of our feature vectors.

**Target vehicle velocity:** We manually analyse each junction using the method described in Section 5.4, enabling us to compare pixel-generated data with the driver's actual behaviour.

**Distance to the merge line or video frame boundary:** This measurement helps us assess the precision of the vehicle's position, direction, velocity, and trajectory as it passes through DUKE. Values for the coordinates of junction dimensions are stored for each junction and then applied to DUKE before inputting the video.

**Vehicle dimensions:** This parameter represents the physical size of the vehicle, including its length, width, and height.
**Target vehicle acceleration:** This value is derived from analysing velocity across various scene segments.6.4.3 Recording features as vectors.
DUKE operates by processing frames collectively, capturing and calculating features for detected vehicles, and then interpreting the subsequent frame. In Figure 39, a single frame is presented, illustrating the identification of a specific target vehicle. In order, the bounding box annotations include id, class, acceleration, distance, and area. Table 19, entry #8, reveals how this frame is represented and stored as a feature vector.


*Figure 39 Junction JM559. Target vehicle showing features of id (9), class (2), acceleration (0.23 px/ms$^2$), distance to merge line (809 px), area (8,742 px$^2$), aligned to # 8 in Table 19.*

Table 16 displays the initial 11 feature vectors corresponding to the subject vehicle depicted in Figure 39. The frame displayed is relative to (#8) in Table 16 and is highlighted. Notably, the values for Approach_b distance and Approach_b class are marked as 0, signifying no identification of a vehicle approaching from the direction of Approach_b, as verified by the

visual evidence in Figure 39, where no vehicular presence is evident heading towards the intersection.

| # | Id | Cls | Distance | Velocity | Acceleration | Area | $P_x$_x | $P_y$_x | Approach_b distance | Approach_b class |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 2 | -999 | 2.21065 | 0.147236 | 7320 | 1551 | 197 | 0 | 0 |
| 2 | 9 | 2 | -975 | 2.40144 | 0.18597 | 7560 | 1531 | 197 | 0 | 0 |
| 3 | 9 | 2 | -949 | 1.19255 | 0.338967 | 7869 | 1508 | 197 | 0 | 0 |
| 4 | 9 | 2 | -923 | 1.30481 | 0.440977 | 7800 | 1483 | 198 | 0 | 0 |
| 5 | 9 | 2 | -892 | 0.914385 | 0.41054 | 8220 | 1459 | 198 | 0 | 0 |
| 6 | 9 | 2 | -865 | 0.875952 | 0.229694 | 8280 | 1433 | 199 | 0 | 0 |
| 7 | 9 | 2 | -837 | 0.869303 | 0.232951 | 8601 | 1408 | 199 | 0 | 0 |
| 8 | 9 | 2 | -809 | 0.850033 | 0.228101 | 8742 | 1380 | 198 | 0 | 0 |
| 9 | 9 | 2 | -781 | 0.880542 | 0.228871 | 8742 | 1352 | 198 | 0 | 0 |
| 10 | 9 | 2 | -750 | 0.772426 | 0.213753 | 9408 | 1327 | 197 | 0 | 0 |
| 11 | 9 | 2 | -721 | 0.726269 | 0.206397 | 9536 | 1300 | 197 | 0 | 0 |

*Table 16 Single vehicle feature vector list captured during each frame iteration (#)*

## 6.5 Initial analysis of feature capture

This section provides a snapshot of the data from Figure 39 and Table 16. Figure 40 shows a plot depicting a vehicle's dynamic motion across 11 frames using the raw data from Table 16. It is important to note that this data has not undergone any smoothing or cleaning processes before analysis. This approach allows for a detailed, granular examination of data points, typically around 25 milliseconds apart. While one potential smoothing method could involve increasing the time interval between data points, such an adjustment would not fully address all the issues identified in this initial data snapshot.

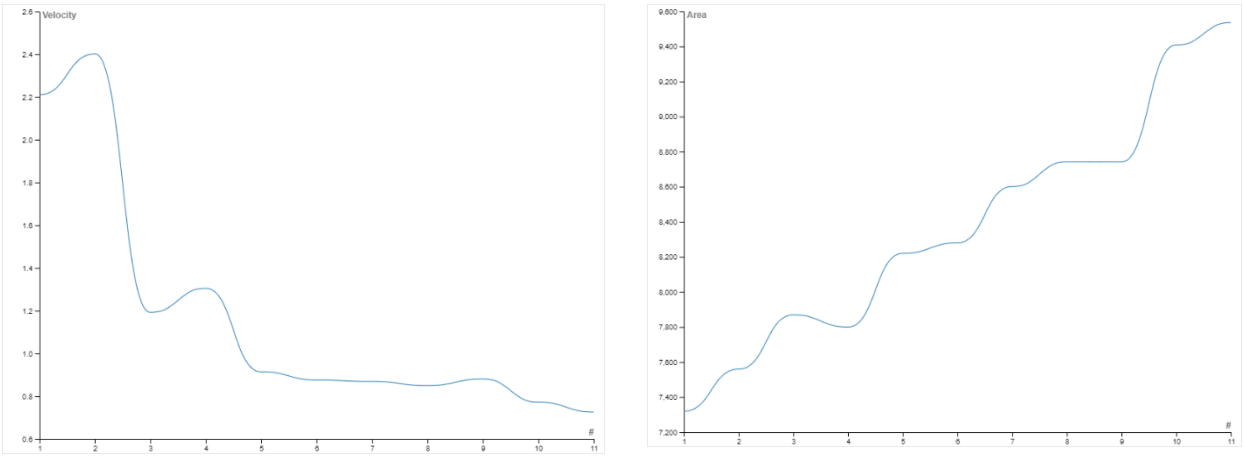



*Figure 40 The features plotted over 11 frames, derived from the data in Table 16, depict the*

*dynamic changes in velocity and area as the vehicle approaches the merge line.*

### 6.5.1 Discussion of Figure 35

**Left: Velocity**

There is a falsely indicated increase in velocity between frames 1 and 2, which can be attributed to the initial detection of and differences in historical data. It is common for inaccuracies to arise in the initial frame due to the lack of historical data in the initial iterations.

**Right: Area**

The area plot shows perturbations, magnified due to irregular height and width bounding box estimation. The area increases as the object approaches the junction due to the camera's location and the vehicle's relative size.

### 6.5.2 Comparison of DUKE-derived data ground truth values

The target vehicle in Figure 39 underwent a manual assessment from its initial detection to its progression towards the merge line, employing the techniques outlined in Section 5.4. This assessment involved the evaluation of various parameters, including acceleration, distance, area, and velocity, which were then compared with the 2D-pixel features extracted by DUKE. This manual evaluation is an initial validation step to gauge the accuracy and dependability of the features acquired through the DUKE pipeline. The results of a comprehensive tracking of the target vehicle, featuring DUKE-extracted features alongside ground truth data, are presented in Table 17.

| Data | Mean Velocity px/ms | Vel (p-value) | Vel (SD) vel | Mean Acc $px/ms^2$ | Acc (p-value) | Acc (SD) vel | Distance px | Mean Area $px^2$ | Acc (p-value) | Acc (SD) vel |
|---|---|---|---|---|---|---|---|---|---|---|
| DUKE | 0.93 | 0.18 | 0.19 | 0.28 | 0.09 | 0.09 | 893 | 8370 | 683 | 716 |
| Ground Truth | 1.06 | 0.17 | 0.2 | 0.33 | 0.09 | 0.09 | 963 | 8259 | 679 | 705 |

*Table 17 compares manually gathered ground truth data with the 2D-pixel features obtained through DUKE.*

Examining Figure 40, which reveals perturbations during the initial detection phase, it is evident that the DUKE-derived data presented in Table 17 gradually becomes more consistent and aligned with the ground truth data as the target vehicle produces more feature data, as a result, has no significant statistical difference between them.

The following steps involve working with more data and refining the IoU thresholds and confidence threshold values to minimise perturbations between feature captures. Additionally, we plan to enhance data smoothing by increasing the time step and disregarding data from

the initial frames, such as the first two frames when considering velocity and the first five frames when evaluating acceleration. This is discussed further in Chapter 7.

## 6.6 Chapter conclusion

This chapter introduces a method for feature extraction from two-dimensional video data, utilising YOLOv5 inference mechanisms. We created a method that enables the extraction of detailed, multidimensional data from vehicles in motion, treated as objects within the dataset. This method leverages YOLOv5's inference capabilities to extract low-level dynamic attributes of vehicles from high-level image frames, organising these attributes into training features within feature vectors. This approach aims to improve the precision of extracted features and enhance the dataset for training machine learning models to predict vehicle behaviour in traffic scenarios better. Through this, we contribute to the ongoing development of automated feature extraction and the training of machine learning models.

We began this chapter with Research Question 3 **(RQ3):** Is obtaining accurate pixel-level features from dynamic vehicles that closely match ground truth data feasible?
This chapter demonstrates our advanced approach to deriving meaningful features from two-dimensional 2D video data. These features closely align with the ground truth data, which serves as a baseline for our research. However, it is imperative to acknowledge that our work is not complete. While our results are promising, our feature data requires further refinement to enhance its quality and reliability. This refinement is essential as we prepare our data for training and testing our prediction model, a topic we address in the upcoming chapter. We aim to ensure that our prediction model is built upon a solid foundation; by refining these features, we hope to improve our predictions' accuracy. Contribution revisited: In this chapter, we explain our approach to generating accurate dynamic vehicle feature vectors for utilisation in real-time prediction.

# Chapter 7: Intent prediction training dataset DYLE

## 7.1 Introduction

This chapter presents DYLE, a novel real-time data handling and learning approach. DYLE stands out because it can dynamically store and update training data in feature vectors, ensuring that the training and prediction models can access the most current data. This system is uniquely designed to be updated in real-time with new data, facilitating continuous learning and adaptation. The process of dynamic enrichment, whereby DYLE accumulates new feature vectors with each iteration using DUKE, as detailed in Section 6.4, underscores the innovative nature of this approach in enhancing machine learning workflows.

As DUKE detects a target vehicle, unique feature vectors specific to the dynamic characteristics of the target vehicle are produced. These vectors are systematically collected and stored within DYLE, as illustrated in Figure 41.

DYLE seamlessly merges manually classified information with machine-learning-based predictions from our prediction model. This amalgamation results in an online training dataset that facilitates the immediate integration of new insights, typically achieved within milliseconds following the classification process.

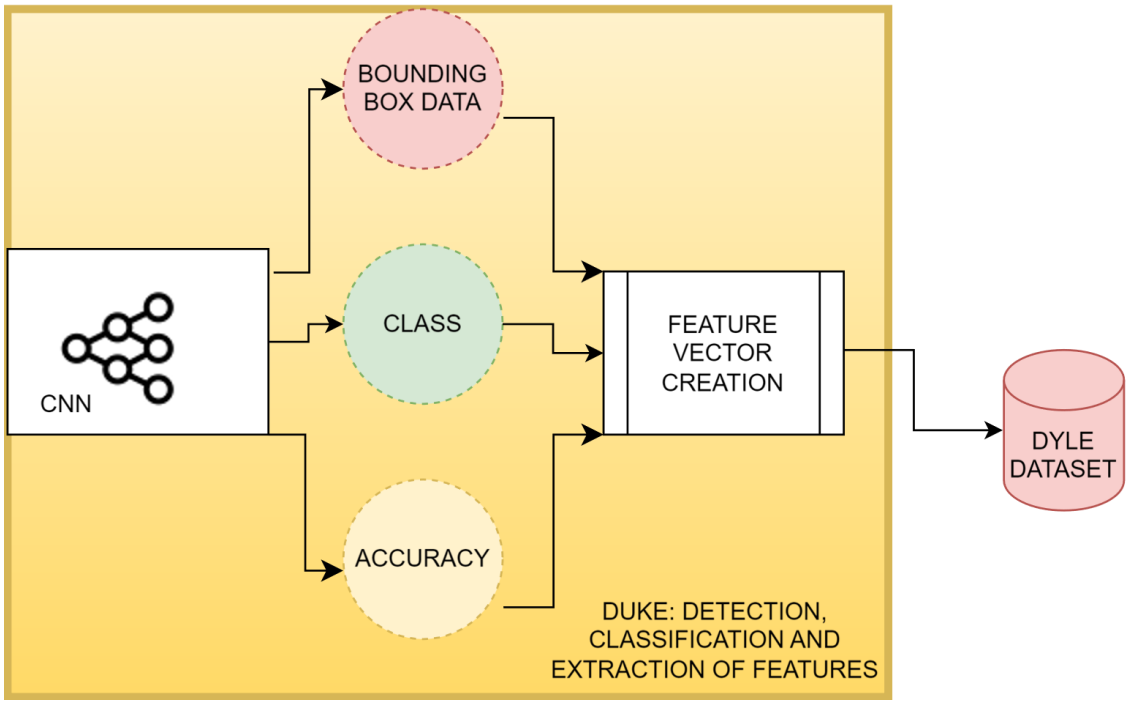



*Figure 41 illustrates the storage process of feature vectors after undergoing DUKE analysis.*

In this chapter, we explore the following question.

**Research Question 4:** Can our feature vectors' inherent generality be observed per the consistent camera positioning hypothesis? This hypothesis posits that recordings from various junctions maintain a similar perspective due to the standardised factors of camera height, position concerning the merge line, and overall camera placement.

**Contribution:** We develop a method for organising and classifying discrete vehicle feature vectors as feature vector arrays both independently and as integral components of a comprehensive general dataset.

## 7.2 Chapter organisation

We start by focusing on extracting training features from Bo's video in Section 7.3. Section 7.4 examines feature training data for single junctions, covering topics such as prediction class, subclasses, and analysis of DYLE video training for single junctions. Section 7.4.4 explores the empirical and quantitative determination of the reliability of feature vectors, while Section 7.4.5 discusses the manual classification of training data. Moving forward, Section 7.5 addresses the creation of datasets for other junctions. Section 7.6 consolidates these datasets through aggregation, with Subsections 7.6.1 presenting the results from k-fold cross-validation on aggregated DYLE and 7.6.2 discussing the obtained results. Finally, Chapter 7 concludes with Section 7.7, which summarises the key findings and insights derived from the detailed exploration of training features, data creation, and dataset aggregation.

## 7.3 Extracting training features from Bo video

Storing the feature vectors extracted through the methods outlined in Chapter 6 imposes an additional computational burden on our model. During the training phase, this is manageable since we are in the data collection stage and not involved in prediction. However, when our model needs to store, update, and subsequently infer from these feature vectors, the increased computational load has implications for real-time predictions. To mitigate this impact, we have constrained the size of each feature within a feature vector to a maximum of 56 bits during development, further refining this to 24 bits in the final model. This reduction in feature size helps optimise computational efficiency without significantly compromising the model's predictive capabilities.

DYLE organises feature vector data in distinct arrays within its structure, each corresponding to a specific junction associated with the data (see Table 18). The procedure involves creating a Bo Vo video dataset for a designated junction and dividing the data into separate subsets for training and testing. Specifically, 60% of the video data is earmarked for the training subset,

while 20% is set aside for testing purposes. The remaining 20% of the video data is reserved for validation, which is crucial to evaluating the model's performance at the individual junction level and in the context of a combined junction dataset. This dataset partitioning strategy facilitates distinct junction training, testing, and validation. Subsequently, the data can be consolidated into a comprehensive junction training dataset, enabling generalised testing and assessment.

| EuroRAP route | Junction location | Total minutes | Manual pred class train data 60% | DAISY Derived pred class 20% | Online Verification 20% |
|---|---|---|---|---|---|
| JM377 | Oxshot Road | 263 | 157.8 | 52.6 | 52.6 |
| JM384 | A248 | 124 | 74.4 | 24.8 | 24.8 |
| JM559 | Petersfield Road | 249 | 149.4 | 49.8 | 49.8 |
| JM454 | Rowhook Road | 207 | 124.2 | 41.4 | 41.4 |
| UO196 | Jacobs Well Road | 119 | | | 119 |

*Table 18 Bo video data $V_o$, split in terms of minutes.*

An identical process is replicated for each junction, resulting in distinct DYLE training datasets comprising feature vectors derived from 60% of the respective junction's video data. Each dataset undergoes a classification phase, where a prediction class and a final action class are manually assigned based on the vehicle's actions at the merge line. Following this classification, each DYLE dataset is integrated and verified with a ground truth observation verification process, with 20% of the video data reserved for validation.

Following individual junction analysis and classification, we aggregate all the unique DYLE datasets to explore the potential for generalisation. The entire junction-based DYLE dataset is then analysed like the individual junction datasets. The 20% video hold-back verification test is repeated on this combined dataset by aggregating the verification video splits.

The primary objective of this analysis is to investigate whether the dataset exhibits inherent generality, driven by consistent camera positioning, per the hypothesis that recordings from differenT-junctions maintain a similar perspective due to the standardisation in camera height, position from the merge line, and overall placement.

## 7.4 Feature training data single junction

The first junction we processed for training data was JM559. We produced a DYLE version unique to JM559, encompassing approximately 18,480 feature vectors. Table 19 presents raw features' initial detection and classification data from a single target vehicle stored in a feature vector array in JM559 DYLE. The final prediction class depends on the vehicle's actions at the merge line, Stop, Hazard or Merge and is manually applied to the training data. We

autonomously apply the final prediction class using our online model as discussed in Chapter 8. This prediction class constitutes one facet of feature association, where all the features within a single feature vector are correlated with a particular outcome and all feature vectors are correlated as a feature vector array, as described in Table 23. Notably, the final prediction class ($F_{pc}$)* feature is intentionally omitted in this phase because this dataset serves as training data and the final prediction classification is done manually.

| id | cls | dis | v | acc | area | px_x | px_y | ap_b_dis | ap_b_cls | $F_{pc}$* |
|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 2 | -949 | 1.05878 | 2.30087 | 7869 | 1508 | 197 | 0 | 0 | |

* *$F_{pc}$ is explained in section 7.4.2*

*Table 19 provides an example of a single feature vector derived from a target vehicle. It is important to note that the final prediction class in the last column has not yet been assigned to this training data.*

A secondary, temporary association relates to the identification (id) of the target vehicle, which serves as the unique identifier and is not part of the training data. The third associational feature pertains to the vehicle's class (cls), and a fourth association includes the approach class (app_b_cls), when applicable, along with the approach distance (app_b_dis). In conjunction with distance, velocity, acceleration, area, px_x, and px_y, we assemble a feature vector at a specific pixel distance from the merge line, corresponding to a specific vehicle class and its associated final prediction class. Table 20 presents an extraction of this online 2D array of feature vectors for a target vehicle starting at 273 pixels from the merge line.

| Correlated array associated features = id, cls and app_b_cls | | | | | | | |
|---|---|---|---|---|---|---|---|
| Distance | Vel | Acc | Area | Px_x | Px_y | Ap_b_dis | Fpc |
| [[-273 | 0.379198 | 0.803161 | 15792 | 891 | 188 | 0] | |
| [-252 | 0.495743 | 0.723585 | 16185 | 877 | 189 | 0] | |
| [-233 | 1.10056 | 0.783731 | 16019 | 856 | 190 | 0] | |
| [-211 | 1.43084 | 0.71151 | 16632 | 839 | 189 | 0] | |
| [-192 | 1.9395 | 0.716536 | 16632 | 820 | 190 | 0] | |
| [-174 | 2.11859 | 0.361123 | 16716 | 803 | 190 | 0] | |
| [-158 | 2.08388 | 0.307875 | 16830 | 786 | 190 | 0]] | |

*Table 20 illustrates pre-classified feature vectors in their online storage state before being written into a file for subsequent classification and, ultimately, for training purposes. However, all the features within this dataset are intricately linked with the vehicle's unique identifier (id), its assigned class (cls), and (if applicable) the distance from which an approaching vehicle, originating from direction b, is associated with the respective feature vector.*

### 7.4.1 Prediction Class

Section 3.3 discussed the rationale behind our selection of junctions for this thesis. A stark reality drove this selection: vehicles emerging from these junctions introduce potential risks to traffic flow on the major road. With this consideration in mind, we devised a prediction classification feature based on three prevalent behaviours observed at the merge line of these T-junctions:

- **Stop,** where the vehicle comes to a complete halt and waits at the junction.

- **Merge,** in which the vehicle reduces its speed to approach the junction and smoothly integrates with the traffic flow.

- **Hazard,** where the vehicle demonstrates a minimal change in trajectory and crosses the merge line, regardless of the presence of other traffic.

These behaviours serve as fundamental building blocks for our predictive classification feature.
In Chapter 2, we delved into examining driver behaviour at T-junctions, drawing from the existing literature to understand the rationale behind the decisions made when entering a major road from a minor road. Our empirical observations confirmed some of the literature's findings, particularly regarding hesitancy, and revealed various behaviours.

Specifically, we observed the following:

- A range of Stop behaviours, including fast approaches followed by a complete stop; slow and steady approaches leading to a stop; and a combination of deceleration, acceleration, and then coming to a halt.

- A diverse range of Merge behaviours, akin to the Stop behaviours, with erratic changes in momentum as drivers made real-time decisions on their actions at the junction.

While these Stop-and-Merge behaviours were dominant in our observations, we noted that instances of Hazard behaviour were comparatively infrequent. However, it is important to note that such Hazard behaviours were more pronounced in busier junctions, such as JM377 and JM454.
We initially applied one of the three final classifications and accumulated loosely independent data at brief intervals, leading to erratic results. For example, as mentioned, no single

behaviour pattern is consistently linked to any of our three predictive classes. If we were to retrospectively apply a predictive class to the initial detection inception feature vectors based on the final batch of feature vectors at the merging point, we would inadvertently eliminate the initial behaviour for another class. This presents a significant challenge because target vehicles may exhibit similar initial behaviours but behave contrarily when approaching the merging point. By implementing associative prediction classes in the form of subclasses, we can individually and interdependently assign prediction labels to each feature vector. This process broadens the spectrum of prediction data and elevates the accuracy of our training dataset by conferring relevance to each feature vector at a given distance from the merge line.

### 7.4.2 Subclasses

The final step in creating our training data involves assigning a prediction classification to the target vehicle based on our empirical observations of the vehicle's action at the merge line. This classification is determined using the Final Prediction Classification ($F_{pc}$).

- If the target vehicle comes to a complete stop, $F_{pc}$ = Stop
- If the target vehicle slows down and safely merges with the traffic, $F_{pc}$ = Merge
- If the target vehicle neither stops nor slows down and instead crosses the merge line with no significant change in momentum, $F_{pc}$ = Hazard.

To enhance the precision of our dataset, we introduced subcategories of predictive classifications. These subcategories were established by associating feature vectors with a definitive predictive classification determined by their proximity to a junction. We devised these subcategories due to conclusions from empirical observations aT-junctions. Our analysis revealed that assigning a single classification to target vehicle feature vectors when they cross the merging point would lead to the arbitrary categorisation of associated feature vectors in a binary manner.

Subclasses of predictive classifications:

$W_{Fpc}$, Weak Association with $F_{pc}$: This classification is applied to feature vectors greater than 70% of the total distance away from the merge line.

$M_{Fpc}$, Moderate Association with $F_{pc}$: This classification is applied to feature vectors greater than 40% and less than 69% of the total distance away from the merge line.

$S_{Fpc,}$ Strong Association with $F_{pc}$: This classification is applied to feature vectors greater than 10% and less than 39% of the total distance away from the merge line.

$F_{pc}$, the final prediction classification, is applied to feature vectors within 9% of the total

distance from the merge line. This classification categorises the features corresponding to the moments before the vehicle takes action at the junction. The distances from the merge line were calculated to establish a hierarchical relationship with the ground truth predictions, with the weakest association being the furthest from the merge line and, consequently, the furthest from the actual ground truth behaviour.

Our sub-classifications help us categorise and differentiate the behaviour of target vehicles at various distances from the junction, enhancing the meaning of our training data. Figures 42–44 inclusively are simplified illustrations of the sub-classification process as the target vehicle approaches the merge line of a T-Junction.

*Figure 42 If the distance of the target vehicle is ≥ 70% of the total distance from the merge line, a feature vector is classed as having a weak association with the final classification prediction $W_{Fpc}$, where $F_{pc}$ is the final classification.*

*Figure 43 If the distance of the target vehicle is >= 40% and <=69% of the total distance from the merge line. The feature vector is classed as having a moderate association with final classification prediction, $M_{Fpc.}$*

*Figure 44 If the distance of the target vehicle is >= 10% and <=39% of the total distance from the merge line. The feature vector is classed as strongly associated with the final classification*

*prediction, $S_{Fpc}$.*

Our use of DUKE is crucial for capturing and storing accurate feature arrays in a file. This file enables the manual classification of the $F_{pc}$ and the association of subclasses with a target vehicle.

Feature vectors are generated when a target vehicle is first detected and classified. This generation process extends from the initial detection point to the merge line and, in specific scenarios, may extend beyond this—especially in cases involving fast Merges or Hazard classes. Due to the relatively short distances covered in terms of pixel measurements and time, capturing a maximum number of features at incremental distances is essential. However, this approach can introduce noise into the model given that bounding box predictions vary at 30 milliseconds; this leads to significant differences in feature values. Smoothing techniques have been applied to the data to mitigate fluctuations and enhance the overall stability of the feature vectors.

### 7.4.3 DYLE video training analysis for single junction

To analyse DUKE's performance in accurate feature generation, we recorded the results in a feature array file and a video file. Visualising bounding box data allows for the manual classification of the target vehicle through empirical observations based on this ID (id) as the target vehicle approaches the merge line. Following this manual classification, we can methodically categorise all feature vectors associated with the target vehicle with a sub-classification.

Empirical observations are taken from the video recording of DUKE processing $V_o$ data. Figures 45–47 inclusively represent frames extracted from the recorded video, illustrating the bounding box (id) and class. While the data classification method is onerous, it is invaluable for identifying misclassifications and tracking errors that might go unnoticed.

As an example, Figure 40 features an evident misclassification. The vehicle approaching from direction b is erroneously classified as a truck (class 7) instead of as a car (class 2). This thorough classification process, though challenging, serves as a critical mechanism for detecting and rectifying such inaccuracies, enhancing the overall accuracy and reliability of the dataset.


*Figure 45 is an example of misclassification as the approaching car from direction b is*

*classed as a truck (7) instead of a car (2).*


*Figure 41. Features, including associated traffic data, are recorded up to the merge line.*


*Figure 46. Full junction traffic data is recorded and stored as features. Approach _b_class and approach_b_distance can vary in the feature array, as seen above.*


*Figure 47. All forms of traffic, including bicycles, are tracked as they approach the merge line and are added to the training data.*

### 7.4.4 Empirically and quantitatively determining the reliability of feature vectors

As discussed in Section 6.5, we conducted experiments to enhance the accuracy of predicting the horizontal bounding box positions. Our iterative process adjusted the initial velocity and acceleration feature vector calculations. In the first four and six iterations, we refined these calculations to address the issues observed in our initial data analysis, where velocity (vel) and acceleration (acc) exhibited erroneous features because the initial detection velocity did not have a historical reference based on the smoothing method described below.

One of the key improvements was adjusting the IoU threshold to 0.75, which effectively helped to smooth out the bounding box data by applying a stricter criterion for bounding box detections. Additionally, we improved the accuracy of our feature vector data by increasing the confidence threshold to 0.8. This adjustment reduced false positive classifications of vehicles as the data became associated with a narrower classification margin. As depicted in Figure

48, a double bounding box indicates that features are being created twice for the same target vehicle, albeit at slightly different locations.


*In Figure 48, double detections are mitigated by adjusting hyperparameters, focusing on the IoU and confidence threshold.*

An additional issue identified in the JM559 junction video is a significant occlusion zone, primarily caused by the double road sign in the image's centre (see Figure 49). Despite this challenge, it was observed that the tracking performed by DeepSORT remained resilient to the occlusion, demonstrating robustness even when smaller vehicles completely vanished for durations of up to 400 milliseconds.

*Figure 49 illustrates a notable occlusion resulting from a road sign located in the central area of our tracking path at junction JM599*.

This resilience suggests that DeepSORT can maintain tracking continuity despite temporary obstructions in the visual field. Handling occlusions effectively is imperative to this work as it is difficult to account for mobile occlusions, such as large vehicles obscuring the target vehicles.

Despite the effective tracking capabilities observed in the JM559 junction video, a quantitative dataset analysis revealed erratic bounding box values for features generated between the start and end of the occlusion-producing road sign. These irregularities in bounding box values may indicate instances where the detection system encounters challenges in accurately estimating the position and size of objects. We applied a smoothing method, the exponential

moving average (EMA), to the features generated at a distance relative to the road signs' dimensions. We utilised this method as it gives more weight to recent data points and is commonly used in the literature.
The formula for calculating the EMA is as follows:

$$EMA_t = \alpha \cdot X_t + (1 - \alpha) \cdot EMA_{t-1} \qquad (21)$$

Where

- $EMA_t$ is the EMA at time $t$, and $t$-1, $EMA_{t-1}$,
- $X_t$ feature at time $t$,
- $\alpha$ is the smoothing factor, which is the constant between 0 and 1, determining the weight given to the most recent observation. The closer $\alpha$ is to 1, the more weight is given to the most recent observations.

We fine-tuned the parameter α to achieve the desired level of smoothing. A smaller α resulted in a smoother curve but reduced responsiveness to recent changes. Conversely, a larger α enhanced the EMA's responsiveness to recent changes at the cost of potentially introducing more noise.
The EMA was implemented specifically at junction JM559 as a function of the distance between occlusion distance points. The implementation involved activating the EMA function whenever a feature deviated beyond the predefined threshold of previous feature values. This function addressed mobile occlusions, which did not directly impact ID tracking but posed challenges for precise bounding box predictions. The EMA approach was then uniformly applied across all junctions as a function, triggered whenever threshold values were surpassed for any recorded feature.
Data collection involved sampling data points throughout entire vehicle feature vector arrays, focusing on distance-velocity profiles. Analysing the behaviour of vehicles as they approached the junction provided valuable insights into ground truth observations and quantitative data.
Figures 50 and 51 illustrate the velocity profile of vehicle id 32, which decelerated quickly towards the merge line and stopped. When traffic cleared, vehicle id 32 accelerated across the merge line, turning right onto the major road. The influence of a road sign occlusion is apparent in Figure 50, which results in irregular fluctuations in the velocity calculation. As seen in Figure 50, the notable surge in velocity originated 526 pixels away from the merge line, which is attributed to the road sign occlusion. This challenge was successfully addressed by applying EMA smoothing, which led to a more stable velocity profile, as demonstrated in the refined depiction presented in Figure 51.

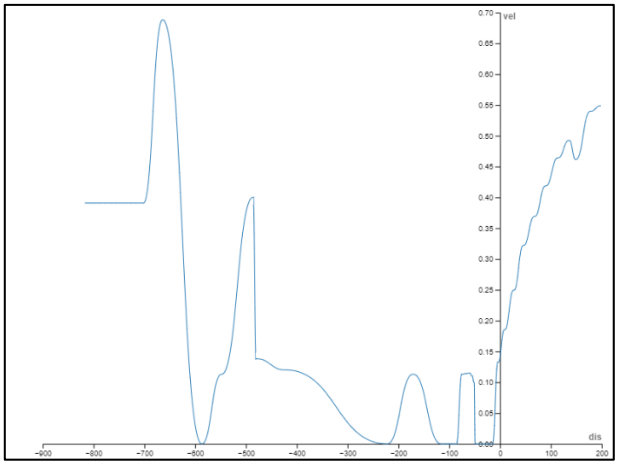



*Figure 50 Velocity profile of id 32 before EMA smoothing application.*

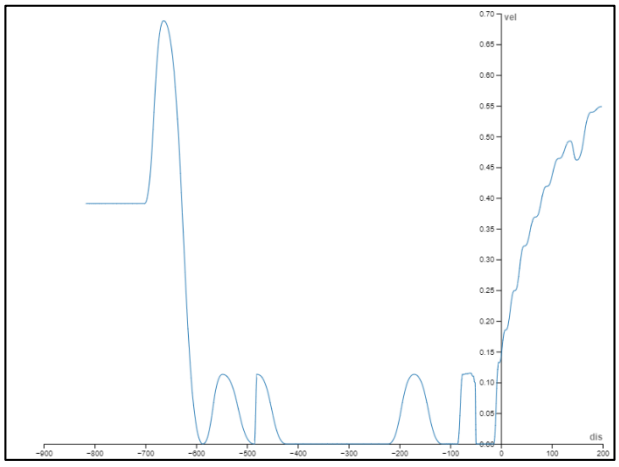



*Figure 51 Velocity profile of id 32 after EMA application and classified as ‘Stop'.*

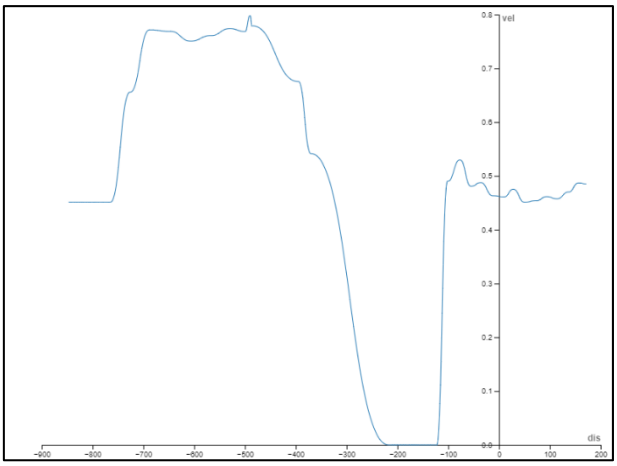



*Figure 52 EMA applied to a velocity distance profile classified as 'Merge'.*

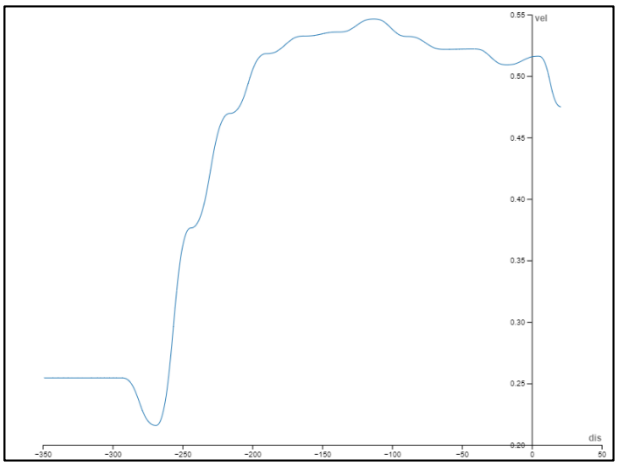



*Figure 53 EMA applied to a velocity distance profile classified as ‘Hazard’.*

An illustration of the need for an extensive dataset covering an extended timeframe is evident in the infrequent incidence of the hazard class. Figure 53 shows the velocity feature profile of a target vehicle presenting a Hazard to traffic on a major road. Initial detection of the target vehicle occurred at 350 pixels from the junction, introducing a challenge due to its late detection, reducing the available time window for accurate prediction.

### 7.4.5 Manual classification of training data

The preceding section illustrated our approaches for ensuring data accuracy and reliability. The subsequent phase involves manually classifying all data within the datasets, assigning each entry an $F_{pc}$ and a subclass based on its proximity to the merge line and action at the merge line. The creation of unclassified training data involved DUKE processing 60% of the Bo video data from Junction JM559. A feature vector array was generated and stored in a CSV file for each target vehicle approaching the junction. A sample is presented in Table 24. Each feature in this table was linked to an ID, a class, and a manually assigned final prediction classification denoted as $F_{pc}$.

In Table 21, the vehicle with ID 133, originating from Junction JM559, was tracked in the recorded video. Observations revealed that this vehicle safely merged onto the major road at the merge line. Consequently, it was manually classified with an $F_{pc}$ of Merge. Additional associated predictions $S_{merge}$, $M_{merge}$, and $W_{merge}$ were applied based on their distance from the merge line. This detailed classification process enriches the training data to increase accuracy in the final model.

| Cls | Dis | Vel | Acc | Area | Px_x | Px_y | App_b_dis | App_b_cls | $S_{Fpc}$/ $F_{pc}$ |
|---|---|---|---|---|---|---|---|---|---|
| 2 | -721 | 0.774123 | 0.219997 | 9465 | 1070 | 197 | 0 | 0 | W_merge |
| 2 | -580 | 2.22272 | 0.562319 | 11082 | 955 | 192 | 93 | 2 | M_merge |
| 2 | -341 | 2.29214 | 0.916604 | 13544 | 922 | 191 | 72 | 2 | S_merge |
| 2 | -273 | 0.379198 | 0.803161 | 15792 | 891 | 188 | 52 | 2 | Merge |

*Table 21 shows sample data from a feature vector array created for a tracked vehicle written to a CSV file and manually classified as merge for an $F_{pc}$ and retrospectively applied associated classifications.*

The complete CSV file, generated from 60% of the video data collected at junction JM599, underwent the classification process outlined earlier. This method facilitated a systematic data sampling for accuracy and reliability, requiring visual confirmation of each feature vector array as a target vehicle while scanning for anomalies in the data. The classified data derived from this task serves as the cornerstone for all predictions in the pipeline, underscoring the paramount importance of Accuracy and reliability at this stage. A closer look into the classes is provided in Figure 54, where distance velocity profiles from Section 7.4.4 illustrate the placement of subclasses within a target vehicle's entire feature vector array.

As emphasised in Section 7.4.2, an arbitrary classification relying solely on merge line actions can misclassify a substantial portion of data, leading to inaccuracies in subsequent analyses. Consequently, the thorough approach employed in the manual classification process fosters

a nuanced comprehension of the dataset. This methodology mitigates the risk of misclassifications, ultimately bolstering the overall reliability of predictions derived from the dataset.

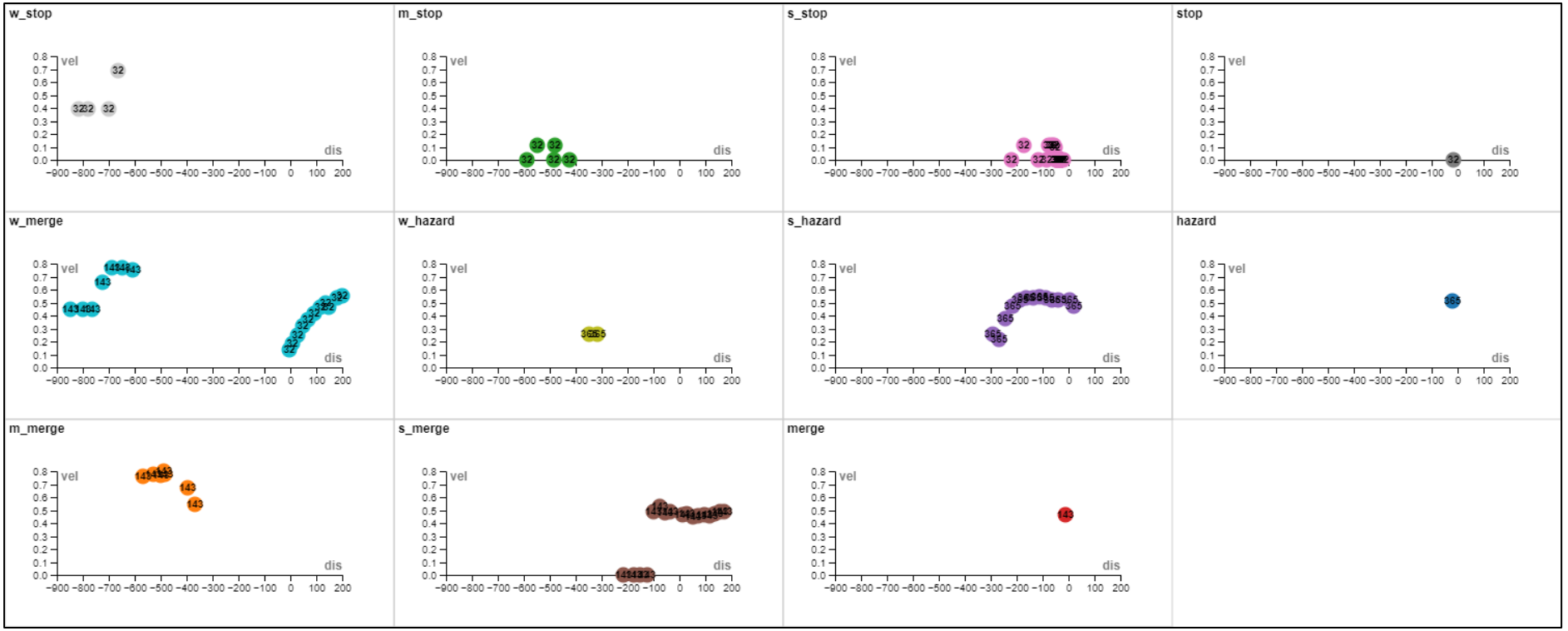



*Figure 54 Each feature is classified within a target vehicle's feature vector array to assist in the probability of predicting a Hazard at a junction at an optimal distance from the merge line.*

The initial velocity detection features were subjected to smoothing with EMA. In the specific case of Hazard classification depicted in Figure 54, we designated these features as Weak_Hazard despite falling within the Strong_Hazard association threshold. This cautious approach aims to prevent biasing the dataset. Subsequently, the sub-classifications of Strong_Hazard were assigned in recognition of the interconnection of all features in a feature vector. The distinctions between consecutive detections (i and i+/-1) are pivotal to establishing these links.

The final classification, represented by the $F_{pc}$, reveals that the target vehicle exhibits a relatively high velocity on the merge line, signalling the impracticality of the vehicle coming to a complete stop. Notably, the Merge class in Figure 54 displays a velocity at the merge line similar to that of the Hazard class. However, ground truth observations align with the data and indicate a divergence: the Merge class target vehicle decelerated to a near stop around 220 pixels from the merge line, indicating that the vehicle driver was observing the major road before accelerating to merge onto it. In contrast, the Hazard class target vehicle accelerated approximately 300 pixels from the merge line without slowing down to cross the merge line. This nuanced differentiation highlights the significance of ground truth observations in refining the accuracy of Hazard classifications.

### *7.4.6 Complete single junction dataset*

Having completed the steps to generate a single-junction classified dataset, we enter the verification phase, which unfolds in three stages. The initial stage employs k-fold accuracy to establish a foundational accuracy baseline for junction JM559. Subsequently, the second stage uses the Pandas Profiling Report (Pandas, 2018). The third stage, covered in the next chapter, utilises the reserved 20% of junction video data to derive ground truth predictions for unseen target vehicles. This process aims to ensure that the training dataset successfully encapsulates the diversity and reliability of the overall data distribution.

### *7.4.7 Accuracy using K-fold cross-validation*

We used K-fold cross-validation to measure our model's performance and generalisation ability by calculating the mean accuracy or the average of correct predictions over total predictions. This approach is widely used in the literature (Wong and Yeh, 2020) as a validation technique to assess how well machine learning performs on an independent dataset. K-fold cross-validation helps to mitigate issues related to the variability of a single train-test split. It provides a more reliable assessment of a model's ability to generalise to new, unseen data by exposing it to multiple training and testing scenarios.

In our k-fold cross-validation procedure, we implemented a naive Bayes algorithm with a probability density function (PDF) model (detailed in Chapter 8), which examines real-time

predictions on live data. The initial step involves dividing the feature vectors into k groups or folds, each of equal size. One fold is assigned as the validation set throughout each iteration while the method undergoes training on the remaining k−1 folds. This iterative process is repeated k times, ensuring each fold is used in the validation set once.

The k signifies the number of subsets the original dataset is partitioned into. The selection of k influences the balance between computational efficiency and the reliability of the performance estimate. Opting for higher values of k generally enhances the robustness of the evaluation, albeit potentially increasing computational costs. This approach allows evaluation of the model's generalisation performance across different subsets of the dataset, contributing to a more dependable estimation of its effectiveness.

The process we used for k-fold cross-validation is as follows:

1. The junction dataset is divided into k equal-sized folds or subsets.
2. The model is trained k times, using k-1 folds for training and the remaining fold for testing. This process is repeated k times, with each k fold used once for the testing data.
3. The proportion of correctly classified instances, an accuracy metric, is calculated for each iteration based on the model's predictions on the testing fold.
4. The k accuracies are then averaged to provide a more robust estimate of the model's performance across different subsets of the data.

Results from k-fold cross-validation on DYLE JM599 are shown in Table 22.

| K value | Mean accuracy JM599 |
|---|---|
| 5 | 0.69 |
| 10 | 0.74 |

Table 22 JM599 K-fold cross-validation accuracy results.

### *7.4.8 Pandas profiling report*

Pandas profiling is a Python library that generates exploratory data analysis (EDA) reports for data frames. Widely utilised in the early stages of data analysis, the Pandas profiling report aids in gaining a rapid understanding of the data and identifying potential issues. Our implementation involved visualising descriptive statistics, such as mean, median, mode, and

various percentiles for each feature as well as addressing missing data.

An essential aspect of this process is the creation of a feature correlation matrix, presented in heatmap format (Figure 55). This matrix provides a visual representation of the correlations between features of target vehicles. The heatmap allows us to identify relationships and dependencies among different variables and to confirm the relationships we already know.

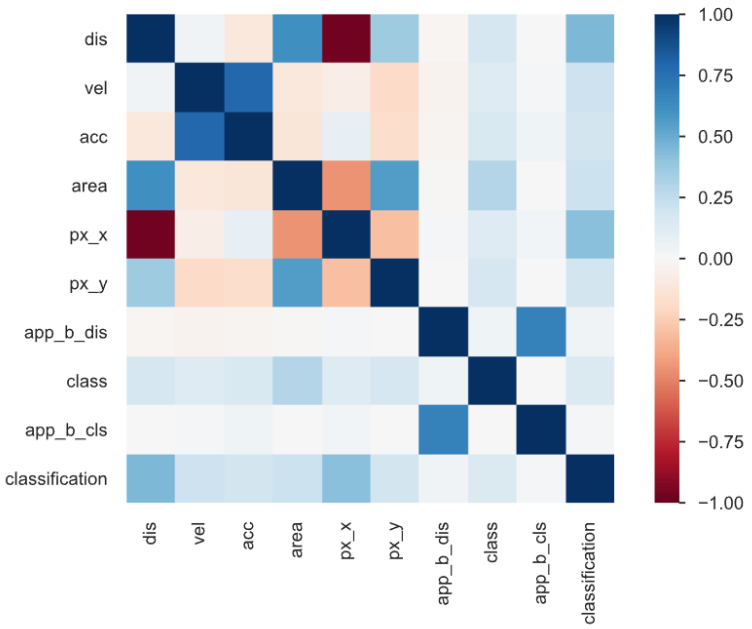



*Figure 55 Feature correlation heat map of JM559 training dataset.*

The relationship between variables is evident in the correlation heatmap in Figure 51. Distance demonstrates a robust positive correlation with area (0.611), indicating that an increase in distance corresponds to an increase in area. As the target vehicle approaches the merge line, the bounding box becomes larger because it is closer to the camera. Conversely, there is a notable negative correlation between distance and pixel X-coordinate (px_x) at -0.966, implying that the pixel X-coordinate tends to decrease as distance increases, confirming that our data is being recorded correctly. Regarding velocity (vel), a strong positive correlation of 0.786 is observed with acceleration (acc), highlighting a significant association between these two factors.

Additionally, velocity exhibits a weaker positive correlation (0.037) with distance due to most target vehicles slowing to the merge line. Moreover, the area displays a substantial positive correlation with distance (0.611) and pixel Y-coordinate (px_y) at 0.562, allowing for the interpretation of approaching vehicles relative to the target vehicles. These correlations confirm that the feature data is reliably recorded and that the correct relationships are apparent across the features. As described in the literature, we could not be confident in the final stages of data analysis without this step.

DYLE JM599 produced 18,480 feature vectors, equating to 403 individually identified target vehicles. There are 12 distinct classes distributed unevenly, as seen in Table 23 and Figure 56. The significant difference between the highest and lowest frequencies reflects the imbalance.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 4,599 | 24.89% |
| w_merge | 4,311 | 23.33% |
| s_merge | 3,214 | 17.39% |
| w_stop | 2,299 | 12.44% |
| m_merge | 1,856 | 10.04% |
| m_stop | 1,789 | 9.68% |
| merge | 209 | 1.13% |
| stop | 191 | 1.03% |
| w_hazard | 3 | 0.02% |
| m_hazard | 3 | 0.02% |
| s_hazard | 3 | 0.02% |
| hazard | 3 | 0.02% |

*Table 23 JM599 class, count, and frequency.*

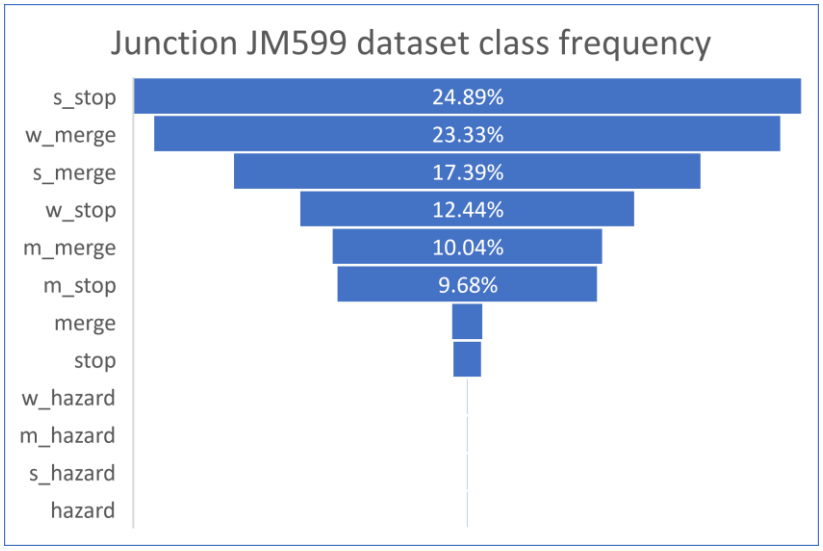



*Figure 56 JM599 class frequency balance.*

The literature highlights the prevalence of imbalances in datasets, and effectively managing these imbalances is a pivotal element in our pipeline. In the case of DYLE, certain classes are underrepresented, such as Hazard classes leaning towards the more frequent classes of Stop and Merge sub-classes. To tackle this issue, we employed two strategies. The initial approach involves generating additional training data, as discussed in Chapter 8. Chapter 9 delves into the utilisation of data augmentation as a means of rectifying this imbalance.

Relatively low cross-validation accuracy is expected in this unbalanced dataset. This is a base dataset for an aggregated dataset combining four junction datasets. In this dataset, there are only 403 $F_{pc}$ classifications, the remaining being subclasses; during the K-fold cross-validation a, the probability of a conclusive classification of Stop, Merge, or Hazard was low.

The remainder of this chapter continues with unique DYLE junction dataset creation and analysis, while the next chapter discusses the steps taken to balance the datasets in favour of a conclusive $F_{pc}$ at an optimal distance from the junction merge line.

## 7.5 Dataset creation for other junctions

Having completed the dataset generation process for JM599, we replicated the procedure for other junctions. The careful fine-tuning of the model—incorporating hyperparameter adjustments and EMA during the training dataset creation and for the data accuracy and reliability investigation performed on the DYLE dataset produced for JM599—facilitated a seamless application of the trained model to the video data from each junction with minimal adjustments.

Each junction obtains a distinct DYLE dataset, complemented by a video recording that tracks the target vehicle identified by its unique ID as it approaches the merge line established by DUKE. Subsequently, each dataset undergoes a comprehensive analysis following the methodology detailed in Section 7.4.6. The classification process is also executed using the methods outlined in Section 7.5.5.

The second junction, JM377, Figure 57, was selected for processing. Situated adjacent to the London Orbital Motorway (M25), JM377 is a busy T-junction. JM377 has no fixed occlusion zones; however, the presence of large vehicles in the foreground introduces a potential challenge by obstructing the view of approaching traffic from direction b.


Figure 57 Junction JM377 shows a van approaching the merge line as traffic approaches from direction b.

Results from k-fold cross-validation on DYLE JM377 are shown in Table 24.

| K value | Mean accuracy JM377 |
|---|---|
| 5 | 0.72 |
| 10 | 0.73 |

Table 24 JM377 K-fold cross-validation accuracy results.

DYLE JM377 produced 25,631 feature vectors, equating to 570 individually identified target vehicles.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 5,466 | 21.33% |
| w_stop | 4,525 | 17.66% |
| m_stop | 4,311 | 16.82% |
| s_merge | 3,775 | 14.73% |
| m_merge | 3,651 | 14.25% |
| w_merge | 3,311 | 12.92% |
| merge | 341 | 1.33% |
| stop | 222 | 0.87% |
| w_hazard | 7 | 0.03% |
| m_hazard | 7 | 0.03% |
| s_hazard | 7 | 0.03% |
| hazard | 7 | 0.03% |

*Table 25 JM377 class, count, and frequency.*

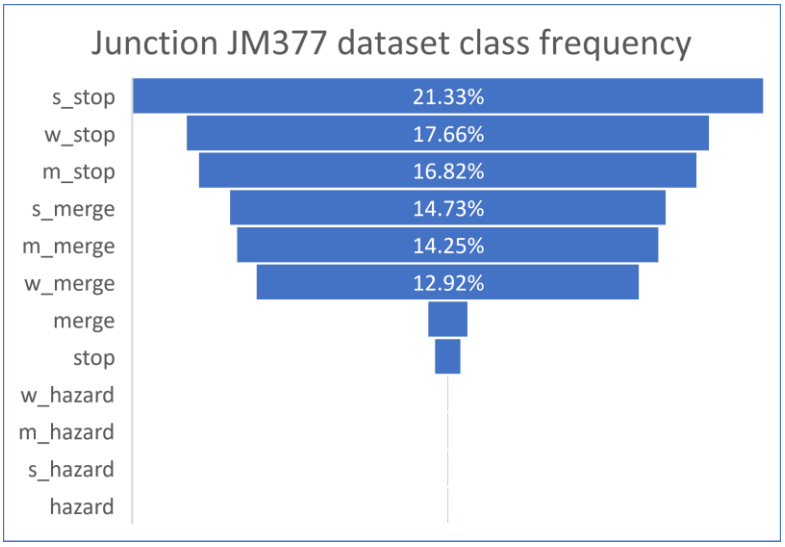



*Figure 58 JM377 class frequency balance.*

The third junction, JM454, is a rural intersection along a bustling major road. Notably, a small occlusion zone exists, as highlighted in the centre of Figure 59. This occlusion is associated with a sign two meters above the ground. It primarily affects larger vehicles, as most smaller vehicles can easily pass underneath the sign.

Figure 59 JM454 with possible occlusion highlighted.

Results from k-fold cross-validation on DYLE JM454

| K value | Mean accuracy JM454 |
|---|---|
| 5 | 0.68 |
| 10 | 0.74 |

*Table 26 JM454 K-fold cross-validation accuracy results.*

DYLE JM454 produced 8,969 feature vectors, equating to 193 individually identified target vehicles.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 1,874 | 20.89% |
| w_stop | 1,741 | 19.41% |
| w_merge | 1,415 | 15.78% |
| m_stop | 1,357 | 15.13% |
| m_merge | 1,231 | 13.73% |
| s_merge | 1,155 | 12.88% |
| stop | 108 | 1.20% |
| merge | 84 | 0.94% |
| w_hazard | 1 | 0.01% |
| s_hazard | 1 | 0.01% |
| m_hazard | 1 | 0.01% |
| hazard | 1 | 0.01% |

*Table 27 JM454 class, count, and frequency.*

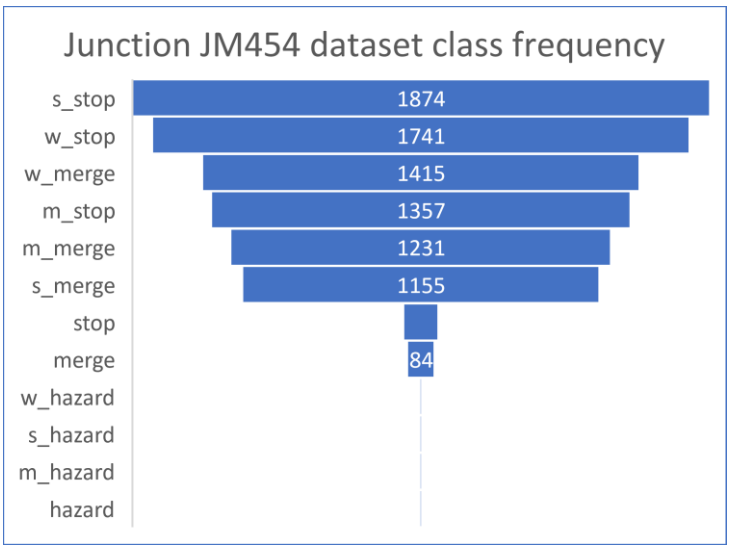



*Figure 60 JM454 class frequency balance.*

The fourth and final junction for which we generated a training dataset is JM384 (Figure 61). Positioned along a major road between two sizable towns, JM384 has no fixed occlusion zones. Large vehicles passing through this junction do not threaten traffic visibility from direction b.

*Figure 61. JM384 has an open topology, allowing clear views of approach b.*

Results from k-fold cross-validation on DYLE JM384 are shown in Table 28.

| K value | Mean accuracy JM384 |
|---|---|
| 5 | 0.71 |
| 10 | 0.71 |

Table 28 JM384 K-fold cross-validation accuracy results.

DYLE JM384 produced 9,982 feature vectors, equating to 217 individually identified target vehicles.

| $F_{pc}$ / $S_{Fpc}$ | count | frequency |
|---|---|---|
| w_stop | 1,945 | 19.49% |
| s_stop | 1,911 | 19.14% |
| w_merge | 1,529 | 15.32% |
| s_merge | 1,522 | 15.25% |
| m_stop | 1,458 | 14.61% |
| m_merge | 1,388 | 13.91% |
| stop | 114 | 1.14% |
| merge | 99 | 0.99% |
| w_hazard | 4 | 0.04% |
| m_hazard | 4 | 0.04% |
| s_hazard | 4 | 0.04% |
| hazard | 4 | 0.04% |

*Table 29 JM384 class, count, and frequency.*

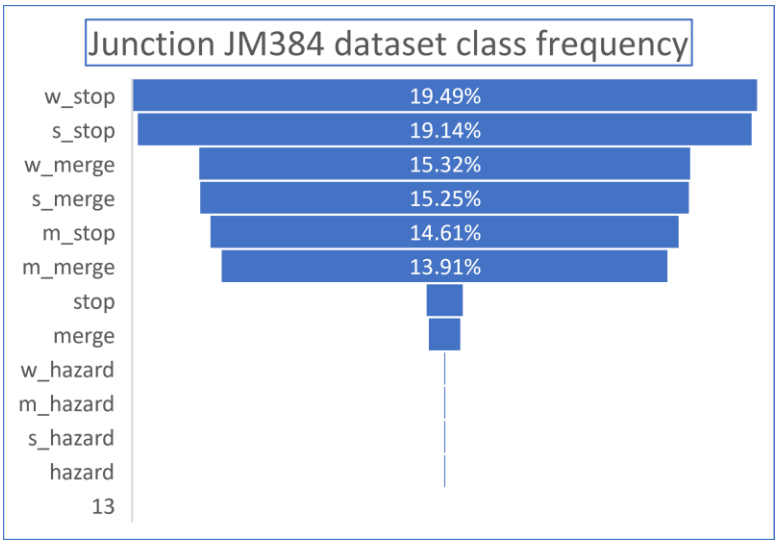



*Figure 62 JM384 class frequency balance.*

The subsequent phase of creating and assessing the training dataset involved consolidating the classified datasets obtained from JM377, JM384, JM559, and JM454. The amalgamation of datasets is a common practice in machine learning, and we implemented methods outlined by Trevizan et al. (2020) and other researchers to enhance our data. These methodologies not only facilitate an expansion of the overall sample size but also improve reliability and generalisability.

One notable advantage of dataset aggregation is the generation of a more representative sample reflecting the behaviour of the target vehicle. If individual datasets exhibit biases or limitations, amalgamating data from diverse sources can counterbalance biases present in

individual datasets, offering a more accurate depiction of T-junction vehicle behaviour. This approach enables a more comprehensive understanding by capturing a wider range of variations, patterns, and trends, thereby presenting a holistic perspective.

Trevizan et al. (2020) state that training models on aggregated datasets can enhance performance because the model can assimilate knowledge from diverse examples, facilitating better generalisation to new, unseen data. Different datasets often contain complementary information, and their aggregation enriches the feature set, yielding a more detailed and nuanced representation of the underlying data.

It is important to note that aggregating datasets introduces more variability into the data, which is beneficial because it enables the analysis or model to capture a broader spectrum of scenarios—a crucial consideration given the inherent variability in our dataset.

### 7.5.1 Results from k-fold cross-validation on Aggregated DYLE

| K value | Mean accuracy aggregated DYLE |
|---|---|
| 5 | 0.77 |
| 10 | 0.79 |

*Table 30 Aggregated junctions DYLE K-fold cross-validation accuracy results.*

The combined DYLE dataset contains 63,062 feature vectors, equating to 1,383 individually identified target vehicles.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 13,850 | 21.96% |
| w_merge | 10,566 | 16.76% |
| w_stop | 10,094 | 16.67% |
| s_merge | 11,051 | 15.33% |
| m_stop | 8,845 | 14.14% |
| m_merge | 8,059 | 12.89% |
| merge | 733 | 1.16% |
| stop | 635 | 1.01% |
| w_hazard | 15 | 0.02% |
| m_hazard | 15 | 0.02% |
| s_hazard | 15 | 0.02% |
| hazard | 15 | 0.02% |

*Table 31 Combined junctions, class, count, and frequency.*

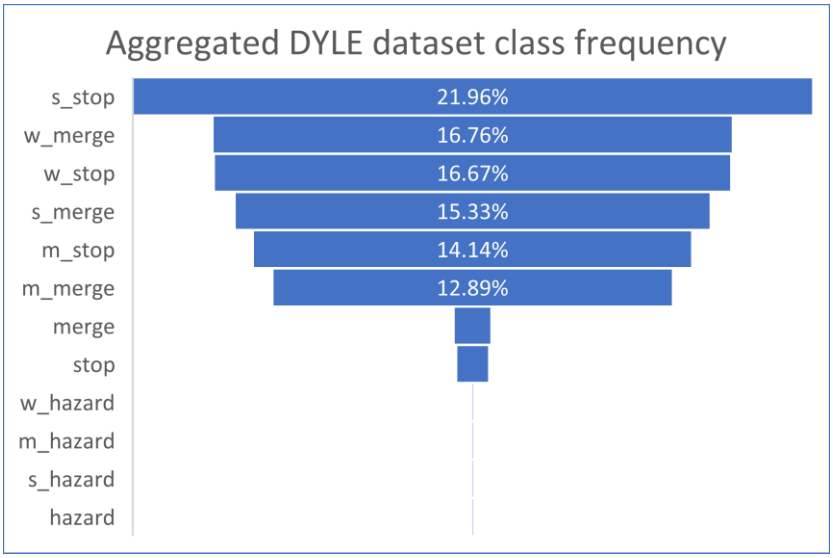



*Figure 63 Combined junction, class frequency balance.*

### 7.5.2 Results discussion

The transition from k = 5 to k = 10 in k-fold cross-validation resulted in a general improvement in mean accuracy. JM599, JM454, and aggregated DYLE exhibited significant increases of 7.2%, 8.8%, and 2.6%, respectively. Notably, the aggregation of datasets further accentuated this improvement, yielding a 10% increase in mean accuracy with k = 5 and an 8.2% increase with k = 10. These findings were compared against the mean averages of the unique junction datasets, as detailed in Table 32 and Figure 64.

| K-value | K-fold cross-validation mean accuracy | | | | |
|---|---|---|---|---|---|
| | JM599 | JM377 | JM454 | JM384 | Aggregated DYLE |
| 5 | 0.69 | 0.72 | 0.68 | 0.71 | 0.77 |
| 10 | 0.74 | 0.73 | 0.74 | 0.71 | 0.79 |

*Table 32 K-fold cross-validation mean accuracy for different datasets.*

The accuracy scores for individual junctions are relatively consistent, with slight variations. JM454 tends to have slightly lower scores in both K=5 and K=10 analysis, suggesting it might

be the most challenging condition or benefitting the most from aggregation. JM454, a rural T-junction, requires vehicles to halt entirely due to limited visibility upon approach. The dataset derived from JM454 predominantly features a 'stop' class, highlighting the benefit of incorporating training data from various other junctions to enrich the dataset.

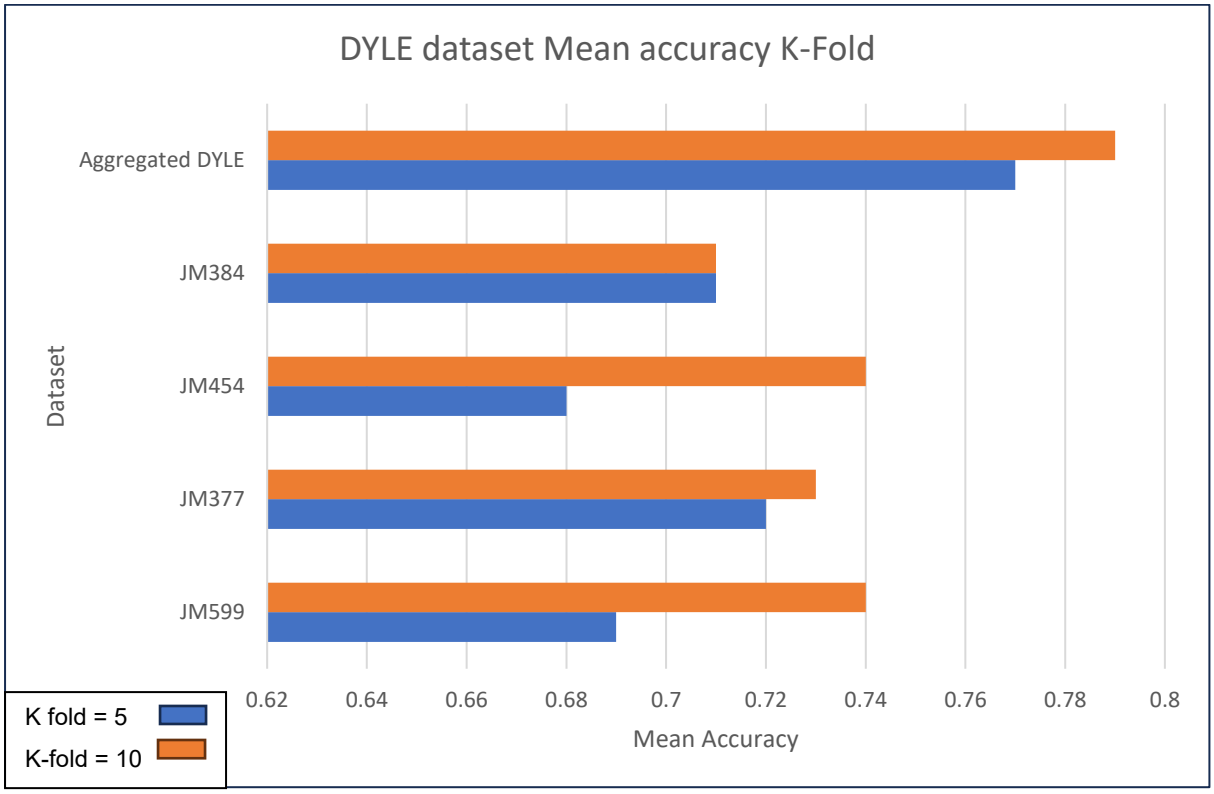



*Figure 64 Chart showing k-fold cross-validation mean accuracy for different datasets.*

Given the nature of hand-classified training data, it was essential to approach any observed improvement cautiously. Despite enhancing mean classification accuracy with the aggregation of junction datasets, it is crucial to acknowledge a significant imbalance in data class distribution. One of our goals is predicting hazardous behaviour at T-junctions, yet the current dataset exhibits a pronounced underrepresentation of the Hazard class, with a scarcity of recorded feature vectors for this particular category.

## 7.7 Chapter conclusion

This chapter illustrated our approaches to generating credible training data, established as feature vectors and arrays. While significant progress has been made, integrating this dataset into a real-world scenario remains challenging. The next chapter introduces our real-time

prediction model to generate predictions from test video data across all junctions and the aggregated dataset. Through this process, we aimed to establish a baseline accuracy to provide a ground truth for metrics that can further determine the reliability of our model.

When we review the research question for this chapter, RQ4, Can our feature vectors' inherent generality be observed per the consistent camera positioning hypothesis?
As we consistently observed target vehicles from a camera positioned at POV x (section 3.4.1) at each test junction, we saw that the results from the aggregated dataset indicate an observable degree of generality in our data, as evidenced by an increase in mean accuracy. Despite slight variations in the perspective of the merge line due to differences in camera placement angles at each junction, our model demonstrated the ability to recognise features from distincT-junctions and successfully apply them to other junctions.
This chapter explored our development of a method for organising and classifying discrete vehicle feature vectors as feature vector arrays, both independently and as integral components of a comprehensive general dataset. To do this, we developed a method for independently organising and classifying target vehicle feature vectors as feature vector arrays to build a comprehensive dataset, DYLE.

## 8.1 Introduction

This section presents the integration of DAISY, our machine learning classifier for predicting intents, into the current workflow. DAISY represents a refinement of the approach described in Chapter 7, which evaluated the precision of DYLE through K-fold cross-validation and naive Bayes classification.

Figure 65 illustrates the integration of DAISY, where feature vectors from DUKE are directly fed into DAISY for real-time intent predictions on target vehicles. This integration allows us to utilise the training data stored in DYLE to base our predictions on new feature vector data.

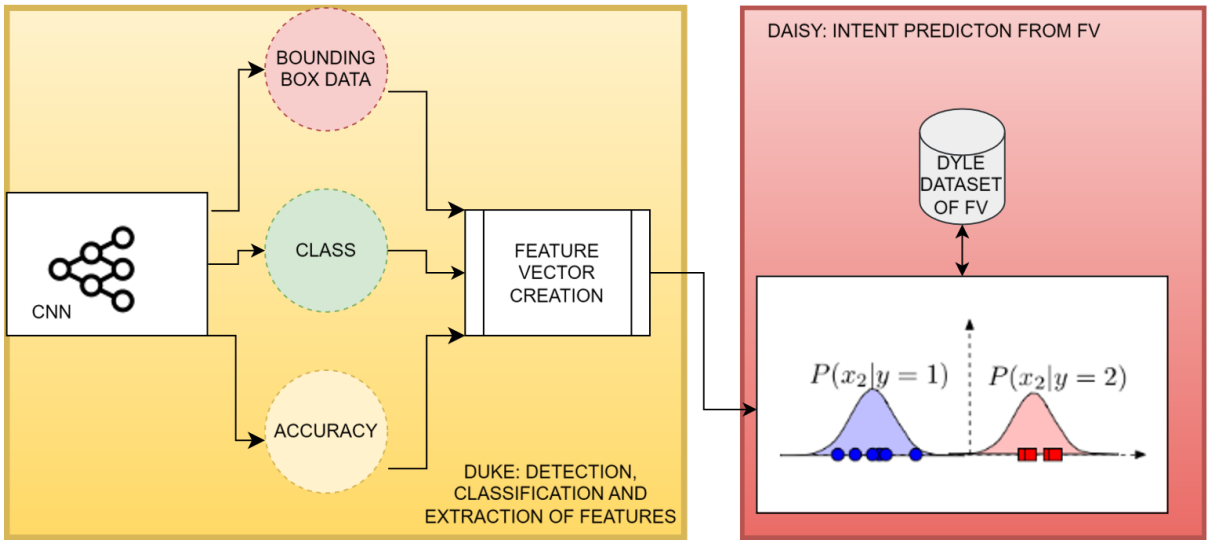



*Figure 65 shows that new and previously unseen feature vectors are transmitted directly to DAISY to classify vehicles' behaviour as they approach the merge line.*

Intent prediction has been extensively researched, as discussed in our review in Chapter 2. The central focus of our work, intent prediction is a complex outcome resulting from a non-trivial problem involving multiple interconnected components in our pipeline. The accuracy and reliability of the results depend heavily on the interdependency of these components, and the methods and results described in the preceding chapters are crucial for ensuring the accuracy of the results achievable using DAISY. Building on the findings outlined in Chapter 2, we adopted a computationally efficient classification method to determine the likelihood of a set of features belonging to a particular class. This choice is pivotal for two key reasons. Firstly, it aims to reduce computational overhead and ensures seamless integration of the classification

code with the rest of the pipeline—a task not without its challenges in Python. The overall efficiency of the pipeline code holds significant importance, given the requirement that classifications must be executed in under 45 ms.

We could use our feature vector data for intent prediction by applying deep-learning or machine-learning classification methods. As highlighted in Chapter 2, the choice of a classification method depends on the classification challenge's specific nature. In our case, the primary challenge revolves around the necessity for rapid classifications of real-time data. Our exploration of suitable intent prediction methods led us to investigate RNN methods, as detailed in Zyner, Worrall, and Nebot (2019) and our work in Chapter 4. Their research on RNN classification involved predicting vehicle intentions at intersections using RNNs with a mixed-density network output layer. They utilised a dataset created with a vehicle-mounted lidar-based tracking system. The model's performance was benchmarked against several baseline models, demonstrating its effectiveness in predicting trajectories with reasonable accuracy. However, it fell short of our requirement for predictions made under 45 ms, with the fastest mean time for prediction being approximately 780 ms.

Additional literature prompted an exploration into the suitability of a linear regression model due to its computational efficiency. Linear regression is a statistical model that predicts the relationship between a dependent variable and one or more independent variables. However, we very quickly shelved the testing of the linear regression model as, upon closer examination, specific weaknesses emerged. Some of the weaknesses we found were highlighted by (Christodoulou et al., 2019). They concluded that the linear regression model's assumption of a linear relationship between independent and dependent variables and sensitivity to outliers impacted estimated coefficients, leading to inaccurate intent estimations. We concluded our analysis by acknowledging that our data generated highly inaccurate predictions when subjected to linear regression. This outcome likely stems from the method's limitations in addressing complex non-linear dynamics, difficulties posed by multicollinearity due to closely related independent variables, and the inherent expectation of constant variance (homoscedasticity) compromised by heteroscedasticity. Consequently, we discontinued further testing with linear regression.

**Research Question 5 (RQ5):** How accurately can a machine learning model, utilising 2D video-derived feature vectors, predict a vehicle's intention at a T-junction?

**Contribution discussed in this chapter:** A computationally efficient approach for predicting vehicle intent at a T-junction using video-derived feature vectors as training data.

## 8.2 Chapter organisation

Section 8.3 introduces the DAISY framework, with subsections providing detailed insights. In 8.3.1, we explore intent prediction in DAISY, highlighting its capabilities. Section 8.3.2 introduces the Gaussian distribution's probability density function. In 8.3.3, we outline the steps of intent prediction in DAISY. Section 8.4 focuses on applying intent prediction to the JM377 junction. Section 8.5 extends this to other junctions, examining system adaptability. Section 8.6 evaluates DAISY's overall performance across all four junctions. In 8.7, we revisit RQ5 and discuss the results.

## 8.3 DAISY

DAISY is a naive Bayes classifier with a probability density function (PDF) model that extends the traditional naive Bayes approach to handle continuous data by modelling the probability distributions of feature vectors assuming a Gaussian distribution. Unlike the traditional naive Bayes, which typically deals with discrete features, a naive Bayes classifier with probability density functions is suitable for continuous features. A PDF models each feature for each class. Instead of directly calculating the likelihood based on observed frequencies, as in the discrete case, likelihood is calculated by evaluating the PDF of the feature for a given class. The 'naïve' assumption of conditional independence given the class label is retained, meaning that the joint probability of observing a set of features is calculated as the product of the individual feature PDFs.

Like traditional naive Bayes, prior probabilities of classes are estimated based on the training data. Given a new observation with continuous features, the classifier calculates the posterior probabilities for each class and predicts the class with the highest probability. The parameters, mean, and variance of the probability density functions are estimated from the training data, which is added to DYLE as the last stage of the pipeline with new online classified feature vectors (see Chapter 9). DAISY can make the final driver intent prediction ($F_{pc}$) and associated subclass predictions ($S_{Fpc}$) based on feature vectors produced by DUKE and those from and added to the DYLE training data. Given the $F_{pc}$ and $S_{Fpc}$, we assume the features are conditionally independent, allowing for comparing pre-classified feature vectors with $S_{Fpc}$s.

We modified DAISY to suit our data by applying Laplace, a smoothing technique (Noto and Saputro, 2022), to handle cases where certain features are calculated as being zero in the training data, such as a stopped vehicle with no approach_b_distance data.

The literature has many examples of using a naive Bayes classifier in machine learning tasks; one paper (Chen et al., 2021) addresses its limitations by incorporating feature weighting and Laplace calibration, resulting in an improved algorithm that achieves over 99% accuracy with a large sample size and remarkable stability; for samples with fewer than 400 attributes and

fewer than 24 categories, the accuracy exceeds 95%.

### 8.3.1 Intent prediction DAISY

Driver intent prediction involves the assignment of categorical labels to a set of input feature vectors. Given the input feature data, this assignment is accomplished by estimating the likelihood of specific class labels. The probability calculation is performed for each potential class label, and the label associated with the highest probability is designated for the input data.

Nevertheless, a direct application of Bayes' theorem for our intent classification encounters impracticalities, primarily due to the computational challenges arising from the many involved features. Consequently, pragmatic approximations are made by estimating class priors and data probabilities from a DYLE. To streamline the computational complexity associated with the conditional probability of the data given the class, a naive Bayes classifier is implemented, operating under the assumption of feature independence.

The naive Bayes model calculates the probability of the input data given the class label by independently computing the conditional probabilities for each input variable. These individual probabilities are then multiplied to obtain the comprehensive probability. DAISY, the model under consideration, determines the posterior probability of a class-given input feature by combining the class's prior probability with the product of the likelihood of observing each feature given the class. Laplace smoothing was applied to mitigate issues related to unseen feature-class combinations during training, ensuring non-zero probabilities and stabilising the data output.

The formula for predicting the probability of a particular class label $Ci$ given the input feature vector $fv$ can be expressed as follows:

$$P(Ci \mid fv) \propto P(Ci)\prod_{j=1}^{n} P(fvj \mid Ci) \qquad (22)$$

Where

- $P(Ci \mid fv)$: The posterior probability of class $Ci$ given the input feature vector *fv*.
- $P(Ci)$: The prior probability of class $Ci$
- $\prod_{j=1}^{n} P(fvj \mid Ci)$: The product of the likelihoods of observing each feature $fvj$ given class $Ci$.

A continuous feature $fvj$ modelled with a probability density function (PDF) and Laplace smoothing is expressed as

$$P(fvj \mid Ci) = \frac{count(fvj\ Ci)+1}{count(Ci)+|Xj|} \quad (23)$$

Where

- $count(fvj, Ci)$: The count of occurrences of the value $fvj$ in DYLE for class $Ci$.
- $count(Ci)$: The total count of instances of class $Ci$ in the training data.
- $|\ Xj\ |$: The total number of unique values for the feature $fvj$ in the DYLE.

Laplace smoothing was initiated to handle cases where an $fv$ was not observed in a given class during training, preventing the assignment of zero probability.

### 8.3.2 Probability density function (PDF) of a Gaussian distribution

The PDF characterises the distribution of fv for each class in terms of mean and standard deviation. To represent the class-specific probability for a feature vector in the dataset DYLE, DAISY uses a Gaussian distribution formula, as shown in *(24),* to represent the distribution of feature vectors for each class.

The equation is as follows:

$$\mathrm{P(fvi)} = (1/\sigma\sqrt{(2\pi)}) * \mathrm{e\text{^}(-(fvi-\mu)\text{^}2\ /\ 2\sigma\text{^}2)} \quad \textit{(24)}$$

- $\mathrm{P(fvi)}$: The probability density function for a given value $\mathrm{fvi}$
- $(1/\sigma\sqrt{(2\pi)}$ The normalisation constant
- $\mathrm{e\text{^}(-(fvi-\mu)\text{^}2\ /\ 2\sigma\text{^}2)}$:: Exponential term
  - $\mu$: The mean of the distribution
  - $\sigma$: The standard deviation

DUKE passes new $f_v$ data to DAISY; relative probability densities are calculated for each input value in an $f_v$ using the Gaussian PDF and the statistics for that column and that class. This process is repeated for each class in the dataset. Relative driver intent probability densities are returned as a value for each class, and intent is defined as the greater of the classes. Densities are often small numbers because they represent the probability per unit of measurement, and the range of possible values in a distribution can be extensive. PDF values describe the relative likelihood of a value occurring in a particular distribution region.

### 8.3.3 Intent classification steps

The first step in the DAISY algorithm involves computing statistics from the dataset DYLE, organised by class. This process establishes the mean and standard deviation for each feature in DYLE and the following probability calculations using class-specific statistics to generate the Gaussian PDF, allowing DAISY to model and effectively represent the distribution of fv for each class.

1) Prior Probability Estimation: Calculate the prior probability of each class based on the training data using equation (25).

$$p(Ci) = \frac{Number\ of\ instances\ of\ class\ Ci}{Total\ number\ of\ instances} \quad (25)$$

2) Feature Likelihood Calculation: For each feature in each class:

- Calculate ($\mu$) and ($\sigma$) of the feature for that class.
- Use the probability density function (PDF) formula to calculate the likelihood of each feature value given the class: equation (24).

3) Laplace Smoothing: Apply Laplace smoothing to handle cases where specific feature values have zero probabilities: equation ( 23).

4) Prediction for new feature vector data: Calculate the posterior probability for each class using the naive Bayes formula: equation (22).

This process utilises PDFs, applies Laplace smoothing to address sparsity issues, and uses the naive Bayes classifier assumptions to compute posterior probabilities for classification.

## 8.4 Live prediction by DAISY

During the data collection phase for training DAISY in Chapter 7, the system classified vehicle intent by displaying secondary classifications ($S_{Fpc}$) and the final primary classification ($F_{pc}$). When reviewing the video recordings of the training process, it was noted that DAISY's predictions primarily consisted of various $S_{Fpc}$ classifications such as W_merge, M_merge, S_Merge, W_stop, M_stop, and S_stop. This predominance of $S_{Fpc}$ classifications in the predictions aligns with the composition of the training data.

The accuracy of our training data is fundamental to the success of our system in making accurate predictions of vehicle intent at various stages in the pipeline. By utilising the

subclasses in the training data, we can now make $F_{pc}$ predictions from the initial point of target vehicle detection rather than solely at the merge line.

In section 7.4.2, we discussed how subclasses are defined in relation to the distance from the merge line. However, DAISY's predictions do not strictly conform to the predefined distance ranges. For instance, we might observe weak predictions in what was designated as a substantial distance range and moderate predictions in what was initially categorised as a weak range. This flexibility allows DAISY to adapt to the nuances of real-world driving behaviour rather than being confined to rigid distance categories.

A feature vector is created when DUKE detects a vehicle and generates the initial features. This vector is immediately passed to DAISY. Based on the vehicle's distance from the merge line, an $S_{Fpc}$ is assigned to this feature vector a feature and added to the feature vector array associated with the vehicle's ID. Predictions are made from this array of associated $S_{Fpc}$s, which evolves and expands with each new data iteration. Real-time predictions are generated using a majority voting technique; each $S_{Fpc}$ is considered equally, and the class that receives more than half of the votes is chosen as the final real-time prediction. If no class receives more than half the votes, no prediction is made for that 40 ms iteration.

We first establish a majority $S_{Fpc}$ prediction from the list during each iteration and then derive an $F_{pc}$. It is important to note that predictions made during initial iterations are less accurate as the dynamic list populates. Over time, as more data is gathered, the predictions become more reliable.

The assignment of an $F_{pc}$ to a vehicle's feature vector array is executed as a separate function once the tracking of the target vehicle concludes. When the vehicle crosses the merge line, an $F_{pc}$ is applied to the final feature vector in the array. This vector is then used to retroactively apply $S_{Fpc}$ to all preceding vectors in the array, ensuring a consistent classification across the vehicle's trajectory.

In cases where DAISY inaccurately predicts the final action of a vehicle, it is crucial to understand that these errors do not contaminate the training data. The actual actions of vehicles at the merge line are precisely recorded and fed into DYLE. This method is hypothesised to gradually cultivate a more accurate dataset, reducing the need for extensive manual checks. As more vehicles are autonomously analysed and classified based on their actual behaviours, the system's predictive accuracy is expected to improve, contributing to more effective and reliable vehicle intent predictions.

We must classify and analyse the errors it makes to address the error types left in DAISY and understand their safety implications. These errors can be categorised into false positives and false negatives.

**False Positives:**

Daisy predicts an action or intent that does not occur. For example, it predicts that a vehicle

will merge when it does not.

Learning:

- Over-cautiousness in prediction models can lead to a higher rate of false positives.
- Possible causes could include overly sensitive thresholds for intent prediction or misinterpreting ambiguous manoeuvres.

Safety Implications:

- Driver Distraction: Frequent false alarms can distract drivers, reducing their attention to critical driving tasks.
- Reduced Trust: Over time, drivers may become desensitised to the system's warnings, potentially ignoring important alerts when they are genuine.

**False Negatives**:

Description: The system fails to predict the occurrence of an action or intent. For instance, it was not predicted that a vehicle would stop when it does.

Learning:

- Indicates a lack of sensitivity or failure to detect subtle cues leading to the intended action.
- This could be due to insufficient training data for specific scenarios or ineffective feature extraction from the input data.

Safety Implications:

- Missed Critical Events: Missing critical events like sudden stops or merges can lead to accidents or near-misses.
- Delayed Reactions: Failure to predict vehicle actions can delay necessary responses, increasing the risk of collisions.

Mitigation:

Incorporate more detailed environmental context, such as road signs and pedestrian presence, to improve stop predictions.

Implement online learning mechanisms where the model can continuously learn and adapt from new data, improving its accuracy over time.

Regularly update the model with new data to keep it current with evolving driving patterns and behaviours.

Safety Implications Summary:

False positives can lead to driver distraction and reduced trust, while false negatives and context-specific errors pose direct safety risks. By enhancing data quality, refining models, and incorporating continuous learning, the predictive accuracy of DAISY can be significantly improved, leading to safer and more reliable vehicle intent predictions.

### 8.4.1 Intent prediction for a single junction JM377

Using the video data (20%) allocated for DAISY-derived predictions, as shown in Table 33, we pass this unseen Bo video data through DUKE to collect feature vectors for analysis by DAISY to predict intent. This procedure follows the methodology outlined in section 7.4.3 and generates a video featuring labelled predictive bounding boxes and visualising the ground truth actions at the merge line.

| EuroRAP route | Junction location | Total minutes | Manual pred class training data 60% | DAISY-derived pred class 20% | Online verification 20% |
|---|---|---|---|---|---|
| JM377 | Oxshot Road | 263 | 157.8 | 52.6 | 52.6 |
| JM384 | A248 | 124 | 74.4 | 24.8 | 24.8 |
| JM559 | Petersfield Road | 249 | 149.4 | 49.8 | 49.8 |
| JM454 | Rowhook Road | 207 | 124.2 | 41.4 | 41.4 |
| UO196 | Jacobs Well Road | 119 | | | 119 |

*Table 33 is a reference copy of Table 21 from Section 7.3, showing the partitions for training, ground truth, and online verification of the entire pipeline.*

To generate an intent prediction for the DAISY-derived data partition in Table 33, we created a new video file, as specified in Section 5.7, using the partitioned 52.6 minutes allocated. Feature vectors are sent from DUKE to DAISY when a vehicle is detected in the video and identified as approaching the merge line of the T-Junction. Using the training data in DYLE and the process detailed in Section 8.3.3, DAISY initiates its predictions with Sfp as a function of distance from the merge line, followed by an $F_{pc}$ prediction at the merge line as the final outcome. Utilising recorded video, we validate the final predictions made by DAISY and document them as ground truth. This provides a list of final predictions and merge line truths for validation. A unique DYLE is also generated for each junction, incorporating $S_{Fpc}$-classified feature vector arrays for each target vehicle in the video segment. This approach enables the creation of new manually classified training data to supplement DYLE. Figures 66–69 inclusively visually depict the predictions made by DAISY as the target vehicle approaches the merge line.

*Figure 66 JM377: DAISY sub-classification of a Weak Merge.*


*Figure 67 JM377: DAISY sub-classification of a Moderate Merge.*


*Figure 68 JM377: DAISY sub-classification of a Strong Merge.*


*Figure 69 JM377: DAISY final classification of a Merge.*

Junction JM377 poses a challenge due to its brief detection distance from the initial identification to the merge line. The examples shown in Figures 62–65 are of the same target vehicle, shown from initial detection in Figure 66, where DAISY has sub-classified ($S_{Fpc}$) the vehicle as a Weak Merge based on the feature vector created by DUKE. The feature vector creation and DAISY classification occur in around 40 ms, allowing multiple predictions to be achieved before the merge line. The results were analysed for accuracy using a confusion matrix (Figure 70) and are tabulated in Table 34.

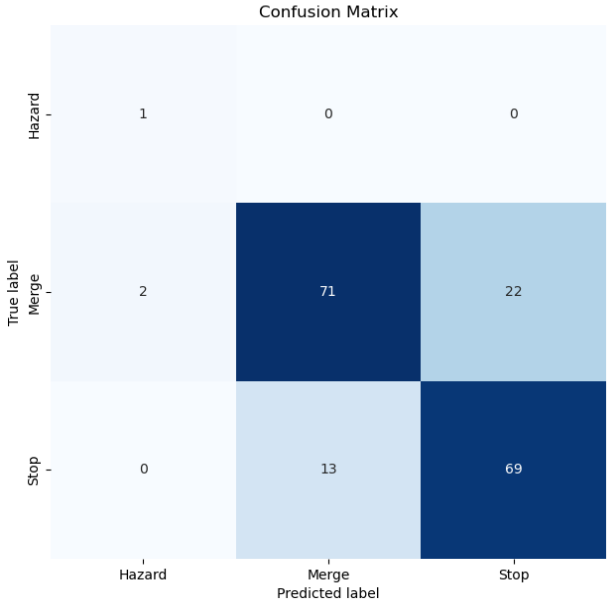



*Figure 70 Confusion matrix of ground truth results against DAISY predictions for $F_{pc}$ in junction JM377.*

Confusion matrix key

- True positives (TPs): Instances correctly predicted as positive; in Figure 63, for example, there are 69 TP Stops and 71 TP Merges.
- False positives (FPs): Instances incorrectly predicted as positive in Figure 63; there are 22 FP Stops and 13 FP Merges.

| Class | Precision | Recall | F1-Score | Quantity |
|---|---|---|---|---|
| Hazard | 0.33 | 1.00 | 0.5 | 1 |
| Merge | 0.85 | 0.75 | 0.79 | 95 |
| Stop | 0.76 | 0.84 | 0.80 | 82 |
| | | | | |
| Accuracy | | | 0.79 | 178 |
| Macro Avg | 0.65 | 0.86 | 0.70 | 178 |
| Weighted Avg | 0.80 | 0.79 | 0.79 | 178 |

*Table 34 Accuracy metrics for Junction JM377.*

### 8.4.2 Initial discussion from results of JM377.

Table 37 shows the accuracy results across three classes, Hazard, Merge, and Stop, with an overall accuracy of 0.79 for 178 instances. The model demonstrates a perfect recall for Hazard

at 1.00 but with low precision (0.33), indicating it correctly identifies all Hazard instances but also misclassifies other classes as Hazard. Merge and Stop classes have more balanced metrics, with precision and recall values indicating a relatively strong ability to correctly identify and classify instances, evidenced by F1 scores of 0.79 and 0.80, respectively. The Macro Average suggests a disparity in class performance with an average precision of 0.65 and recall of 0.86, hinting at the model's tendency to favour recall over precision. The Weighted Average precision and recall, closely mirroring the overall accuracy, indicate a model that performs well overall but may benefit from adjustments to improve precision, especially in less frequent classes like Hazard, without compromising its high recall.
The outcomes from JM377, despite the limited data volume, indicate a positive trajectory. With a more evenly distributed dataset, it is anticipated that the accuracy of DYLE will be enhanced.

### 8.4.2 Evaluation metrics

Accuracy and the F1 Score are both metrics widely used to evaluate a classification model's performance, but focus on different aspects. Accuracy is the most intuitive performance measure, and it is simply a ratio of correctly predicted observations to the total observations. It is the number of correct predictions the model makes overall predictions made. The F1 Score is the Harmonic Mean between precision and Recall. It is a way of combining the precision and Recall of the model. The F1 Score is a metric we use to balance Precision and Recall, and there is an uneven class distribution. The main difference between accuracy and the F1 score is that accuracy is used when the true positives and true negatives are more important, while the F1-Score is used when the false negatives and false positives are crucial, which is imperative in our study. However, in a multi-class confusion matrix, the true negatives (TN) are not explicitly stated for each class because they are the sum of all correct predictions for the other classes.
So we simplify the formula for multi-class classification to:

Accuracy= Sum of the diagonal (True Positives for each class) / Total number of predictions

Accuracy can be misleading if the data set is imbalanced when the number of observations in different classes varies greatly. For example, if you have a test set of 100 instances and 95 belong to one class and 5 to another, a model that always predicts the majority class will have an accuracy of 95% despite being unable to identify the minority class.
F1 Score does not take true negatives into account. It focuses on the model's ability to correctly classify instances for a given class (or for all classes if calculating the macro or

weighted average F1), which makes it more beneficial as we are more interested in the balance between precision and Recall.

Precision, Recall, and F1 Score are pivotal for understanding the model's capabilities in classification tasks. These metrics offer valuable insights into DAISY's ability to accurately identify positive instances while minimising false positives and false negatives. Precision and Recall, fundamental metrics in binary classification, play distinct roles in assessing DAISY's performance. Precision measures the accuracy of DAISY's positive predictions, calculated as the ratio of true positives to the sum of true positives and false positives. Conversely, Recall, also known as sensitivity or the true positive rate, evaluates DAISY's proficiency in capturing all positive instances by calculating the ratio of true positives to the sum of true positives and false negatives. When applied to DAISY, these metrics provide a nuanced and comprehensive evaluation of its effectiveness in making accurate positive predictions and capturing all relevant instances in a classification task.

Table 34 provides performance metrics, including Precision, Recall, and F1-score, for DAISY using Junction JM377 video data. The classes represent the labels DAISY is predicting. There are three $F_{pc}$ classes: Hazard, Merge, and Stop. Precision is the ratio of correctly predicted positive observations (true positives) to the total predicted positives (true positives + false positives). Precision measures the accuracy of positive predictions made by the DAISY. For example, Precision is 0.33 for the Hazard class, indicating that 33% of the instances the model predicted as Hazards were true positives. Recall, or true positive rate, is the ratio of correctly predicted positive observations (true positives) to the total actual positives (true positives + false negatives). Recall measures DAISY's ability to capture all positive instances. For example, the Recall for the Merge class is 0.75, indicating that DAISY correctly identified 75% of the Merge class. F1-score is the harmonic mean of Precision and Recall, providing a balance between the two metrics. The F1-score is instrumental as we have a significant imbalance between classes. The Quantity column indicates the number of instances in each class.

We use these metrics to evaluate DAISY's performance for each class and its overall effectiveness in making predictions across different categories for single and a combination of junctions.

## 8.5 Intent prediction for other junctions

We followed the same procedures described in 8.4 for the remaining four junctions, recorded the results, produced a confusion matrix, and tabulated accuracy metrics.

### 8.5.1 Junction JM384

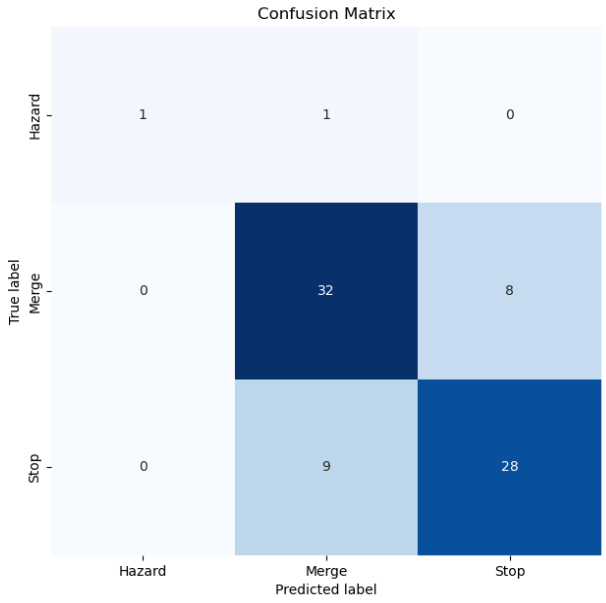



*Figure 71 Confusion matrix of ground truth results against DAISY predictions for $F_{pc}$ in junction JM384.*

| Class | Precision | Recall | F1-Score | Quantity |
|---|---|---|---|---|
| Hazard | 1.00 | 0.5 | 0.67 | 2 |
| Merge | 0.76 | 0.80 | 0.78 | 40 |
| Stop | 0.78 | 0.76 | 0.77 | 37 |
| | | | | |
| Accuracy | | | 0.77 | 79 |
| Macro Avg | 0.85 | 0.69 | 0.74 | 79 |
| Weighted Avg | 0.78 | 0.77 | 0.77 | 79 |

*Table 35 Accuracy metrics for Junction JM384. The table presents each class's Precision, Recall, and F1-score metrics and the total incidences.*

#### *8.5.1.2 Initial discussion from results of JM384*

The Macro average scores (0.85 for Precision, 0.69 for Recall, and 0.74 for F1-Score) suggest that, on average, DAISY performs well in terms of precision but struggles more with recall. The Weighted average scores closely align with the DAISY accuracy (0.77 for Precision and F1-Score and 0.77 for Recall), indicating a consistent performance across classes when adjusted for their representation in the test set. DAISY demonstrates a decent performance overall, with solid precision but weaker recall, especially for the Hazard class, due to the small number of Hazard instances. The balanced performance on Merge and Stop suggests that DAISY has learned these classes well.

### 8.5.2 Junction JM599

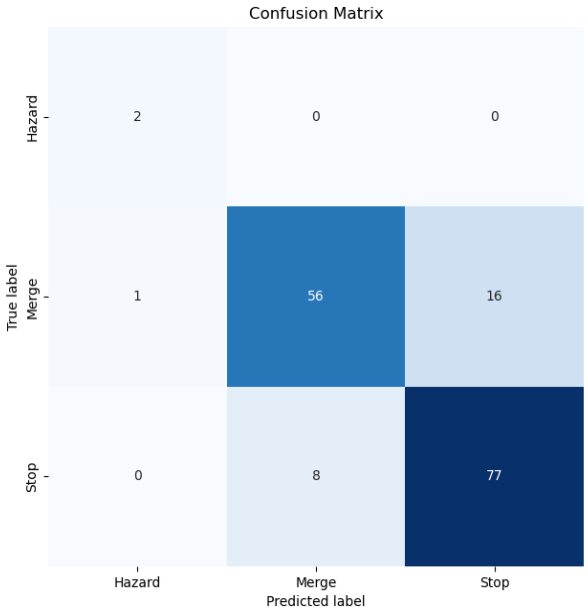



*Figure 72 Confusion matrix of ground truth results against DAISY predictions for $F_{pc}$ in junction JM599.*

| Class | Precision | Recall | F1-Score | Quantity |
|---|---|---|---|---|
| Hazard | 0.67 | 1.00 | 0.80 | 2 |
| Merge | 0.88 | 0.77 | 0.82 | 73 |
| Stop | 0.83 | 0.91 | 0.87 | 85 |
| | | | | |
| Accuracy | | | 0.84 | 160 |
| Macro Avg | 0.79 | 0.89 | 0.83 | 160 |
| Weighted Avg | 0.85 | 0.84 | 0.84 | 160 |

*Table 36 Accuracy metrics for Junction JM599. The table presents each class's Precision, Recall, and F1-score metrics and the total incidences.*

#### *8.5.2.1 Initial discussion from results of JM599*

The Hazard class, with the lowest quantity of instances (2), achieved perfect Recall (1.00) and a good Precision (0.67), resulting in an F1-Score of 0.80. The Merge class, with a significant quantity of instances (73), showed high Precision (0.88) but slightly lower Recall (0.77), leading to an F1-Score of 0.82. The Stop class, having the highest number of instances (85), demonstrated strong performance with a Precision of 0.83 and a Recall of 0.91, achieving the highest F1-Score of 0.87. Overall, the DAISY exhibits an Accuracy of 0.84 across 160 instances, with a Macro Average indicating a balanced performance across classes (Precision: 0.79, Recall: 0.89, F1-Score: 0.83) and a Weighted Average reflecting the influence of class imbalance (Precision: 0.85, Recall: 0.84, F1-Score: 0.84).

### 8.5.3 Junction JM454

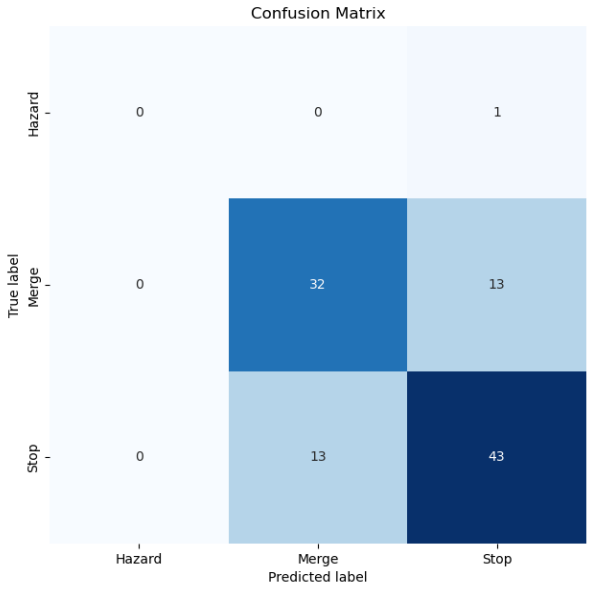



*Figure 73 Confusion matrix of ground truth results against DAISY predictions for $F_{pc}$ in junction JM454.*

| Class | Precision | Recall | F1-Score | Quantity |
|---|---|---|---|---|
| Hazard | 0.00 | 0.00 | 0.00 | 0.00 |
| Merge | 0.71 | 0.71 | 0.71 | 45 |
| Stop | 0.75 | 0.77 | 0.76 | 56 |
| | | | | |
| Accuracy | | | 0.74 | 102 |
| Macro Avg | 0.49 | 0.49 | 0.49 | 102 |
| Weighted Avg | 0.73 | 0.74 | 0.73 | 102 |

*Table 37 Accuracy metrics for junction JM454. The table presents each class's Precision, Recall, and F1-score metrics and the total incidences.*

#### 8.5.3.1 Initial discussion from results of JM454

The Hazard class shows no instances and has a performance score of 0.00 across Precision, Recall, and F1-Score. DAISY performs moderately for the Merge class with Precision, Recall, and F1-Score at 0.71 across 45 instances, suggesting a balanced ability to identify and classify Merge instances correctly. The Stop class performs slightly better with Precision at 0.75,

Recall at 0.77, and F1-Score at 0.76 over 56 instances, indicating good accuracy in identifying Stop instances. Overall, an accuracy of 0.74 over 102 instances, with a Macro Average and Weighted Average across metrics at 0.49 and 0.73, respectively. Weighted Average accounts for class imbalance, suggesting that despite the poor performance on Hazard, DAISY performs reasonably well on Merge and Stop instances.

## 8.6 DAISY performance with aggregated data for all four junctions

The validation process of DAISY, utilising empirical ground truth observations for each junction, facilitated the generation of supplementary training data. To ensure diversity, we allocated 20% of the video data for each junction that had not been used for training. DUKE generated feature vector arrays for each target vehicle while validating individual junctions, as detailed in Section 8.5. We subsequently validated DAISY's intent predictions for each feature vector within the target vehicle array. This iterative validation process enabled the incorporation of manually classified features into DYLE, expanding the number of training examples available for DAISY's predictions. Only true positive data was utilised as additional training feature vectors, enhancing the model's accuracy prediction capacity. We implemented the approach outlined in Section 7.4.6.1, employing k-fold cross-validation on the revised DYLE dataset. Subsequently, we amalgamated the ground truth observations for each junction and conducted a comprehensive accuracy analysis. Similar to our assessments for individual junctions, we utilised a confusion matrix and accuracy calculations to evaluate the model's overall performance across the combined dataset. This methodology allowed us to gauge the model's effectiveness in a broader context, considering the diverse data encompassed by the updated DYLE dataset.

### 8.6.1 K-fold cross-validation of updated DYLE training dataset

We appended the previously aggregated DYLE from 7.6.1 with the feature vectors from these experiments to create an updated DYLE constituting 79,461 feature vectors, equating to 1,901 individually identified target vehicles.

After augmenting the DYLE dataset with freshly classified ground truth data, we observed a marginal improvement in k-fold cross-validation accuracy, rising from the initial value of 0.79 (Section 7.6.1) to the updated score of 0.81 (with k = 10, Table 38). This enhancement underscores the positive impact of incorporating additional manually classified data on the overall performance and accuracy of the model during cross-validation.

| K value | Mean accuracy updated DYLE |
|---|---|
| 5 | 0.79 |
| 10 | 0.81 |

*Table 38 Updated aggregated junctions DYLE k-fold cross-validation accuracy results.*

Analysing the updated DYLE dataset in Table 39 and Figure 74, we observed improved balance and class distribution in the first eight classes compared to the aggregated DYLE in Section 7.6.1. The additional manually classified ground truth data contributed to a slightly more even representation of classes, potentially enhancing the model's generalisation capabilities.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 16,620 | 20.92% |
| w_merge | 12,679 | 15.96% |
| w_stop | 13,012 | 16.38% |
| s_merge | 13,261 | 16.69% |
| m_stop | 11,025 | 13.87% |
| m_merge | 10,995 | 13.84% |
| merge | 968 | 1.19% |
| stop | 913 | 1.07% |
| w_hazard | 18 | 0.02% |
| m_hazard | 19 | 0.02% |
| s_hazard | 17 | 0.02% |
| hazard | 20 | 0.03% |

*Table 39 Updated DYLE.*

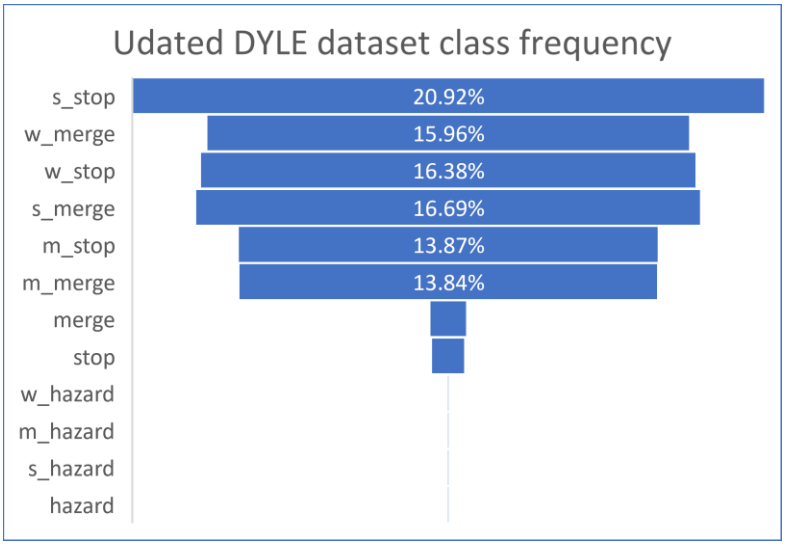



*Figure 74 Updated DYLE class frequency.*

### 8.6.2 Analysing ground truth and prediction data

The subsequent stage involved consolidating the ground truth data from individual junctions into a unified file. The aim was to conduct an accuracy analysis that allowed evaluation of DAISY's overall performance when confronted with data from diverse junctions, providing insights into its accuracy across the entire dataset.

### 8.6.3 Precision-Recall and F1-Score

The accuracy metrics of the results of the combined dataset (Table 40) indicate higher Precision in each class and, thus, fewer false positives. Notably, the Merge class

demonstrates the highest Precision, with values of 0.56 for Hazard, 0.86 for Merge, and 0.82 for Stop. A higher Recall, signalling fewer false negatives, is observed in the Stop class, boasting values of 0.83 for Hazard, 0.80 for Merge, and 0.86 for Stop. Additionally, a higher F1-Score, reflecting a superior trade-off between Precision and Recall, stands out in the Stop class, with scores of 0.67 for Hazard, 0.83 for Merge, and 0.84 for Stop.

The overall accuracy for the combined dataset reaches 83%. The Macro Average, which independently calculates metrics for each class and then takes the average, provides an overview of the model's performance across all three classes, with a Precision of 0.74, a Recall of 0.83, and an F1-Score of 0.78.

Moreover, the weighted average considers the number of instances for each class for Precision, Recall, and F1-Score. This approach assigns more significance to classes with larger instances and results in a Precision of 0.84, a Recall of 0.83, and an F1-Score of 0.83, offering a more detailed evaluation considering class distribution.

| Class | Precision | Recall | F1-Score | Quantity |
|---|---|---|---|---|
| Hazard | 0.56 | 0.83 | 0.67 | 6 |
| Merge | 0.86 | 0.80 | 0.83 | 253 |
| Stop | 0.82 | 0.86 | 0.84 | 260 |
| | | | | |
| Accuracy | | | 0.83 | 519 |
| Macro Avg | 0.74 | 0.83 | 0.78 | 519 |
| Weighted Avg | 0.84 | 0.83 | 0.83 | 519 |

*Table 40 Combined junction accuracy metrics show each class's Precision, Recall, and F1-score metrics and the total incidences.*

### 8.6.4 Metric comparison of single and combined junctions

We evaluated the four junctions individually and collectively, summarising the results in Table 41 and Figure 75. The overall accuracy is 0.83 in the updated DYLE, suggesting a balanced trade-off between Precision and Recall. Hence, there was accurate classification of 83% of instances. The k-fold cross-validation result of 0.81 attests to the model's consistent performance across diverse subsets of the dataset.

Examining each specific junction (JM377, JM384, JM599, JM454), accuracy values range from 0.71 to 0.84, reflecting variations in junction dynamics, traffic volume, and behaviour. The k-fold cross-validation results for individual junctions (ranging from 0.71 to 0.74) indicate stable performance across different folds.

The updated DYLE dataset and the combined ground truth data exhibit enhanced accuracy compared with the average of the four junctions and the aggregated DYLE metrics from Chapter 7. The k-fold cross-validation results underscore the model's reliability, consistent performance, and ability to generalise effectively. This observation is particularly relevant to our exploration of the hypothesis in Research Question 4 (RQ4), where we sought to investigate and analyse specific aspects related to how our model can generalise across differenT-junctions.

| Metric | | F1-Score | | | Accuracy | K-fold (10) |
|---|---|---|---|---|---|---|
| Junction | Class | Hazard | Merge | Stop | | |
| Updated DYLE | | 0.67 | 0.83 | 0.84 | 0.83 | 0.81 |
| JM377 | | 0.5 | 0.79 | 0.8 | 0.79 | 0.73 |
| JM384 | | 0.5 | 0.8 | 0.76 | 0.77 | 0.71 |
| JM599 | | 0.8 | 0.82 | 0.87 | 0.84 | 0.74 |
| JM454 | | 0 | 0.71 | 0.76 | 0.74 | 0.74 |
| Aggregated Training DYLE | | | | | | 0.79 |

*Table 41 Comparison of single and combined junction and DYLE dataset accuracy metrics.*

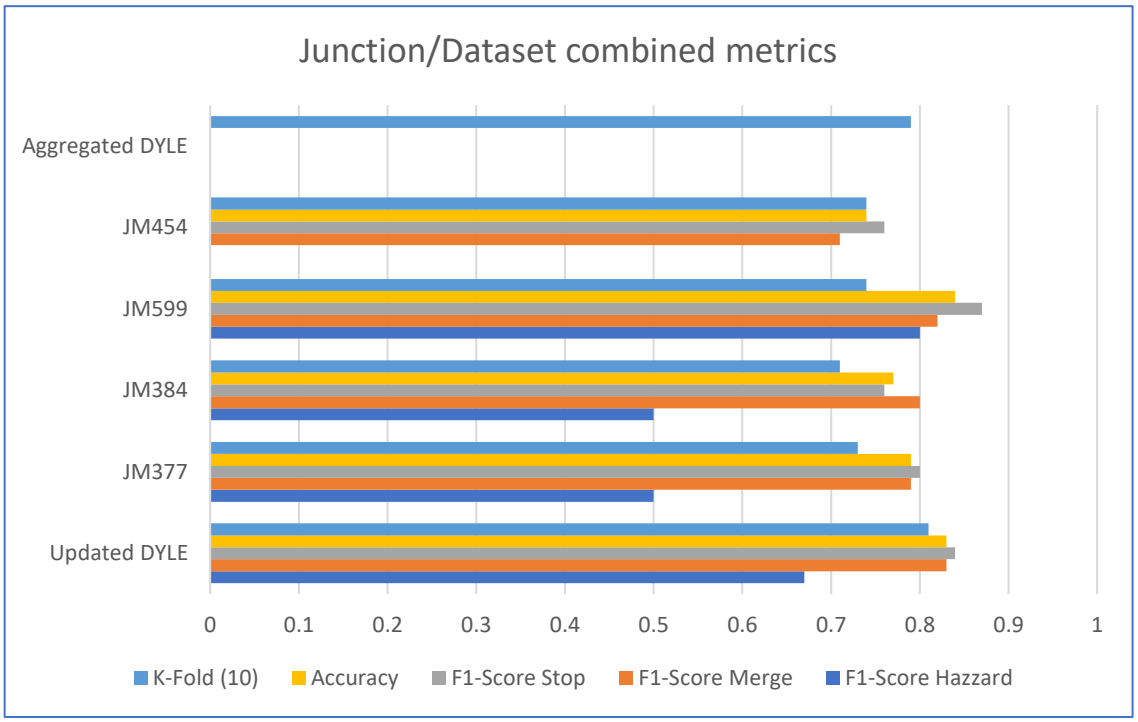



*Figure 75 K-fold training accuracy compared to ground truth F1-Score per class and overall accuracy.*

## 8.7 Discussion

Upon reviewing the updated DYLE dataset and comparing it to the aggregated DYLE from Section 7.6.1, it became evident that the dataset's composition had improved. Incorporating new manually classified ground truth data led to a slightly more balanced representation of the main classes. This enhancement in data balance suggests a positive impact on the model's performance, indicating that it is potentially better at generalisation across differenT-junctions. In summary, the results indicate that the dataset refinement contributed to a more robust and representative training set for the model.

This chapter addressed **Research Question 5 (RQ5),** '*How accurately can a machine learning model, utilising 2D video-derived feature vectors, predict a vehicle's intention at a T-junction?*' We have shown reasonable accuracy using data from a single junction, achieving an F1 score of 0.87 for the Stop class and 0.82 for the Merge class at JM599. By amalgamating ground truth data and manually incorporating newly classified data, we observe enhanced k-fold cross-validation, resulting in higher F1 scores than the averages for individual junctions. The current accuracy level stands at 0.83.

**Contribution discussed in this chapter:** *A computationally efficient approach for predicting vehicle intent at a T-junction using video-derived feature vectors as training data*. We have demonstrated that in terms of efficiency, we can predict an intention in approximately 40 ms, demonstrating an efficient pipeline backed with an accuracy of 83% on a small dataset.

In the next chapter, we transition the entire pipeline online, activating DAISY to transmit predictions directly to DYLE through classified feature vectors. This transformation renders the pipeline a fully remote system, enabling real-time updates and inferences on new data within 100 milliseconds and dynamically building the training dataset.

# Chapter 9: Pipeline Autonomy

## 9.1 Introduction

In the preceding chapters, we discussed the methodologies and methods constituting the individual components in the pipeline depicted in Figure 76. This sequential process explains the means through which we predict vehicle intent.

The pipeline feeds 2D video as an input, employs deep learning for detection, classification, and tracking, and uses a machine learning prediction model, DAISY. DAISY analyses historical vehicle actions using verified and categorised feature vectors, ultimately generating an intent classification as output.

We now possess a reasonably accurate trained model and can generalise effectively across diverse junctions. Instead of merely freezing this model and deploying without incorporating additional training data, we introduce an additional phase: autonomy.

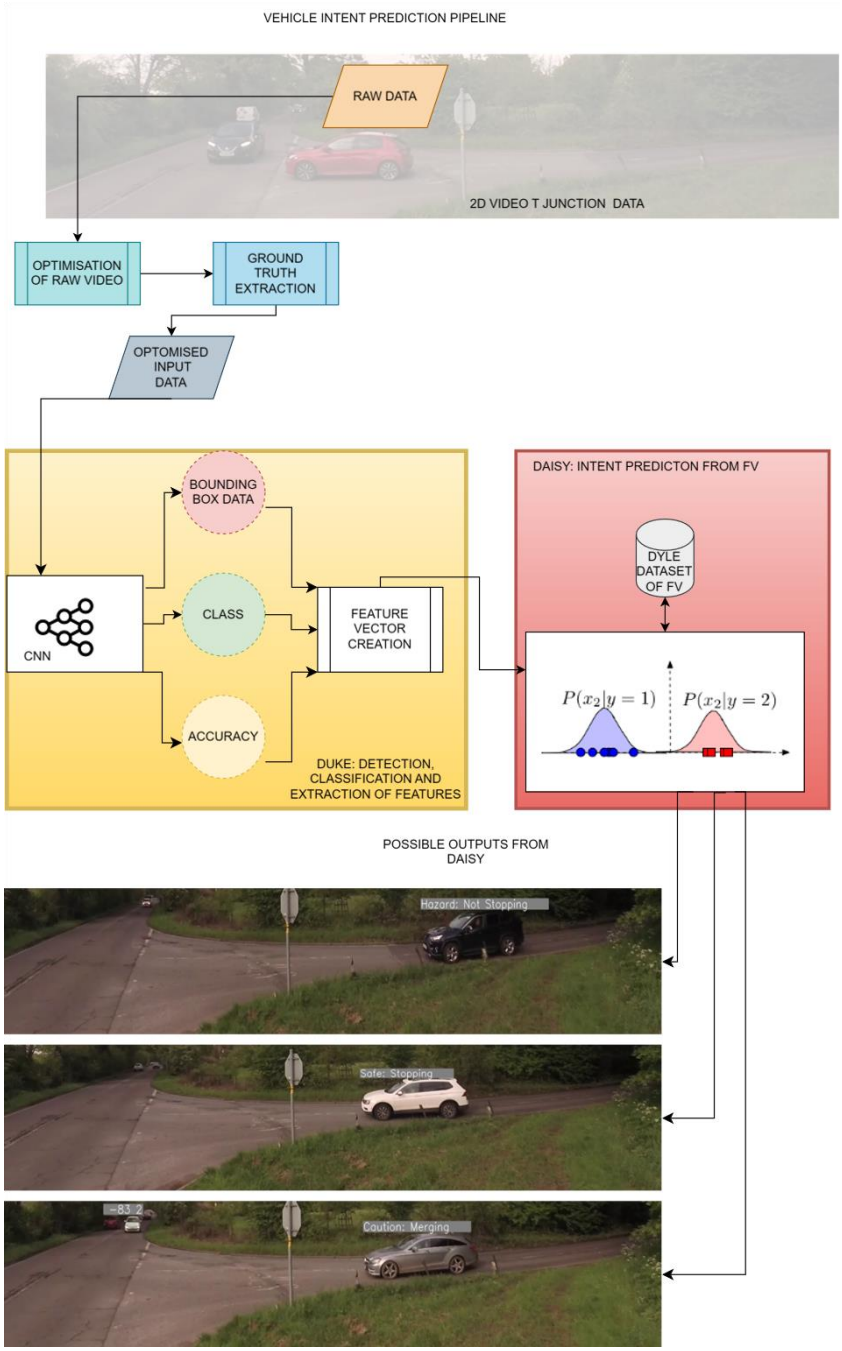



*Figure 76 illustrates the complete pipeline, encompassing the stages of video data capture and processing, the generation of DUKE feature vectors, the three possible predictions of vehicle intent by DAISY, and examples of possible outputs. Additionally, the diagram highlights the dynamic updating of DYLE post-DAISY prediction, representing a continuous and adaptive refinement process.*

This chapter focuses on the online phase, exploring how DAISY autonomously generates ground truth-validated feature vector training data, passed from DUKE and trained using the DYLE dataset.

This validated data then updates DYLE training data by incorporating accurately classified feature vector arrays. These arrays consist of $S_{Fpc}$ and $F_{pc}$ predictions. They are efficiently transmitted to DYLE within 45 ms from when the target vehicle is physically classified and integrated as classified feature vectors in real-time. This process enriches DAISY's model with new training data, enhancing its predictive capabilities for the subsequent intent of the following target vehicle.

Until now, we have conducted manual verifications for each target vehicle action at the merge line within our experimental junctions and retrospectively applied the corresponding $F_{pc}$ and $S_{Fpc}$ classification features to the training data. Despite being a labour-intensive approach, this method assured an association between the target vehicles' $S_{Fpc}$ feature vectors and the target vehicle's $F_{pc}$. The product of this work is DYLE, our dataset of classified feature vectors, which we verified and analysed in Chapters 7 and 8.

In our subsequent experiments, we understand the impact of using our novel sub-classification ($S_{Fpc}$) method on all associated target vehicle feature vectors. Without the $S_{Fpc}$ classification, our dataset has a better class frequency balance and the possibility of greater accuracy of intent prediction at the merge line. Before we put the pipeline online, in this chapter, we conduct a final ablation study by removing the sub-classifications from the junction updated DYLE dataset, described in section 8.6, to ascertain the effect of the subclasses on the training data.

**Research Question 6 (RQ6):** Can a trained machine learning model accurately predict vehicle intent at a T-Junction using new data, and what is its effective prediction range from the junction?

**Research Question 7 (RQ7):** Can our online model infer and append intent predictions as new inference data in real-time without negatively affecting the accuracy or F1 score?

Contribution:

A quantitative examination of how accurately DAISY, trained on progressively larger datasets, can predict driver intentions and determine the practical distance from the junction at which predictions remain viable. This exploration contributes to understanding the limits and capabilities of machine learning in the context of driver behaviour prediction at critical road intersections.

We create and evaluate an online model capable of inferring and appending new data in real-time while maintaining base accuracy and F1 score.

### 9.1.2 Chapter organisation

This chapter provides a comprehensive overview of various studies and analyses related to the classification and verification of feature vectors in autonomous systems. Section 9.1 introduces the chapter and outlines its organisation. In Section 9.2, an ablation study explores the sub-classification of feature vectors. Section 9.3 discusses the autonomous application of Fpc and Sfpc classification features, including a case study on autonomous merge line data recording (DUKE). Interactions with other vehicles are examined in Section 9.4.

The focus then shifts to online data verification and analysis in Section 9.5, which includes several subsections detailing specific aspects of online verification, accuracy, and comparison of F1 scores using the JM454 system. Section 9.6 extends this analysis to other systems - JM599, JM377, and JM384, focusing on online distance accuracy experiments and class distribution.

Section 9.7 addresses the balancing of training data, including the generation of new hazard class samples and the cross-validation of different DYLE models. Section 9.8 explores unseen data online with a new junction, UO196, examining class distribution, accuracy, and F1 scores and comparing results from various DYLE iterations. The chapter concludes in Section 9.9, summarising the findings and implications of the studies presented.

## 9.2 Ablation study, the effect of sub-classification of feature vectors

In light of the manual integration of subclasses into our training data, post-classification of target vehicles' behaviours at the merge line, and their absence in our initial analyses, it is imperative to assess their impact on the training dataset before the online deployment of our system. Our approach involves real-time writing of Fpc and Sfpc classifications, necessitating understanding their influence on the DAISY prediction model. It is crucial since, in an online setting, we can only control the training data, with verification and adjustments reserved for offline periods.

We undertook an ablation study using the combined DYLE dataset to gauge the effect of our sub-classification categories. Our first step involved removing all subclasses, resulting in feature vectors solely labelled with $F_{pc}$ ground truth classifications. Consequently, each feature vector was categorised as $F_{pc}$: Stop, Merge, or Hazard, devoid of any $S_{Fpc}$ classification, irrespective of the vehicle's distance from the merge line.

Following this, we replicated the analysis method outlined in section 7.4.6.1, applying K-fold cross-validation on the modified DYLE dataset. This procedure aimed to understand how the

removal of subclasses influences the model's predictive accuracy and to evaluate the necessity of these subclasses in enhancing the model's performance in real-world scenarios.

| K value | Mean Accuracy Updated DYLE $F_{pc}$ only |
|---|---|
| 5 | 0.66 |
| 10 | 0.69 |

*Table 42 Ablation study update DYLE K-fold cross-validation accuracy results removing $S_{Fpc}$* from the updated DYLE dataset.

| $F_{pc}$ | Count | Frequency |
|---|---|---|
| merge | 968 | 50.92% |
| stop | 913 | 48.03% |
| hazard | 20 | 1.05% |

*Table 43 Updated DYLE with Fpc class only. Count/Frequency*

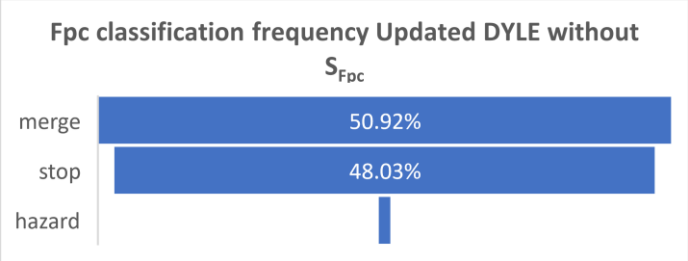



*Figure 77 Updated DYLE Class frequency*

The results shown in Table 44 indicate that DAISY's predictive performance on the training data has decreased following the removal of $S_{Fpc}$ classifications. This decline in predictive accuracy could be attributed to the diminished distinction in the dataset caused by the reduction in distance-based features. With removing these subclassifications, features such as distance, area, Px, and Py across different feature vectors now exhibit more similar values, leading to less distinct data characteristics.

Another important observation is that this abridged DYLE DAISY's prediction does not incorporate data from feature vectors manually classified as greater than 9% of the distance to the merge line. By removing the feature vectors recorded from 10% of the total distance to the merge line, the dataset no longer includes feature vectors with a distance feature that exceeds 9% of the distance from the merge line. While we can focus our dataset construction on only three classifications (Stop, Merge, or Hazard) and gather precise data to enhance predictions at the merge line, this approach presents a significant limitation. It restricts our predictive capacity to a maximum of 9% of the distance from the merge line, which translates to approximately 400 milliseconds before the final action at the merge line. This timeframe is

insufficient for our objective.

We aim to predict vehicle intent as far away from the merge line as possible. Therefore, while simplifying the dataset to three classifications might streamline the data and potentially improve accuracy at close distances, it would not align with our aim of early prediction. Early prediction is crucial for practical applications, such as advanced driver-assistance systems (ADAS), where timely alerts and decisions could significantly enhance road safety. Thus, while the ablation study provides valuable insights into the impact of subclassifications, it also underscores the necessity of a more comprehensive approach that includes a broader range of distances for effective prediction.

## 9.3 Autonomously applying Fpc and Sfpc classification features to feature vectors

In our existing methodology, we manually classify each feature vector array based on target vehicles' visible ground truth actions at the merge line, a process involving tracking each vehicle from initial detection to the merge line, recording an $F_{pc}$ and adding an $S_{Fpc}$ to each feature vector in the target vehicle array and adding it to DYLE.

Up to this point, DAISY utilises manually curated training data to make real-time predictions about a target vehicle's intent as it approaches the merge line. It is possible to use these predictions to classify each DUKE-derived feature vector with a specific DAISY-predicted $S_{Fpc}$ and an $F_{pc}$ and then integrate these classifications into DYLE in an autonomous and real-time manner. This process would incrementally enrich the dataset for subsequent analyses.

However, as outlined in previous chapters, DAISY's prediction accuracy currently stands at 83%, implying that approximately 17% of its predictions could be erroneous. Maintaining the integrity and accuracy of our training data is crucial, so we use only the verified ground truth actions, as determined by the DUKE system, for input into DYLE. This method helps preserve the quality and reliability of the training data that DAISY accesses, regardless of its current prediction accuracy.

### 9.3.1 DUKE: Autonomous Merge line data recording

Determining the $F_{pc}$ for a target vehicle approaching the merge line involves a threshold-based model as a function within DUKE. This model measures features such as the distance from the merge line and the vehicle's velocity and acceleration at the merge line and compares these values against predefined thresholds for stop, merge, and hazard actions. When the thresholds are met, DUKE triggers the $F_{pc}$ feature as the target vehicle nears the merge line. The $F_{pc}$, which could be either Stop, Merge, or Hazard, is assigned to the final feature vector in the current array representing the target vehicle. This classification is based on the vehicle's

performance against the set thresholds for the key features at the critical point of the merge line. Once the $F_{pc}$ is determined, the subsequent step involves classifying the other features within the array with $S_{Fpc}$. These $S_{Fpc}$ classifications are assigned based on their proximity to the merge line, providing a more detailed understanding of the vehicle's behaviour as it approaches the merge line. After these classifications are applied, the fully classified feature vector is appended to DYLE as individual classified feature vectors. This autonomous process enriches the training data, DYLE, for DAISY, refining and enhancing its real-time predictive capabilities for subsequent target vehicles.

The threshold model is an inline function described in the algorithm below; the implementation of this function is designed to replicate the manual classification of the feature vector process described in section 7.4.5 and create a fully autonomous system capable of generating and learning from new data in an unsupervised mode.

The threshold model follows these basic steps: checks if the vehicle is within 9% of the total distance to the merge line, categorises the vehicle's action as "Merge," "Hazard," or "Stop" based on its acceleration and velocity and labels the feature vectors associated with the vehicle as "Weak," "Moderate," or "Strong" based on the merge line action and distance from the merge line.

***Algorithm: Threshold_Model_Function***

*Input: distanceToMergeLine, totalDistanceToMergeLine, acceleration, velocity, thresholds (x, y, a,b)*

*Output: $F_{pc}$, Associated_ $S_{Fpc}$*

*Function:*

*1. Set thresholdDistance = 9% of totalDistanceToMergeLine*

*2. If distanceToMergeLine <= thresholdDistance, then*

- *a. If acceleration > x and velocity > x and (acceleration < y and velocity < y), then*
  - *i. $F_{pc}$ = "Merge"*
- *b. If acceleration > x and velocity > x and (acceleration > y or velocity > y), then*
  - *i. $F_{pc}$ = "Hazard"*
- *c. Else*
  - *i. $F_{pc}$ = "Stop"*

*3. If distanceToMergeLine is >= thresholdDistance(a), then*

- *a. $S_{Fpc}$ = "Strong"+ $F_{pc}$*

*Else, If distanceToMergeLine is >= thresholdDistance(b), then*

- *a. $S_{Fpc}$ = "Moderate" + $F_{pc}$*

*Else*

*a.* $S_{Fpc}$ *= "Weak" +* $F_{pc}$

*4. Return* $F_{pc}$*, Associated_* $S_{Fpc}$

Accurately categorising feature vector arrays is pivotal in enhancing DAISY's real-time forecast ability. Sampling the autonomous operation without direct supervision is essential, and this can be achieved by consistently reviewing video logs and juxtaposing the predictions with actual outcomes. Additionally, conducting K-fold cross-validation on the DYLE system is crucial to assess overfitting and generalisation ability, followed by fine-tuning the threshold model to ensure optimal performance.

## 9.4 Interactions with other vehicles

As highlighted in the literature review, analysing traffic behaviour at a junction necessitates considering more than just the road's topology. It's also crucial to consider the various road users who may impact traffic flow, including drivers, cyclists, pedestrians, and other road users in the traffic environment. Their behaviours, interactions, and movements significantly shape traffic dynamics and must be factored into any analytical model or traffic management strategy to ensure accuracy and effectiveness.

We collected a set of feature vectors that activated the 'approach_b' feature to investigate this. This feature is integrated into the feature vectors of target vehicles to indicate the presence of another vehicle approaching the T-junction from direction b on the major road. The 'approach_b' feature is paired with 'approach_dis_b', a metric indicating the distance of the approaching vehicle from the junction's merge line at each time step.

Figure 78 indicates target vehicles nearing the merge line and the measured distance ('app_b_dis') at which a vehicle from direction b is detected. This distance is represented in pixels from the merge line. The classifications are based on $S_{Fpc}$ classes, providing a snapshot of how approaching vehicles from direction b influences the behaviour of target vehicles at the junction.

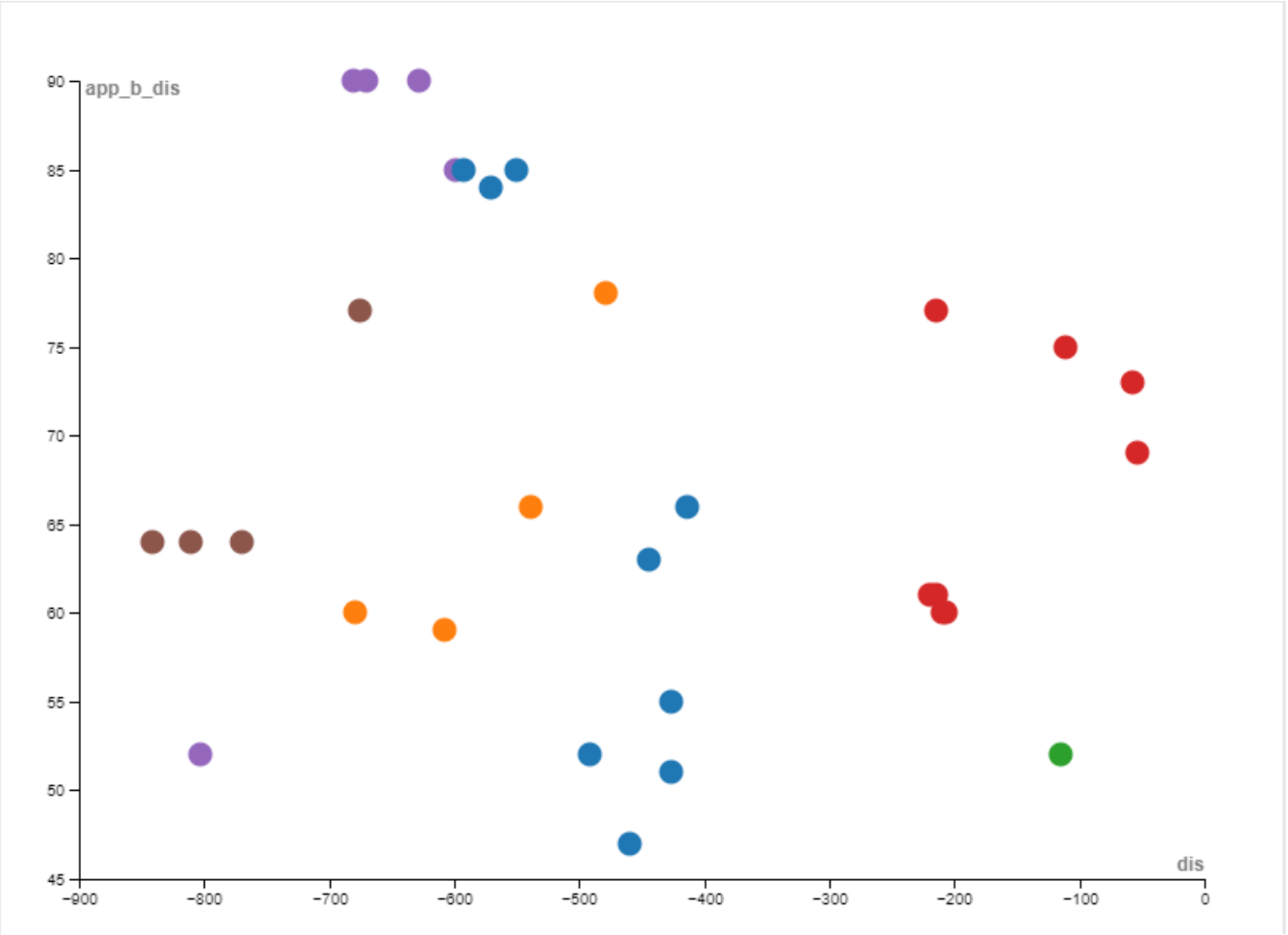



*Figure 78 $S_{Fpc}$ of feature vectors showing target vehicles approaching the merge line when a vehicle approaches from direction b at a distance from the merge line in pixels.*

Target vehicles nearing the merge line to enter the main road are impacted by traffic from direction b. Figure 75 shows data primarily classified as a 'strong stop' within approximately 220 pixels of the merge line. The predominant stop classification is likely due to patterns in the training data, where vehicles with a positive approach_b feature indicating a vehicle on the main road typically have stop class when within a set distance range. By incorporating the 'approach_b' feature, a clear pattern emerges in the feature vector: the presence of an approaching vehicle often leads to the classification of the target vehicle as more likely to stop. While our current dataset is not extensive enough to verify this trend conclusively, we believe that the 'approach_b' feature is a valuable enhancement. As DYLE is enriched with additional feature vectors, this feature is expected to improve the accuracy of intent predictions.

## 9.5 Online data Verification and analysis

In the following experiments, we utilised the remaining 20% of the video data set aside for online verification, as Table 21, Section 7.3 outlined. The complete pipeline is now fully active; we enabled DYLE to incorporate new autonomously classified feature vectors added by DUKE, as detailed in Section 9.3.1.

As we verify each junction, the size of the online DYLE increases with the autonomous addition of newly classified feature vectors. We change the order of the test junction experimentation as new data is appended to DYLE; this way, we can balance out bias, which may be present if we append data from the same junction in the same order during each batch of experiments. We use this method as it may help to alleviate issues since the first junction will not be training from the same amount of data as the last junction; as we append during each experiment, the quantity of training data available increases. After completing the Verification of each Junction, we obtain two key sets of data: a video log of the junction during the verification period and an updated DYLE dataset enriched with newly classified data. Additionally, we gather insights on the predictions made by DAISY during this process based on final intent predictions and ground truth actions at the merge line.

Following the online Verification of each Junction, we analyse DYLE. We examine the newly added feature vectors for class distribution and employ K-fold cross-validation to assess the training data's accuracy changes. We also validate the autonomous classifications made by DUKE, using the video logs and vehicle IDs to link and evaluate the accuracy of the corresponding feature vectors.

### 9.5.1 Single Junction Online Verification JM454

Junction JM454 is a T-junction on the busy rural Junction A29 in southeast England. At this junction, we have a short distance from the initial detection to the merge line.

The junction-specific video log was used to verify incremental predictions made by DAISY starting from the first detection, as shown in Figure 76, where there is a time series of video screenshots. The initial intent prediction DAISY output is  Stop. This is the merge line intent prediction that DAISY predicts vehicle (id=77) will take when it gets to the merge line—generated from processing $S_{Fpc}$ in the method described in section 8.3. As the target vehicle approached the merge line, multiple predictions were made based on the associated feature vectors generated by DUKE. Ground truth verification confirmed that the target vehicle picture in Figure 79 stopped at the merge line as predicted. DUKE updated DYLE with the target vehicles' classified feature vector array. In the top image in Figure 76, a vehicle is detected approaching from direction b, classified as a car (2) and 41 pixels from the junction entrance merge line. This feature may assist in the accurate early predictions of target vehicle intent, as discussed above. However, we require more junction data with approach_b  features to confirm this.

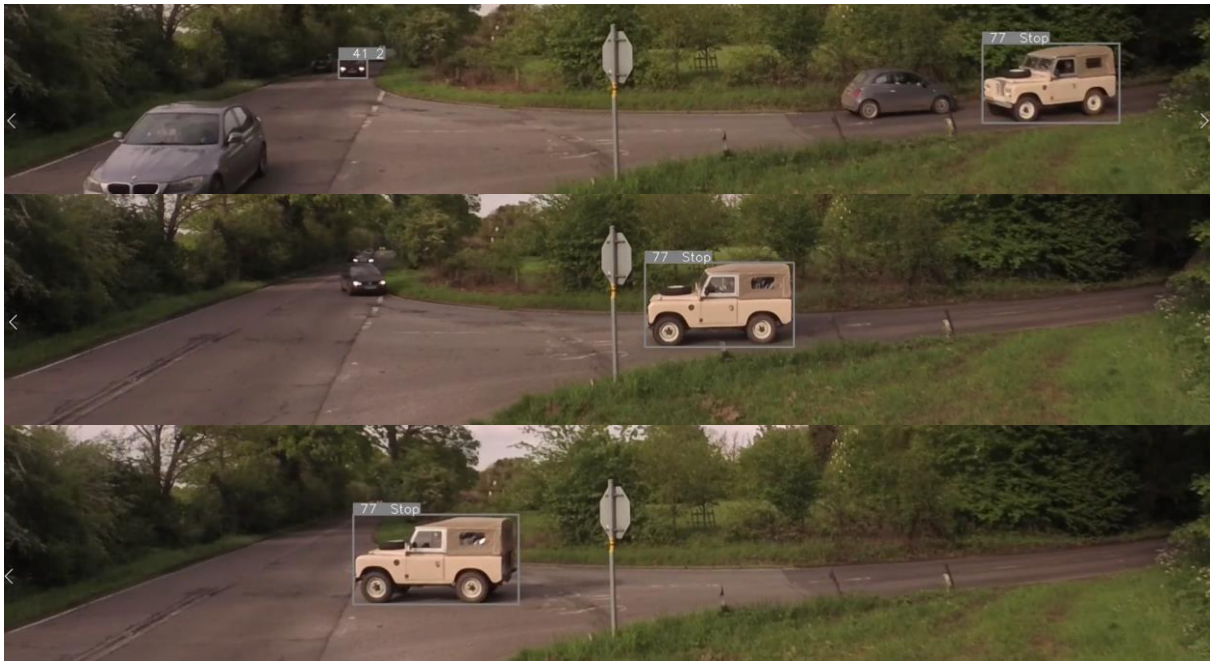



*Figure 79 is a chronological series of video screenshots that capture the intent prediction of a target vehicle from initial detection. As the target vehicle (id=77) approaches the merge line, DAISY outputs further predictions and a final prediction at the merge line. In this case, the predictions were accurate with ground truth validation.*

The following example is of a Merge prediction; in Figure 80, a time series screenshot from

our video shows that DAISY's initial prediction is for the target vehicle (id=47) to Stop at the merge line; ground truth verification showed that vehicle 47 merge onto the major road without stopping. This is an example of initial prediction inaccuracies due to the method we use to classify the $S_{Fpc}$ generated by DAISY, where we have very few examples for the initial prediction classification. However, as more features are generated and are sub-classified, the accuracy improves towards the merge line.

Even though the initial error in Daisy's prediction DUKE updated DYLE with vehicle 47 feature vector array classified as $F_{pc}$ Merge and the associated $S_{Fpc}$ subclassification applied to the associated feature vectors in the array.

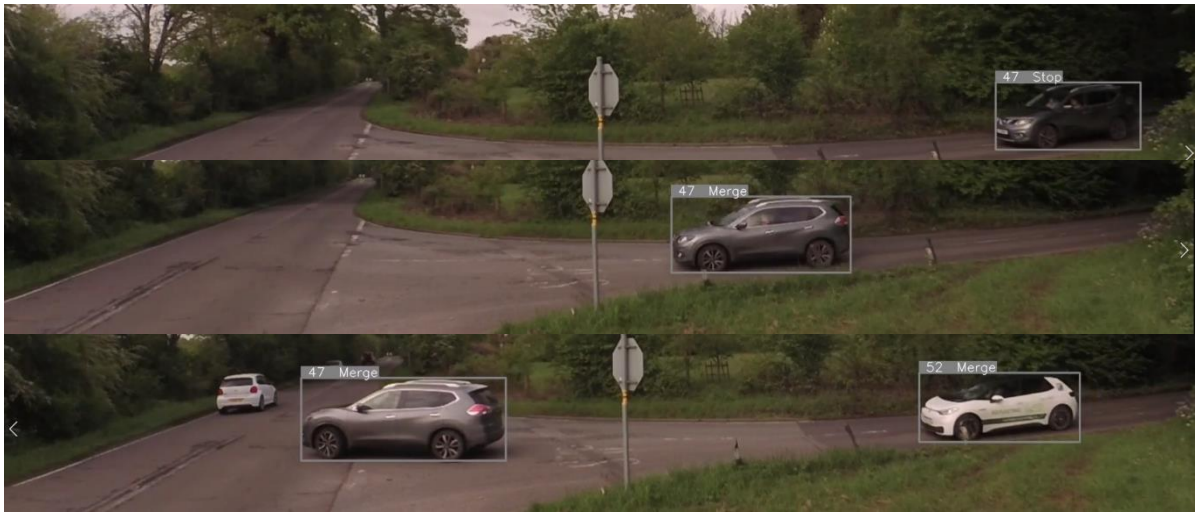



*Figure 80 The initial intent prediction of vehicle 47 made by DAISY is incorrect regarding the target vehicle 47. Ground truth confirmation showed that vehicle 47 merged without stopping as predicted.*

Prediction of a hazard class intent is more difficult due to the imbalance in the dataset; there are very few examples of hazard-classified feature vectors in the training data; this may well be an advantage because it not only reduces the chance of a dangerous false hazard prediction, it a; so means that any feature vector that falls outside of the probability of being a stop or merges will be classified as a hazard, catering for data that has never been seen for example an out of control vehicle travelling erratically towards the merge line.

However, the example shown in the time series video screenshots in Figure 78 shows that DAISY can correctly predict a Hazard intent class from around 50% of the distance to the merge line. The vehicle (id=265) in Figure 81 entered the major road carelessly. The ground truth video log showed that vehicle 256 entered in front of a fast-moving vehicle from direction b, causing it to slow down. DUKE updated DYLE with the hazard-classified feature vectors

from vehicle 256 feature vector array, instantly adding to the training data.

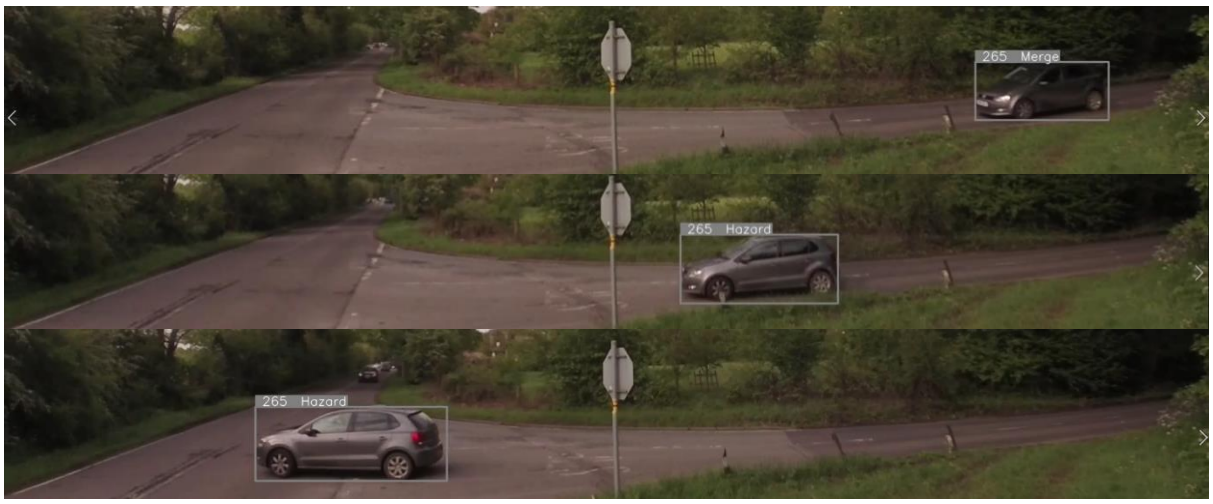



Figure 81 Despite vehicle 265 being incorrectly classified initially, the correct final classification was predicted at around 50% of the distance to the merge line.

### 9.5.2 Updaing DYLE with autonomously classified feature vectors from JM454

In the post-video analysis of the ground truth data, we verified the new data in DYLE and then measured accuracy using our K-fold method. Table 44 lists the accuracy results for the Online DYLE dataset, a combination of the updated DYLE from 8.6.4, Table 41, and autonomously classified feature vectors from JM454.

| K value | Mean accuracy Online DYLE |
|---|---|
| 5 | 0.79 |
| 10 | 0.82 |

*Table 44 Online DYLE K-fold cross-validation accuracy results with Junction JM454 feature vectors autonomously classified and appended.*

The accuracy of our model shows no improvement when setting k to 5 and only a marginal improvement of 0.01 when k is increased to 10. This outcome aligns with our expectations, as the expansion of our dataset has not been as substantial as in earlier updates, where there

was a significant augmentation in the count of classified feature vectors. Notably, the addition of classifications made autonomously has not altered the fundamental characteristics of the dataset.

### 9.5.3 Verification of autonomous online intent predictions JM454

During the above experiments, DUKE created classified feature vectors. Before experimenting with the remaining data from our test T-junctions, we cross-referenced the merge line action classifications made autonomously by DUKE with the video log of the experiment on JM454. As discussed in 9.5.2, the addition of autonomous classification did not impact the k-fold cross-validation accuracy of DYLE; however, as more feature vectors are appended, any miss classification errors will quickly degrade the accuracy of DAISY predictions.

When we reviewed the video log, we found that DUKE correctly classified all the Merge actions at the merge line; however, we had to correct four Stop predictions and alter the classification to Merge. We then adjusted the threshold values (x,y), discussed in 9.3.1, to account for this in the subsequent experiments. So far, our method for assessing ground truth accuracy has involved using $F_{pc}$ at the merge line. By comparing final predictions with ground truth data, we establish a reliable metric for accuracy in predicting vehicle intent at the merge line. We aim to predict this intent as early as possible and as far from the merge line as possible. This approach aims to maximise the available time for generating alerts when necessary.

### 9.5.4 Accuracy and distance from merge line junction JM454

In our analysis of Junction JM454, we utilised the video log to track DAISY's initial prediction when the target vehicle was first detected and then monitored its subsequent predictions up to the merge line. Following the methodology outlined in section 7.4.2, we defined specific distance ranges and correlated these with subclass associations relative to the vehicle's proximity to the merge line. We recorded DAISY's predictions within these predetermined ranges and compared them to the ground truth ascertained from the video log. This approach allowed us to evaluate the accuracy of DAISY's predictions in relation to how far the prediction is made from the merge line.

| Dis_Range | >= 70% | | | >= 40% and <=69% | | | >= 10% and <=39% | | | <= 9% | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Class** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Qty** |
| **Hazard** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1 |
| **Merge** | 0.59 | 0.79 | 0.67 | 0.66 | 0.84 | 0.74 | 0.75 | 0.91 | 0.82 | 0.74 | 0.91 | 0.82 | 104 |
| **Stop** | 0.42 | 0.22 | 0.29 | 0.63 | 0.40 | 0.49 | 0.82 | 0.56 | 0.66 | 0.81 | 0.53 | 0.64 | 72 |
| | | | | | | | | | | | | | |
| **Accuracy** | | | 0.55 | | | 0.66 | | | 0.77 | | | 0.76 | 177 |
| **Macro Avg** | 0.34 | 0.34 | 0.32 | 0.43 | 0.41 | 0.41 | 0.85 | 0.82 | 0.83 | 0.85 | 0.81 | 0.82 | 177 |
| **W-Avg** | 0.52 | 0.55 | 0.51 | 0.65 | 0.66 | 0.63 | 0.78 | 0.77 | 0.76 | 0.77 | 0.76 | 0.74 | 177 |

*Table 45 JM454 Accuracy metrics based on intent predictions made from a distance range from the total distance to the merge line.*

**Key:**

Pr (Precision): The ratio of correctly predicted positive observations to the total predicted positives.

Re (Recall): The ratio of correctly predicted positive observations to all observations in the actual class.

The F1 Score is the weighted average of Precision and Recall.

Qty: Number of actual instances in each class.

Macro vs. Weighted Averages: The Macro Average is consistently lower than the Weighted Average, suggesting that the model performs better on classes with more instances.

Summary of the results of distance and prediction accuracy from Table 45.

The results erroneously show excellent performance for 'Hazard', as this is based on a single result. The Performance for 'Merge' is good, but for 'Stop', there are poor results, especially in recall. The overall accuracy and averages indicate a well-performing model, but the 'Stop' class, having a lower recall, suggests a potential area for model refinement. Since the 'Hazard' class has only one instance, we require more data for this class to ensure that the model's performance is genuinely robust and not a result of limited exposure.

The class 'Hazard' is still problematic due to the lack of instances (Qty=1). The overall model accuracy and weighted average F1 score are the highest and closest to the merge line, which might indicate a tendency of the model to be more conservative in class predictions.

### 9.5.5 Comparison of F1 scores and accuracy for given distance ranges JM454

We collated the F1 scores and accuracy metric from the individual distance range experiments above and compared them, as seen in Figure 82. The data shows that the F1 score for the

Stop class increases as the distance from the merge line decreases. This suggests that predictions become more accurate for stopping vehicles closer to the merge line. The Merge class maintains a relatively high F1 score across all distance ranges, slightly increasing as the distance from the merge line decreases. This indicates consistent prediction accuracy for merging vehicles, with a marginal improvement at closer ranges. A single Hazard class data point cannot be analysed for a trend without more Hazard classified feature vectors.

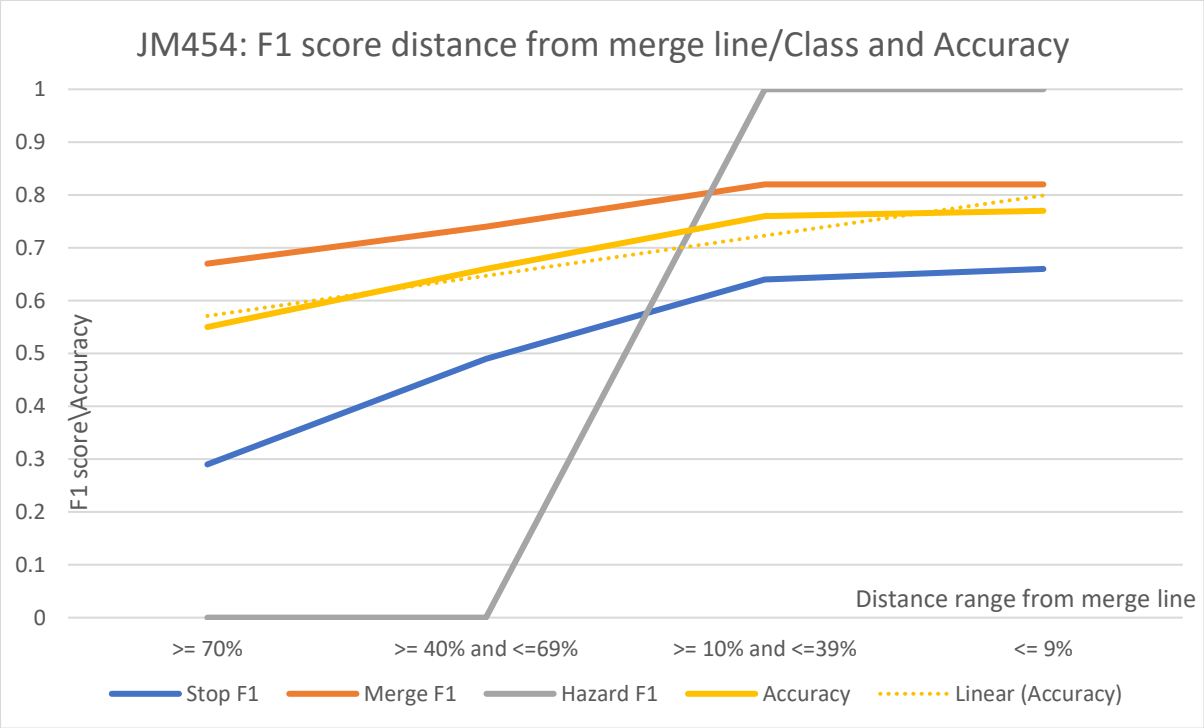



*Figure 82 F1 score and accuracy based on class and distance from merge line JM454*

The average F1 score across classes remains relatively stable across different distances, with a slight upward trend as the vehicle approaches the merge line. The linear trend indicates a modest increase in overall predictive performance as the distance to the merge line decreases. The model's performance in predicting stopping and merging behaviour improves as vehicles approach the merge line, possibly due to more apparent feature vectors representing the vehicle behaviours or increased data quality at closer distances.

Since the 'Hazard' class lacks multiple data points, no trend can be established, which also means that the model's performance for hazard prediction is not well-represented in this graph.

The overall stability of the accuracy score suggests that the model performs reasonably well across all distances but is slightly better from 49% of the total distance to the merge line.

## 9.6 Online distance accuracy experiments for JM599, JM377 and JM384

We have evaluated the precision of predicting vehicle intentions based on their proximity to the merge line at a single test junction. Employing the techniques described in Section 9.5, we extended this analysis to three additional test T-junctions. We aimed to assess each junction independently and then aggregate the data from all four junctions to assess the system mean accuracy at each given distance range. Data is autonomously and incrementally added to the online DYLE database during each junction experiment, sampled and verified to correct misclassifications.

We utilise video logs to validate the autonomous predictions of vehicle intentions, from the initial detection of the target vehicle to its eventual action at the merge line. This process involves manually correcting misclassifications in DYLE and fine-tuning threshold settings as needed. After these adjustments, we calculate the accuracy of the revised online DYLE database using a k-fold cross-validation method. We subsequently verified the ground truth actions against the predicted and at each junction using the distance ranges we established above and used benchmark metrics to compare.

### 9.6.1 JM599 Online verification and distance from merge line accuracy

Once we verified the ground truth actions with the DAISY predictions using the video log of JM599 verification partition video data and corrected the misclassifications, we carried out a k-fold cross-validation using the now updated online DYLE, as seen in Table 46.

| K value | Mean accuracy Online DYLE |
|---|---|
| 5 | 0.79 |
| 10 | 0.83 |

*Table 46 Online DYLE K-fold cross-validation accuracy results with Junction JM599 feature vectors autonomously classified and appended*

The online appending of autonomously classified feature vector data using JM599 video to DYLE has shown a slight increase in k-fold cross-validation accuracy when k = 10 and no improvement when k = 5, as seen in Table 46. Accuracy and F1 scores are also improved, as seen in Table 47, where we have tabulated the metrics of predictions made by DAISY from each distance range.

| Dis_Range | >= 70% | | | >= 40% and <=69% | | | >= 10% and <=39% | | | <= 9% | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Class** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Qty** |
| **Hazard** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 2 |
| **Merge** | 0.65 | 0.69 | 0.67 | 0.73 | 0.78 | 0.75 | 0.79 | 0.84 | 0.82 | 0.85 | 0.94 | 0.89 | 103 |
| **Stop** | 0.52 | 0.48 | 0.50 | 0.67 | 0.62 | 0.64 | 0.77 | 0.70 | 0.73 | 0.92 | 0.78 | 0.84 | 73 |
| | | | | | | | | | | | | | |
| **Accuracy** | | | 0.60 | | | 0.70 | | | 0.78 | | | 0.87 | 178 |
| **Macro Avg** | 0.39 | 0.39 | 0.39 | 0.47 | 0.46 | 0.46 | 0.69 | 0.68 | 0.68 | 0.76 | 0.74 | 0.75 | 178 |
| **W-Avg** | 0.59 | 0.60 | 0.59 | 0.70 | 0.70 | 0.70 | 0.78 | 0.78 | 0.78 | 0.88 | 0.87 | 0.87 | 178 |

*Table 47 JM599 Accuracy metrics based on intent predictions made from a distance range from the total distance to the merge line.*

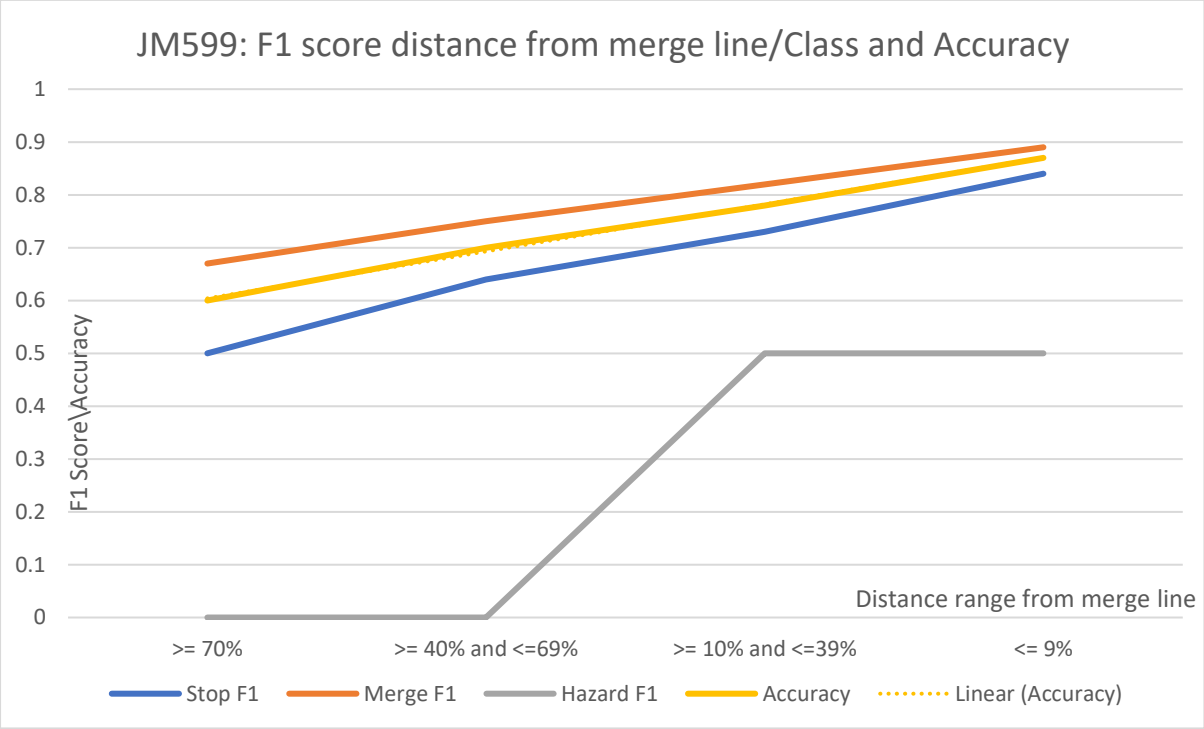



*Figure 83. F1 score and accuracy based on class and distance from merge line for JM599*

Initial discussion from the results of JM599: The Merge class consistently has the highest F1 scores across all distance ranges, indicating that the model is most accurate for this class. Accuracy increases as the distance to the merge line decreases, suggesting that the model's overall performance improves with a more inclusive threshold. The linear trend line of the average F1 scores is almost flat the further from the merge line and begins to ascend as we move towards the merge line, suggesting a relationship between the inclusivity of the distance ranges, sub-classifications ($S_{Fpc}$) and the model's performance.

### 9.6.2 JM384 Online verification and distance from merge line accuracy

Once we verified the ground truth actions with the DAISY predictions using the video log of JM384 verification partition video data and corrected the misclassifications, we carried out a k-fold cross-validation using the now updated online DYLE, as seen in Table 48 and compared this to the results from the previous iteration of Online DYLE from junction JM599. We saw a slight increase from 0.79 to 0.81 when k = 5 and no change when k = 10. An accuracy increase when using a smaller k value may be due to reduced variance in model evaluation; with a smaller k, each fold comprises a larger portion of the dataset. As a result, the variance in the evaluation metric across different folds may decrease, potentially providing a more stable estimate of model performance. However, using fewer folds can also mean that each training dataset is smaller, potentially leading to a higher bias in the model training process because the model is trained on a less diverse data set in each iteration.

| K value | Mean accuracy Online DYLE |
|---|---|
| 5 | 0.81 |
| 10 | 0.83 |

*Table 48 Online DYLE K-fold cross-validation accuracy results with Junction JM384 feature vectors autonomously classified and appended*

Accuracy and F1 scores are shown below for JM384, as seen in Table 49, where we have tabulated the metrics of predictions made by DAISY from each distance range.

| Dis_Range | >= 70% | | | >= 40% and <=69% | | | >= 10% and <=39% | | | <= 9% | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Class** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Qty** |
| **Hazard** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **Merge** | 0.48 | 0.55 | 0.51 | 0.60 | 0.66 | 0.62 | 0.73 | 0.71 | 0.72 | 0.74 | 0.82 | 0.78 | 38 |
| **Stop** | 0.71 | 0.63 | 0.67 | 0.79 | 0.73 | 0.76 | 0.84 | 0.84 | 0.84 | 0.90 | 0.83 | 0.86 | 63 |
| | | | | | | | | | | | | | |
| **Accuracy** | | | 0.60 | | | 0.70 | | | 0.79 | | | 0.82 | 101 |
| **Macro Avg** | 0.40 | 0.40 | 0.39 | 0.46 | 0.46 | 0.46 | 0.52 | 0.52 | 0.52 | 0.54 | 0.55 | 0.54 | 101 |
| **W-Avg** | 0.63 | 0.60 | 0.61 | 0.72 | 0.70 | 0.71 | 0.80 | 0.79 | 0.80 | 0.84 | 0.82 | 0.83 | 101 |

*Table 49 JM384 Accuracy metrics based on intent predictions made from a distance range from the total distance to the merge line. There is a clear trend of improving performance as the distance range decreases. The Stop class is predicted with relatively high accuracy,*

*especially in the <= 9% range. The overall performance improves in lower ranges, similar to the previous junction analysis. The quantity of data per class appears to impact model performance, as seen with the Stop class.*

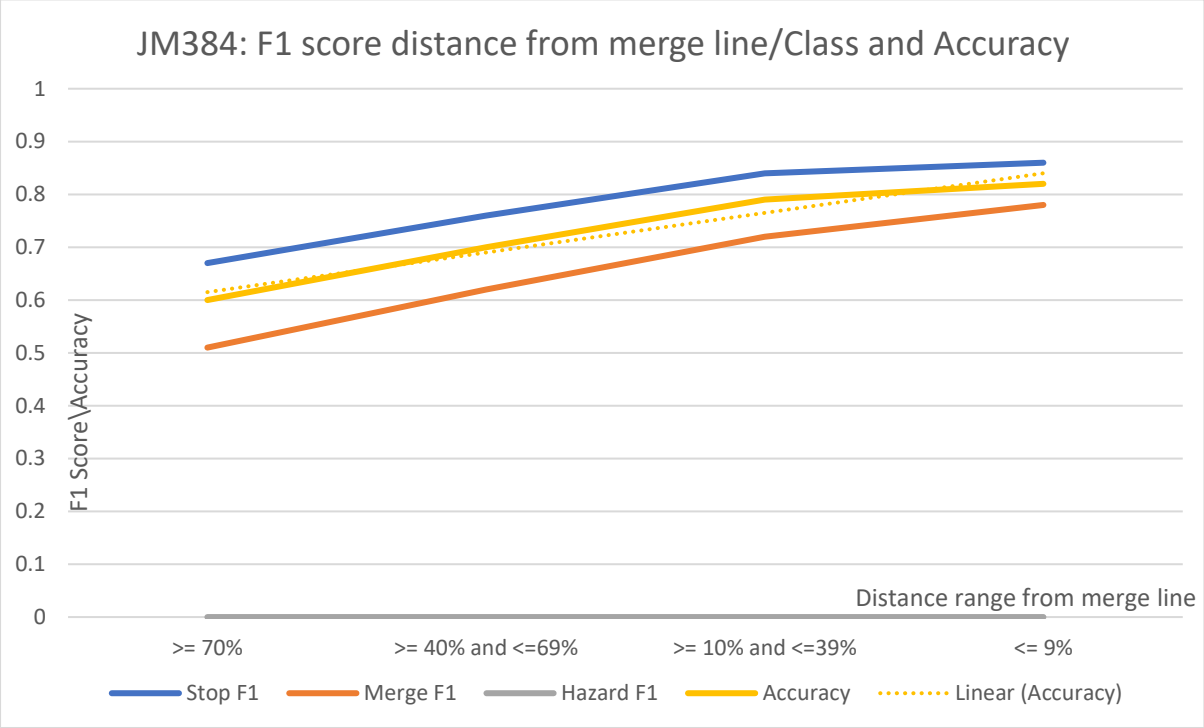



*Figure 84 F1 score and accuracy based on class and distance from merge line JM384*

Initial discussion of the results from JM384: The yellow dashed line shows the overall accuracy trend of the model. The line is relatively straight, suggesting a consistent improvement in accuracy across distance ranges and a proportional increase in accuracy as the distance from the merge line decreases. DAISY performs better for Stop and Merge classes when closer to the merge line, but there is no data to analyse for the 'Hazard' class. The accuracy improves at closer distances, as seen in the other junctions.

### 9.6.3 JM377 Online verification and distance from merge line accuracy

Once we verified the ground truth actions with the DAISY predictions using the video log of JM377 verification partition video data and corrected the misclassifications, we carried out a k-fold cross-validation using the now fully updated online DYLE, as seen in Table 50 and compared this to the results from the updated DYLE.

| K value | Mean accuracy Online DYLE |
|---|---|
| 5 | 0.83 |
| 10 | 0.85 |

*Table 50 Online DYLE K-fold cross-validation accuracy results with Junction JM377 feature vectors autonomously classified and appended*

Initial result discussion based on Table 50.

From 0.81 to 0.83 with K=5 could result from the specific way the data gets split in the 5 folds, which might result in each fold being a good representation of the overall dataset; hence, when combined, they result in a better model. Ongoing refinement of the threshold model parameters or adding more representative features by adding new data contribute to improved accuracy. The increase in accuracy from 0.83 to 0.85 with k = 10 suggests that there may be reduced variance; more folds mean that each test is less variable and potentially less biased towards any particular subset of data, making the estimated accuracy more reliable.

The model is trained on 90% of the data each time, which might prevent overfitting compared to training on 80% (as with K=5). Also, more folds can lead to more diverse training and validation sets, which may help the model learn a more general pattern.

JM377 Accuracy and F1 scores, as seen in Table 51 and Figure 85, where we have tabulated the metrics of predictions made by DAISY from each distance range.

| Dis_Range | >= 70% | | | >= 40% and <=69% | | | >= 10% and <=39% | | | <= 9% | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Class** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Qty** |
| **Hazard** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.75 | 1.0 | 0.86 | 0.67 | 0.67 | 0.67 | 3 |
| **Merge** | 0.58 | 0.58 | 0.58 | 0.69 | 0.70 | 0.70 | 0.68 | 0.78 | 0.72 | 0.76 | 0.84 | 0.79 | 67 |
| **Stop** | 0.71 | 0.72 | 0.71 | 0.80 | 0.80 | 0.80 | 0.82 | 0.72 | 0.76 | 0.88 | 0.81 | 0.84 | 88 |
| | | | | | | | | | | | | | |
| **Accuracy** | | | 0.65 | | | 0.74 | | | 0.75 | | | 0.82 | 158 |
| **Macro Avg** | 0.43 | 0.43 | 0.43 | 0.50 | 0.50 | 0.50 | 0.75 | 0.83 | 0.78 | 0.77 | 0.77 | 0.77 | 158 |
| **W-Avg** | 0.64 | 0.65 | 0.64 | 0.74 | 0.74 | 0.74 | 0.76 | 0.75 | 0.75 | 0.82 | 0.82 | 0.82 | 158 |

*Table 51 JM377 Accuracy metrics based on intent predictions made from a distance range from the total distance to the merge line.*

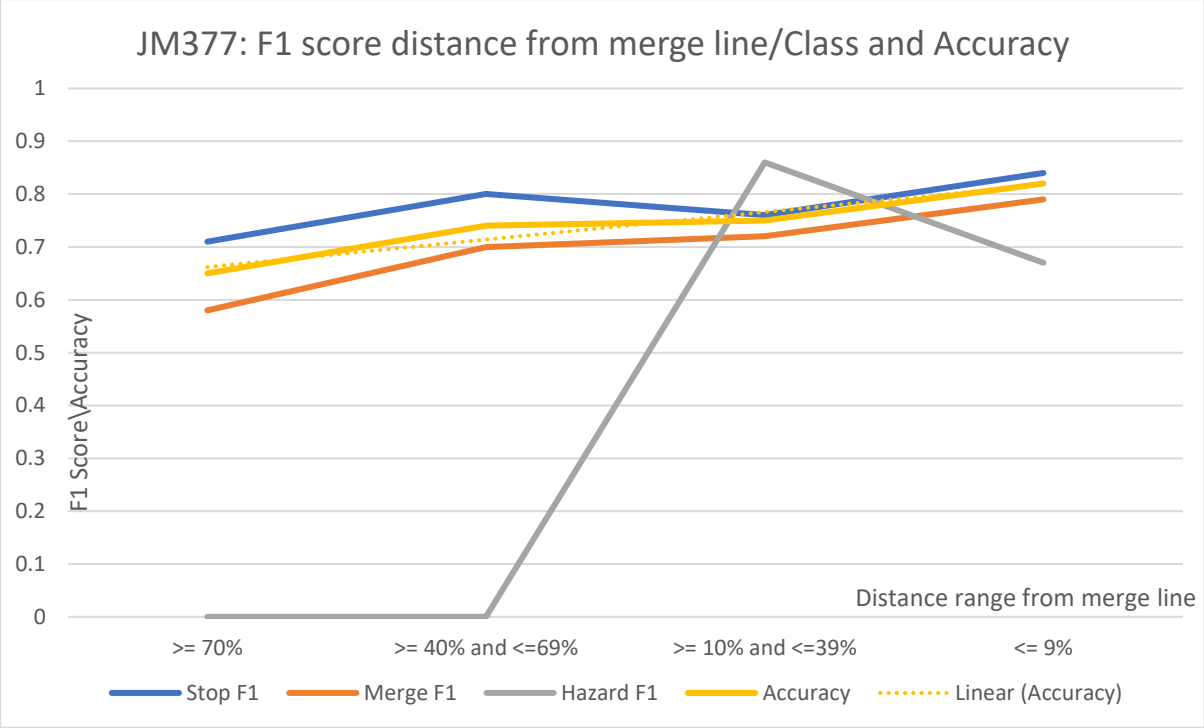



*Figure 85 F1 score and accuracy based on class and distance from merge line JM377*

DAISY struggles with accurate predictions in the range (>= 10% and <= 39%) where the accuracy dips. JM377 is a very busy junction, with traffic held close to the merge line waiting

to join the major road. The range that DAISY struggles in is where traffic backs up, causing overlapping or ambiguous cues that could indicate multiple intents. A method of alleviating this is to pause inference when traffic is stopped, moving very slowly or backed up.

### 9.6.4 Online DYLE class distribution

We have now appended the previously updated DYLE from 8.6.1 with the feature vectors from the online junction experiments to create an online DYLE constituting 99,851 feature vectors, equating to 2,515 individually identified target vehicles. Table 52 and Figure 86 show the class distributions and highlight the lack of Hazard classified feature vectors.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 21,420 | 21.45% |
| w_merge | 16,056 | 16.08% |
| w_stop | 15,671 | 15.69% |
| s_merge | 16,194 | 16.22% |
| m_stop | 14,170 | 14.19% |
| m_merge | 13,751 | 13.77% |
| merge | 1280 | 1.28% |
| stop | 1209 | 1.21% |
| w_hazard | 24 | 0.02% |
| m_hazard | 28 | 0.03% |
| s_hazard | 22 | 0.02% |
| hazard | 26 | 0.03% |

*Table 52 Online DYLE.*

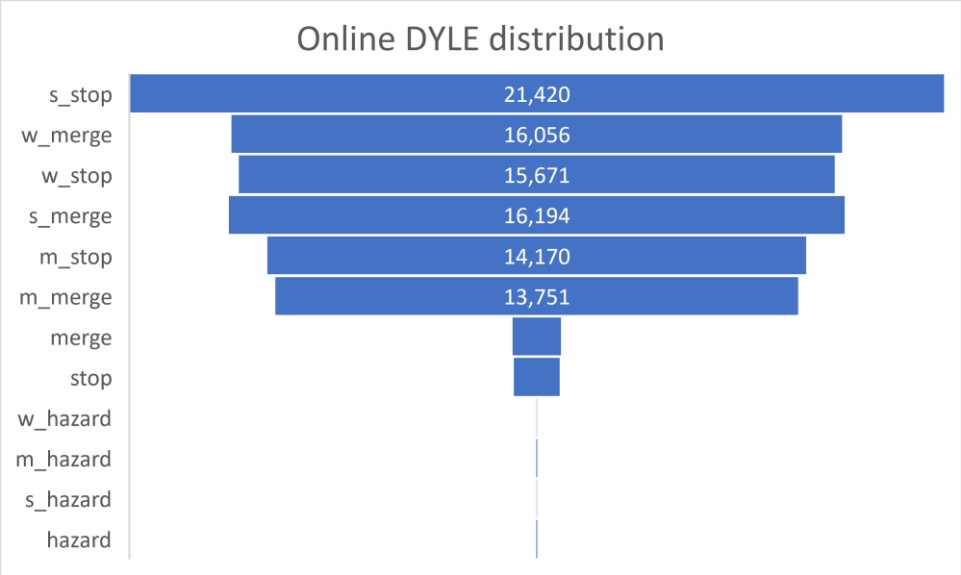



*Figure 86 Online DYLE class frequency count*

### 9.6.5 Analysis of the resulting metrics from the online Verification video data

We now have a comprehensive mixed junction dataset called online DYLE, consisting of feature vectors from the initial 60% of the video data covering all junctions, designated as the training subset. Subsequently, we incorporated a 20% segment for manual testing, followed by the final 20% segment of video data integrated autonomously into the online pipeline. We sampled and rectified any incorrect classifications before proceeding with a K-fold cross-validation. Table 53 presents updated metrics for each junction and the newly compiled online DYLE dataset; these updates are depicted in Figure 87.

| Metric | | $F_{pc}$ F1-Score (<=9%) | | | Accuracy | K-fold (10) |
|---|---|---|---|---|---|---|
| Junction | Class | Hazard | Merge | Stop | | |
| Online DYLE + Combined* | | 0.54 | 0.82 | 0.80 | 0.82 | 0.85 |
| Updated DYLE + Combined* | | 0.67 | 0.83 | 0.84 | 0.83 | 0.81 |
| JM377 | | 0.67 | 0.79 | 0.84 | 0.82 | 0.85 |
| JM384 | | 0.0 | 0.78 | 0.86 | 0.82 | 0.83 |
| JM599 | | 0.5 | 0.89 | 0.84 | 0.87 | 0.83 |
| JM454 | | 1.0 | 0.82 | 0.64 | 0.76 | 0.82 |
| Aggregated Training DYLE | | | | | | 0.79 |

**Combined = JM599, JM384, JM377 and JM454*

*Table 53 Comparison of $F_{pc}$ of single junction metrics with previous DYLE iterations. Online DYLE has the final 20% video verification data feature vectors appended.*

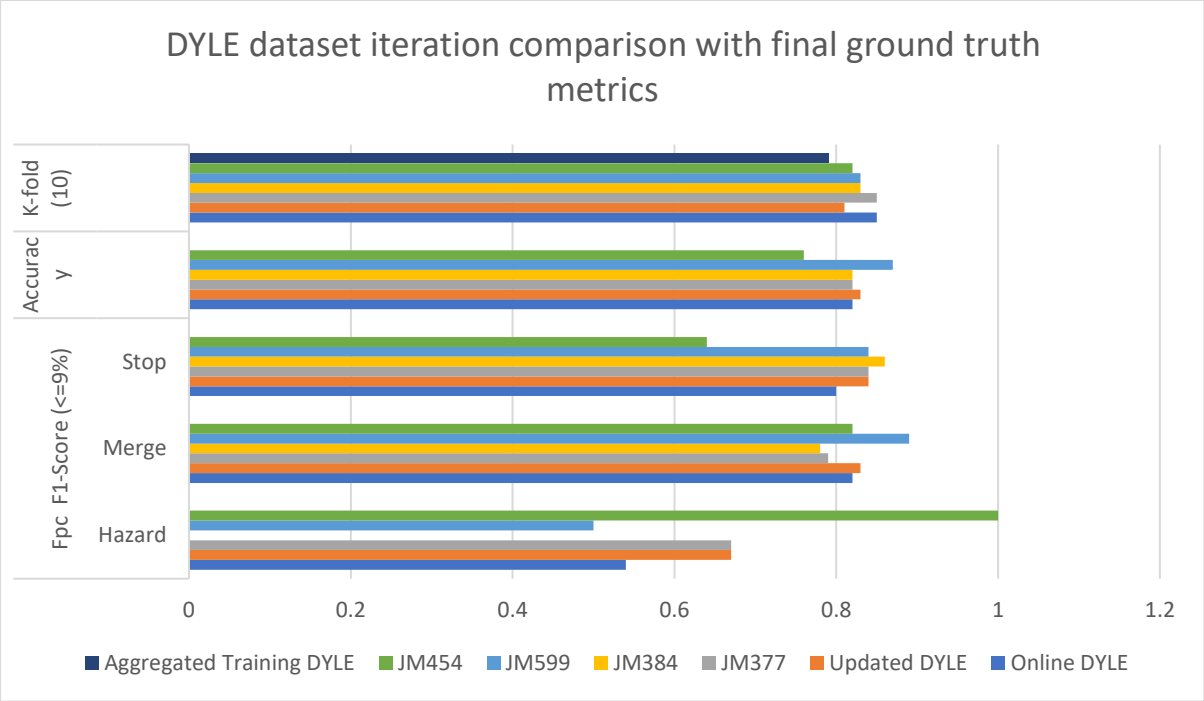



*Figure 87 K-fold training accuracy compared to ground truth F1-Score per class and overall accuracy.*

Table 53 shows the $F_{pc}$ against the actual ground truth actions for each junction, with the details found in Sections 9.5.8 and 9.6.1, .2 and.3 The initial training dataset, Aggregated DYLE, was established in Section 7.5, while the Updated DYLE is detailed in Section 8.6.1. Although the Online DYLE is not performing as well as the Updated DYLE iteration, it's important to note that the latest data added to the Online DYLE was incorporated autonomously, and samples were manually verified post-integration which demonstrates a

positive outcome for the online pipeline in the future. Isolated examination of the Online DYLE results reveals a promising trend, with relatively good accuracy and a good balance across F1 scores for all classes.

Figure 88 is a comparison of the latest iteration of DYLE against the previous Updated DYLE, and it highlights the increase in the k-fold score; all junctions demonstrate steady performance in cross-validation, especially for JM377 and JM384, which have the highest k-fold scores, suggesting their performance is consistent across various data subsets. Accuracy measures the proportion of true results (both true positives and true negatives) among the total number of cases examined, and JM599 has the highest accuracy, suggesting that DAISY accurately classifies a high percentage of instances at this junction. JM454 has the lowest accuracy, which may indicate higher misclassifications overall. As more data is needed, we could not thoroughly evaluate the Hazard class's accuracy and determine its impact on overall performance. Given the live traffic setting of this study, instances of behaviour that DAISY would categorise as a Hazard are scarce. The training data comprises approximately 1% of the total feature vectors identified as Hazard. This scarcity of Hazard-classified examples presents a significant challenge in conducting a comprehensive analysis.

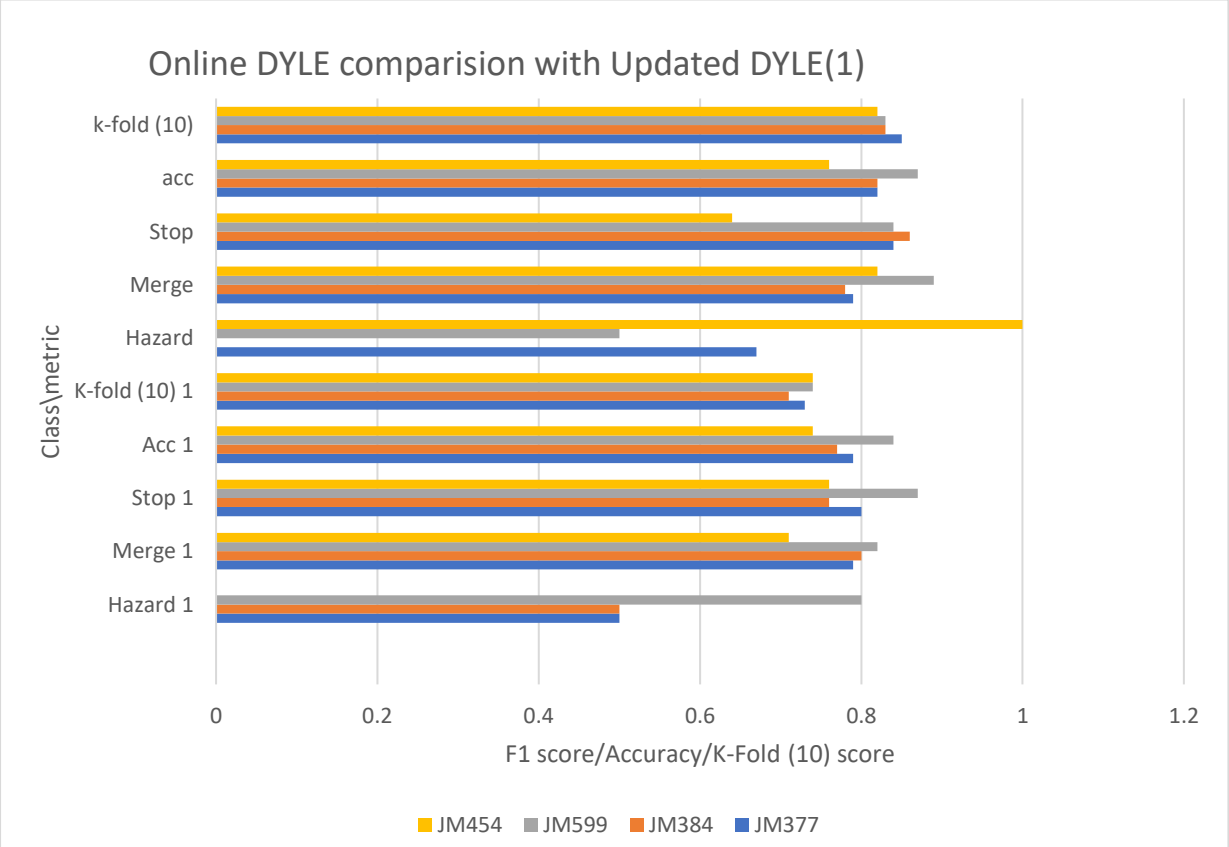



*Figure 88 Performance comparison from previous DYLE iteration for junctions JM377, JM454, JM599 and JM384.*

### *9.6.6 Accuracy and distance from merge line*

When integrating data on distance ranges with the revised DYLE dataset to evaluate our model's effectiveness across various junctions, we observed a more uniform categorisation of stop and merge actions and an improvement in overall mean accuracy, in contrast to data from isolated junctions. This data smoothing improves our ability to answer **Research Question 5 (RQ5):** Can a trained machine learning model accurately predict vehicle intent at a T-Junction using new data, and what is its effective prediction range from the junction?

Based on the data presented in Figure 89, general intent predictions regarding vehicle behaviour improved when the vehicle was less than 40% away from the total distance to the merge line. This level of prediction accuracy could potentially extend to class-specific intents, such as merging, at approximately 70% distance from the merge line. However, it's important to note that the impact of data imbalance on this predictive capability is not fully understood.

Furthermore, the analysis shows that the model's ability to predict stopping behaviour is not as strong as its predictions for merging or responding to hazards. The model's accuracy notably increases as the vehicle gets closer to the merge line, with a marked improvement observed when the vehicle is 39% or less away from the merge line. Table 54 and Figure 85 show the combined metrics of the online verification experiments.

| Distance Range | >= 70% | >= 40% and <=69% | >= 10% and <=39% | <= 9% |
|---|---|---|---|---|
| Metric | | | | |
| Stop F1 | 0.54 | 0.67 | 0.74 | 0.80 |
| Merge F1 | 0.61 | 0.70 | 0.77 | 0.82 |
| Hazard F1 | 0.00 | 0.00 | 0.59 | 0.54 |
| Accuracy | 0.60 | 0.70 | 0.77 | 0.82 |

*Table 54 Mean junction class F1 score and mean accuracy using the Online DYLE dataset*

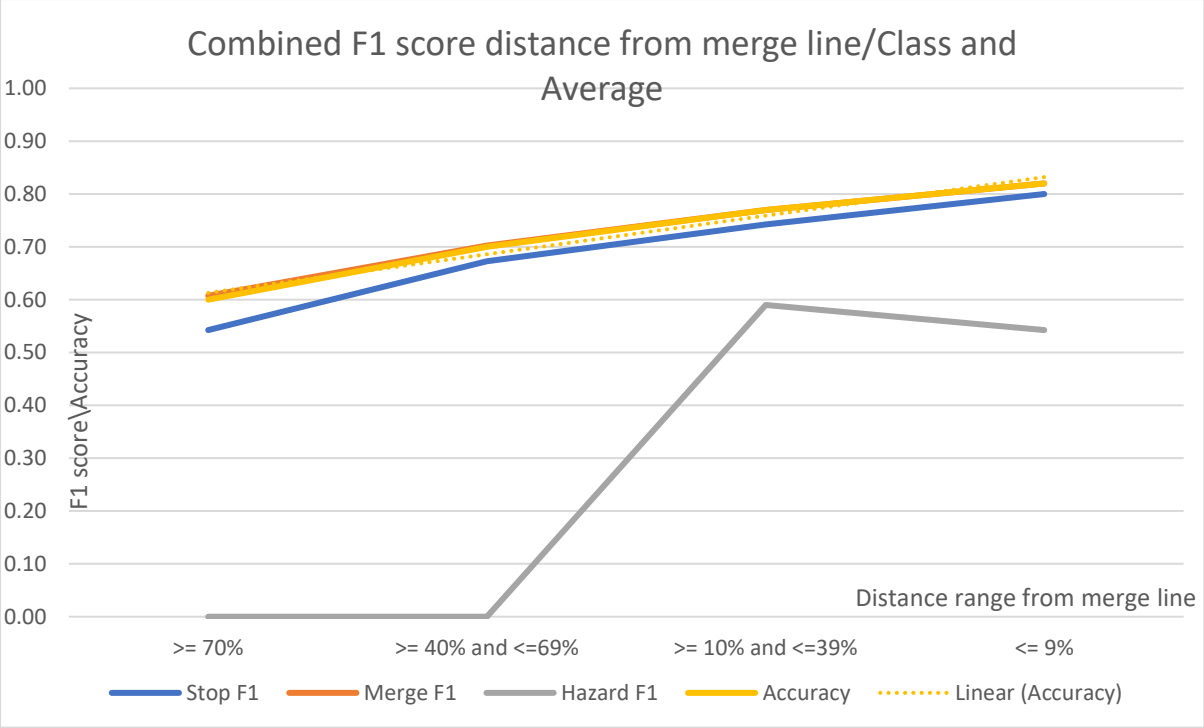



*Figure 89 Mean F1 score and accuracy based on class and distance from merge line for the combined junctions JM377, JM454, JM599 and JM384. The Stop F1 score is consistently the lowest across all distance ranges, indicating that the model is less accurate at predicting stops than Merge, which is consistently the highest, suggesting that the model is better at predicting Merge with a high confidence level. Accuracy shows an overall positive trend, meaning that the model's predictions become more accurate as the vehicle gets closer to the merge line. However, a significant increase in accuracy seems to occur when the distance range from the merge line is less than or equal to 39%. The Linear (Accuracy) trend line indicates that, on average, accuracy improves as the vehicle gets closer to the merge line. This trend line smooths out fluctuations in the actual accuracy data to show the general direction of change*.

## 9.7 Balancing the Training Data

Our work developing a general T-junction dataset has produced extensive vehicle behaviour feature data containing over 2,500 individually classified vehicles with over 99,000 classified feature vectors. Due to the nature of our data gathering in live traffic, there is an imbalance in our dataset. As previously discussed, we have a minority class, Hazard, and associated Hazard $S_{Fpc,}$ which is underrepresented in our data set online DYLE. Balancing data is crucial in machine learning to prevent the model from being biased toward the majority class and to improve overall performance. To address this issue before we move on to our subsequent experiments, we use a method based on the work of (Kovács, 2019), a well-researched

method called Synthetic Minority Over-sampling Technique (SMOTE) to create synthetic instances of the minority class.

SMOTE is a statistical technique for increasing the number of cases in your dataset in a balanced way. SMOTE creates synthetic samples from the minor class instead of copies. For each sample in the minority class, SMOTE finds its k-nearest neighbours (k-NN), where k is typically a sample from the k-nearest neighbours is randomly chosen, and a synthetic sample is created at a point along the line segment connecting the minority class sample and its chosen neighbour. The values for the synthetic sample are interpolated between the two existing samples. The SMOTE algorithm generates synthetic samples by interpolating between the positive of the minority class instances in the feature space. Given a minority class instance $fv_i$ and one of its nearest neighbours $fv_{nn}$, the synthetic sample $fv_{new}$ is created by the following equation:

$$fv_{new} = fv_i + \lambda \cdot (fv_{nn} - fv_i) \qquad (25)$$

Where $\lambda$ is a random number between 0 and 1.

This scalar $\lambda$, is how far the new synthetic sample is to be placed along the line segment between $fv_i$ and $fv_{nn}$ .

The value of $\lambda$ is generated again for each feature in the data point, which means that the synthetic samples can be scattered in the space around the original minority instances, contributing to variance and potentially leading to better generalisation for DAISY trained on this augmented dataset.

This process is repeated until the desired level of balance is reached in the dataset. For example, if the goal is to have an equal number of instances in both classes, SMOTE would keep generating synthetic samples from the minority class until this balance is achieved.

### 9.7.1 Generating new samples of Hazard class and sub-classes

We generated new samples of the Hazard class and associated sub-classes of W_Hazard, M_Hazard and S_Hazard, which were appended to the online DYLE dataset, creating a new dataset called SMOTE DYLE The new distribution can be seen in Table 55 and Figure 90, where all the subclasses are now distributed more evenly. More feature vectors are generated in the subclass (SFpc) S_Stop because vehicles in this distance zone often stop or move very slowly, resulting in a higher volume of data processing in this zone compared to other subclassification zones. In these other zones, vehicles tend to move more quickly and usually

pass through without stopping and generating multiple feature vectors over a short distance.

| Fpc / SFpc | Count | Frequency |
|---|---|---|
| s_stop | 21,420.00 | 14.52% |
| w_merge | 16,056.00 | 10.89% |
| w_stop | 15,671.00 | 10.62% |
| s_merge | 16,194.00 | 10.98% |
| m_stop | 14,170.00 | 9.61% |
| m_merge | 13,751.00 | 9.32% |
| merge | 1,280.00 | 0.87% |
| stop | 1,209.00 | 0.82% |
| w_hazard | 14,998.00 | 10.17% |
| m_hazard | 15,897.00 | 10.78% |
| s_hazard | 15,651.00 | 10.61% |
| hazard | 1,196.00 | 0.81% |

*Table 55 SMOTE DYLE*

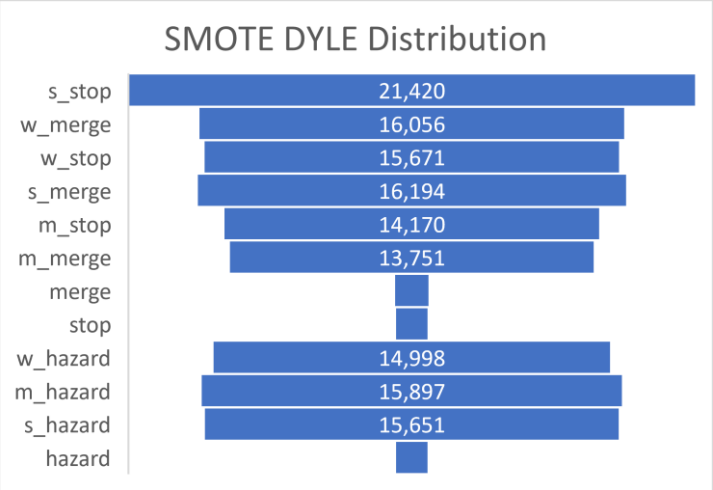



*Figure 90 SMOTE DYLE class frequency count*

SMOTE generates synthetic samples that are not duplicates of existing minority class instances. Instead, it blends characteristics from the minority class, allowing classifiers to form broader and less precise decision boundaries. This approach can enhance the DAISY's ability to generalise. However, it is crucial to understand that SMOTE's effectiveness in achieving a balanced dataset does not automatically translate to improved classifier performance. The reason is that these synthetic samples are created within the feature space and may introduce noise or unrepresentative patterns, not accurately reflecting the true distribution of the minority class. Therefore, as the literature suggests, we combine SMOTE with k-fold cross-validation to confirm that the DAISY's performance is better on new, unseen data.

### 9.7.2 K-Fold cross-validation of SMOTE DYLE

We apply K-fold cross-validation on the SMOTE DYLE dataset using the method outlined in section 7.4.6.1. This procedure aimed to understand how the synthetic addition of Hazard subclasses influences the model's predictive accuracy and to evaluate the necessity of these additional subclasses in enhancing DAISY's predictive accuracy.

| K value | Mean accuracy SMOTE DYLE |
|---|---|
| 5 | 0.86 |
| 10 | 0.89 |

*Table 56 SMOTE  DYLE K-fold cross-validation accuracy results with synthetic feature vectors for Hazard sub-classes appended*

Based on the K-fold cross-validation results seen in Table 56, DAISY's predictive accuracy has improved using the SMOTE method for generating synthetic samples. The mean accuracy of the SMOTE DYLE dataset has increased from 0.83 to 0.86 when k=5 and from 0.85 to 0.89 when k=10. The increase in mean accuracy suggests that both a higher number of folds in cross-validation and the application of SMOTE aid DAISY are probably due to better generalisation and handling of class imbalances. It's important to note that while increasing k can lead to more reliable estimates of model performance, it also increases computational cost. As discussed, adding subclasses to associate feature vectors with a ground-truth action is one of our contributions to this thesis. Without the sub-classes, we would depend on a prediction at the merge line, which, with the correct amount of accurately verified data, should prove very accurate based on our findings in previous chapters. However, our goal is to predict vehicle intent as far from the merge line as possible, and by creating associative sub-classes, we saw that accuracy is improving now that these sub-classes are balanced.

### 9.7.3 Generating new samples of minority $F_{PC}$ classes

In SMOTE DYLE, an imbalance persists in the less frequent $F_{pc}$ classes of Stop and Merge. These classes are crucial, established based on concrete ground truths, and form the basis for sub-class classification. Each $F_{pc}$ class encapsulates a verified feature vector from a target vehicle derived from our extensive video logs at various junctions. These feature vectors were initially classified either manually or using DUKE, followed by verification through our video logs. The Hazard $F_{pc}$ class is also a minority class and now consists mainly of synthetically generated feature vectors, as detailed in section 9.7.1. The next step to further balance the dataset involves generating synthetic feature vector samples for both the Stop and Merge classes and additional synthetic samples for the Hazard class to ensure a more uniform distribution across all classes. We generated the new feature vectors for all the $F_{pc}$ classes using the SMOTE method above and appended them to the new Uniform DYLE dataset. From the frequency and count details in Table 57 and Figure 91 below, we can now see a more

uniform distribution of the classes.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 21,420.00 | 11.78% |
| w_merge | 16,056.00 | 8.83% |
| w_stop | 15,671.00 | 8.62% |
| s_merge | 16,194.00 | 8.90% |
| m_stop | 14,170.00 | 7.79% |
| m_merge | 13,751.00 | 7.56% |
| merge | 12,606.00 | 6.93% |
| stop | 11,999.00 | 6.60% |
| w_hazard | 14,998.00 | 8.25% |
| m_hazard | 15,897.00 | 8.74% |
| s_hazard | 15,651.00 | 8.61% |
| hazard | 13,453.00 | 7.40% |

*Table 57 SMOTE $F_{pc}$ DYLE*

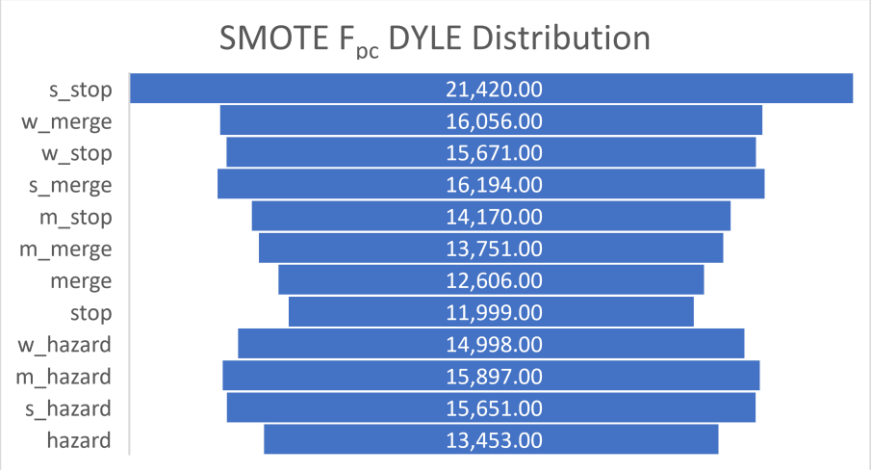



*Figure 91 SMOTE $F_{pc}$ DYLE class frequency count*

### 9.7.4 K-Fold cross-validation of Uniform DYLE

We apply K-fold cross-validation on the Uniform DYLE dataset to help understand how the synthetic addition of $F_{pc}$ classes influences the model's predictive accuracy.

| K value | Mean accuracy Uniform DYLE |
|---|---|
| 5 | 0.87 |
| 10 | 0.91 |

*Table 58 SMOTE DYLE K-fold cross-validation accuracy results with synthetic feature vectors for $F_{pc}$ appended*

Based on the K-fold cross-validation results seen in Table 58, DAISY's predictive accuracy has marginally improved using the SMOTE method for generating synthetic samples. The mean accuracy of the Uniform DYLE dataset has increased from 0.86 to 0.87 when k=5 and from 0.89 to 0.91 when k=10. Based on k= 10 with cross-validation by appending synthetic feature vector examples created using the SMOTE method, we see the uniform DYLE has a 7.06% accuracy increase from the Online DYLE dataset.

| K value | Mean accuracy Online DYLE | Mean accuracy SMOTE DYLE | Mean accuracy Uniform DYLE |
|---|---|---|---|
| 5 | 0.83 | 0.86 | 0.87 |
| 10 | 0.85 | 0.89 | 0.91 |

*Table 59 compares the DYLE datasets as synthetic data is appended.*

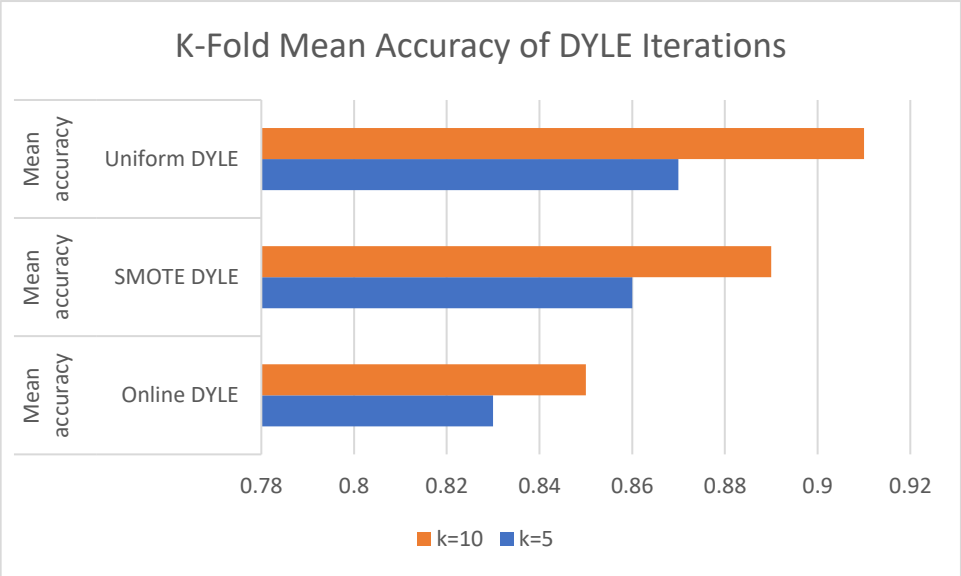



*Figure 92 Comparison of the incremental improvement of the DYLE dataset using SMOTE to generate synthetic samples.*

Our Initial analysis of the SMOTE method and results from appending synthetic data shows that due to the Online DYLE dataset having an imbalance of classes, it appears that DAISY tended to be biased towards the majority class, often neglecting the minority class. By creating synthetic samples of the minority class, the class distribution is balanced. The results in Table 59 and Figure 92 suggest that DAISY can now learn more effectively from both the minority and majority classes. This balanced learning environment allows the DAISY to understand each class's characteristics better, leading to more accurate predictions; showing a 7% increase in accuracy indicates that the model is now better at predicting outcomes across all classes, not just the majority class. The overall impact of using SMOTE is a more robust and accurate model, which is particularly important in our work, where correctly predicting the minority class is as crucial as predicting the majority class. The results reflect the effectiveness of SMOTE in enhancing Daisy's performance by mitigating the effects of class imbalance, as evidenced by a measurable improvement in accuracy. However, without a ground truth analysis, the apparent improvement in accuracy is not verified. The next step is to carry out a

complete unseen data experiment on the entire pipeline.

## 9.8 Unseen data online with a new junction

In section 7.3, Table 21, we demonstrated how we partitioned our video data for training and verification, and chapters 8 and 9 explore four of our junction video data. As detailed in 7.3, we have a hold-back junction consisting of 119 minutes of video data reserved for when we could apply the most accurate model and robust dataset to this video data to answer our primary research question. **(RQ6):** Can a trained machine learning model accurately predict vehicle intent at a T-Junction using new data, and what is its effective prediction range from the junction?

**Research Question 7 (RQ7):** Can our online model infer and append intent predictions as new inference data in real-time without negatively affecting the accuracy or F1 score?

### 9.8.1 UO196 Junction preparation

We created the optimal video ($V_o$) of junction UO196 as a single video file based on the specifications discussed in Chapter 5. We use Uniform DYLE as our dataset, adjust the ground truth junction parameters for detection zones, and merge line distance in relation to the junction topology and camera perspective. We also adjust the threshold model from section 9.3.1 to ensure that the autonomous $F_{pc}$ classification generated by DUKE and subsequent post-$F_{pc}$ classification of the $S_{Fpc}$ class are recorded accurately during the online appending of the classified feature vectors. As the UO196 $V_o$ data is input into our pipeline and target vehicles are detected and tracked, feature vectors are created, inferred, classified and appended to DYLE in a mean time of 43 ms. The latest iteration is called Alpha DYLE and contains Uniform DYLE data with the autonomously appended data from UO196.

### 9.8.2 Distribution of classes and K-fold score of Alpha DYLE

After processing all the video data through our pipeline, we created an updated version of the Alpha DYLE dataset. This new dataset increased Uniform DYLE in size by approximately 5% with the addition of newly classified feature vectors. Our analysis showed a modest improvement in performance metrics, with an increase from 0.91 to 0.92 when setting k = 10. However, there was no noticeable change in performance when k was set to 5, compared to the results obtained with the SMOTE DYLE dataset. Furthermore, as indicated in Table 60, introducing our online autonomous data appending approach did not negatively affect the dataset's quality or integrity.

| K value | Mean accuracy Alpha DYLE |
|---|---|
| 5 | 0.87 |
| 10 | 0.92 |

*Table 60 Alpha DYLE K-fold cross-validation accuracy results with synthetic feature vectors for $F_{pc}$ appended*

The class distribution of Alpha DYLE can be seen in Table 61 and visually in Figure 93.

| $F_{pc}$ / $S_{Fpc}$ | Count | Frequency |
|---|---|---|
| s_stop | 23,458 | 12.24% |
| w_merge | 17,687 | 9.22% |
| w_stop | 17,302 | 9.02% |
| s_merge | 17,621 | 9.19% |
| m_stop | 15,495 | 8.08% |
| m_merge | 15,280 | 7.97% |
| merge | 12,722 | 6.64% |
| stop | 12,118 | 6.32% |
| w_hazard | 15,018 | 7.83% |
| m_hazard | 15,912 | 8.30% |
| s_hazard | 15,662 | 8.17% |
| hazard | 13,455 | 7.02% |

*Table 61 Alpha DYLE*

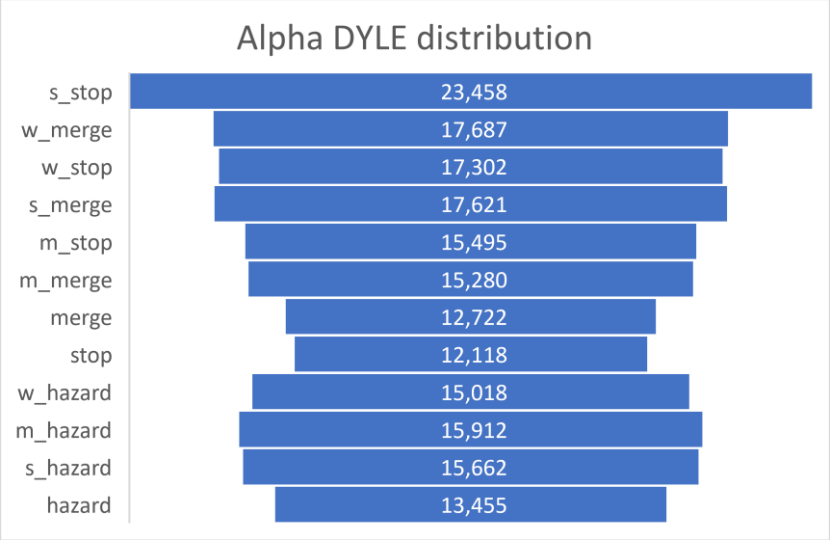



*Figure 93 Alpha DYLE class frequency count*

Figure 89 shows that the class distribution of Alpha DYLE has remained intact during the addition of autonomous feature vectors from junction UO196.

### 9.8.3 Accuracy and F1 score based on UO196 video log ground truths

The K-fold score for the Alpha DYLE model remained broadly consistent, prompting us to conduct a thorough validation through video log ground truth verification. Additionally, we recorded the DAISY predictions at various distance zones, enabling us to gather performance indicating data on accuracy and F1 scores from all distance zones and the $F_{pc}$, as detailed in Table 62 and illustrated in confusion matrixes in Figures 94-96 inclusively.

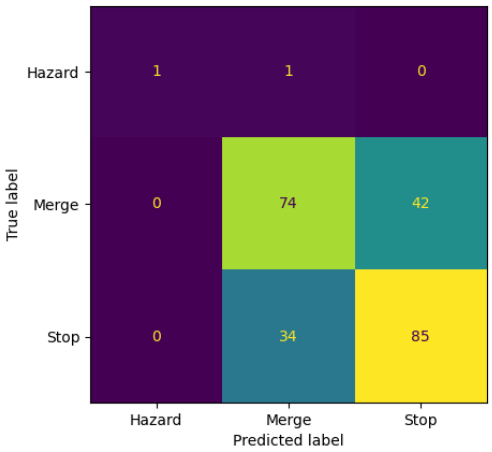



*Figure 94 UO196 Confusion matrix >= 70% Zone*

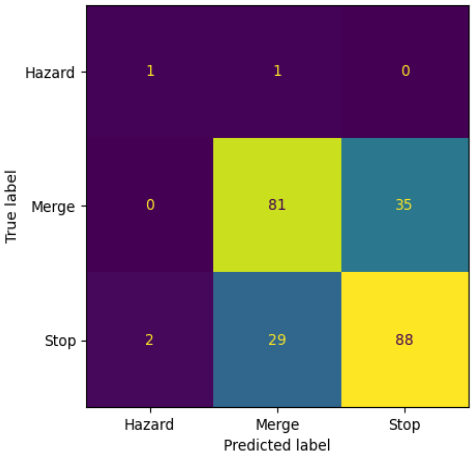



*Figure 95 UO196 Confusion matrix >= 40% and <=69% zone*

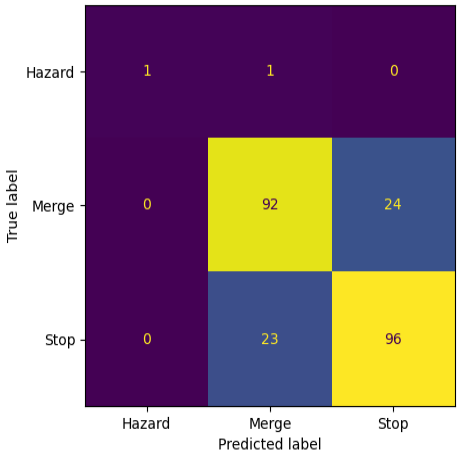



*Figure 96 UO196 Confusion matrix >= 10% and <=39% zone*

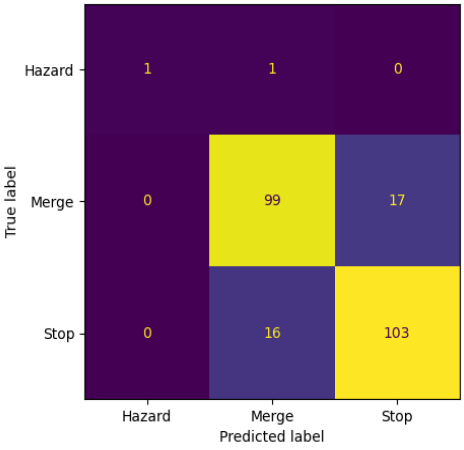



*Figure 97 UO196 Confusion matrix Fpc <=9% zone*

Accuracy and F1 scores are shown in Table 62 and Figure 98, where we have tabulated the metrics of predictions made by DAISY from each distance range.

| Dis_Range | >= 70% | | | >= 40% and <=69% | | | >= 10% and <=39% | | | <= 9% | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Class** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Pr** | **Re** | **F1** | **Qty** |
| **Hazard** | 1.0 | 0.5 | 0.67 | 0.33 | 0.5 | 0.4 | 1.0 | 0.5 | 0.67 | 1.0 | 0.5 | 0.67 | 2 |
| **Merge** | 0.68 | 0.64 | 0.66 | 0.73 | 0.70 | 0.71 | 0.79 | 0.79 | 0.79 | 0.85 | 0.85 | 0.85 | 116 |
| **Stop** | 0.67 | 0.71 | 0.69 | 0.72 | 0.74 | 0.73 | 0.80 | 0.81 | 0.80 | 0.86 | 0.87 | 0.86 | 119 |
| | | | | | | | | | | | | | |
| **Accuracy** | | | 0.68 | | | 0.72 | | | 0.80 | | | 0.86 | 237 |
| **Macro Avg** | 0.78 | 0.62 | 0.67 | 0.59 | 0.65 | 0.61 | 0.86 | 0.70 | 0.75 | 0.90 | 0.74 | 0.79 | 237 |
| **W-Avg** | 0.68 | 0.68 | 0.67 | 0.72 | 0.72 | 0.72 | 0.80 | 0.80 | 0.80 | 0.86 | 0.86 | 0.86 | 237 |

*Table 62 Accuracy metrics based on intent predictions made from a distance range from the total distance to the merge line using unseen data from UO196*

Initial discussion of results from experiments using UO196:

Table 62 shows the Metrics Precision (Pr), Recall (Re), F1-Score (F1), and Quantity (Qty) at the Discrimination distance ranges as a total distance from the merge line., and the accuracy of DAISY prediction based on Fpc at the given distance.

>= 70%

>= 40% and <= 69%

>= 10% and <= 39%

<= 9%

Hazard: Low sample size (Qty = 2), potentially leading to less reliable metrics. Consistent F1 scores across different discrimination ranges but low precision in the 40-69% range.

Merge: Largest sample size (Qty = 116), providing more reliable metrics. Generally, higher scores in higher discrimination ranges indicate better performance as the discrimination threshold increases.

Stop: The sample size is similar to Merge (Qty = 119) and consistently increases all metrics as the discrimination threshold increases.

DAISY Overall Performance: Accuracy: Increases with higher discrimination thresholds, from 0.68 to 0.86.

Macro Average: Considers each class equally and shows a general improvement in metrics with higher discrimination thresholds.

Weighted Average (W-Avg): Accounts for class imbalance and mirrors the trend in accuracy, improving as discrimination thresholds increase. Precision, recall, and F1-score are consistently higher in the <= 9% discrimination range. Due to its low quantity, the Hazard class required more instances to ascertain its true performance. For the "Merge" and "Stop" classes, there is a clear trend of improvement in model metrics with the increase in discrimination threshold.

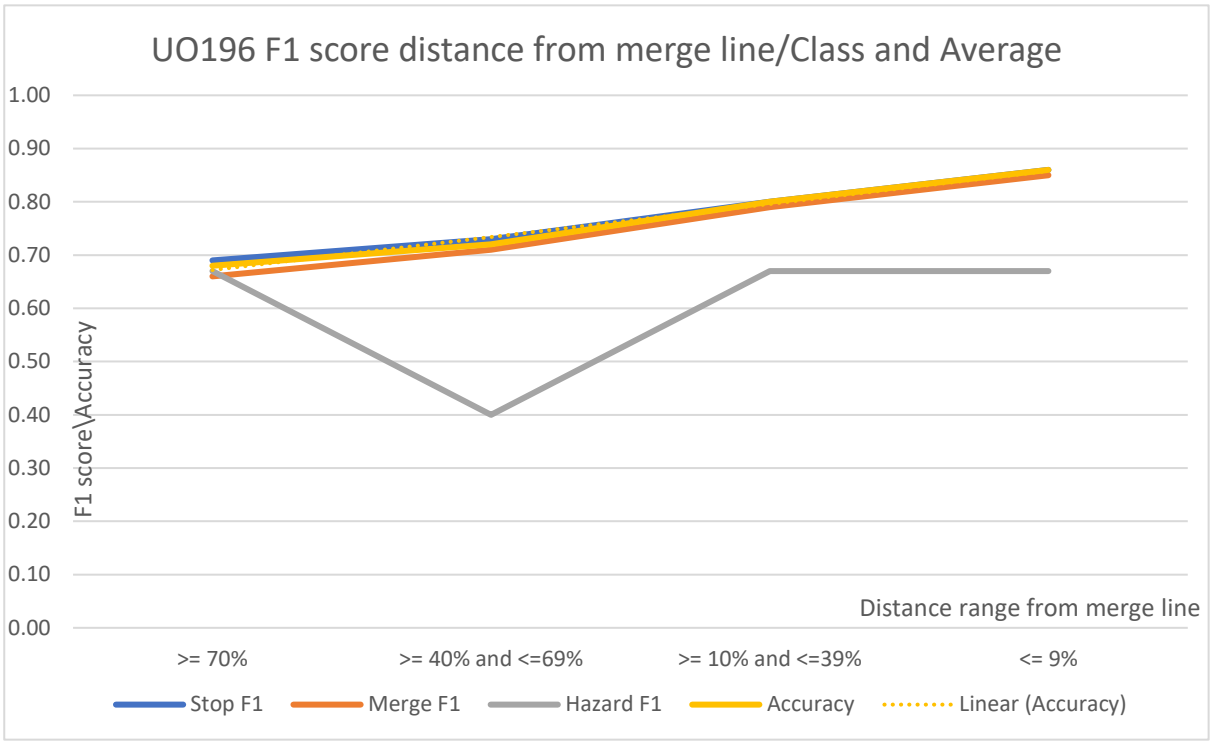



*Figure 98 Mean F1 score and accuracy based on class and distance from merge line UO196. DAISY trained on Alpha DYLE demonstrates robustness in F1 scores across different distance discrimination thresholds for the Stop and Merge classes. There is a steady increase in overall accuracy as the distance discrimination threshold decreases. The hazard class does not have enough representative samples to analyse.*

### 9.8.4 Comparison of results from Alpha DYLE + UO196 and other DYLE iterations

| Metric | | $F_{pc}$ F1-Score (<=9%) | | | Accuracy | K-fold (10) |
|---|---|---|---|---|---|---|
| Junction | Class | Hazard | Merge | Stop | | |
| Alpha DYLE + UO196 | | 0.67 | 0.85 | 0.86 | 0.86 | 0.92 |
| Online DYLE + Combined* | | 0.54 | 0.82 | 0.80 | 0.82 | 0.85 |
| Updated DYLE + Combined* | | 0.67 | 0.83 | 0.84 | 0.83 | 0.81 |
| Aggregated Training DYLE | | | | | | 0.79 |
| Uniform DYLE (SMOTE) | | | | | | 0.91 |

**Combined = JM599, JM384, JM377 and JM454*

*Table 63 Comparison of $F_{pc}$ of single junction metrics with previous DYLE iterations. Uniform DYLE has been balanced with SMOTE-created feature vectors.*

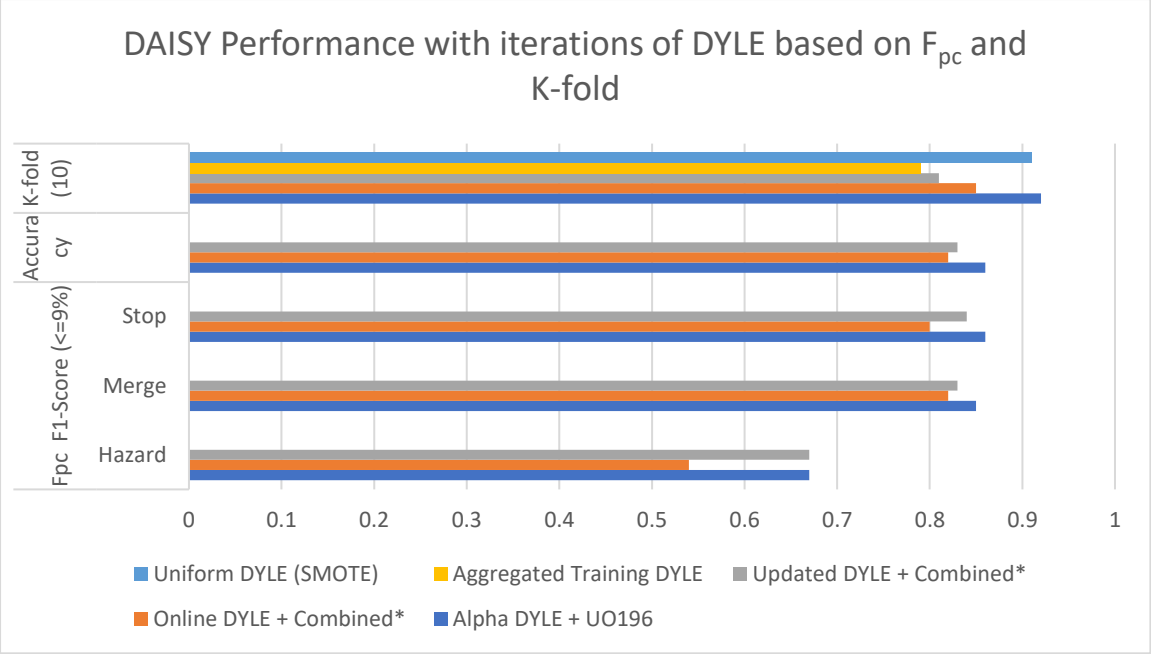



*Figure 99 K-fold score, accuracy compared to ground truth $F_{pc}$ and F1-Score per class and overall*

## 9.9 Chapter conclusion

This chapter demonstrates the methods and methodologies we used to conduct a comprehensive interactive series of experiments to answer the following research questions.

**Research Question 6 (RQ6):** Can a trained machine learning model accurately predict vehicle intent at a T-Junction using new data, and what is its effective prediction range from the junction?

**Research Question 7 (RQ7):** Can our online model infer and append intent predictions as new inference data in real-time without negatively affecting the accuracy or F1 score?

We progressively trained, evaluated and improved our dataset DYLE using our experimental junction video data and created a dataset called Alpha DYLE. We then processed our hold-back junction video data from UO196, which had not been used in our training series of experiments and passed this video through our entire pipeline. RQ6 focuses on two interlinked components from our final results:

1) Accuracy of intent prediction: Our results show that Alpha DYLE has a k-fold (k=10) of 0.92 and a ground truth average of 0.82 for $F_{pc}$ classifications on new unseen data.

2) What is the effective prediction range from the junction? We demonstrated that the effective prediction range increases as the target vehicle approaches the merge line. Accuracy improves from less than 40% of the total distance from the merge line.

RQ7 focuses on the impact of autonomous appending and inferring from new data; our results showed an improvement in $F_{pc}$ F1 scores for Stop and Merge, K-fold score and overall accuracy with no detrimental effect.

Our Contribution in this Chapter.
A quantitative examination of how accurately DAISY, trained on progressively larger datasets, can predict driver intentions and determine the practical distance from the junction at which predictions remain viable. This exploration contributes to understanding the limits and capabilities of machine learning in the context of driver behaviour prediction at critical road intersections.
Creating and evaluating an online model capable of inferring and appending new data in real time while maintaining base accuracy and F1 score.

# Chapter 10: Conclusion and Future Work

This chapter provides a comprehensive overview of how each section in this thesis plays a pivotal role in addressing the fundamental questions that drove this research. By leveraging the knowledge acquired from preceding chapters, we illuminate the contributions to answering the main research questions.

To address our core questions, we embarked on creating an experimental pipeline. The chapters in this thesis collectively delineate the evolution of a pipeline designed to explore predicting vehicle intent at T-junctions. This pipeline is established in 2d video-derived feature vectors, emphasising key aspects such as dataset creation, model training, and real-time prediction capabilities.

## 10.1 Thesis summary

**Chapter 2** This chapter delves into the concept of vehicle intent prediction, which is pivotal for forecasting future actions or movements of vehicles in various scenarios, such as navigating intersections or traffic or during autonomous vehicle operations like lane changes. The term "vehicle intent" is preferred over "driver intent" to focus on the vehicles' dynamics, whether controlled by humans or autonomously.

Our review combines computer vision and machine learning with sensor data analysis—employing 2D and 3D cameras, radar, Lidar, and GPS—to understand the vehicle's environment and movements. Machine learning models leverage this data, historical trends and behavioural modelling of human driving habits to predict future actions. This predictive framework also integrates environmental factors, such as road conditions and nearby traffic, offering a holistic approach to anticipating vehicle behaviour and enhancing road safety and efficiency.

We found that the advantages of these technologies include their consistency, reliability, and ability to process large volumes of data without fatigue, contrasting with human susceptibilities to distraction and emotional factors. Machine learning models also have the potential to learn and improve over time, possibly exceeding human prediction accuracy.

However, challenges remain, particularly in interpreting complex human behaviours, situational awareness, and ethical considerations in critical decision-making scenarios. Despite these hurdles, vehicle intent prediction aims to match or surpass human capabilities in predicting vehicle actions, contributing significantly to road safety and the advancement of autonomous vehicle technologies.

**Chapter 3** describes the creation of a data-rich video dataset for unsignalised UK T-junctions

to extract accurate vehicle feature vectors. The methodologies used to craft the dataset are explained, setting the stage for subsequent chapters focusing on object detection, classification, and prediction of driver intent. We emphasise the importance of setting a standard point of view for processing 2D videos and the necessity for meticulous camera placement from all junctions during recording to maintain a robust dataset.

**Chapter 4** addresses Research Question 1, examining the impact of a specialised dataset on the real-time performance of object detection and classification. Contributions include quantitative comparisons of YOLOv5 and Faster R-CNN models using the video dataset and constructing a target-based vehicle image dataset. Our tailored dataset slightly enhanced the confidence in vehicle detection, as shown in the copy of Table 9 below from our video data. This demonstrates that training for a specific object detection task can be enhanced through transfer learning. We utilised COCO as a foundational dataset in our instance and transferred our specialised image training data as weights to the YOLOv5m model. Ultimately, this results in more accurate ground truth observations throughout the object detection and classification, reducing computational power waste associated with false positives.

| Model YOLOv5 m | Benchmark mAP | With our dataset, mAP |
|---|---|---|
| mAP 0.5 | 0.45 | 0.45 |
| mAP 0.5 0.95 | 0.64 | 0.65 |
| Class Confidence cars (tab 3) using Bo video. | 0.91 | 0.93 |

*Copy of Table 9 (4.5.3). Post-training evaluation metrics for transfer learning using YOLOv5 m with IoU 0.5*

**Chapter 5,** Research Question 2, explores how pixel density and frame rate variations affect real-time object detection and classification models. The chapter reveals the relationship between data quantity, computational resource requirements, and the neural network's performance. An optimal model and input specification are identified for efficiently capturing vehicle features and generating feature vectors. We observed that a deep neural network like YOLOxl achieves accurate vehicle classifications with high-resolution images at a high frame rate. Conversely, a small neural network like YOLOn, when given low-resolution images at a low frame rate, tends to produce inconsistent predictions or entirely miss vehicle detections. Our model needs to fall between these two points. We pinpointed a model and input specification by conducting iterative experiments that struck the right balance, swiftly capturing the required vehicle detail within our existing models' constraints.

**Chapter 6** introduced an advanced approach to extracting feature vectors from 2D video data, addressing Research Question 3: Is obtaining accurate pixel-level features from dynamic vehicles that closely match ground truth data feasible? The data presented in Table 17 below

indicates that the feature values derived from DUKE do not exhibit significant statistical variance when compared to our ground truth data.

| Data | Mean Velocity px/ms | Vel (p-value) | Vel (SD) vel | Mean Acc $px/ms^2$ | Acc (p-value) | Acc (SD) vel | Distance px | Mean Area $px^2$ | Acc (p-value) | Acc (SD) vel |
|---|---|---|---|---|---|---|---|---|---|---|
| DUKE | 0.93 | 0.18 | 0.19 | 0.28 | 0.09 | 0.09 | 893 | 8370 | 683 | 716 |
| Ground Truth | 1.06 | 0.17 | 0.2 | 0.33 | 0.09 | 0.09 | 963 | 8259 | 679 | 705 |

Copy of *Table 17 compares manually gathered ground truth data with the 2D-pixel features obtained through DUKE.*

While promising, we concluded that further refinement of the feature data was necessary to enhance quality and reliability before training and testing the prediction model.

**Chapter 7** discussed approaches to generating credible training data in the form of feature vectors and arrays. The chapter also explores the generality of feature vectors across different experimental T-junctions. It introduces the concept of organising and classifying discrete vehicle feature vectors as feature vector arrays to classify all target vehicle data from initial detection to merge line prediction. Chapter 7 addressed **RQ4:** Can our feature vectors' inherent generality be observed per the consistent camera positioning hypothesis?
The final results from the copy of Table 32 below demonstrate an improvement when DYLE is trained by transferring data from multiple junctions into a single dataset.

| K-value | K-fold cross-validation means Accuracy | | | | |
|---|---|---|---|---|---|
| | JM599 | JM377 | JM454 | JM384 | Aggregated DYLE |
| 5 | 0.69 | 0.72 | 0.68 | 0.71 | 0.77 |
| 10 | 0.74 | 0.73 | 0.74 | 0.71 | 0.79 |

*Copy of Table 32 K-fold cross-validation means Accuracy for different datasets.*

In conclusion, the analysis of feature vectors within the context of the consistent camera positioning hypothesis indicates a positive correlation between the standardisation of data collection methods and the enhancement of model performance. Specifically, employing a consistent point of view (POV) and camera angle across various data collection points enables the aggregation of higher-quality data, as evidenced by the improved aggregated DYLE scores for K=5 and K=10 compared to individual mean accuracies. This suggests that such a standardised approach facilitates the use of data from a single junction for training across multiple junctions and significantly contributes to achieving superior overall performance in data analysis and model training.

**Chapter 8** reviews the updated DYLE dataset, noting improvements in composition and balance due to new manually classified ground truth data. In this chapter, we examined Research Question 5 **RQ5**: How accurately can a machine learning model, utilising 2D video-derived feature vectors, predict a vehicle's intention at a T-junction?

Based on the data presented in the copy of Table 41, it's evident that the Updated DYLE dataset demonstrates consistent performance across various classifications, with F1-Scores between 0.67 and 0.84 and a peak Accuracy of 0.83. This indicates that our machine learning model, DAISY, is performing well, achieving an accuracy of 0.83 despite the limited data available. This also enabled us to advance in completing the pipeline, ensuring we established a solid foundation in our machine-learning methodology.

| Metric | | F1-Score | | | Accuracy | K-fold (10) |
|---|---|---|---|---|---|---|
| Junction | Class | Hazard | Merge | Stop | | |
| Updated DYLE | | 0.67 | 0.83 | 0.84 | 0.83 | 0.81 |
| JM377 | | 0.5 | 0.79 | 0.8 | 0.79 | 0.73 |
| JM384 | | 0.5 | 0.8 | 0.76 | 0.77 | 0.71 |
| JM599 | | 0.8 | 0.82 | 0.87 | 0.84 | 0.74 |
| JM454 | | 0 | 0.71 | 0.76 | 0.74 | 0.74 |
| Aggregated Training DYLE | | | | | | 0.79 |

*Copy of Table 41 Comparison of single and combined junction and DYLE dataset accuracy metrics.*

**Chapter 9** details experiments addressing Research Questions 6 and 7. Results show that the online model, DAISY, can accurately predict driver intentions at a T-junction, with an effective prediction range increasing as the target vehicle approaches the merge line. The chapter contributes to understanding the limits and capabilities of machine learning in predicting driver behaviour at critical road intersections. **RQ6:** Can a trained machine learning model accurately predict vehicle intent at a T-Junction using new data, and what is its effective prediction range from the junction? We achieved an accuracy of intent prediction of k-fold (k=10) of 0.92 and a ground truth average of 0.82 for $F_{pc}$ classifications on new unseen data, and we demonstrated that the effective prediction range increases as the target vehicle approaches the merge line. Accuracy improves from less than 40% of the total distance from the merge line.

**RQ7:** Can the online model infer and append intent predictions as new inference data in real-time without negatively affecting the Accuracy or F1 score? We found a positive impact of autonomous appending to DYLE and inferring from new data; our results showed an improvement in $F_{pc}$ F1 scores for Stop (0.86) and Merge (0.85), K-fold score (0.92) and overall

Accuracy (0.86) with no detrimental effect to the DYLE data.

## 10.2 Research questions discussed

With an overview of the whole thesis, we can now return to my initial research questions. The overarching question was:

*How effectively can computer vision and machine learning methods be utilised to predict the intentions of vehicles at T-junctions in real-time, and to what degree of Accuracy and effectiveness can these predictions be achieved?*

We then dissected this question into eight focused research questions; the first key question was.

*How does employing a constrained and focused dataset affect the real-time performance of object detection and classification?* We discovered that, as expected, there was no significant change in detection accuracy mAP, as we were creating additional vehicle class data and not a new dataset. However, the class confidence increased, demonstrating an improvement in real data predictions when using our video dataset and suggesting that a focused dataset could improve the performance. However, further tests and model training will yield a more conclusive answer.

The second question was in relation to our choice of the YOLO model and the relationship between performance and input values. *Considering the neural network's characteristics in use, how do pixel density and frame rate variations affect real-time object detection and classification models?* We discovered a well-defined relationship between the quantity of data and the computational resource requirements for our vehicle detection and classification model. It became evident that an extensive neural network, such as YOLOxl, delivers accurate vehicle classifications when provided with high-resolution images at a high frame rate. In contrast, a simple neural network, such as YOLOn, fed low-resolution images at a low frame rate either exhibits erratic predictions or fails to detect vehicles. The YOLO models we experimented with were trained using transfer learning and our focused dataset from the previous question.

The third question queried the efficacy of obtaining accurate pixel-level features from dynamic vehicles that closely match ground truth data. *Is obtaining accurate pixel-level features from dynamic vehicles that closely match ground truth data feasible?* Our original approach to deriving meaningful features from two-dimensional 2D video data closely aligns with the ground truth data. The quality of our feature vectors hinged on the experiments undertaken in assessing YOLO models in the previous question. We obtained ground truth-based

accurate vectors in real-time by selecting a YOLO model with accuracy and inference time performance.

The fourth question examined the extent of our model's generalisation; the model's ability to infer from all our tesT-junctions is based on standardising recording settings. *Can our feature vectors' inherent generality be observed per the consistent camera positioning hypothesis?* The results from the aggregated dataset indicate an observable degree of generality in our data, as evidenced by an increase in mean Accuracy. Despite variations in the perspective of the merge line due to differences in camera placement angles at each junction, our model demonstrated the ability to recognise features from distincT-junctions and successfully apply them to other junctions.

The fifth question assessed the above components at a single experimental junction before moving to the remaining junctions. *How accurately can a machine learning model, utilising 2D video-derived feature vectors, predict a vehicle's intention at a T-junction?* We have shown reasonable Accuracy using data from a single junction, achieving an F1 score of 0.87 for the Stop class and 0.82 for the Merge class at JM599.

By amalgamating ground truth data and manually incorporating newly classified data, we observe enhanced k-fold cross-validation, resulting in higher F1 scores than the averages for individual junctions. The accuracy level stands at 0.83 at the end of these experiments.

Question 6 explored the real-world testing on new unseen data. We found that DAISY could predict unseen data with an accuracy of 0.82 and that Accuracy deteriorated from distances >40% from the merge line. Question 7 focused on the effect of self-learning the autonomous appending of intent classifications, and we found a positive impact with no detrimental effect on the DYLE data.

### 10.3 Comparison with current state of the art real time intent prediction

Key Metrics for Comparison:

1. Accuracy: The proportion of correct predictions made by the model.
2. Precision and Recall: Precision measures the accuracy of positive predictions, while recall measures the ability to identify all relevant instances.
3. F1-Score: The harmonic mean of precision and recall, providing a single metric for overall performance.
4. Latency: The model's time to process input data and produce predictions.
5. Robustness: The model's ability to handle diverse and unseen driving scenarios.
6. Scalability: How well the model performs with increasing data and complexity.

Current State-of-the-Art Models:

**ChauffeurNet (Waymo):**

Description: A deep learning approach is used to predict the behaviour of surrounding vehicles. It employs a combination of LSTM and convolutional neural networks (CNNs) for spatiotemporal processing.

Performance: High accuracy and robustness in diverse urban environments.

Challenges for Comparison: Waymo's extensive proprietary dataset and highly optimized hardware make direct comparisons challenging.

**Tesla Autopilot:**

Description: Uses a suite of cameras, radar, and ultrasonic sensors, along with neural networks, to predict vehicle actions and navigate complex environments.

Performance: Demonstrates high accuracy in many driving scenarios with rapid updates via over-the-air improvements.

Challenges for Comparison: Proprietary data and frequent updates mean performance metrics are continuously evolving, making static comparisons difficult.

**NVIDIA Drive:**

Description: Utilizes deep neural networks for vehicle intent prediction, leveraging high-performance GPUs for real-time processing.

Performance: Known for low latency and high processing power, it is suitable for real-time applications.

Challenges for Comparison: Requires significant computational resources, which may not be directly comparable to DAISY's implementation.

**Mobileye (Intel):**

Description: Uses computer vision and machine learning algorithms to predict vehicle intents. Focuses on providing real-time predictions using efficient processing techniques.

Performance: High accuracy in detecting and predicting vehicle behaviour, with a focus on scalability.

Challenges for Comparison: Mobileye's system architecture and proprietary datasets can make direct performance comparisons complex.

**Performance Comparison:**

DAISY vs. ChauffeurNet (Waymo):

- Accuracy: DAISY demonstrates comparable accuracy but may slightly lag due to Waymo's extensive dataset.
- Latency: DAISY's real-time processing capabilities are competitive, though Waymo benefits from specialized hardware.
- Robustness: Both systems handle urban environments well, but Waymo has the advantage of more comprehensive data.

DAISY vs. Tesla Autopilot:

- Accuracy: DAISY's accuracy in predicting vehicle intents is similar, but Tesla's frequent updates provide a continuously improving model.
- Latency: Both systems are designed for real-time operation, with comparable latency.
- Scalability: DAISY may have an edge in scalability due to its modular approach, whereas Tesla's proprietary system may be more optimized for specific hardware.

DAISY vs. NVIDIA Drive:

- Accuracy: Comparable accuracy, but NVIDIA's system may have higher precision due to its extensive hardware acceleration.
- Latency: NVIDIA likely has lower latency due to GPU acceleration, though DAISY is competitive with efficient algorithm design.

DAISY vs. Mobileye:

- Accuracy: DAISY and Mobileye show similar levels of accuracy, with Mobileye possibly having a slight edge due to its specialized vision algorithms.
- Latency: Both systems are designed for low latency, making them suitable for real-time applications.
- Scalability: DAISY's approach may offer better scalability due to its emphasis on modularity and extensibility.

**Systematic Differences Affecting Direct Comparison:**

Datasets:

Proprietary Data: Many state-of-the-art models use proprietary datasets that are not publicly available, making direct comparisons difficult.

Data Diversity: Differences in the diversity and volume of training data can significantly impact model performance.

Hardware:

Specialized Hardware: Some systems benefit from specialized hardware (e.g., NVIDIA's GPUs), which may not be directly comparable to the hardware used by DAISY.

Optimization: Hardware-specific optimizations can lead to performance differences that are not solely due to the algorithm.

Algorithm Complexity:

Model Architecture: Variations in model architecture (e.g., LSTM vs. CNN) can lead to differences in performance metrics.

Feature Engineering: Different feature extraction and engineering approaches can impact model accuracy and latency.

Real-Time Capabilities:

Processing Speed: Differences in processing speed due to algorithm efficiency and hardware can affect real-time performance.

Latency Requirements: Varying latency requirements for different applications (e.g., highway driving vs. urban environments) can influence model comparisons.

Comparing DAISY with state-of-the-art vehicle intent prediction models reveals that while DAISY is competitive in accuracy, latency, and robustness, systematic differences such as datasets, hardware, and algorithm complexity make direct comparisons challenging. Nonetheless, DAISY's real-time processing capabilities, modular design, and focus on scalability position it as a strong contender in autonomous vehicle intent prediction.

**Gap Analysis: Identifying Opportunities for Contribution**

Compared with state-of-the-art vehicle intent prediction models, several gaps and opportunities for contribution emerge.

**Real-Time Processing Efficiency:**

**Gap:** While existing models like ChauffeurNet and Mobileye demonstrate high accuracy, their real-time processing efficiency may vary due to hardware optimisation and algorithmic complexity differences.

**Contribution:** This thesis optimises real-time processing efficiency in DAISY, ensuring rapid prediction of vehicle intents without compromising accuracy. By leveraging efficient algorithms and hardware-agnostic design principles, DAISY aims to excel in accuracy and low-latency processing.

**Scalability and Adaptability:**

**Gap:** Current models may lack scalability or adaptability, limiting deployment across diverse platforms and environments.

**Contribution:** This thesis emphasises DAISY's modular and scalable design, enabling seamless integration into various autonomous vehicle systems and hardware configurations. By addressing scalability challenges and ensuring adaptability to evolving technological landscapes, DAISY aims to set a new standard for versatility in real-time intent prediction models.

This thesis bridges existing gaps in real-time vehicle intent prediction by addressing processing efficiency, scalability, robustness, and sensor integration challenges. Through the development and optimisation of DAISY, this research contributes to advancing autonomous driving technologies, particularly in enhancing safety, reliability, and adaptability in real-world driving scenarios. By focusing on these critical areas of improvement, this thesis aims to establish DAISY as a benchmark for future developments in vehicle intent prediction, setting new standards for efficiency, scalability, and robustness in autonomous vehicle systems.

## 10.4 Discussion and future work

**Impact of a Constrained Dataset on Real-time Performance**: It was found that focusing on creating additional vehicle class data without forming a new dataset did not significantly change the detection accuracy (measured by mean Average Precision, mAP) but did increase class confidence. This implies that a dataset with a narrower focus could potentially improve performance. By gathering training images of vehicles from our point-of-view perspective and initially using a dataset specifically tailored to vehicles instead of a broad object dataset (such as COCO) for transfer learning, we can establish the foundational step towards creating a more targeted image training dataset.

**YOLO Model Performance Analysis**: This thesis revealed a clear relationship between data quantity, computational resources, and model performance. New improved YOLO versions are becoming available every year, and the next step would be to integrate YOLOv6, 7 and 8 into DUKE and repeat the analysis undertaken in Chapter 5 to define the performance of each model.

**Accuracy of Pixel-level Features from Dynamic Vehicles**: This thesis demonstrated the feasibility of obtaining accurate pixel-level feature vectors from dynamic vehicles that align closely with ground truth data, especially when using a suitably accurate and efficient YOLO model. Combining an updated focused image dataset and an updated YOLO model may increase inference rate and Accuracy performance, allowing for a deeper neural network for tracking.

**Model Generalisation Across T-junctions**: The model showed an ability to generalise across differenT-junctions, as indicated by increased mean Accuracy. This was attributed to the consistent camera positioning, suggesting the model could recognise and apply features from various junctions. This raises the question of how much data is required from discrete junctions to generalise across any T-junction in the UK?
Future work will entail the collection of video data from another selection of experimental junctions and the amalgamation of all our test data to infer unseen junction data.

**Real-world Testing on Unseen Data**: DAISY demonstrated the ability to predict unseen data accurately. However, Accuracy decreased for data captured in relation to greater distances from the merge line. Future work would remove the constraints of real-time inference to focus solely on intent prediction accuracy at varying distances from the merge line and then build the pipeline around a solid accuracy base.

We would experiment with the following models: GPT (Generative Pre-trained Transformer) Series. OpenAI's GPT series (Roumeliotis and Tselikas, 2023), especially the latest iterations like GPT-4, offer highly accurate predictions for a range of NLP tasks, including intent detection; this could be applied to our feature vector arrays. Their ability to generate human-like text makes them suitable for complex intent understanding. LSTM networks are particularly effective for sequence prediction problems, making them ideal for applications where intent needs to be inferred from a sequence of actions. CNNs for Image-Based Intent Prediction In contexts where the intent is derived from visual cues, such as predicting the intent of a vehicle in autonomous driving, CNNs have been highly effective due to their ability to extract features from images; we would experiment with combining detection, classification and intent prediction into a single shot CNN.

CRF (Conditional Random Fields) combined with RNNs for sequence labelling tasks (Leevy, Khoshgoftaar and Villanustre, 2020) where context is crucial for predicting the intent, combining CRFs with RNNs or LSTMs has proven to be highly effective, offering a balance between understanding sequence context and making accurate predictions. Transformer-based models for Multimodal Intent Prediction Models like ViLBERT (Vision-and-Language BERT) (Hong et al., 2020) that utilise the transformer architecture to process both text and visual input have shown high Accuracy in predicting intent where visual cues and textual information are essential.

**Self-learning and Intent Classification**: We introduced a self-learning mechanism for appending intent classifications and enriching live training data that showed a positive impact without harming the data quality. Future work would involve collecting data from a static site on a 24-hour basis to generate sufficient data to assess the impact of the self-learning mechanism.

**Continuous Improvement**

**Feedback Loop for Safety Enhancements**: The vehicle intent model can continuously learn from new data, improving its predictions over time. This dynamic improvement ensures that the safety layer it represents becomes more effective as more data is collected and analysed.

## 10.5 Future work: The Swiss cheese model of safety

Integrating a vehicle intent model, particularly in the context of motorcycle safety aT-junctions, can significantly contribute to the Swiss Cheese safety model (Akuh and Atombo, 2019) by adding an innovative layer of defence against accidents.

Our vehicle intent prediction model serves as an early warning system, predicting potential

conflicts before they occur. By understanding the likely actions of vehicles at T-junctions, the system can alert riders and drivers to possible hazards, allowing for preemptive action.
This can be integrated with traffic signals and control systems; our vehicle intent prediction model can optimise traffic flow, reducing the chances of accidents by managing vehicle movements more effectively.
For motorcycles, integrating intent prediction models into Advanced Rider Assistance Systems (ARAS) can enhance the functionality of these systems, providing riders with real-time information about the intentions of surrounding vehicles, thereby allowing for safer navigation through junctions. By providing alerts to riders and drivers about the predicted movements of other vehicles, our model directly addresses human errors, such as failure to notice an approaching vehicle or misjudging its speed and direction.
The insights gained from the vehicle intent model can be used to develop better training materials and simulations, teaching riders and drivers about common risk patterns at T-junctions and how to avoid them. Data and insights from our model can inform more effective traffic policies and junction designs that inherently reduce conflict points and improve safety for all road users, especially vulnerable ones like motorcyclists.
By providing a predictive capability, our vehicle intent model fills gaps that other safety measures might not address, such as unpredictable human behaviour or the limitations of the physical infrastructure to enforce safe interactions.

Other practical use considerations.
Traffic Management Systems:
Utilising aggregated intent prediction data from multiple vehicles to optimize traffic flow and reduce congestion.
Benefit: More efficient traffic management, leading to reduced travel times and lower emissions.
Challenge: Collecting and processing large volumes of data in real-time while maintaining privacy and security.
Fleet Management:
Enhancing the management of commercial vehicle fleets by predicting vehicle behaviour and optimizing routes.
Benefit: Improved efficiency and safety of fleet operations, leading to cost savings and better service quality.
Challenge: Integrating predictive systems with existing fleet management software and ensuring scalability.
Smart Infrastructure:
Integrating intent prediction with smart city infrastructure, such as connected traffic lights and

road signs.

Benefit: Enhancing infrastructure responsiveness to real-time traffic conditions, improving overall urban mobility.

Challenge: Coordinating between various stakeholders (e.g., municipalities, tech providers) and ensuring interoperability of systems.

Driver Training and Simulation:

Using intent prediction models in driving simulators for training new drivers or assessing driver behaviour.

Benefit: Providing realistic and challenging scenarios for training, leading to better-prepared drivers.

Challenge: Develop high-fidelity simulation environments that accurately reflect real-world driving conditions.

Emergency Response:

Assisting emergency vehicles in navigating traffic by predicting the actions of surrounding vehicles.

Benefit: Faster and safer routes for emergency responders, potentially saving lives.

Challenge: Ensuring that prediction models can handle the unique dynamics of emergency scenarios.

Future Outlook:

As vehicle intent prediction systems like DAISY evolve, their integration into various aspects of transportation and urban mobility is expected to grow. The potential for these systems to enhance safety, efficiency, and overall user experience is significant. However, achieving this potential requires addressing the technical, operational, and regulatory challenges associated with their implementation. By focusing on these areas, we can pave the way for more intelligent and responsive transportation systems in the future.

### 10.5.1 Contribution to the Swiss Cheese Model for Motorcycle Safety

Integrating a vehicle intent model into motorcycle safety strategies represents a proactive and dynamic approach to safety. It does not rely solely on reactive measures (e.g., helmets, protective gear) or static infrastructure (e.g., road signs, physical barriers) but adds a sophisticated layer that actively predicts and mitigates risks. In the context of the Swiss Cheese Model, it is a layer that not only enhances the effectiveness of existing layers but also evolves to address emerging safety challenges, making it a valuable addition to the multifaceted approach required for improving road safety, particularly aT-junctions where the interaction dynamics are complex, and the stakes are high.

# Bibliography

A2D2 (2022). *Driving Dataset*. [online] a2d2.audi. Available at: https://www.a2d2.audi/a2d2/en.html.

Afifah, F., Guo, Z. and Abdel-Aty, M. (2023). System-level impacts of en-route information sharing considering adaptive routing. *Transportation Research Part C: Emerging Technologies*, 149(104075), p.104075. doi:https://doi.org/10.1016/j.trc.2023.104075.

Akhtar, N. and Ragavendran, U. (2019). Interpretation of intelligence in CNN-pooling processes: a methodological survey. *Neural Computing and Applications*, 32. doi:https://doi.org/10.1007/s00521-019-04296-5.

Akuh, R. and Atombo, C. (2019). Road Transport Accident Analysis from A System-Based Accident Analysis Approach Using Swiss Cheese Model. *International Journal of Engineering Education*, 1(2), pp.99–105. doi:https://doi.org/10.14710/ijee.1.2.99-105.

Amini, E., Omidvar, A. and Elefteriadou, L. (2021). Optimizing operations at freeway weaves with connected and automated vehicles. *Transportation Research Part C: Emerging Technologies*, 126(126), p.103072. doi:https://doi.org/10.1016/j.trc.2021.103072.

Ammar Al-Taie, Yasmeen Abdrabou, Shaun Alexander Macdonald, Pollick, F.E. and Brewster, S. (2023). Keep it Real: Investigating Driver-Cyclist Interaction in Real-World Traffic. *Enlighten: Publications (The University of Glasgow)*, (769). doi:https://doi.org/10.1145/3544548.3581049.

Ayush Dodia and Kumar, S. (2023). A Comparison of YOLO Based Vehicle Detection Algorithms. *IEEE*. doi:https://doi.org/10.1109/icaia57370.2023.10169773.

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*.

Chai, J., Zeng, H., Li, A. and Ngai, E.W.T. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, p.100134. doi:https://doi.org/10.1016/j.mlwa.2021.100134.

Charles-Éric Noël Laflamme, Pomerleau, F. and Philippe Giguère (2019). Driving Datasets Literature Review. *arXiv (Cornell University)*.

Chen, H., Hu, S., Hua, R. and Zhao, X. (2021). Improved naive Bayes classification algorithm for traffic risk management. *EURASIP Journal on Advances in Signal Processing*, 2021(1). doi:https://doi.org/10.1186/s13634-021-00742-6.

Chen, Z., Guo, H., Yang, J., Jiao, H., Feng, Z., Chen, L. and Gao, T. (2022). Fast vehicle detection algorithm in traffic scene based on improved SSD. *Measurement*, 201, p.111655. doi:https://doi.org/10.1016/j.measurement.2022.111655.

Cheng, B., Wei, Y., Shi, H., Feris, R., Xiong, J. and Huang, T.S. (2018). Revisiting RCNN: On Awakening the Classification Power of Faster RCNN. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.1803.06799.

Christodoulou, E., Ma, J., Collins, G.S., Steyerberg, E.W., Verbakel, J.Y. and Van Calster, B. (2019). A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. *Journal of Clinical Epidemiology*, [online] 110, pp.12–22. doi:https://doi.org/10.1016/j.jclinepi.2019.02.004.

Ciaparrone, G., Luque Sánchez, F., Tabik, S., Troiano, L., Tagliaferri, R. and Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. *Neurocomputing*, [online] 381, pp.61–88. doi:https://doi.org/10.1016/j.neucom.2019.11.023.

Crundall, D., Crundall, E., Clarke, D. and Shahar, A. (2012). Why do car drivers fail to give way to motorcycles at t-junctions? *Accident Analysis & Prevention*, 44(1), pp.88–96. doi:https://doi.org/10.1016/j.aap.2010.10.017.

Crundall, D., Howard, A. and Young, A. (2017). Perceptual training to increase drivers' ability to spot motorcycles at T-junctions. *Transportation Research Part F: Traffic Psychology and Behaviour*, 48, pp.1–12. doi:https://doi.org/10.1016/j.trf.2017.05.003.

Dong, X., Yan, S. and Duan, C. (2022). A lightweight vehicles detection network model based on YOLOv5. *Engineering Applications of Artificial Intelligence*, 113, p.104914. doi:https://doi.org/10.1016/j.engappai.2022.104914.

Ecker, H. and Wassermann, J. (2001). *Braking Deceleration of Motorcycle Riders*.

Elharrouss, O., Akbari, Y., Almaadeed, N. and Al-Maadeed, S. (2022). *Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches*.

Ergys Ristani, Solera, F., Zou, R.S., Cucchiara, R. and Tomasi, C. (2016a). Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. *arXiv (Cornell University)*.

F. Frank Chen, Zhang, L., Kang, S., L. Leon Chen, Dong, H., Dan, L. and Wu, X. (2023). Soft-NMS-Enabled YOLOv5 with SIOU for Small Water Surface Floater Detection in UAV-Captured Images. *Sustainability*, 15(14), pp.10751–10751. doi:https://doi.org/10.3390/su151410751.

Forward Development (2023a). *City Car Driving - Car Driving Simulator, PC Game*. [online] citycardriving.com. Available at: https://citycardriving.com [Accessed 13 Oct. 2023].

Foundation, R. (2022). *RSF EuroRAP 2021 Results Data Portal*. [online] rsfmaps.agilysis.co.uk. Available at: http://rsfmaps.agilysis.co.uk/.

Fox, C.W., Camara, F., Markkula, G., Romano, R.A., Madigan, R. and Merat, N. (2018). When Should the Chicken Cross the Road? - Game Theory for Autonomous Vehicle - Human Interactions. *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems*. doi:https://doi.org/10.5220/0006765404310439.

Geiger, A. (2023). *The KITTI Vision Benchmark Suite*. [online] www.cvlibs.net. Available at: https://www.cvlibs.net/datasets/kitti/.

Girshick, R., Kokkinos, I., Laptev, I., Malik, J., Papandreou, G., Vedaldi, A., Wang, X., Yan, S. and Yuille, A. (2017). Editorial- Deep Learning for Computer Vision. *Computer Vision and Image Understanding*, 164, pp.1–2. doi:https://doi.org/10.1016/j.cviu.2017.11.006.

GNU Image Manipulation Program (2023). *GNU Image Manipulation Program*. [online] www.gimp.org. Available at: https://www.gimp.org/about/.

Hadi Ghahremannezhad, Shi, H. and Liu, C. (2023). Object Detection in Traffic Videos: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(7), pp.6780–6799. doi:https://doi.org/10.1109/tits.2023.3258683.

Hamuda, E., Mc Ginley, B., Glavin, M. and Jones, E. (2018). Improved image processing-

based crop detection using Kalman filtering and the Hungarian algorithm. *Computers and Electronics in Agriculture*, [online] 148(148), pp.37–44. doi:https://doi.org/10.1016/j.compag.2018.02.027.

Hamzenejadi, M.H. and Mohseni, H. (2023). Accurate and Real-Time Vehicle Detection in Uav Imagery Based on Small-Size and Improved Yolov5. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.4341617.

Haufe, S., Kim, J.-W., Kim, I.-H., Marra, M., Lucci, C., Huertas-Leyva, P., Baldanzini, N., Pierini, M. and Savino, G. (2021). The future of the Autonomous Emergency Braking for Powered-Two-Wheelers: field testing . *IOP Conference Series: Materials Science and Engineering*. doi:https://doi.org/10.1088/1757-899X/1038/1/012016.

He, Q., Mei, Z., Zhang, H. and Xu, X. (2023). Automatic Real-Time Detection of Infant Drowning Using YOLOv5 and Faster R-CNN Models Based on Video Surveillance. *Journal of Social Computing*, [online] 4(1), pp.62–73. doi:https://doi.org/10.23919/jsc.2023.0006.

Hesse, R., Schaub-Meyer, S. and Roth, S. (2023). *Content-Adaptive Downsampling in Convolutional Neural Networks*.

Hong, Y., Wu, Q., Qi, Y., Cristian Rodriguez-Opazo and Gould, S. (2020). A Recurrent Vision-and-Language BERT for Navigation. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2011.13922.

Hsu, Y.-C., Swaminathan Gopalswamy, Srikanth Saripalli and Shell, D.A. (2020). A POMDP Treatment of Vehicle-Pedestrian Interaction: Implicit Coordination via Uncertainty-Aware Planning. doi:https://doi.org/10.1109/iros45743.2020.9341320.

Hu, Y., Zhan, W. and Tomizuka, M. (2018). Probabilistic Prediction of Vehicle Semantic Intention and Motion. *2018 IEEE Intelligent Vehicles Symposium (IV)*. doi:https://doi.org/10.1109/ivs.2018.8500419.

Huang, J., Rathod, V., Sun, C., Zhu, M., Anoop Korattikara, Fathi, A., Fischer, I.S., Zbigniew Wojna, Song, Y., Guadarrama, S. and Murphy, K. (2016). Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.1611.10012.

Jabir, B., Falih, N. and Rahmani, K. (2021). Accuracy and Efficiency Comparison of Object Detection Open-Source Models. *International Journal of Online and Biomedical Engineering (iJOE)*, 17(05), p.165. doi:https://doi.org/10.3991/ijoe.v17i05.21833.

Jeong, Y., Kim, S. and Yi, K. (2020). Surround Vehicle Motion Prediction Using LSTM-RNN for Motion Planning of Autonomous Vehicles at Multi-Lane Turn Intersections. *IEEE Open Journal of Intelligent Transportation Systems*, 1, pp.2–14. doi:https://doi.org/10.1109/ojits.2020.2965969.

Jocher, G. (2021). *YOLOv5 Documentation*. [online] docs.ultralytics.com. Available at: https://docs.ultralytics.com [Accessed 26 Oct. 2021].

Kaggle (2022). *Kaggle: Your Home for Data Science*. [online] Kaggle.com. Available at: https://www.kaggle.com/.

Kalman, R.E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, [online] 82(1), p.35. doi:https://doi.org/10.1115/1.3662552.

Kannan, R. and Lasky, R.C. (2020). Autonomous Vehicles Still Decades Away: 2019. *2020 Pan Pacific Microelectronics Symposium (Pan Pacific)*, 4. doi:https://doi.org/10.23919/panpacific48324.2020.9059394.

Kaysi, I.A. and Abbany, A.S. (2007). Modeling aggressive driver behavior at unsignalized intersections. *Accident Analysis & Prevention*, 39(4), pp.671–678. doi:https://doi.org/10.1016/j.aap.2006.10.013.

Kim, J., Sung, J.-Y. and Park, S. (2020). Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition. *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*. doi:https://doi.org/10.1109/icce-asia49877.2020.9277040.

Kovács, G. (2019). Smote-variants: A python implementation of 85 minority oversampling techniques. *Neurocomputing*. doi:https://doi.org/10.1016/j.neucom.2019.06.100.

Krajewski, R., Bock, J., Kloeker, L. and Eckstein, L. (2018b). The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. doi:https://doi.org/10.1109/itsc.2018.8569552.

Leddartech (2024). *Leddar PixSet Dataset*. [online] LeddarTech. Available at: https://leddartech.com/solutions/leddar-pixset-dataset/.

Lee, N., Choi, W., Vernaza, P., Choy, C., Philip and Manmohan Chandraker (2017a). DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. *Computer Vision and Pattern Recognition*. doi:https://doi.org/10.1109/cvpr.2017.233.

Leevy, J.L., Khoshgoftaar, T.M. and Villanustre, F. (2020). Survey on RNN and CRF models for de-identification of medical free text. *Journal of Big Data*, 7(1). doi:https://doi.org/10.1186/s40537-020-00351-4.

Li, Y., Li, A., Li, X. and Liang, D. (2022b). Detection and Identification of Peach Leaf Diseases based on YOLO v5 Improved Model. *2022 The 5th International Conference on Control and Computer Vision*. doi:https://doi.org/10.1145/3561613.3561626.

Liao, Y., Xie, J. and Geiger, A. (2021). KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3), pp.1–1. doi:https://doi.org/10.1109/tpami.2022.3179507.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. and Cornell (n.d.). *Microsoft COCO: Common Objects in Context*.

Liu, H., Sun, F., Gu, J. and Deng, L. (2022). SF-YOLOv5: A Lightweight Small Object Detection Algorithm Based on Improved Feature Fusion Mode. *Sensors*, 22(15), p.5817. doi:https://doi.org/10.3390/s22155817.

Maddern, W., Pascoe, G., Linegar, C. and Newman, P. (2016). 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research*, [online] 36(1), pp.3–15. doi:https://doi.org/10.1177/0278364916679498.

Mahendrakar, T., Ekblad, A., Fischer, N., White, R., Wilde, M., Kish, B. and Silver, I. (2022). *Performance Study of YOLOv5 and Faster R-CNN for Autonomous Navigation around Non-Cooperative Targets*. [online] IEEE Xplore. doi:https://doi.org/10.1109/AERO53065.2022.9843537.

Mittal, U., Chawla, P. and Tiwari, R. (2022). EnsembleNet: a hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models. *Neural*

*Computing and Applications*, 35(6), pp.4755–4774. doi:https://doi.org/10.1007/s00521-022-07940-9.

Mohd, A., Lee Vien Leong, Shee Tian Hao and Choon Wah Yuen (2022). Assessing the risky riding behavior and the effect of entrance behavior of right-turning motorcyclists on critical gap at T-junctions. *Transportation Engineering*, 10(100154), pp.100154–100154. doi:https://doi.org/10.1016/j.treng.2022.100154.

Moreno, E., Denny, P., Ward, E., Horgan, J., Eising, C., Jones, E., Glavin, M., Parsi, A., Mullins, D. and Deegan, B. (2023). Pedestrian Crossing Intention Forecasting at Unsignalized Intersections Using Naturalistic Trajectories. *Sensors (Basel, Switzerland)*, [online] 23(5), p.2773. doi:https://doi.org/10.3390/s23052773.

Muhammad Jehanzaib Yousuf, Kanwal, N., Mohd. Samar Ansari, Mamoona Naveed Asghar and Lee, B. (2022). Deep Learning based Human Detection in Privacy-Preserved Surveillance Videos. *BCS Learning & Development*, 103514. doi:https://doi.org/10.14236/ewic/hci2022.33.

Nafiseh Zarei, Payman Moallem and Shams, M. (2023). Real-time vehicle detection using segmentation-based detection network and trajectory prediction. *Iet Computer Vision*. doi:https://doi.org/10.1049/cvi2.12236.

Noto, A.P. and Saputro, S. (2022). Classification data mining with Laplacian Smoothing on Naïve Bayes method. *Nucleation and Atmospheric Aerosols*. [online] doi:https://doi.org/10.1063/5.0116519.

Pai, C.-W. (2009). Motorcyclist injury severity in angle crashes at T-junctions: Identifying significant factors and analysing what made motorists fail to yield to motorcycles. *Safety Science*, 47(8), pp.1097–1106. doi:https://doi.org/10.1016/j.ssci.2008.12.007.

Pai, C.-W. and Saleh, W. (2008). Modelling motorcyclist injury severity by various crash types at T-junctions in the UK. *Safety Science*, 46(8), pp.1234–1247. doi:https://doi.org/10.1016/j.ssci.2007.07.005.

Pandas (2018). *Python Data Analysis Library — pandas: Python Data Analysis Library*. [online] Pydata.org. Available at: https://pandas.pydata.org/.

Panero Martinez, R., Schiopu, I., Cornelis, B. and Munteanu, A. (2021). Real-Time Instance

Segmentation of Traffic Videos for Embedded Devices. *Sensors*, 21(1), p.275. doi:https://doi.org/10.3390/s21010275.

Parkers (2021). *Fiat 500 Hatchback 2008 specs & dimensions | Parkers*. [online] www.parkers.co.uk. Available at: https://www.parkers.co.uk/fiat/500/hatchback-2008/specs/ [Accessed 14 Oct. 2023].

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016a). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] doi:https://doi.org/10.1109/cvpr.2016.91.

Ren, S., He, K., Girshick, R. and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [online] 39(6), pp.1137–1149. doi:https://doi.org/10.1109/tpami.2016.2577031.

Renjun, X., Junliang, Y., Yi, W. and Mengcheng, S. (2022). Fault Detection Method Based on Improved Faster R-CNN: Take ResNet-50 as an Example. *Geofluids*, 2022, pp.1–9. doi:https://doi.org/10.1155/2022/7812410.

Ristani, E., Solera, F., Zou, R., Cucchiara, R. and Tomasi, C. (2016b). Performance Measures and a Data Set for Multi-target, Multi-camera Tracking. *Lecture Notes in Computer Science*, 9914, pp.17–35. doi:https://doi.org/10.1007/978-3-319-48881-3_2.

Robbins, C.J., Allen, H.A. and Chapman, P. (2018). Comparing drivers' gap acceptance for cars and motorcycles at junctions using an adaptive staircase methodology. *Transportation Research Part F: Traffic Psychology and Behaviour*, 58, pp.944–954. doi:https://doi.org/10.1016/j.trf.2018.07.023.

Robbins, C.J., Allen, H.A., Miller, K.V. and Chapman, P.M. (2019). The 'Saw but Forgot' error: A role for short-term memory failures in understanding junction crashes?. *PLOS ONE*, 14(9), pp.e0222905–e0222905. doi:https://doi.org/10.1371/journal.pone.0222905.

Robocar, O. (2020). *Oxford RobotCar Dataset*. [online] Oxford RobotCar Dataset. Available at: https://robotcar-dataset.robots.ox.ac.uk/ [Accessed 13 Oct. 2023].

RoSPA Road Safety Research Common motorcycle crash causes. (2017). Available at:

https://www.rospa.com/rospaweb/docs/advice-services/road-safety/motorcyclists/common-motorcycle-crash-causes.pdf.

Roumeliotis, K.I. and Tselikas, N.D. (2023). ChatGPT and Open-AI Models: A Preliminary Review. *Future Internet*, [online] 15(6), p.192. doi:https://doi.org/10.3390/fi15060192.

rsfmaps.agilysis.co.uk. (2023). *RSF EuroRAP 2021 Results Data Portal*. [online] Available at: http://rsfmaps.agilysis.co.uk/.

S. P. Lakshmi Priya, T. Karunya, R. Praveen Kumar and Arumugam, S. (2023). Vehicle Detection in Autonomous Vehicles Using Computer Vision. *Advances in intelligent systems and computing*, pp.17–34. doi:https://doi.org/10.1007/978-981-99-3608-3_2.

Savino, G., Pierini, M., Thompson, J., Fitzharris, M. and Lenné, M.G. (2016a). Exploratory field trial of motorcycle autonomous emergency braking (MAEB): Considerations on the acceptability of unexpected automatic decelerations. *Traffic Injury Prevention*, 17(8), pp.855–862. doi:https://doi.org/10.1080/15389588.2016.1155210.

Senserrick, T., McRae, D., P, P.W., Rome, L. de, Rees, P. and Williamson, A. (2017). Enhancing Higher-Order Skills Education and Assessment in a Graduated Motorcycle Licensing System. *Safety*, 3(2), p.14. doi:https://doi.org/10.3390/safety3020014.

Sezer, V., Bandyopadhyay, T., Rus, D., Frazzoli, E. and Hsu, D. (2015). Towards autonomous navigation of unsignalized intersections under uncertainty of human driver intent. *IEEE*. doi:https://doi.org/10.1109/iros.2015.7353877.

Shorten, C. and Khoshgoftaar, T.M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1). doi:https://doi.org/10.1186/s40537-019-0197-0.

Sowmya, V. and Radha, R. (2021). Heavy-Vehicle Detection Based on YOLOv4 featuring Data Augmentation and Transfer-Learning Techniques. *Journal of Physics: Conference Series*, 1911(1), p.012029. doi:https://doi.org/10.1088/1742-6596/1911/1/012029.

Statista. (2023). *Road accidents at junctions Great Britain 2019*. [online] Available at: https://www.statista.com/statistics/325327/road-accidents-at-junctions-great-britain-uk/.

Trevizan, B., Chamby-Diaz, J., Bazzan, A.L.C. and Recamonde-Mendoza, M. (2020). A comparative evaluation of aggregation methods for machine learning over vertically

partitioned data. *Expert Systems with Applications*, [online] 152, p.113406. doi:https://doi.org/10.1016/j.eswa.2020.113406.

ultralytics (2023). *GitHub - ultralytics/yolov5 at blog.roboflow.com*. [online] GitHub. Available at: https://github.com/ultralytics/yolov5?ref=blog.roboflow.com.

Vellenga, K., Steinhauer, H.J., Karlsson, A., Falkman, G., Rhodin, A. and Koppisetty, A.C. (2022). Driver Intention Recognition: State-of-the-Art Review. *IEEE Open Journal of Intelligent Transportation Systems*, 3, pp.602–616. doi:https://doi.org/10.1109/ojits.2022.3197296.

Waldner, F. and Diakogiannis, F.I. (2020). Deep learning on edge: Extracting field boundaries from satellite images with a convolutional neural network. *Remote Sensing of Environment*, 245(245), p.111741. doi:https://doi.org/10.1016/j.rse.2020.111741.

Walker, I. (2005). Signals are informative but slow down responses when drivers meet bicyclists at road junctions. *Accident Analysis & Prevention*, 37(6), pp.1074–1085. doi:https://doi.org/10.1016/j.aap.2005.06.005.

Wang, C.-Y., Hong-Yuan Mark Liao, Yeh, I-Hau., Wu, Y.-H., Chen, P.-Y. and Hsieh, J.-W. (2019). CSPNet: A New Backbone that can Enhance Learning Capability of CNN. doi:https://doi.org/10.48550/arxiv.1911.11929.

Wang, F. and Shi, D. (2020). *IEEE TRANSACTIONS ON 1 Index Terms-Unsignalized intersection, decision mak- ing, autonomous vehicles, deep reinforcement learning, deep Q-learning, double deep Q-learning* .

Waymo LLC (2019). *Open Dataset – Waymo*. [online] Waymo. Available at: https://waymo.com/open/.

Wen, Z.G., Su, J., Zhang, Y., Li, M., Gan, G., Zhang, S. and Fan, D. (2023). A lightweight small object detection algorithm based on improved YOLOv5 for driving scenarios. *International Journal of Multimedia Information Retrieval*, 12(2). doi:https://doi.org/10.1007/s13735-023-00305-5.

Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J., Ramanan, D., Carr, P., Hays, J., Ai, A. and Tech, G. (2021). *Argoverse*

*2: Next Generation Datasets for Self-Driving Perception and Forecasting*. [online] Available at: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/4734ba6f3de83d861c3176a6273cac6d-Paper-round2.pdf [Accessed 7 Jan. 2024].

Wojke, N., Bewley, A. and Paulus, D. (2017). *Simple online and realtime tracking with a deep association metric*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICIP.2017.8296962.

Wong, T. and Yeh, P. (2020). Reliable Accuracy Estimates from k-Fold Cross Validation. *IEEE Transactions on Knowledge and Data Engineering*, [online] 32(8), pp.1586–1594. doi:https://doi.org/10.1109/TKDE.2019.2912815.

Yee Mun Lee, Sheppard, E. and Crundall, D. (2015). Cross-cultural effects on the perception and appraisal of approaching motorcycles at junctions. *Transportation Research Part F: Traffic Psychology and Behaviour*, 31, pp.77–86. doi:https://doi.org/10.1016/j.trf.2015.03.013.

Yusuf Gladiensyah Bihanda, Chastine Fatichah and Anny Yuniarti (2023). Comparative Analysis of ConvNext and Mobilenet on Traffic Vehicle Detection. doi:https://doi.org/10.1109/icsecs58457.2023.10256339.

Zhan, W., Sun, L., Wang, D., Shi, H., Clausse, A., Naumann, M., Kummerle, J., Konigshof, H., Stiller, C., de La Fortelle, A. and Tomizuka, M. (2019). INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088 [cs, eess]*. [online] Available at: https://arxiv.org/abs/1910.03088.

Zhang, T., Song, W., Fu, M., Yang, Y. and Wang, M. (2021). Vehicle Motion Prediction at Intersections Based on the Turning Intention and Prior Trajectories Model. *IEEE/CAA Journal of Automatica Sinica*, 8(10), pp.1657–1666. doi:https://doi.org/10.1109/jas.2021.1003952.

Zhong, Y., Wang, J., Peng, J. and Zhang, L. (2020). *Anchor Box Optimization for Object Detection*. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 1286-1294).

Zhou, F., Zhao, H. and Nie, Z. (2021). *Safety Helmet Detection Based on YOLOv5*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICPECA51329.2021.9362711.

Zhou, Y., Wen, S., Wang, D., Mu, J. and Richard, I. (2021). Object Detection in Autonomous Driving Scenarios Based on an Improved Faster-RCNN. *Applied Sciences*, 11(24), p.11630. doi:https://doi.org/10.3390/app112411630.

Zyner, A., Worrall, S. and Nebot, E. (2018). A Recurrent Neural Network Solution for Predicting Driver Intention at Unsignalized Intersections. *IEEE Robotics and Automation Letters*, [online] 3(3), pp.1759–1764. doi:https://doi.org/10.1109/LRA.2018.2805314.