

# Improved Perceptions of Autonomous Vehicles In Urban Mobility Environment Through Deep learning Object Detection Algorithms

A thesis submitted to Trinity College Dublin, the University of Dublin in  
candidature for the Degree of

Doctor of Philosophy in Civil, Structural and  
Environmental Engineering

2024

By

Afnan Hmood Alshkeili



**Trinity College Dublin**  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin

Under the Supervision of Dr. Bidisha Ghosh

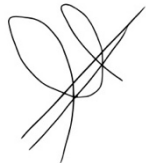
Co-supervisor Dr. Wenliang Qiu

# Declaration

*I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my work.*

*I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.*

Signature:

A handwritten signature consisting of several overlapping loops and a long horizontal stroke extending to the right.

Date: January 31, 2024



# Abstract

Autonomous Vehicles (AVs) or self-driving cars have become the most exciting area of research in the field of transportation in the last decade. Implementation of AVs in real-world requires further investigation especially in the contexts of safety and comfort of vulnerable road users (pedestrians and cyclists). This thesis focusses on exploring the impact of autonomous vehicles (AV) and their abilities from a computer-vision-based perspective.

The thesis initiates with an extensive literature review, delving into the current landscape of detecting and embracing AV. The primary objective is to explore the utilization of deep learning methodologies for detection of vulnerable road users.

Following the review of the state-of-the-art, a technology acceptance model is employed to evaluate behavioural attitudes both before and after exposure to self-driving cars. The objective is to glean insights into the perceived concerns associated with AV. These initial investigations set the groundwork for carrying out experiments on detecting pedestrians and cyclists using deep-learning based object detection algorithms analysing images captured from AVs.

This thesis rigorously compared various deep-learning-based object detection algorithms using benchmark datasets (Cityscapes, Eurocity Person, and Kitti) to ensure efficiency and real-time safety for AV development. Five distinct algorithms (Faster RCNN, Cascade RCNN, FCOS, Deformable, DETR, and RetinaNet) were compared for the detection of urban road objects across various datasets. Additionally a diverse traffic benchmark dataset was constructed combining several benchmark datasets incorporating various weather, lighting, and traffic scenarios, evaluating state-of-the-art detection algorithms in traffic situations and adapting them accordingly. Lastly, the research made a significant contribution by developing a unified algorithm for simultaneous estimation, tracking, and detection of multiple road objects, advancing AV technology and promoting safer autonomous transportation systems. This algorithm was tested on real-life traffic images collected through a field study.

This thesis underscores the paramount importance of comprehending public perception and acceptance of autonomous vehicles (AVs). Simultaneously, it emphasizes the necessity for the development of robust and efficient detection

algorithms to ensure the safety of AVs in diverse, real-world environments. The creation of a comprehensive traffic benchmark dataset is a notable contribution, addressing the need for varied and challenging test cases in the advancement of autonomous vehicle technology. Furthermore, the establishment of a unified algorithm for multi-object estimation and tracking fills a crucial research gap, providing a foundational element for holistic approaches to AV technology. Collectively, this research marks a substantial leap forward in the continual progress of autonomous vehicles, offering valuable insights into societal acceptance, safety enhancement through advanced algorithms, and the provision of essential tools for the ongoing development and integration of AVs into contemporary transportation systems.

# Acknowledgements

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to my research supervisor, Dr. Bidisha Ghosh, for giving me the opportunity to do research and providing invaluable guidance throughout this research. Her dynamism, vision, sincerity and motivation have deeply inspired me. It was a great privilege and honor to work and study under her guidance.

I would like to acknowledge and give my warmest thanks to my co-supervisor Dr. Wenliang Qiu who made this work possible. His guidance and advice carried me through all the stages of completing my project.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. I am very much thankful for their love, understanding, prayers and continuing support to complete this research work. Also, I express my thanks to my sisters for their support and valuable prayers. My thanks also go to my friends for the keen interest shown to complete this thesis successfully.

Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

# Contents

<i>Declaration</i> .....	<i>iii</i>
<i>Abstract</i> .....	<i>iv</i>
<i>Acknowledgements</i> .....	<i>vi</i>
<i>List of Figures</i> .....	<i>x</i>
<i>List of Tables</i> .....	<i>xiii</i>
<b>Chapter 1</b> .....	<b>15</b>
<b>1 Introduction</b> .....	<b>16</b>
<b>1.1 BACKGROUND AND MOTIVATION</b> .....	<b>16</b>
<b>1.2 AUTONOMOUS VEHICLES</b> .....	<b>18</b>
1.2.1 History of automated driving.....	18
1.2.2 Challenges.....	19
1.2.3 Automation levels.....	20
1.2.4 How Autonomous Vehicles work.....	22
1.2.5 Advanced Driver Assistance Systems (ADAS) and autonomous vehicle sensors.....	23
1.2.6 Developments.....	25
1.2.7 The regulatory and legal framework.....	29
1.2.8 Ongoing work in the EU.....	29
<b>1.3 RESEARCH OBJECTIVES</b> .....	<b>30</b>
<b>1.4 ORGANIZATION OF THE THESIS</b> .....	<b>31</b>
<b>1.5 DISSEMINATION FROM THESIS</b> .....	<b>33</b>
<b>Chapter 2</b> .....	<b>35</b>
<b>2 Literature review</b> .....	<b>36</b>
<b>2.1 TECHNOLOGY ACCEPTANCE MODEL</b> .....	<b>36</b>
2.1.1 Autonomous Vehicle acceptance.....	40
<b>2.2 COMPUTER VISION</b> .....	<b>49</b>
<b>2.3 IMAGE PROCESSING</b> .....	<b>50</b>
2.3.1 Image enhancement .....	51
2.3.2 Image restoration .....	51
2.3.3 Image analysis.....	52
2.3.4 Applications of image processing .....	53
<b>2.4 DEEP LEARNING-BASED OBJECT DETECTION</b> .....	<b>54</b>
2.4.1 Object detection and tracking .....	59
2.4.2 Detection of Autonomous Vehicles .....	64
2.4.3 Pedestrian and cyclist detection .....	65
<b>2.5 OBJECT DETECTION ALGORITHMS</b> .....	<b>70</b>
2.5.1 Overview of Object Detectors.....	71
2.5.2 Convolutional Neural Network (CNN).....	73
2.5.3 DESCRIPTION OF ALGORITHMS.....	81
2.5.4 DESCRIPTION OF DATASETS.....	105
2.5.5 PERFORMANCE MEASURES .....	115
2.5.6 INFERENCE TIME MEASUREMENT .....	118
<b>2.6 Research Scope</b> .....	<b>119</b>

<b>Chapter 3</b> .....	<b>121</b>
<b>3 The Acceptance of Autonomous Vehicles in a Mixed Mode Environment....</b>	<b>122</b>
<b>3.1 INTRODUCTION</b> .....	<b>122</b>
<b>3.2 TECHNOLOGY ACCEPTANCE MODEL (TAM)</b> .....	<b>123</b>
3.2.1 Perceived usefulness and perceived ease of use .....	125
3.2.2 Mathematical theory .....	125
<b>3.3 EXPERIMENT AND DATA COLLECTION</b> .....	<b>127</b>
3.3.1 Survey Design.....	127
3.3.2 Administration Method.....	128
<b>3.4 DATA DESCRIPTION AND MEASUREMENTS</b> .....	<b>129</b>
3.4.1 Demography data analysis .....	129
3.4.2 Reliability and Validation of the Measurement Model.....	132
<b>3.5 ANALYSIS AND RESULTS</b> .....	<b>134</b>
3.5.1 Frequency and Comparative Analysis .....	134
3.5.2 Applying TAM TAM to autonomous vehicles .....	145
<b>3.6 DISCUSSION AND CONCLUSION</b> .....	<b>149</b>
<b>Chapter 4</b> .....	<b>151</b>
<b>4 Comparative Evaluation of Algorithms</b> .....	<b>152</b>
<b>4.1 DESIGN OF EXPERIMENT</b> .....	<b>152</b>
4.1.1 Hardware Requirements .....	153
4.1.2 Software Requirements .....	154
<b>4.2 DATASET ORGANISATION AND TRAINING</b> .....	<b>154</b>
4.2.1 Experiment Summary.....	154
4.2.2 Data Flow .....	154
4.2.3 Data Pre-processing .....	156
4.2.4 Parametrization of Object Detection Algorithms and Training Details .....	156
4.2.5 Comparison of Performance Across Datasets.....	157
4.2.6 Training of ODAs .....	158
<b>4.3 TRAINING AND EVALUATION ON INDIVIDUAL DATASETS</b> .....	<b>161</b>
4.3.1 Cityscapes Dataset .....	161
4.3.2 KITTI Dataset .....	164
4.3.3 EuroCity Person dataset.....	169
<b>4.4 DISCUSSION</b> .....	<b>186</b>
<b>4.5 CONCLUSION</b> .....	<b>186</b>
<b>Chapter 5</b> .....	<b>188</b>
<b>5 Benchmarking Framework for Training &amp; Evaluation</b> .....	<b>189</b>
<b>5.1 TRAINING AND EVALUATION ACROSS DATASETS</b> .....	<b>189</b>
5.1.1 Cross-Validation Results.....	191
<b>5.2 DEVELOPMENT OF A UNIFIED BENCHMARKING EVALUATION DATASET</b> .....	<b>193</b>
5.2.1 Collation of datasets .....	195
5.2.2 Variation in Scenes.....	196
<b>5.3 TRAINING, TESTING AND VALIDATION OF UBED</b> .....	<b>197</b>
5.3.1 Training of UBED .....	198
5.3.2 Testing and Validation.....	198

<b>5.4</b>	<b>PERFORMANCE EVALUATION .....</b>	<b>199</b>
5.4.1	Faster RCNN .....	199
5.4.2	Cascade RCNN .....	200
5.4.3	RetinaNet .....	201
5.4.4	FCOS .....	202
5.4.5	Deformable DETR .....	203
<b>5.5</b>	<b>DISCUSSION &amp; CONCLUSION .....</b>	<b>204</b>
<b>Chapter 6 .....</b>		<b>207</b>
<b>6</b>	<b><i>Detection, Estimation &amp; Tracking Road Objects for Assisting Driving.....</i></b>	<b>208</b>
<b>6.1</b>	<b>PROPOSED METHODOLOGY OF DTE ALGORITHM .....</b>	<b>208</b>
<b>6.2</b>	<b>DESCRIPTION OF DETECTION, ESTIMATION, AND TRACKING ALGORITHMS .....</b>	<b>211</b>
6.2.1	Object Detection Algorithm: Traffic RetinaNet .....	211
6.2.2	Object Distance Estimation .....	213
6.2.3	Object Tracking Algorithm: Simple, Online and Realtime Tracking (SORT).....	215
6.2.4	Count and Speed estimation .....	217
<b>6.3</b>	<b>DATA .....</b>	<b>218</b>
6.3.1	Datasets used for training .....	218
6.3.2	Dataset used for testing .....	219
<b>6.4</b>	<b>EVALUATION OF DTE .....</b>	<b>220</b>
6.4.1	Evaluation of Traffic RetinaNet .....	220
6.4.2	Evaluation of Tracking .....	223
6.4.3	Distance, Speed and Flow estimation .....	226
6.4.4	Evaluating DET on the field data .....	228
<b>6.5</b>	<b>DISCUSSION &amp; CONCLUSION .....</b>	<b>231</b>
<b>Chapter 7 .....</b>		<b>232</b>
<b>7</b>	<b><i>Conclusion .....</i></b>	<b>233</b>
<b>7.1</b>	<b>RESEARCH CONTRIBUTIONS .....</b>	<b>233</b>
<b>7.2</b>	<b>RESEARCH LIMITATIONS.....</b>	<b>236</b>
<b>7.3</b>	<b>FUTURE RESEARCH DIRECTION.....</b>	<b>237</b>
<b><i>Bibliography.....</i></b>		<b>239</b>
<b><i>Appendix A .....</i></b>		<b>262</b>
<b><i>Appendix B.....</i></b>		<b>270</b>
<b><i>Appendix C .....</i></b>		<b>271</b>
	Challenges .....	274
A)	Challenges.....	274
B)	Overcoming challenges.....	275
<b><i>Appendix D.....</i></b>		<b>276</b>
	Files to modify .....	276

# List of Figures

<b>Figure 1.1</b> The Firebird II concept car .....	19
<b>Figure 1.2</b> Guangzhou L4 bus shuttle .....	22
<b>Figure 1.3</b> Functional perspective of vehicular data .....	23
<b>Figure 1.4</b> ADAS sensors used .....	24
<b>Figure 1.5</b> Masdar city PRT.....	26
<b>Figure 1.6</b> AV market forecast (Lavasani et al., 2016).....	27
<b>Figure 1.7</b> Navya autonomous bus in Tokyo .....	27
<b>Figure 1.8</b> Boss the Tartan robot, developed by Carnegie Mellon University and General Motors (Urmson et al., 2007).....	28
<b>Figure 1.9</b> Thesis roadmap .....	33
<b>Figure 2.1</b> Theory of Reasoned Action (TRA) .....	36
<b>Figure 2.2</b> Technology Acceptance Model (TAM) .....	37
<b>Figure 2.3</b> TAM 2 .....	38
<b>Figure 2.4</b> TAM 3 .....	39
<b>Figure 2.5</b> Computer vision .....	50
<b>Figure 2.6</b> Relationship between Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL).....	55
<b>Figure 2.7</b> Deep learning timeline by Favio Vazquez: from 1940 to 2017.....	55
<b>Figure 2.8</b> Supervised learning architect.....	57
<b>Figure 2.9</b> Unsupervised learning architect .....	58
<b>Figure 2.10</b> Semi-supervised learning architect.....	58
<b>Figure 2.11</b> Two-stage object detection pipeline.....	61
<b>Figure 2.12</b> One-stage object detection pipeline .....	62
<b>Figure 2.13</b> YOLO object detection system .....	63
<b>Figure 2.14</b> Taxonomy of object detectors.....	72
<b>Figure 2.15</b> CNN architecture.....	74
<b>Figure 2.16</b> Convolution layer process(Martin, 2018) .....	75
<b>Figure 2.17</b> Pooling layer (2x2 max-pooling kernel).....	76
<b>Figure 2.18</b> ReLU equation depiction .....	77
<b>Figure 2.19</b> Fully connected layer.....	77
<b>Figure 2.20</b> A Softmax layer within a neural network.....	79
<b>Figure 2.21</b> Regions with convolutional neural network architecture .....	84
<b>Figure 2.22</b> Fast RCNN architecture .....	85
<b>Figure 2.23</b> Faster RCNN architecture.....	87
<b>Figure 2.24</b> Region Proposal Network .....	88
<b>Figure 2.25</b> RoI pooling layer .....	89
<b>Figure 2.26</b> Cascade RCNN architecture.....	91
<b>Figure 2.27</b> Faster RCNN architecture.....	92
<b>Figure 2.28</b> Iterative BBox at Inference.....	94
<b>Figure 2.29</b> Integral Loss .....	95
<b>Figure 2.30</b> Featurized image pyramid .....	96
<b>Figure 2.31</b> Different type of pyramids architecture.....	97
<b>Figure 2.32</b> RetinaNet network architecture .....	98
<b>Figure 2.33</b> FCOSarchitecture .....	100
<b>Figure 2.34</b> DETR framework .....	103

<b>Figure 2.35</b> Deformed DETR framework.....	104
<b>Figure 2.36</b> Sample of COCO dataset images.....	106
<b>Figure 2.37</b> High quality cityscapes image annotation.....	108
<b>Figure 2.38</b> Coarse cityscape annotation .....	108
<b>Figure 2.39</b> Sample of Cityscapes dataset .....	109
<b>Figure 2.40</b> Sample of KITTI dataset .....	111
<b>Figure 2.41</b> Sample of COCO dataset images.....	112
<b>Figure 3.1</b> Technology Acceptance Model (TAM). .....	124
<b>Figure 3.2</b> Which category describes you the best?.....	129
<b>Figure 3.3</b> Gender & Age .....	130
<b>Figure 3.4</b> Education levels.....	131
<b>Figure 3.5</b> Employment sector and years of experience.....	131
<b>Figure 3.6</b> Safety statistics: (a) Statistical representation of the notion that self-driving vehicles generate fewer accidents, (b) Statistical representation of the claim that self-driving vehicles decrease traffic congestion, (c) Statistical representation of the idea that self-driving vehicles will reduce risky driving behaviours, (d) Statistical representation of the assertion that self-driving vehicles outperform humans in detecting dangerous situations.....	135
<b>Figure 3.7</b> Environmental statistics: (a) Statistical representation of the notion that self-driving vehicles will reduce energy consumption, (b) Statistical representation of the idea that self-driving vehicles are environmentally friendly, (c) Statistical representation of the notion that self-driving vehicles cause an increase in car use and emissions, (d) Statistical representation of the idea that self-driving vehicles will increase the number of miles people travel. ....	137
<b>Figure 3.8</b> Environmental statistics: Statistical representation of the notion that self-driving vehicles will free up public spaces and promote clean air. ....	138
<b>Figure 3.9</b> Conjunction statistics:(a) Statistical representation of the idea that self-driving vehicles offer more convenience and productivity, (b) Statistical representation of the point that self-driving vehicles offer more personal freedom and independence, (c) Statistical representation of the idea that mobility is more affordable with self-driving vehicles through ridesharing, (d) Statistical representation of the assertion that self-driving vehicles will reduce driving efforts. ....	139
<b>Figure 3.10</b> Anxiety statistics: (a) Statistical representation of the statement that 'before using an AV, I doubted if I would be able to control the vehicle if an ethically complicated situation arises', (b) Statistical representation of the statement that 'after using an AV, I became confident that all my journeys with AVs would be successful', (c) Statistical representation of the statement that 'before using an AV, I was worried that interacting with the vehicle would require much mental effort', (d) Statistical representation of the statement that 'after using an AV, I became confident that all my journeys with AVs would be successful'.....	142
<b>Figure 3.11</b> Anxiety statistics: (a) Statistical representation of the statement that 'After using AV, I became confident that all of my journeys with AVs will be successful', (b) Statistical representation of the statement that 'Interacting with the AV did not require a lot of mental effort', (c) Statistical representation of the statement that 'I would trust the vehicle', (d) Statistical representation of the statement that 'I am afraid that I won't be able to react in case an emergency occurs'. ....	143



<b>Figure 3.12</b> Statistical representation of the notion ‘Before using AV, I was afraid an emergency would arise because the vehicle would malfunction’.....	144
<b>Figure 3.13</b> Results of TAM, determinants, variables, and perceptions.....	145
<b>Figure 3.14</b> Regression analysis results for before using AVs. ....	147
<b>Figure 3.15</b> Regression analysis results for after using AVs. ....	148
<b>Figure 4.1</b> Experiment framework .....	153
<b>Figure 4.2</b> Dataset structure .....	155
<b>Figure 4.3</b> Cityscapes ODA where the images are obtained from testing using faster RCNN, RetinaNet and cascade RCNN.....	164
<b>Figure 4.4</b> KITTI ODA where the images are obtained from testing using faster RCNN. ....	166
<b>Figure 4.5</b> ECP FCOS confusion matrix .....	172
<b>Figure 4.6</b> FCOS detection results on ECP dataset .....	173
<b>Figure 4.7</b> ECP RetinaNet confusion matrix .....	174
<b>Figure 4.8</b> RetinaNet detection results on ECP .....	175
<b>Figure 4.9</b> Faster RCNN ECP confusion matrix .....	176
<b>Figure 4.10</b> Faster RCNN detection results on ECP .....	178
<b>Figure 4.11</b> Cascade RCNN ECP confusion matrix .....	180
<b>Figure 4.12</b> Cascade RCNN results on ECP .....	181
<b>Figure 4.13</b> Deformable DETR results on ECP. ....	183
<b>Figure 4.14</b> Comparing detection result obtained on ECP, (a) FCOS detection results, (b) RetinaNet detection results, (c) Faster RCNN detection result, (d) Deformable DETR detection result, (e) Cascade RCNN detection result.....	185
<b>Figure 5.1</b> Framework .....	194
<b>Figure 6.1</b> Framework of detection, tracking and estimation.....	210
<b>Figure 6.2</b> Framework of RetinaNet with ResNeXt backbone (Traffic RetinaNet). ....	212
<b>Figure 6.3</b> Distance estimation in 2D image plane.....	214
<b>Figure 6.4</b> Framework of improved SORT. ....	217
<b>Figure 6.5</b> Detection of multipule classes of objects in MOT16 dataset.....	217
<b>Figure 6.6</b> Tracking results in multiple scenes over three consecutive frames.....	225
<b>Figure 6.7</b> Flow and Speed estimation.....	227
<b>Figure 6.8</b> Number and Speed of Pedestrian and Vehicle in Scene (a).....	228
<b>Figure 6.9</b> Percentage error of distance estimation.....	229
<b>Figure 6.10</b> Detection, Tracking & Estimation result. ....	230

# List of Tables

<b>Table 1.1</b> Levels of automation based on SAE J3016. The word ‘system’ refers to the automated driving system. ....	21
<b>Table 2.1</b> Summary of AV acceptance survey paper .....	41
<b>Table 2.2</b> Summary of reviewed papers on pedestrian and cyclist detection .....	66
<b>Table 2.3</b> Chosen algorithms unique aspects and reason to be used for evaluation.	82
<b>Table 3.1</b> <b>Questions asked for TAM before and after analysis</b> .....	128
<b>Table 3.2</b> Reliability and Validity Tests .....	133
<b>Table 4.1</b> Hyperparameter Settings .....	156
<b>Table 4.2</b> Comparison of datasets .....	158
<b>Table 4.3</b> Datasets and algorithms used for training and testing in this chapter..	161
<b>Table 4.4</b> Results of different ODAs on Cityscapes .....	161
<b>Table 4.5</b> Results of detection on cityscape provided in literature. ....	162
<b>Table 4.6</b> KITTI dataset training results .....	165
<b>Table 4.7</b> Ground truths (GTs), Detection(Dets), AP and Recall values for the classes .....	167
<b>Table 4.8</b> Ground truths (GTs), Detection(Dets), AP and Recall values for the classes .....	167
<b>Table 4.9</b> KITTI literature testing result (mAP) .....	168
<b>Table 4.10</b> ECP performance results when trained on 2080Ti GPU.....	169
<b>Table 4.11</b> ECP performance results when trained on 3090Ti GPU.....	169
<b>Table 4.12</b> ECP literature testing results (mAP ) .....	171
<b>Table 4.13</b> Average percentage and average recall of FCOS .....	171
<b>Table 4.14</b> Average percentage and average recall of RetinaNet .....	174
<b>Table 4.15</b> Average percentage and average recall of Faster RCNN .....	176
<b>Table 4.16</b> Average percentage and average recall of Cascade RCNN .....	179
<b>Table 4.17</b> Average percentage and average recall of Deformable DETR.....	182
<b>Table 5.1</b> Matrix of changes involved in unifying datasets.....	190
<b>Table 5.2</b> Cross dataset evaluation on Cityscapes, KITTI and EuroCity Person (mAP). A→B, refer to training on A and testing on B. ....	191
<b>Table 5.3</b> Summary of dataset and algorithms used in this chapter.....	197
<b>Table 5.4</b> Evaluation of UDEB trained and tested using Faster RCNN .....	199
<b>Table 5.5</b> Result obtained from testing UBED on Faster RCNN .....	200
<b>Table 5.6</b> Evaluation of UDEB trained using Cascade RCNN .....	200
<b>Table 5.7</b> Result obtained from testing UBED on Cascade RCNN .....	201
<b>Table 5.8</b> Evaluation of UDEB trained using RetinaNet.....	201
<b>Table 5.9</b> Result obtained from testing UBED on RetinaNet.....	202
<b>Table 5.10</b> Evaluation of UDEB trained and tested using Faster RCNN .....	202
<b>Table 5.11</b> Result obtained from testing UBED on FCOS.....	203
<b>Table 5.12</b> Evaluation of UDEB trained and tested using Deformable DETR .....	203
<b>Table 5.13</b> Result obtained from testing UBED on Deformable DETR.....	204
<b>Table 6.1</b> Datasets and algorithms used for training and testing in this Chapter...	209
<b>Table 6.2</b> Association Status .....	217
<b>Table 6.3</b> Compare traffic RetinaNet to the original RetinaNet (COCO dataset) ...	220
<b>Table 6.4</b> Compare traffic RetinaNet to the original RetinaNet (Cityscapes dataset) .....	220

<b>Table 6.5</b>	Compare traffic RatinaNet to the original RetinaNet (KITTI dataset) ....	221
<b>Table 6.6</b>	Average percentage and average recall of the different categories .....	221
<b>Table 6.7</b>	Comparing results of the original SORT to the improved. ....	224
<b>Table 6.8</b>	MOT evaluation matrix.....	224
<b>Table 6.9</b>	Statistical Estimation Results of Videos .....	224
<b>Table 6.10</b>	Computational performance of the framework .....	225
<b>Table 0.1</b>	Descriptive Statistics. ....	270
<b>Table 0.1</b>	ECP literature testing evaluation (LAMR).....	273

---

# Chapter 1

---

# 1 Introduction

## 1.1 BACKGROUND AND MOTIVATION

Autonomous vehicles (AVs), self-driving cars, or driverless cars are widely used phrases to describe vehicles that can sense the environment and safely perform driving with no or little human input. These cars were introduced to reduce driving efforts, especially on urban roads (Aneesh, Shine, Pradeep, & Sajith, 2019), because of their safety considerations. For instance, the United States is considered a country built on wheels; with over 289.5 million registered passenger cars by 2021 (US VIO Vehicle Registration Statistics: See How Many Cars in The US n.d.), it is evident that people in the U.S. like to have their own cars. A statistical study in 2015 showed that around 94% of traffic accidents in the United States were caused by human errors (Deb et al., 2017). It is estimated that the rate of penetration of AVs into the traffic fleet can reduce traffic conflicts proportionately, with a reduction of over 90% if all vehicles on the road are AVs (Papadoulis, Quddus, & Imprialou, 2019). Apart from improving safety, the additional benefits of AV would include reductions in the stress levels of drivers/passengers, reductions in emissions through improved driver behaviour, and improvements in overall efficiency in network management.

With regard to AVs, moving object detection, classification, and tracking algorithms have received extensive research attention. The search terms ‘object detection’ and ‘autonomous driving’ led to the identification of 321 articles in Scopus. Many of these articles focused on multiple sensors, including Light Detection and Ranging (LiDAR), multiple cameras, radar, laser, etc., employing sensor fusion methodologies along with advance machine learning and deep-learning algorithms. Object detection (Papageorgiou, Oren, & Poggio, 1998), segmentation (Haralick & Shapiro, 1985), tracking (Jarraya, 2018), semantic segmentation (Thoma, 2016), optical flow (Krapp & Hengstenberg, 1996), depth estimation (Torralba & Oliva, 2002), and 3D scene reconstruction (Häne, Zach, Cohen, Angst, & Pollefeys, 2013) are some of the algorithms used by researchers in analysing the environmental information in the surroundings for autonomous driving.

The object detection algorithms studied in the literature considered multiple classes of objects, primarily focusing on pedestrians and road signs. Region-based Convolutional Neural Network (R-CNN) is a common deep learning algorithm used for object detection.

Yet, this cannot be implemented for real-time traffic object detection on road because of the long computational times. Nevertheless, faster R-CNN was developed to overcome this issue (X. Zhao et al., 2016). Newer algorithms like Single Shot Multi-Box Detector (SSD) (W. Liu et al., 2016) and You Only Look Once (YOLO) (Redmon, Divvala, Girshick, & Farhadi, 2016) achieved high model performance as they have been able to overcome some of the disadvantages present in CNN and R-CNN, such as selective search. The selective search uses local cues like texture, intensity, and colour to find the Region of Interest (RoI), which is a slow and time-consuming process. Gavrila (Gavrila, 2000) proposed a prototype system for pedestrian detection from a moving vehicle using a two-step approach algorithm, wherein the first step contour features are used in a hierarchical template matching approach. The second step involves utilizing the richer set of intensity features in a pattern classification approach. Lee et al. (Lee, Chiu, Lin, & Hung, 2009) developed an object detection algorithm in 3D cues for detecting pedestrians and vehicles. They proposed an object detection algorithm for detecting pedestrians and vehicles from a moving video, using background modelling to extract frame information into a 3D space. The objects are then verified to identify whether or not they are pedestrians (vehicles) by the class-specific Support Vector Machine (SVM), based on the 3D cues. This thesis focuses on detection, as it is the initial step for all other applications. For instance, almost all tracking algorithms require detection of the objects, either in the first frame or in every frame; the estimation algorithm also requires a similar approach. Therefore, detection algorithms are a must to precisely locate and categorize objects. Hence, the main algorithm I shall focus on to increase safety and efficiency is detection.

For instance, in traffic tracking scenes, an object detection mechanism—either in every frame, or when the object first appears in the video—is required. A common approach for object detection is to use information in a single frame. However, some object detection methods use the temporal information computed from a sequence of frames to reduce the number of false detections. This temporal information is usually in frame difference, which highlights changing regions in consecutive frames. Given the object regions in the image, the task of the tracker is to perform object correspondence from one frame to the next to generate the tracks.

In addition, the field of AVs is developing rapidly; therefore, efficient, complex, and cost-effective detection and tracking algorithms, which focus mainly on pedestrian and cyclist safety, need to be investigated and properly developed. In order to implement these

complex computational technologies in a real-world environment, AV connectivity technology should be introduced for computational purposes. Furthermore, an urban mixed-mode environment is an ecosystem that involves the different components of a traffic system, viz., vehicles, cyclists, and pedestrians.

Moreover, acceptance after making this technology accessible to the public is an area for investigation and research. For this reason, the Technology Acceptance Model (TAM) is a technique that is being deployed by researchers to analyse public acceptance. However, an analysis of their acceptance after the use of such technologies has not been investigated yet. This research will highlight the perspective of people on the use of AVs in two phases: before their introduction to AV technology and after their experience of it. Hence, in this research, the acceptance of AVs before and after making them accessible to the public will be assessed, along with the proposal of a real-time object tracking detection algorithm.

## **1.2 AUTONOMOUS VEHICLES**

### **1.2.1 History of automated driving**

‘Autonomous vehicles’, ‘self-driving cars’, or ‘robocalls’ are phrases used to describe vehicles that are capable of sensing their environment and of moving safely with little help, or no human involvement. A historical review of autonomous vehicles carried out by Joe et al. covers the period from 1925 to 2019. It stated that the first driverless vehicle was unveiled by Francis Houdina back in 1925 (Janai, Güney, Behl, & Geiger, 2020): ‘American Wonder’ was driven remotely along Broadway in New York City by an operator using a radio control in a nearby vehicle. The experiment led General Motors (GM) to explore the field of automation, where they proposed several prototypes that used radio streams for manoeuvres to be performed. Again, in 1956, Firebird II was unveiled; however, nearly all the prototypes were banned from being tested on the road. Owing to the high cost and limited scalability, this technology was restricted to ground transportation, airports, park shuttles, or automated facilities. Figure 1.1 presents the Firebird II concept car, which was never meant to be produced; it was, nevertheless, created to show how technology would change the automobile sector.



**Figure 1.1** The Firebird II concept car<sup>1</sup>

In 1995, a new development occurred, in which a dynamic driving control technology—electronic stability control (ESC)—was announced (Babak, Hussain, Karakas, & Cetin, 2017). Several driving assistance technology developments arose in the 1990s, based on sensors that measure the status of the respective vehicles, thereby providing relevant information and warnings to drivers. For instance, in 1995, Mitsubishi presented the first LiDAR-based distance control (Janai et al., 2020) (Moters, M. 2008), while in 1999, Mercedes-Benz implemented the radar-assisted adaptive cruise control. Several approaches to the issue of autonomous vehicles continued, and manufacturing companies continued to develop technologies. Even though driver assistance technologies became a commercial success, enriching driving comfort and safety, the goal of producing fully driverless vehicles has remained unattained till date.

### **1.2.2 Challenges**

---

<sup>1</sup> 1956 *Firebird II* [Photograph], by General Motor <https://www.gm.com/heritage/collection/gm-concept/1956-firebird-II>



Many challenges confront the implementation of AVs: some theoretical, some practical. The theoretical ones are those related to computer vision—for instance, road object detection and recognition (Martínez-Díaz & Soriguera, 2018). The computing framework implemented in AV is a key aspect when the theoretical characteristics of self-driving cars are considered. Extracting relevant and accurate information from the raw data provided by sensors in real time, and telling the vehicle how it must proceed to take decisions, is challenging. The use of the cloud and preclusion of hacking are issues that concern computer scientists. Regulation and policymakers find it challenging to devise laws and policies that protect AV users and manufacturers. The introduction and acceptance of AVs on the roads is a challenge; not all individuals can accept the concept of self-driving vehicles. Researchers argued that AV will contribute positively to traffic flow by optimizing vehicle operations and reduce crashes and delays (C. Xu, Ding, Wang, & Li, 2019). However, (Meyer, Becker, Bösch, & Axhausen, 2017) stated that AV would cause increased car use demand and consequent lack of road capacity, wherein travel demand by the elderly, the handicapped, and children would increase, since it would be no burden for them to travel on their own. A study by (Park, Byun, Kim, Ahn, & Shin, 2021) concluded that the use of AVs will potentially improve traffic flow. Other challenges like liability and insurance raise these questions: how insurance companies will deal with AVs, who will constitute the ‘driver’, who has ultimate ‘control’, and so on and so forth. Software failure (Koopman & Wagner, 2016) is a crucial challenge that confronts AVs. Hence, the implementation of AV remains a challenging topic and requires further studies.

### **1.2.3 Automation levels**

The Society of Automotive Engineers (SAE) is an international company based in the United States since 1905, with over 138,000 members worldwide. The company is a professional association: it is a standards-developing organization, mainly for the different engineering industries. In 2018, the organization defined the six levels of driving automation, ranging from Level 0, with no driving automation, to Level 5, which is fully autonomous (Standard, 2018). On the other hand, in 2013, the National Highway Traffic Safety Administration (NHTSA) divided automation levels into five levels, ranging from no automation (Level 0) to fully self-driving automation (Level 4)(National Highway

Traffic Safety Administration and others, 2017). The main difference between the two approaches is that in Levels 0 to 3, according to SAE, the human driver monitors the road. In the levels above that, autonomous driving monitors the road. But NHTSA states that the levels from 0 to 2 entail human control driving, with no or two automation features. Level 3 vehicles can have full control in a specific environment. Finally, Level 4 is designed to perform all safety-critical driving functions and monitor all different road conditions for the entire trip. The table below highlights the different automation levels, based on the SAE J3016 (Standard, 2018) standards published in 2018. These define the terms related to driving the automation system of the road motor vehicles.

**Table 1.1** Levels of automation based on SAE J3016. The word ‘system’ refers to the automated driving system.

Level	Name	Definition	Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of dynamic driving task	System capability (driving modes)
Human driver monitors the driving environment						
0	No automation	The full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	–
1	Driver assistance	The driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task.	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial automation	The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task	System	Human driver	Human driver	Some driving modes
Automated driving system monitors the driving environment						
3	Conditional automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task (including latitudinal and longitudinal control) with the expectation that the human driver will respond appropriately to a request to intervene	System	System	Human driver	Some driving modes
4	High automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene. If the human driver fails to take control of the vehicle, the system steers the vehicle to the side of the road in a controlled manner and stops it	System	System	System	Most driving modes
5	Full automation	The full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver	System	System	System	All driving modes

Based on the proposed driving level standards, no vehicle is considered as having Level 5 autonomy. Although car manufacturers have developed many systems to reach full

autonomation, experts and politicians doubt whether full automation (Level 5) with driverless private vehicles able to operate on any road is achievable in the conceivable future (Nieuwenhuijsen, 2015; Van Witsen, 2016).

It is worth mentioning that these automation levels have already been implemented in transport fleets today—for example, Tesla’s Autopilot system at Level 2 and the Guangzhou bus shuttle at Level 4. Figure 1.2 shows the automated bus shuttle in Guangzhou, which can accommodate up to 6 passengers.



**Figure 1.2** Guangzhou L4 bus shuttle <sup>2</sup>

#### **1.2.4 How Autonomous Vehicles work**

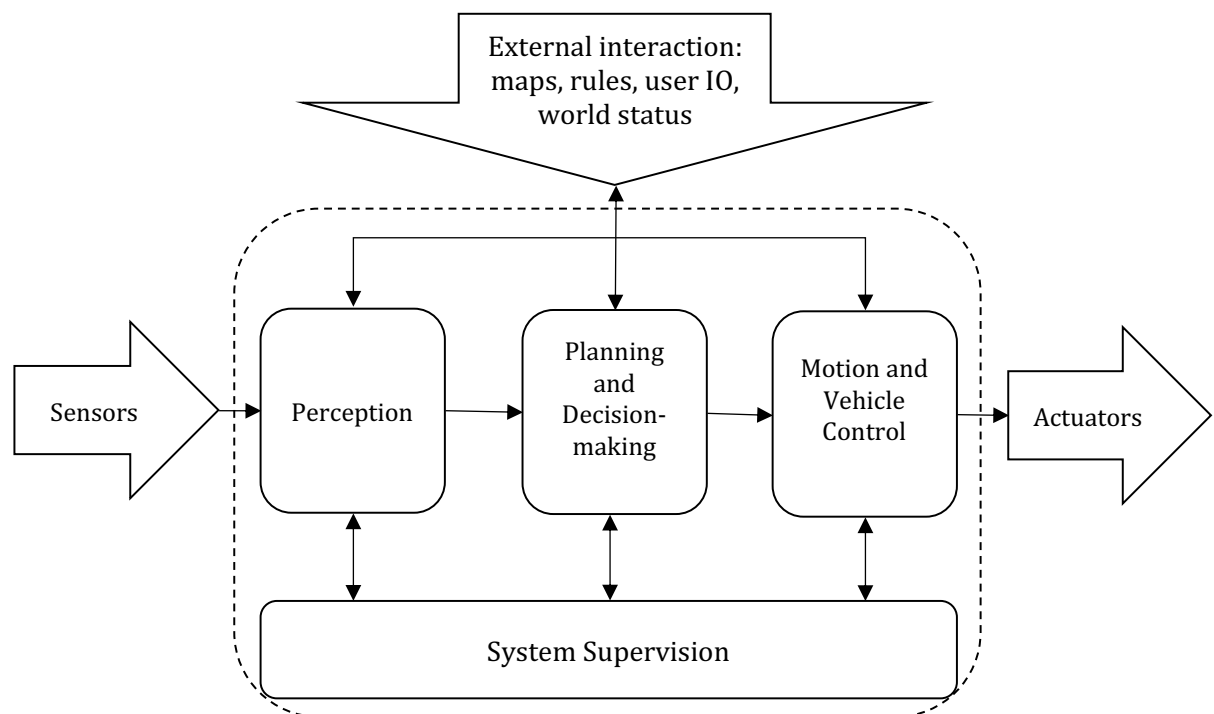
Autonomous vehicles rely on sensors, actuators, complex algorithms, machine learning systems, and powerful processors to execute software. Vehicles create and maintain a map of their surroundings on the basis of a variety of sensors, located in different parts of the vehicle. Radar sensors monitor the positions of nearby vehicles. Video cameras detect traffic lights, read road signs, track other vehicles, and look for pedestrians. Other sensors like LiDAR, ultrasonic, and radar are also used to perform

---

<sup>2</sup> *Guangzhou L4 bus shuttle* [Photograph], by the city of Guangzhou’s official WeChat account <https://mobility-innovators.com/guangzhou-launched-first-driverless-bus-line-with-l4-autonomous-driving-buses>

the different driving tasks (refer to Section 1.2.4 for information on the functionality of different sensors).

After receiving all information from the different sensors, a software processes all inputs, plots the path, and sends the instruction to the vehicle actuators, which control acceleration, braking, and steering. Hard-coded rules, obstacle avoidance algorithms, predictive modelling, and object recognition help the software follow traffic rules and navigate obstacles. Figure 1.3 illustrates the processing of vehicular data collected from the sensors by the four main functional modules of the AV system (Ignatious, Sayed, & Khan, 2021).



**Figure 1.3** Functional perspective of vehicular data

### 1.2.5 Advanced Driver Assistance Systems (ADAS) and autonomous vehicle sensors

Advanced Driving Assistance Systems (ADAS) are technical elements that increase the safety of cars on the roads. They can be also defined as digital systems that help drivers in routine navigation and parking, using computer networks to enable more data-driven and safer driving experiences. These systems increase the driver’s potential to adapt to road hazards, through early warning and automated systems.

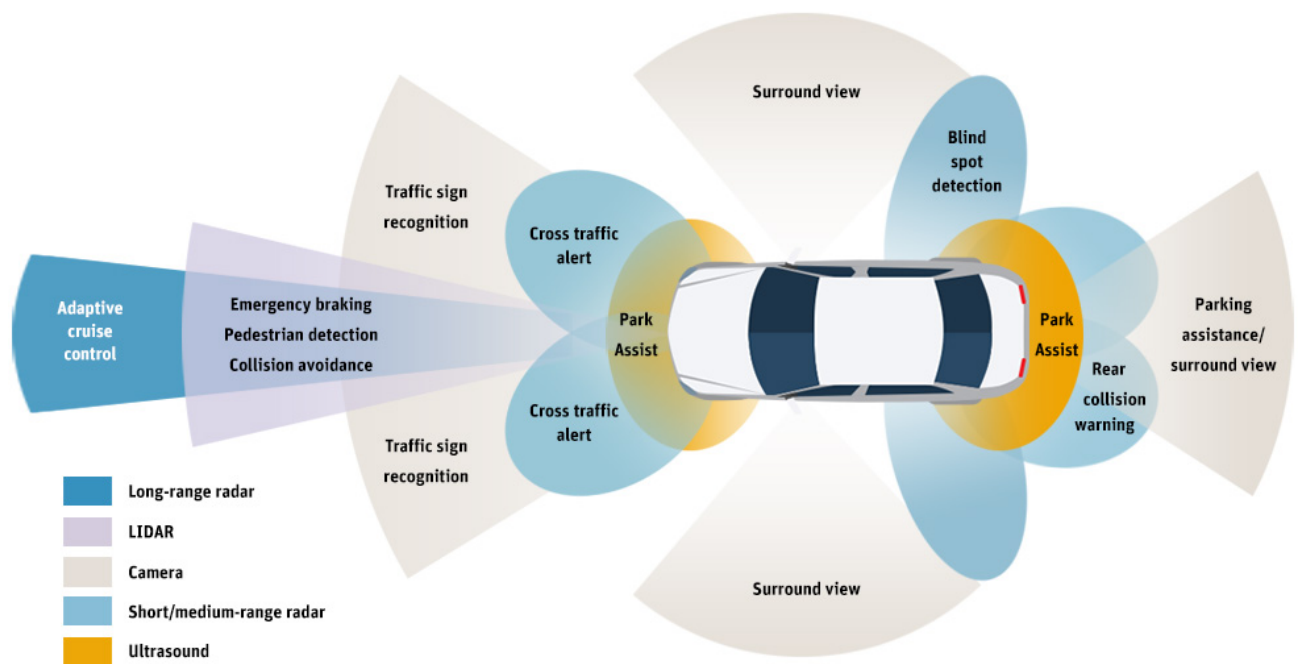
ADAS aim to minimize the incidence and severity of automotive accidents that cannot be averted to prevent deaths and injuries. In this, systems are developed to automate,

adapt, and enhance vehicle technology to ensure safety and better driving. Such systems are used to detect human driver weariness and distraction, and issue cautionary signals to analyse driving performance and offer recommendations like cruise control systems.

Moreover, ADAS have been adopted in nearly all automobiles nowadays. Some examples of the available systems are:

- Adaptive cruise control (ACC)
- Forward collision alert
- Lane departure alert
- Traffic light traction control recognition
- Anti-lock braking systems

These systems rely on different sensors technologies to monitor and move through the surroundings. Figure 1.4 illustrates the use of the different sensors.



**Figure 1.4** ADAS sensors used

ADAS systems are grouped into two classes:

- 1 Passive safety systems protect vehicle occupants from injuries after a crash. Seat belts, padded dashboards, whiplash protection system, SOS system, and airbags are some of the examples of passive systems (Kumar Kukkala, Tunnell, Pasricha, & Bradley, 2018).

- 2 Active systems are those in which the car provides advance warning e.g., automatic emergency braking (AEB), wherein the system detects an impending accident and applies the brakes without the driver's assistance. Functional features include adaptive cruise control (ACC), lane-keeping assist (LKA), lane centring (LC), blind spot detection (BSD), and traffic jam assist. These systems are important in the sense that they provide passive and active safety mechanisms that eliminate human driving errors.

### **1.2.6 Developments**

The success of the automation technologies in the market has been traced by the US Defence Advanced Research Projects Agency (DARPA) (Shubbak, 2017). Leading companies that provide web-based services, such as Google, have announced that they are developing and testing autonomous cars, which will be in the market in the coming years. For instance, a Google partner company, Waymo, an American autonomous driving technology development company that provides self-driving taxis, has travelled more than one million miles between 2009 and 2015 (Lavasani, Jin, & Du, 2016). Again, ParkShuttle—a Level 4 bus shuttle that has been operating since 2006—deploys driverless buses to connect a metro station with a business park in Rotterdam. This automatic transport system has been developed further and is currently being applied in several places worldwide, such as Masdar City (Abu Dhabi), as a Personal Rapid Transit (PRT) system (2getthere, 2016) (Figure 1.5).



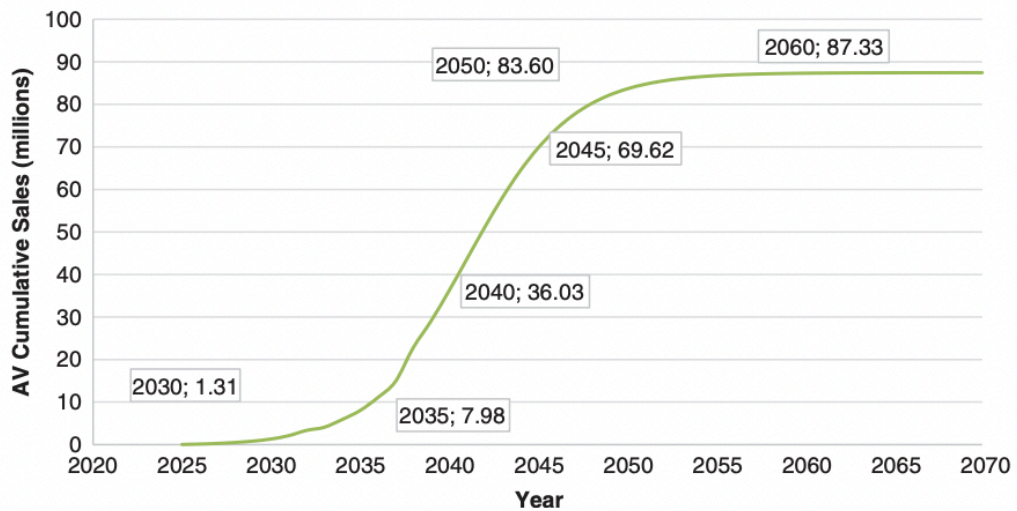


**Figure 1.5** Masdar city PRT<sup>3</sup>

Market studies forecast that the adoption of AVs holds promise. Lavasani et al. pointed out that AVs will significantly change individual lives. The AV market should be, therefore, considerably large (Lavasani et al., 2016). An AV market forecast proposed by Lavasani et al. can be seen in Figure 1.6 (Lavasani et al., 2016); it shows that a slight increase in purchases will be noticed in from 2030 to 2035; however, a proportionate increase in sales will follow, before tending to stabilize by 2060.

---

<sup>3</sup> *Masdar city PRT* [Photograph] by Jeannette <https://www.2getthere.eu/news/masdars-prt-system-functions-99-since-launch-2010/>



**Figure 1.6** AV market forecast (Lavasani et al., 2016)

Navya is one of the leading companies in autonomous mobility systems; it has already provided the world market more than 170 units in all types of environments (mix mode environment and controlled environment). Figure 1.7 provides an example of a Navya shuttle operating in Tokyo.



**Figure 1.7** Navya autonomous bus in Tokyo<sup>4</sup>

<sup>4</sup> The autopilot bus is seen in Tokyo's Chiyoda Ward [Photograph] by Mainichi/Tatsuya Michinaga <https://mainichi.jp/english/articles/20210310/p2a/00m/0bu/006000c>



It is worth mentioning that challenges like the DARPA Grand Challenge in 2004 forced researchers to compete to develop complex systems and automated algorithms to achieve the aims of their competition. For these, different sensors—radars, LiDAR, ultrasounds, and video sensors—are imbedded in vehicles to capture the information from the surroundings and compute distance. Algorithms are then developed to process traffic data information to calculate distance, path, and flow, which accordingly enable control of the actuators (Thrun, 2010). In 2007, the third DARPA challenge took place: teams from different research groups demonstrated the capability of robotic cars to concurrently perceive the environment of the vehicle, stabilize its motion, and to react with suitable driving manoeuvres (Campbell, Egerstedt, How, & Murray, 2010; Kammel et al., 2008). In this challenge, robotic cars drove driverless in a closed urban environment with a predefined traffic scenario. Tartan Racing was the first place winner in DARPA 2007: it is a collaborative effort between Carnegie Mellon University and General Motors (Urmson et al., 2007). Their vehicle was ‘Boss’, wherein their system was loaded on a Chevrolet Tahoe chassis, as shown in Figure 1.8. LiDAR, radar, and visual sensors were incorporated to safely navigate the urban environment.



**Figure 1.8** Boss the Tartan robot, developed by Carnegie Mellon University and General Motors (Urmson et al., 2007).

The Dubai World Challenge for Self-Driving Transport is one of the largest international platforms for companies and research and development centres to implement scenarios and applications of self-driving technology on Dubai roads. This challenge was organized by Dubai's Roads and Transport Authority (RTA), the aim of which is to develop the transportation fleet and achieve 2030 Dubai Self-Driving vision to transfer 25% of all passenger trips in Dubai so that they can be smart and driverless.

### **1.2.7 The regulatory and legal framework**

Autonomous vehicles are the future of transportation; regulations and policies are yet to be put in place in this respect. However, countries like the US and Australia have devised some legal regulations. In the US, for instance, the policy of the National Highway Traffic Safety Administration (NHTSA) is the active policy, under which seven states and Washington DC have passed legislation aimed at regulating autonomous vehicles while one state has issued an executive order. However, those regulations do not concern the operations, sales, and use of autonomous vehicles for the public but rather regulate the testing of vehicles (Tarpley & Jesma, 2016). In Australia, AVs must get an exception from current regulations if they intend to operate on public roads. Having said this, however, The Automated Vehicle Safety Law, due in 2026, will establish an 'in-service regulator' for the technology and national standards for autonomous vehicles.

Germany is the first country in the world to create a legal framework for fully automated driving with autonomous driving functionality for cars and trucks (Bianco Levrin Giovanna Larini, 2017). Across the European Union, the European Commission is in the process of developing regulations to support the implementation of connected and autonomous vehicles, in which all European countries, including Ireland, will significantly influence the legislation adopted (Barrett, Rageade, Wallis, & Weil, 2021).

### **1.2.8 Ongoing work in the EU**

In the international context, the Transport Ministers of the G7 States and the European Commissioner for Transport underlined the need to take appropriate steps

to establish a harmonized regulatory framework that would enable safe deployment of automated and connected driving technologies across national borders.

In the framework of the Digital Agenda for Europe (Force, 2015), enabling intelligent transport systems and automated and connected vehicles is a horizontal task for transport policy and economic policy. The Connecting Europe Facility and the Investment Plan for Europe (Pillath, 2016) have set as important targets the stimulation of investments in broadband networks and transport infrastructure, which are necessary for effective Connected Intelligent Transport Systems (C-ITS).

### **1.3 RESEARCH OBJECTIVES**

The thesis investigates aspects of implementation of AV in real-world mobility environment. The brief objectives of this thesis are:

- To analyse people's perceptions as regards AVs once they have used one, utilising a Technology Acceptance Model (TAM).
- To conduct a comprehensive performance evaluation of five detection algorithms—Faster R-CNN, Cascade R-CNN, RetinaNet, FCOS, and Deformable DETR—utilizing benchmark datasets such as Cityscapes, KITTI, and Eurocity Person. The goal is to conduct a comprehensive comparative analysis, evaluating how these algorithms perform in scenarios that closely mimic real-world complexities. The unique aspect of this work lies in its extensive approach, comparing the algorithms across multiple datasets and benchmarking them against existing urban road datasets. The overarching goal is to scrutinize the efficiency of the detection algorithms and ensure their suitability for real-time applications, thereby contributing to advancements in safety measures for autonomous systems and intelligent transportation.
- To create a unique traffic benchmark dataset after taking into consideration different kinds of weather, lighting, and traffic scenes, thereby testing the efficiency of the state-of-art detection algorithms in traffic scenarios, and further modifying them to suit traffic scenes.
- To develop a single end-to-end algorithm to estimate, track, and detect different road objects.

The work in the thesis was carried out to achieve all the proposed research objectives. Initially, a study was conducted to carry out an in-depth analysis of people's perceptions concerning Autonomous Vehicles (AVs) by utilizing the Technology Acceptance Model (TAM), shedding light on crucial insights into the public's acceptance and concerns regarding this emerging technology. This study was followed by comparison of performances of various deep-learning based object detection algorithms (Faster RCNN, Cascade RCNN, RetinaNet, FCOS, Deformable DETR) using benchmark datasets (Cityscapes, Eurocity Person and Kitti), comparing results to ensure not only efficiency but also real-time application safety, thus advancing the development of AVs. For improved comparison a diverse traffic benchmark dataset collating images depicting various weather conditions, lighting scenarios, and traffic scenes was developed. This dataset serves as a vital resource for evaluating state-of-the-art detection algorithms in complex traffic scenarios and facilitating necessary modifications. Additionally, a field dataset which consist of about 300 images was developed for distance estimation evaluation problem, the data was collected and annotated manually. The last contribution involved development of a unified algorithm capable of estimating, tracking, and detecting multiple road objects These contributions collectively support the understanding and advancement of AV technology, paving the way for safer and more efficient autonomous transportation systems.

#### **1.4 ORGANIZATION OF THE THESIS**

This thesis is presented in seven chapters following this one. In **Chapter 2**, a short description of the available literature in this field has been provided. The review has been followed by description of the algorithms and bench-mark image datasets used in this thesis.

**Chapter 3** presents a qualitative study to comprehensively measure the acceptance of autonomous vehicles, to identify participant concerns, to develop an experimental evaluation of AV acceptance yet to propose solutions for the future adaptation of AVs.

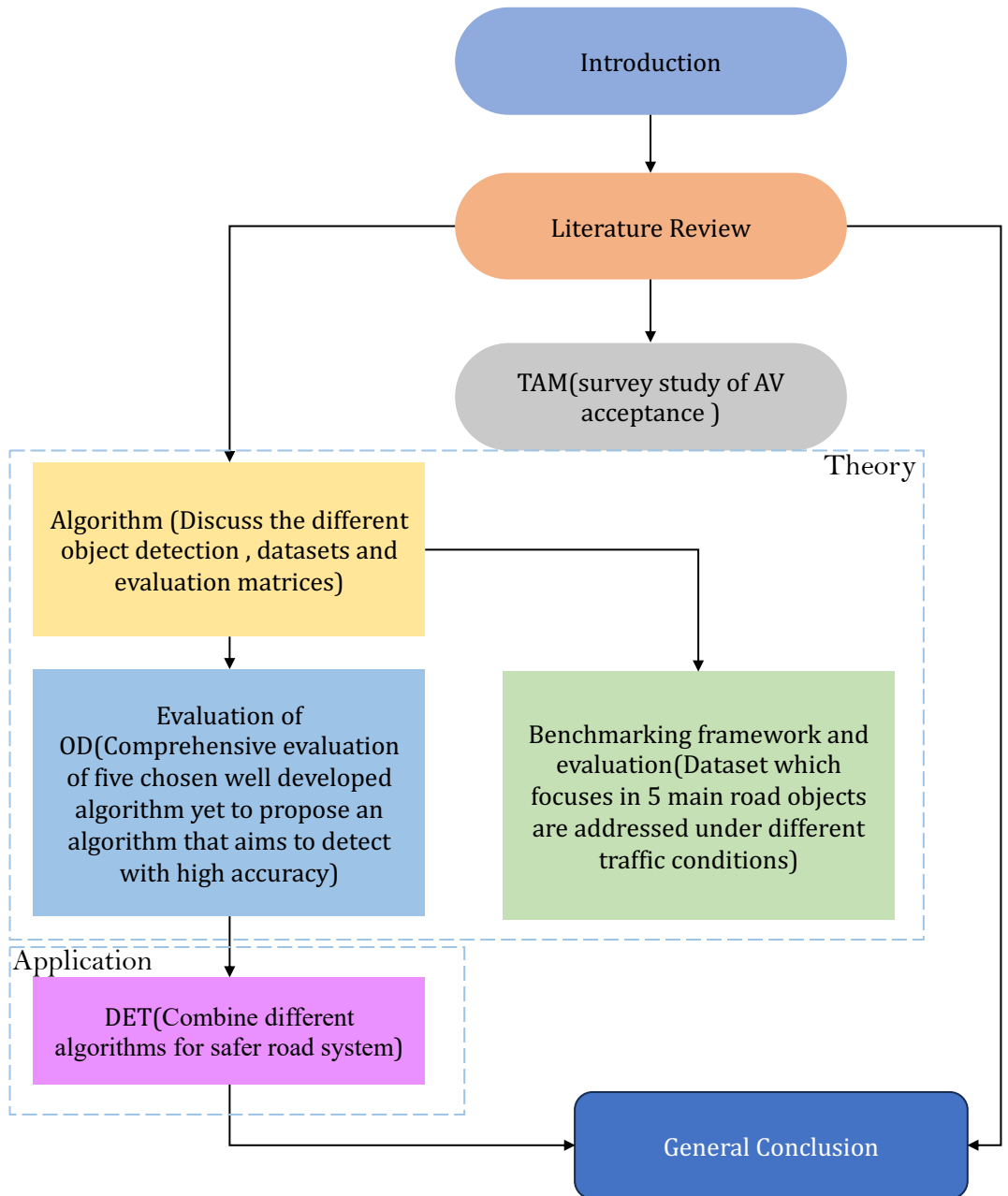
Detection is an area of investigation as it is the first and main step to achieve the aim of this thesis. In **Chapter 4** will build on the fundamentals of the previous chapters. This chapter will evaluate detection performance using different evaluation matrices under different conditions of datasets and different object sizes.

**Chapter 5** will develop an urban road dataset where it will standardize the performance. It will highlight the unique aspects of the dataset and the different performance matrices used for evaluation.

A combined model will be developed in **Chapter 6** that is based on the perceptions gained from the previous chapters. The model is a combination of three aspects: algorithm detection, tracking, and estimation. The aim of this model is to increase the safety of pedestrians and cyclists on the road, and to overcome the literature gap where no combined algorithm has been developed yet for road objects, mainly cyclists and pedestrians.

**Chapter 7** will conclude the thesis by summarizing the contributions, evaluating the overall performances of the proposed algorithm, discussing the implications, and indicating directions for future research.

The following figure (Figure 1.9) illustrates the roadmap of the thesis.



**Figure 1.9** Thesis roadmap

## 1.5 DISSEMINATION FROM THESIS

Papers from different chapters in the thesis have been presented at the following international transportation conferences:

- Alshkeili, A., and Ghosh, B.

Detection of Pedestrian and Cyclists from Automated Vehicles using Image Analysis the 55th UTSG Annual Conference 2023 - Cardiff

- Alshkeili, A., Qiu, W. and Ghosh, B.  
Cyclist and Pedestrian in Autonomous Vehicles View  
In Proceedings of the Irish Transport Research Network (ITRN2023)
- Alshkeili, A., and Ghosh, B.  
User Acceptance of Adoption of Autonomous Vehicles  
Transportation Research Procedia  
Transport Research Arena (TRA) Conference (TRA 2022)
- Alshkeili, A., Qiu, W. and Ghosh, B.  
Detection, Estimation & Tracking Road Objects for Assisting Driving.  
In Proceedings of the 7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)
- Alshkeili, A., Qiu, W. and Ghosh, B.  
Cyclist and Pedestrian in Autonomous Vehicles View  
In Proceedings of the Irish Transport Research Network (ITRN2019)

---

# Chapter 2

---



## 2 Literature review

This research looks at the topic from two different aspects: one is object tracking and detection using machine-learning algorithms while the other is the technology acceptance of the adoption of autonomous vehicles in the transport system. The most relevant research in each of these fields is discussed in this chapter, so as to provide context for the original work to follow.

### 2.1 TECHNOLOGY ACCEPTANCE MODEL

The Technology Acceptance Model (TAM) is an information systems theory that illustrates how users accept and use technology. It was tailored from the Theory of Reasoned Action (TRA). TRA is a social psychology model used for the determination of consciously intended behaviour (Davis, Bagozzi, & Warshaw, 1989). In this model, a person's behaviour is determined by his/her behavioural intention (BI), which determines the person's attitude (A) and the subjective norm (SN) related to the behaviour, with a typical weight being calculated by:

$$BI = A + SN.$$

2.1

The following figure illustrates the path used to estimate the actual behaviour on the basis of TRA.

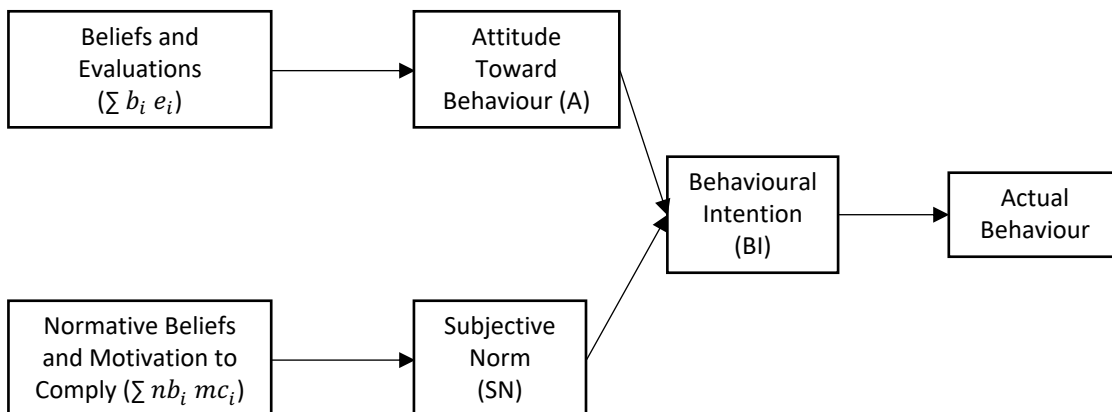


Figure 2.1 Theory of Reasoned Action (TRA)

The TRA model was used in various research areas that aim to understand the limitations of theory, test assumptions, and analyse several refinements and extensions (Davis et al., 1989).

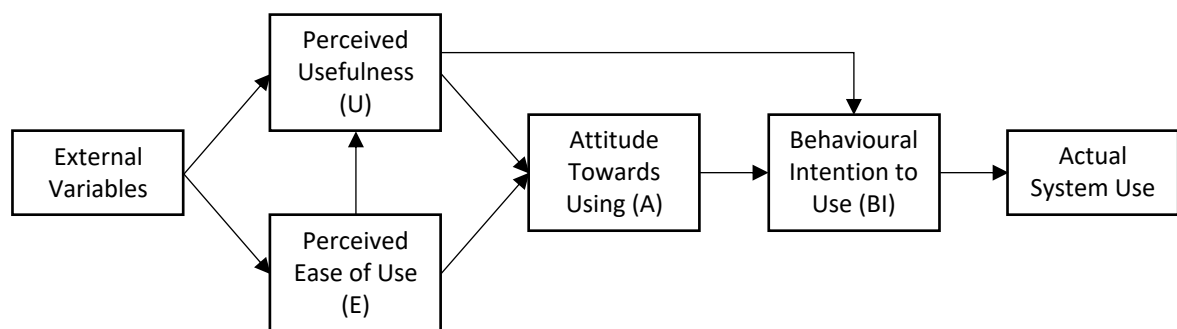
The TAM model, which is an adoption of TRA, was developed initially by Davis in 1986 to explain computer use behaviour. The model was adopted to examine users' acceptance of information systems (Davis et al., 1989). The goal of TAM was to explain the determination of computer acceptance. TAM was also developed to explain user acceptance, deploying theoretical and mathematical justifications to use fewer input variables. An essential purpose was to provide a basis for tracing the input of external factors on internal beliefs, attitudes, and intentions. The model consists of several variables that explain the behavioural intentions in relation to the use of technology (Scherer, Siddiq, & Tondeur, 2019). Moreover, TAM proposed two theories: perceived usefulness and perceived ease of use relevant to computer acceptance. Perceived usefulness (U) is the perspective of the subjective probability of users, which refers to the fact that the use of such a system will increase their job performance within an organizational context. Perceiver ease of use (EOU) refers to the degree to which potential users expect the target system to be free of effort.

In TAM, in order to estimate BI, personal attitude towards the use of System (A) and perceived usefulness (U) are joined, and then the following equation is used to calculate the estimated weight:

$$BI = A + U.$$

2.2

The following figure illustrates the TAM computer acceptance behaviour.

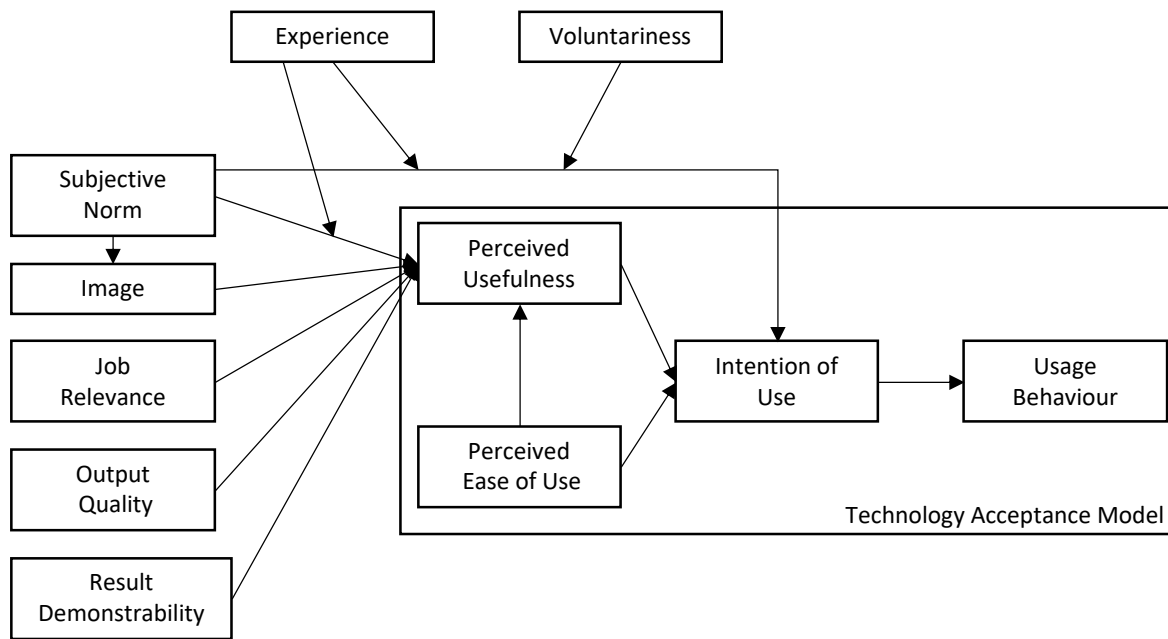


**Figure 2.2** Technology Acceptance Model (TAM)

This acceptance model has gone through several extended phases. In 2000, Venkatesh and Davis conducted a study to extend TAM, which assessed how the perceived usefulness and intention of use constructs change with continued information system (IS) use (Venkatesh & Davis, 2000). TAM 2 included more

information on the acceptance; it shows that subjective norms exert a significant direct effect on intentions of use over and above perceived usefulness (PU) and perceived ease of use (PEOU) for mandatory systems.

The following figure demonstrates the modification made by Venkatesh and Davis for the initial TAM model.



**Figure 2.3 TAM 2**

The proposed TAM 2 (Venkatesh & Davis, 2000), as provided in Figure 2.3, gave explanations in greater detail. For these reasons, customers discovered a given system that is useful in respect of three factors in time: pre-implementation, one-month post-implementation, and three months post-implementation. TAM2 theorizes that the mental evaluation on the part of users of the fit between essential goals at work and the consequences of performing job duties using the device serves as a foundation for forming perceptions related to the usefulness of the system (Venkatesh & Davis, 2000). TAM2 performed well in every voluntary and obligatory environment (Venkatesh & Bala, 2008). Although this model had positively impacted the understanding of user behaviours towards the adoption of IT, the model identified general determinants of the perceived ease of use. It was developed separately from the initial TAM; hence, there was no idea if it is valid to combine them to overcome the backdrops of understanding behaviour intention. Therefore, an integrated model of determinants of perceived usefulness and perceived ease of use that empirically validates the model and uses the integrated

model as a springboard to propose future directions for research on interventions (Venkatesh & Bala, 2008) was implemented.

In 2008, Venkatesh and Bala carried out research that looks at a broader view of IT acceptance. Their main aim was to enable managers in organizations to decide how to implement the IT system that can lead to its greater user acceptance and effective utilization. TAM3 presents a numerical network of the determinants of individuals' IT adoption and use.

They developed an integrated model of acceptance TAM3 (Figure 2.4) by combining TAM2 and the model of the determinants of perceived ease of use (Davis et al., 1989).

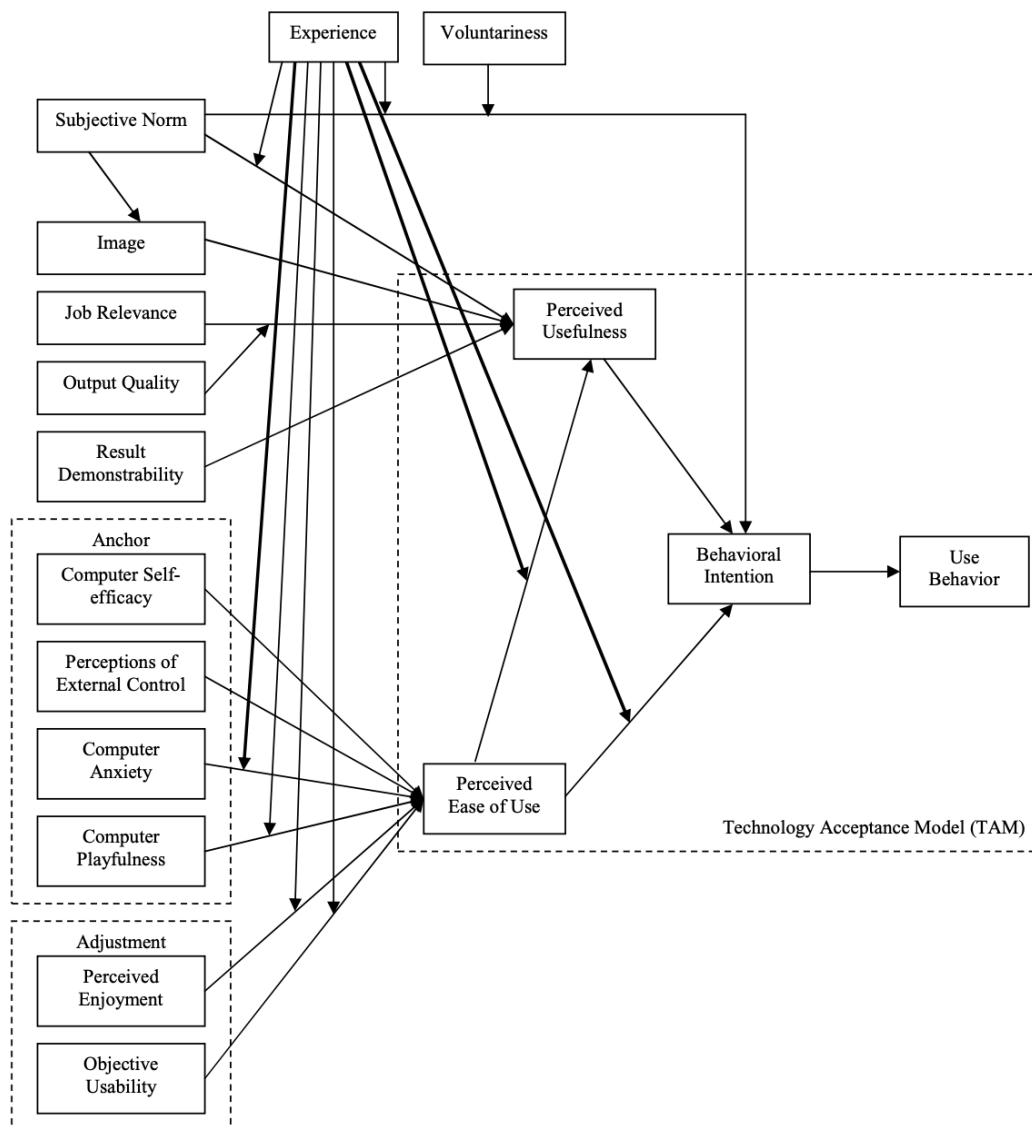


Figure 2.4 TAM 3

TAM3 consisted of four different types: individual differences, system characteristics, social influence, and facilitating conditions, which were the determinants of perceived usefulness and perceived ease of use. This model was then tested in a real-time setting for IT implementation.

Moreover, a broader acceptance model—Unified Theory of Acceptance and Use of Technology (UTAT)—was then developed by Venkatesh et al. (Venkatesh, Morris, Davis, & Davis, 2003). This model was found to understand the organizational outcomes associated with new technology.

### **2.1.1 Autonomous Vehicle acceptance**

The technology acceptance model, which aims to study the public acceptance of adapting to autonomous vehicles as one of the modes of transportation, is a research area that researchers are interested in. The term 'Autonomous AND Vehicle AND Survey' received more than 2,000 results on Scopus. Studies showed that people are willing to adopt to AV, however some concerns regarding safety and cost are yet to be dealt with. For instance, a study by P. Liu et al. (P. Liu, Guo, Ren, Wang, & Xu, 2019) which was held in China showed that about 39.3% of participants are willing to pay less than \$2,900 while 34.3% are willing to pay \$2,900. Moreover, most surveys were conducted online using different online tools. Liljamo, Liimatainen, and Pöllänen (Liljamo, Liimatainen, & Pöllänen, 2018) have conducted a study using focused group in different countries across the globe with a total participation of 2036 to examine whether people are ready for automated vehicles and what concerns people have that hinder the adoption of these vehicles and conclude that traffic safety and ethical perspectives play a key role in the acceptance of automated vehicles. A web based survey study carried in the USA by Bansal and Kockelman (Bansal & Kockelman, 2017) to study the long-term AV adaptation in the US and the willingness to pay for technology. 2167 participated in the survey and results showed that by 2045, Level 4 AVs are likely to be adopted to the extent of 24.8–87.2% of the vehicle fleet. Yuen et al. (Yuen, Cai, Qi, & Wang, 2021) have developed a TAM study with a total of 274 participant to examine the factors influencing a user's behavioural intention to use AVs. In Table

2.1 a concise list of research studies conducted to investigate the acceptance of autonomous vehicles has been presented. This compilation serves as a valuable resource for understanding the evolving landscape of autonomous vehicle acceptance, highlighting the multifaceted perspectives, and contributing to a broader comprehension of this critical aspect of emerging transportation technology. Notably, it is worth mentioning that the table underscores a unique aspect of this thesis: a departure from the conventional reliance on hypothetical scenarios in previous studies. Instead, this research distinguishes itself by directly involving participants with access to autonomous vehicles, thereby analyzing acceptance based on real-life experiences. This distinctive approach enriches the field by offering a more authentic and experiential perspective.

**Table 2.1** Summary of AV acceptance survey paper

<i>Author</i>	<i>Location</i>	<i>Target Participants</i>	<i>Method</i>	<i>Objectives</i>	<i>Insights and Conclusion</i>	<i>Sample Size</i>
(Acheampong & Cugurullo, 2019)	Dublin	Random Sample	Two-step survey process	Capture the possible behavioural influences on individuals' AV adoption decisions.	A two-step survey was conducted, wherein the first step involved an online pilot survey with 50 participants. The second step entailed posting the survey link on social media (Twitter and Facebook), distributing leaflets with scannable QR codes, and sending emails to students and staffers of all major universities in Dublin. The study showed (a) AV interest and adoption intentions and (b) user adoption decisions regarding three different AV modes: ownership, sharing, and public transport.	507
(Bansal, Kockelman, & Singh, 2016)	USA	Random Sample	Internet-based	Understand participants' opinions on smart car technologies and strategies.	The study showed a willingness to pay for adding full (Level 4) automation, and the results of the survey can be used to develop smarter transportation systems for more efficient and sustainable travel.	374

(P. Liu, Guo, et al., 2019)	China	Random Sample		A survey to investigate the willingness to pay in order to adopt AV technology in daily life.	The aim was to provide insights to assess the value of self-driving vehicle technology in the vehicle market. It concluded that in China, younger generations with high incomes are willing to adopt this technology; however, the older generation has some uncertainty. Results show that about 39.3% of participants are willing to pay less than \$2,900 while 34.3% are willing to pay \$2,900.	1,355
(Nordhoff, De Winter, Kyriakidis, Van Arem, & Happee, 2018)	116 countries	Random Sample	Online survey	A survey to study the acceptance of driverless vehicles and sociodemographic characteristics.	The study concluded that domain-specific attitudes more strongly determine self-reported acceptance of driverless vehicles than sociodemographic characteristics.	10,000



(Bansal & Kockelman, 2017)	USA	Participants had a good understanding of AV	Web-based survey	A survey to study long-term AV adoption in the US and the willingness to pay for AV technology to help traffic engineers, planners, and policymakers propose long-term plans.	It concluded that by 2045, Level 4 AVs are likely to be adopted to the extent of 24.8–87.2% of the vehicle fleet	2167
(Brell, Philipsen, & Ziefle, 2019)	Germany		Focus group	A survey to explore risk perceptions towards connected and autonomous driving compared to conventional driving.	The results will foster the successful implementation of AVs on the German market and develop public information strategies.	516
(Kyriakidis, Happee, & De Winter, 2015)	109 countries	Random Sample	Internet-based	A survey that aimed to investigate user acceptance, concerns, and willingness to buy partially, highly, and fully automated vehicles.	The results showed that the public has concerns about legislation, legal issues, and software hacking.	5,000

(Liljamo et al., 2018)	Finland	Random Sample (from China, India, Japan, the USA, the UK, and Australia)	Focus group	This study aimed to examine whether people are ready for automated vehicles and what concerns people have that hinder the adoption of these vehicles.	The study dealt with the concerns that people have which hinder the adoption of AVs. The results indicated that traffic safety and ethical perspectives play a key role in the acceptance of automated vehicles.	2,036
(P. Liu, Yang, & Xu, 2019)	China	Random Sample of Tianjin citizens	Direct intercepts by well-trained interviewers	This study aimed to understand the acceptance of AV	The survey aimed to understand that acceptance is based on three main aspects: general acceptance, willingness to pay, and behavioural intentions. The results proved that psychological factors influence initial public acceptance.	441

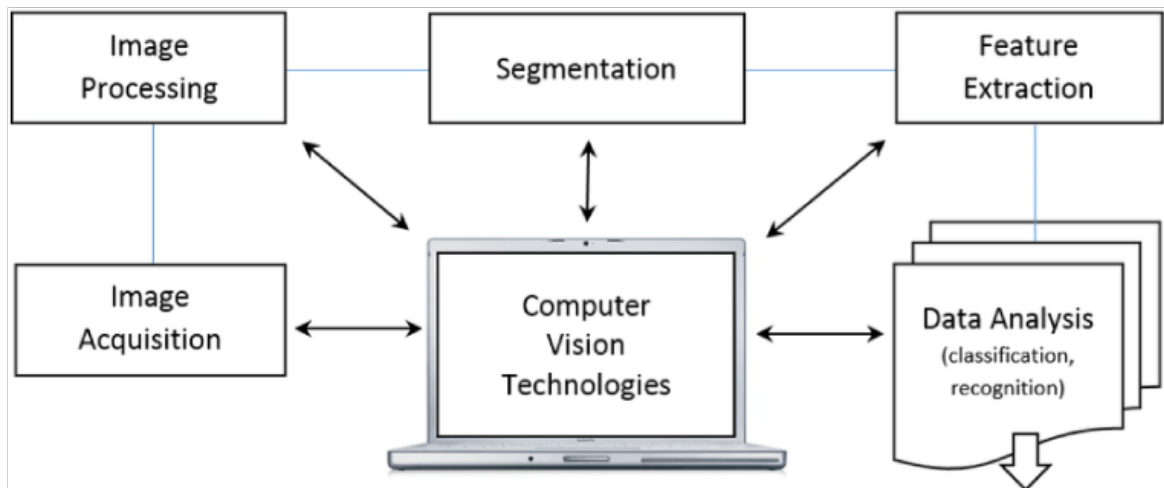
(Sanbonmatsu, Strayer, Yu, Biondi, & Cooper, 2018)	USA	Paid survey: Amazon Mechanical Turk Workers	Online survey	A survey to measure consumers' beliefs about fully automated vehicles and their confidence in their beliefs.	The survey results showed that AV tends to be viewed favourably among knowledgeable customers compared to customers with less knowledge regarding AV, and 72% of respondents indicated that they are 'somewhat certain' or 'highly certain' about AV beliefs. The study concluded that educating the nation about AV will lead to wider acceptance and adoption of such technology.	114
(Huang & Qian, 2021)	China	Random sample	Online survey	A survey that aimed to examine customers' negative attitudes and intentions towards the adoption of AV and how their psychological traits moderate the relationships.	The survey concludes that the psychological trait of the need for uniqueness strengthens the association between consumers' reasoning for AVs and their intention to adopt.	849

(Yuen et al., 2021)	China	-	TAM	This study applied an integrated model based on innovation diffusion theory (IDT) and the technology acceptance model (TAM) to examine the factors influencing a user's behavioural intention to use AVs.	perceived usefulness (PU) and perceived ease of use (PEOU) positively influence users' behavioural intention to use AV	274
(Rejali, Aghabayk, Esmali, & Shiwakoti, 2023)	Iran	Drivers with valid license	Online survey	A survey to assess user acceptance of fully automated vehicles in a middle-income country	comparing three popular user acceptance models: Technology Acceptance Model (TAM), Theory of Planned Behavior (TPB), and Unified Theory of Acceptance and Use of Technology (UTAUT)	1381

(Hógye-Nagy, Kovács, & Kurucz, 2023)	University of Debrecen and Széchenyi István University in Győr	Member of a university community	Online Survey	investigates the acceptance of autonomous cars based on the role of attitudes toward autonomous vehicles, acceptance of technology, previous experiences, and gender.	investigates the acceptance of autonomous cars based on the role of attitudes toward autonomous vehicles, acceptance of technology, previous experiences, and gender. The optimism factor of technology adaption propensity affected the acceptance.	1273
--------------------------------------	--	----------------------------------	---------------	---	--	------

## 2.2 COMPUTER VISION

Computer vision (Keller & Wang, 1995) is a field of computer science that focuses on replicating parts of the complexity of the human visual system and authorizes computers to identify and process objects in both images and videos in the same manner as humans do. For computers to reach a high level of analysis, it involves several lower-level processes like artificial intelligence, which, in turn, consists of several lower-level processes like deep learning and others. Computer vision is based on algorithms that use mathematical equations to compute 3D image components. Figure 2.5 illustrates an example of a simple computer vision process in which several inputs are fed into the algorithms for the outputs to be produced. Richard Szeliski, in his book *Computer Vision: Algorithms and Applications* (Szeliski, 2010), presents a historical review of computer vision. In the early 1970s, computer vision was viewed as a visual perception component of an ambitious agenda to mimic human intelligence and empower robots with intelligent behaviours (Szeliski, 2010). Many universities like Massachusetts Institute of Technology (MIT), Stanford, and Carnegie Mellon University (CMU) focused on developing the technology. In the 1980s, a more complicated mathematical technique was developed to perform quantitative analysis for both image and scene. In the 1990s, developments in computer vision continued; however, a significant development in tracking, optical flow, and a multi-view stereo algorithm was noticed. In the 2000s, a profound interplay between vision and the graphics field was noticed. Progress in this field continues till date.



**Figure 2.5** Computer vision

### 2.3 IMAGE PROCESSING

Image processing is a sub-category of signal processing and is a common area of research; it is used in most of the different fields that use technology for analytical purposes. These include signal processing, object detection, and medical screening. Hence, it helps improve pictorial information for human interpretation and the processing of scene data for autonomous machine perception. This improvement is achieved by applying some operations on images to enhance or useful information extraction for the purposes of study or analysis (K. Chen, Lui, & Modersitzki, 2019). Image processing is of two main types: digital and analogue processing. Analogue image processing is used for hard copies like photos and printouts. The manipulation of information using computers is the digital image processing explored in more detail in this literature.

Digital image processing—as shown in Figure 2.5—is where the camera captures an image, and then sends it to a digital system in which processing occurs; a processed output is finally obtained. It involves three main phases: pre-processing, enhancement, and display of information extraction. It can be divided into image enhancement, image restoration, image analysis, and image compression (da Silva & Mendonca, 2005). Image enhancement entails extracting or improving the perception of information in images for human viewers that can be used for other image processing techniques (Aber, Marzloff, Ries, & Aber, 2019; Maini & Aggarwal, 2010). Image restoration is the technique used to re-establish an

entrapped image—usually a blurred or noisy image—so that it can be reverted (Reeves, 2014). The image analysis technique provides numerical outputs that are generated automatically for extracting information from processing images (da Silva & Mendonca, 2005). Examples of image analysis are image segmentation, edge extraction, and texture and motion analysis.

### **2.3.1 Image enhancement**

As mentioned, image enhancement is the processing of an image, so that the output image is enhanced/improved without losing the information for specific applications (Maini & Aggarwal, 2010). This technique is used to increase the quality of the image (Ackar, Almisreb, & Saleh, 2019). It is applied to different fields in which images must be understood and analysed (Maini & Aggarwal, 2010). Several techniques are used in image enhancement, based on application, image type, greyscale, or coloured image scale. It is used in various field of imaging, such as medical imaging (Kaur, Chawla, Khiva, & Ansari, 2017), underwater imaging (Sahu, Gupta, & Sharma, 2014), video and image defogging, intelligent transport systems (Bubeníková, Muzikářová, & Halgaš, 2012), and much more.

### **2.3.2 Image restoration**

Researchers have defined image restoration as the restoration of images that have been corrupted by an additional blueness because of a lack of focus (Woods, 2012). It is considered the first step before image analysis; it, therefore, improves the quality of the image by smoothing noise without smoothing edges (Descombes, 2018). Image restoration is used in different fields, such as the medical field, optics, and astronomy. M. Banham and A. Katsaggelos published an extended review of digital image restoration, wherein they mentioned the different fields where this technology is used. They focused on astronomy involving several mathematical linear and non-linear equations (Banham & Katsaggelos, 1997). An initial step in restorations is to decode signal to digital images: many research papers have proposed mathematical equations to compute this transformation. For instance, Schulz stated that a common approach to solve an image restoration is to solve an





edge detection, and classification, according to (Blaschke, 2010). Moreover, microscopic images have complex and nonuniform structures; hence, many images have to be taken for analysis to take place. However, the use of image processing—specifically image analysis—enables the extraction of relative information like the compositions of image, particle size and density, and intensity profile (“Image Processing and Image Analysis,” 2008). Fourier transformation is a special image analysis in which algorithms are applied to determine the periodicities in micrographs; its main purpose is to overcome the speed limit issue indicated by using the traditional image analysis algorithms (Goda & Jalali, 2013). In transportation image analysis, systems are used to automatically analyse traffic data collected from road CCTVs. It is used for particular applications, such as to monitor speed, the number of vehicles, length, and lane occupancy information (Waterfall & Dickinson, 1984).

#### **2.3.4 Applications of image processing**

Transportation has integrated the IT system into a much more sophisticated, developed intelligent system. Transporters have introduced algorithms, sensors, and software that help detect and track road objects to make the transportation sector comprehensive and much more reliable. The use of image processing enabled research and developers to obtain specific and accurate locational and navigational information.

Waterfall et al. (Waterfall & Dickinson, 1984) conducted a study on image processing for traffic management to monitor and measure the flow of vehicles and pedestrians on roads and public transport systems. A combination of image processing and measurement techniques is used. Monochrome sensors are used to measure the flow; this technology offers high reliability at a low cost, which is why it is adopted. The other technology adopted is based on image processing. They have developed an image primitive algorithm relatively unaffected by variations in lighting to achieve the aim of the study.

Oliver et al. (Oliver, Baxter, & Wallace, 1996) presented the most common positioning techniques, which involve the use of different distance- or direction-dependent measurements. They covered the received signal strength (RSS)

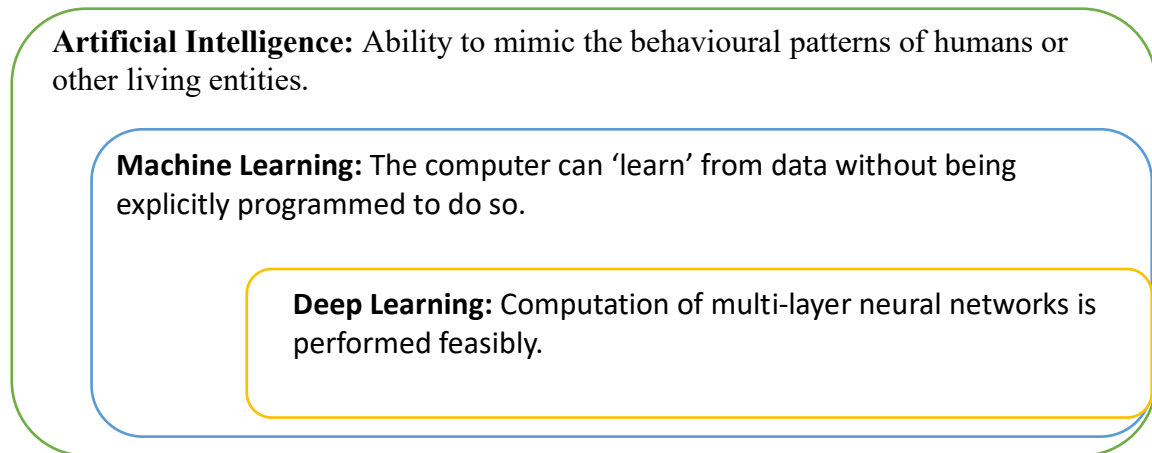
techniques, which can estimate the distance between objects, given the signal-transmitted power on the distance between a receiver and the transmitting source. This technique is used in spoofing attacks (Figueiredo & Jain, 2002). Signal propagation time—a terminology that refers to time of arrival (TOA)—is used to describe a distance-dependent technique that can be directly inferred if the medium propagation speed is known. The angle of arrival observation (AOA) provides locational information about the emitter.

Locational information is mainly provided using the global positioning system (GPS), where signals are received, processed, and both location and distance are obtained. This technology is used widely in ITS, apart from detection and tracking.

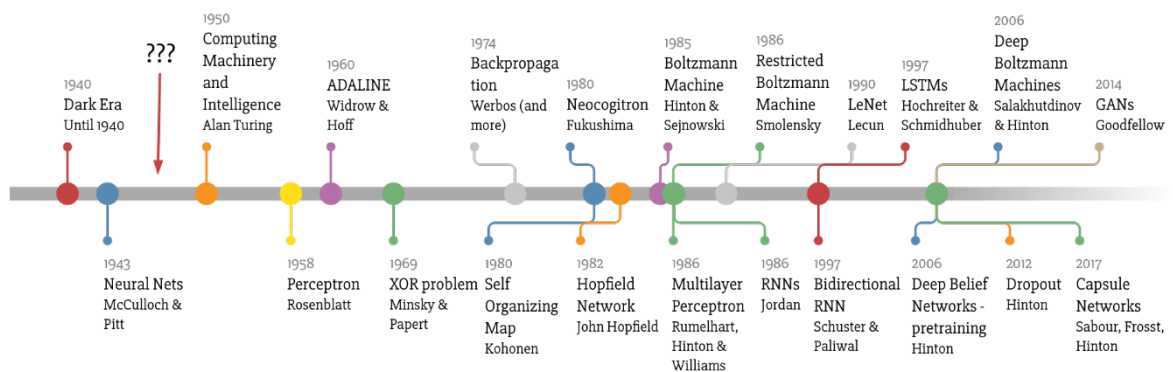
## 2.4 DEEP LEARNING-BASED OBJECT DETECTION

Deep learning is an artificial intelligence (AI) function; it is a subset of machine learning, as shown in Figure 2.6. AI is an umbrella discipline that encompasses related technologies involved in making machines more intelligent. Machine Learning (ML) refers to an AI system that can self-learn based on the algorithm, and, hence, it refers to systems that get smarter over time without human interpolation. Deep Learning (DL) is a machine learning (ML) applied to large datasets. Most AI work involves ML because intelligent behaviour requires considerable knowledge. *Deep learning* is a multi-layer neural network that is used for decision-making. It is treated as a brain in which it receives new information/data and performs several computations to produce useful output information (Le et al., 2011). Deep learning can also be referred to as a neural network. The early development of the neural network was in the 1940s, where the initial objective was to simulate the human brain system to solve general learning problems in accordance with a set of principles. Werbos (Werbos, 1974), in his doctoral dissertation of 1974, determined the method used to train artificial neural networks by deploying the backpropagation of errors. Hinton et al. (Rumelhart, Hinton, & Williams, 1986) proposed a backpropagation algorithm that was popular in the 1980s and 1990s. Salakhutdinov et al. (Salakhutdinov & Hinton, 2009) in 2006 introduced unsupervised pretraining and a deep belief net. Figure 2.7 (Liao & Poggio, 2016) shows a deep learning timeline, in which the

development of deep learning is highlighted over a time period, based on several deep learning papers between 1940 and 2017.



**Figure 2.6** Relationship between Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL)

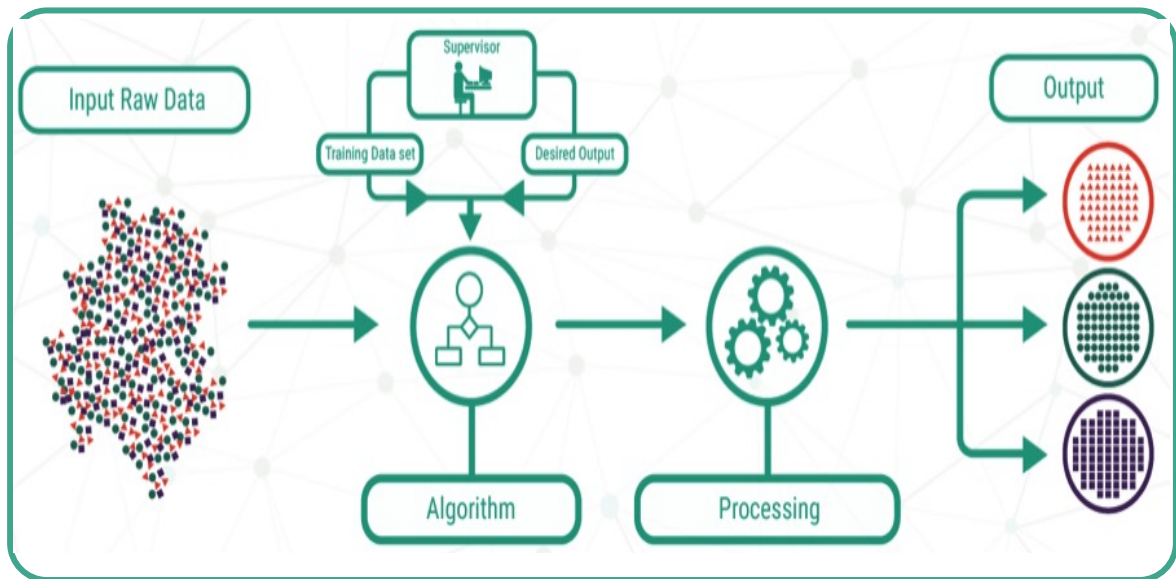


Made by Favio Vázquez

**Figure 2.7** Deep learning timeline by Favio Vazquez: from 1940 to 2017

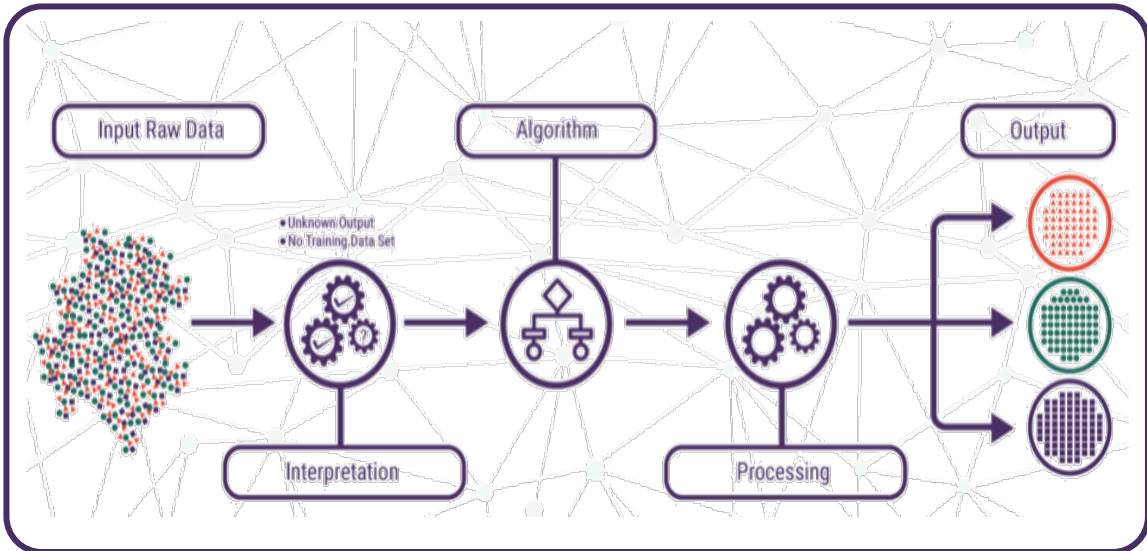
Feeding the output into the system and mapping input to output to produce more accurate predictions is how supervised learning operates. Figure 2.8 shows the process that supervised learning follows, wherein several mapping processes are applied to the input data; it, therefore, finally produces a classified output. This type of learning can be used for different purposes, such as bioinformatics. In (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, Fei-Fei, et al., 2015) the authors proposed a non-discrimination from the perspective of supervised learning, wherein the proposed method aimed to predict

an actual output  $Y$  from features  $X$  based on labelled trained data. Limiting disparate impact was proposed by Feldman et al. (Hardt, Price, & Srebro, 2016). Cunningham et al. (Cunningham, Cord, & Delany, 2008) summarized supervised learning algorithm in six steps: establishing the training type, converge a training set, resolving the input feature, resolving the formation of the learned function and comparable machine learning algorithm, embrace the design and execute the learning algorithm on the collected training set, and, finally, evaluate the accuracy of the learned function. P. Viola et al. (Viola & Jones, 2004) published a book in which supervised learning is the focus. Graves (Graves, 2012) pointed out that the nature and degree of supervision provided by the targets vary greatly between supervised learning tasks. (Waterfall & Dickinson, 1984) supervised learning algorithms induce models from training data, and these models then can be used to classify unlabelled data. The main limitation of supervised learning based on (Chaitanya et al., 2020) is the dependence on human labelling, where it limits the number of categories. Secondly, the discriminative models lack interpretability because they do not produce mid-level representations. Furthermore, for learning to be completed, the system needs data for training models. Benchmarking datasets like Caltech (Dollár, Wojek, Schiele, & Perona, n.d.), KITTI (Geiger, Lenz, Stiller, & Urtasun, 2013), ImageNet (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, & Fei-Fei, 2015), PASCAL VOC, MS COCO(Caesar, Uijlings, & Ferrari, 2018), and Open Images V5 (Kuznetsova et al., n.d.) are used for training.



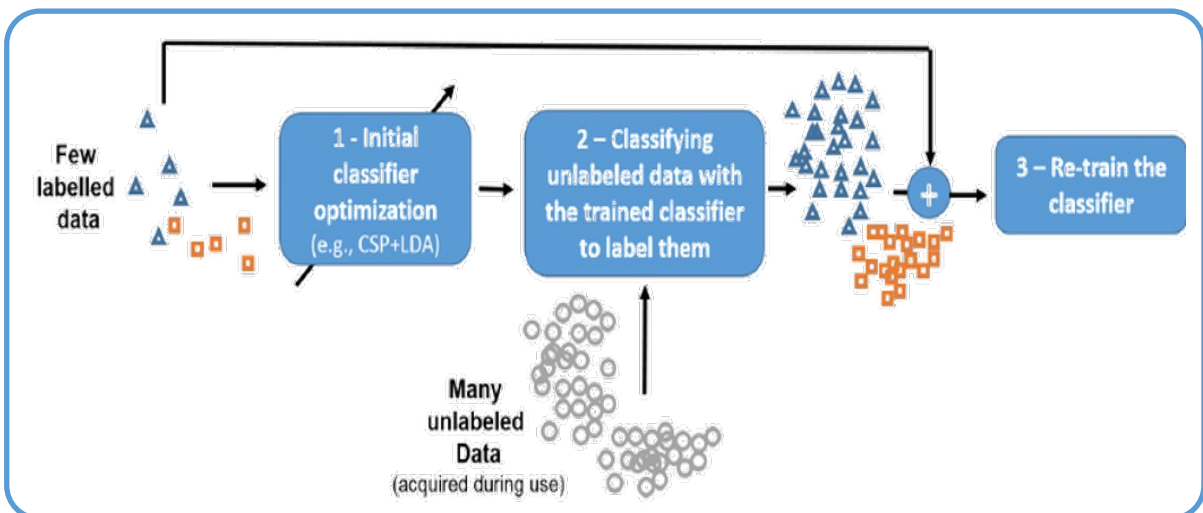
**Figure 2.8** Supervised learning architect

Unsupervised learning is the method concerned with how the system can learn to represent particular input patterns that reflect the statistical structure of the overall collection of input patterns. (Oliver et al., 1996) applied a Minimum Message Length using unsupervised learning methodology to estimate the number of components  $k$ . (Figueiredo & Jain, 2002) used unsupervised learning to address multivariable data for a finite mixture model. (Le et al., 2011) proposed high-level, class-specific feature detectors using unlabelled data. Figure 2.9 represents an unsupervised learning method, where raw data is fed as input and a classified output is produced. The main difference between both methods (supervised and unsupervised) is that under supervised learning, the output is known and fed into the system. In the unsupervised system, in contrast, if the outputs are unknown and the data unlabelled, the system must learn on its map outputs.



**Figure 2.9** Unsupervised learning architect

(Goldberg, 2009) defined semi-supervised learning as the learning model concerned with studying how computers and natural systems (i.e., humans) learn in the presence of labelled and unlabelled data. This learning aims to change the learning behaviour and proposes algorithms that consider both types of data. F. Du et al. (Du, Zhu, Liu, & Yang, 2020) proposed a semi-supervised method for productive mapping: they applied their proposed algorithm in two soil mapping case studies, and the proposed method achieved high accuracy. However, some drawbacks can be seen, such as less sensitivity to certain field samples. K. Chaitanya et al. (Chaitanya et al., 2020) also used this learning method for medical imaging, wherein they performed an experiment using open data; their findings showed that such technology is not very useful for such application performances. Figure 2.10 illustrates a brief workflow of a semi-supervised learning framework.



**Figure 2.10** Semi-supervised learning architect

### 2.4.1 Object detection and tracking

Object detection aims to determine whether there are any instances of objects from categories (i.e., humans, vehicles, animals, etc.) in an image frame, and to return specific aspects like location, category, and extent of each instance of such objects, using boundary boxes (Gerónimo, López, Sappa, & Graf, 2010; Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, & Fei-Fei, 2015). Object detection consists of several subtasks, including face detection, pedestrian detection, and collision detection. These are especially used in fields requiring accurate analysis of objects, such as military uses, surveillance systems, and self-driving cars. A. Gavulová et al. in 2011 (Gavulová, Pirník, & Hudec, 2011) established a national traffic information system that uses real-time images and videos for traffic control. (Viola & Jones, 2004) used object detection for face recognition, in which algorithms were applied that involved three main parts to achieve the experiment detection goal with a high rate of accuracy.

Basic deep learning-based object detection consists of three main parts: backbone, which refers to the feature extraction network; a neck, which refers to the feature computational region; and, finally, the dense region, wherein classification is performed. For a complete understanding of object detection, a review of the traditional approach and the modern approach will be discussed in the following subsections.

#### ***A) Traditional object detection approach***

The traditional object detection pipeline follows three main stages: informative region selection, feature extraction, and classification of the object, wherein the first stage—the location of the objects in the image plane—is determined. The next entails categorizing objects; finally, the object is set to the relevant category.

**Informative region selection** is the stage in which the object location is obtained, and sliding windows do this. In a frame, objects appear at different locations and have different sizes and aspect ratios; because of this issue, images are scanned using a multiscale sliding window. However, this method results in high computational costs and may produce irrelevant candidates (Z.-Q. Zhao, Zheng, Xu, & Wu, 2018).



**Feature extraction** is used to extract visual features for object recognition, wherein several techniques, such as histogram of oriented gradient (HOG) (Dalal & Triggs, 2005), Haar-like (Papageorgiou et al., 1998), and Scale-Invariant Feature Transform (SIFT) (Lowe, 1999), are applied to provide a semantic and robust representation. Owing to the different viewpoints and backgrounds, and on account of the limitations, it is a challenge to design a manual robust feature descriptor to describe the various object types (Z.-Q. Zhao et al., 2018).

**Classification** is the stage in which a classifier is used to distinguish/classify the objects into categories to make the representations more hierarchical, semantic, and informative for visual recognition.

The traditional approach is that it is computationally expensive and requires many sliding window techniques for the bounding boxes that are generated. It also requires several manual engineering features and might be insufficient to describe all object categories.

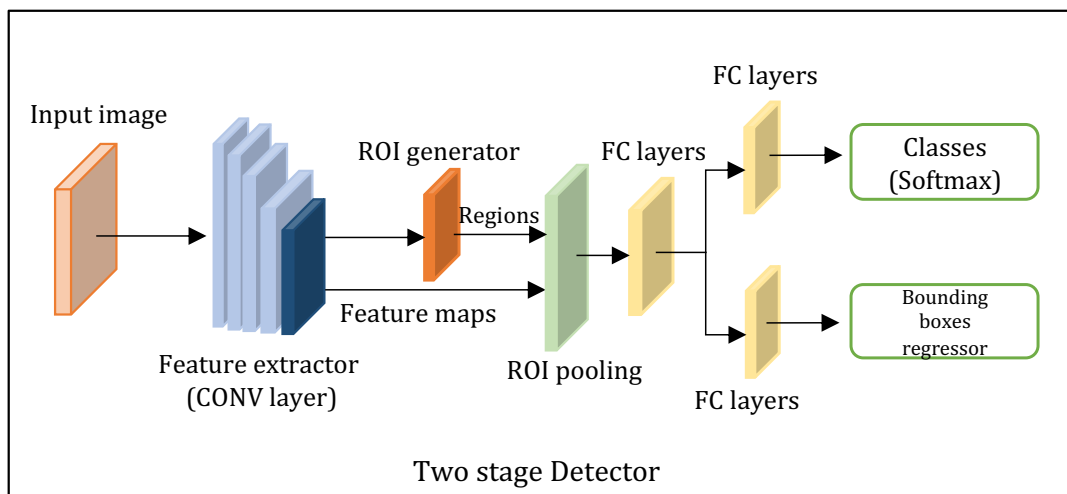
### ***B) Modern object detection approach***

The emergence of deep learning has overcome some of the drawbacks of the traditional approach. Hence, as mentioned earlier, deep learning can learn more complex features using several classification layers. In addition, the expressivity and robust training algorithms applied allow learning informative object representations without the need to design a manual features extraction.

Several object detections based on deep learning algorithms have been developed. These can be divided into two categories based on their framework. The first comprises region proposals, such as Fast RCNN (Girshick, 2015a), Faster RCNN (X. Zhao et al., 2016), and FPN (Lin et al., 2016)—these are also known as two-stage object detectors. The second category, based on regression, includes YOLO (Redmon et al., 2016), SSD (W. Liu et al., 2016), and RetinaNet (Lin, Goyal, Girshick, He, & Dollár, 2017)—these are also known as one-stage object detection.

i) Two-stage detectors

Two-stage detection or region proposal-based detection achieves a high recognition accuracy and localization precision compared to the one-stage detection, which will be discussed later. In two-stage detection, the network will have an additional stage added to the main three parts; this produces a much more accurate detection result, as shown in Figure 2.11. For this section, we will focus on the regional convolution network presented in Fast RCNN (Girshick, 2015a), Faster RCNN (X. Zhao et al., 2016), and FPN (Lin et al., 2016).



**Figure 2.11** Two-stage object detection pipeline

(Girshick, Donahue, Darrell, & Malik, 2014) proposed a method in which a selective search is used to extract 2,000 features from an image called a regional proposal. However, this proposed algorithm is not suitable for real-time application as it requires approximately 47 seconds for each image to be tested and requires a long time to train; also, a selective search algorithm is a fixed algorithm. Therefore, it might produce bad candidate region proposals. In 2015, Girshick et al. (Girshick, 2015b) proposed an improved algorithm for fast RCNN, which overcomes some of the drawbacks found in the initial RCNN algorithm.

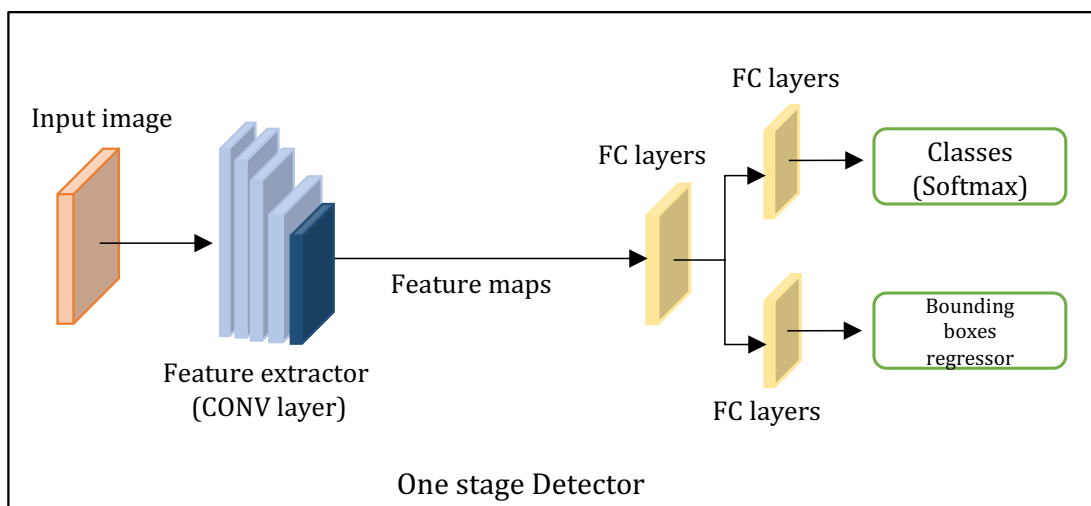
Fast RCNN (Girshick, 2015b) has an approach similar to RCNN, but, instead of feeding 2,000 regional proposals to the convolutional neural network every time, the convolution operation is completed once per command, and a feature map is generated based on it. This improvement has a significant advantage in

computational time, reducing as it does the time required for both testing and training.

(X. Zhao et al., 2016) in 2016 proposed a faster RCNN which eliminates the selective search algorithm and enables the network to learn the region proposals. The network is similar to Fast RCNN, in which an image is given as an input to the convolution feature map, and a separate network is used to predict the region proposals. These are then reshaped using the region of interest (RoI) pooling layer, which is used to classify images within the proposed region and predict the offset values for the bounding boxes.

## ii) One-stage detector

A single deep neural network is used in a one-stage detection. The main advantage of this method is the high inference speed, and, so, computational cost related to time expense is reduced (Jiao et al., 2019; Z.-Q. Zhao et al., 2018). Figure 2.12 shows a general pipeline of a one-stage detector. YOLO (Redmon et al., 2016; Redmon & Farhadi, 2017, 2018), SSD (W. Liu et al., 2016), FCOS (Tian, Shen, Chen, & He, 2019), RetinaNet, and DetectNet are some examples of regression detectors. Both SSD and YOLO will be explored further.

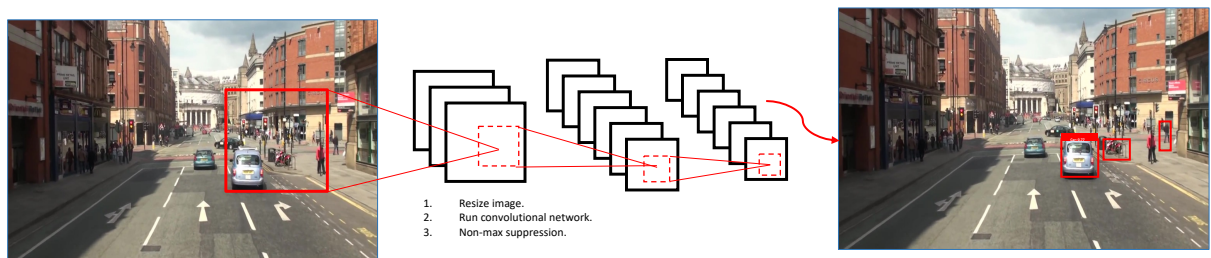


**Figure 2.12** One-stage object detection pipeline

You Only Look Once (YOLO) (Redmon et al., 2016) is a fast single-stage architecture. It overcomes the disadvantage of two-stage detection. The input must

undergo complex pipelines that are slow and hard to optimize because each component must be trained separately. However, in YOLO, the input pixels directly pass to the bounding box coordinates and class probabilities. This is done by dividing the input image into  $S \times S$  grids; each cell in the grid is used for predicting the object centre in that particular cell, and then each cell predicts bounding boxes and their corresponding confidence scores. In this system, you only look once to predict what is presented in the image and its location. In addition, the system is capable of processing 45 frames per second—it achieves relatively high accuracy for real-time detection.

Furthermore, the system uses the topmost feature map to predict both confidences for multiple categories and bounding boxes. In Figure 2.13, a YOLO detection system is presented. The system receives an image pixel, resizes the image, then undergoes a single convolution network, and, finally, thresholds the resultant detections by the model's confidence. Moreover, YOLO has gone through several modifications, which can be seen in YOLO v2 and YOLO v3. However, some limitations can be found in allocating small objects caused by strong spatial constraints imposed on bounding box predictions. It struggles to generalize the objects in new or unusual aspect ratios or configurations.



**Figure 2.13** YOLO object detection system

(W. Liu et al., 2016) proposed a single-stage object detector that overcomes some of the limitations found in YOLO, called Single Shot Detection (SSD). SSD is made of anchors adopted in the Region Proposal Networks (RPN), MultiBox, and multiscale representation. In SSD, a fixed feature map is used instead of fixed grids, where it also takes advantage of a set of default anchor boxes with different aspect ratios and scales to discretize the output space of bounding boxes. As a result, the system can detect objects of different sizes, and the network manages to predict objects from multiple feature maps with different resolutions.

RetinaNet is one stage-dense object detector that was proposed by (Lin, Goyal, et al., 2017); it achieved high-performance accuracy using anchor. It was introduced because of RPN and uses a feature pyramid similar to SSD.

#### **2.4.2 Detection of Autonomous Vehicles**

This section presents a review of recent literature on detection for autonomous vehicles, which is the focus of this thesis. However, as there are not many papers in this field, the number of resources found and analysed is limited.

Deep learning was adopted for many road detection applications such as pedestrian detection. Zhao et al.(Z. Q. Zhao, Bian, Hu, Cheng, & Glotin, 2017) used RCNN to extract robust features of a pedestrian in a complicated environment. They imply the Edge-Boxes algorithm to generate effective region proposals from an image, as the quality of extracted region proposals can significantly affect the detection performance. BoBo et al. (Bo Bo, Slembrouck, Veelaert, & Philips, 2020) conducted a study where they used a multi-camera for tracking purposes where data are fed into YOLO object detector for classification and analysis. The algorithm successfully detects using a trajectory in which the accuracy detection is 83%, and the pedestrian's accuracy is 93%. Finally, Tomè et al.(Tomè et al., 2015) proposed research for pedestrian detection based on convolution neural networks. The proposed system with a reasonable computational complexity involves a combination of Locally Decorrelated Channel Features (LDCF) as a region proposal algorithm and the fine-tuned deep convolutional neural network.

A recent paper by Chand et al. (Janai et al., 2020) proposed a framework for autonomous vehicles to detect accidents between other cars in order to take appropriate decisions, i.e., to slow down or stop. Feng et al. (Feng, Liu, Jiang, & Wang, 2020) proposed an accurate and fast object detection and localization system based on binocular vision. They based their algorithm on YOLOv3, in which they utilized MobileNet as a backbone and tested their algorithm on the KITTI dataset. Johari et al. (Johari & Swami, 2020) compared human-centred autonomy and full autonomy for self-driving cars. They also compared three object detection-based deep learning algorithms (SSD, R-CNN, and R-FCN) under different weather conditions. DeepTrackNet (Amara, Karthika, & Soman, 2020) is an end-to-end

unified deep learning framework for real-time detection, localization, and tracking for autonomous vehicles: they used Mobilenet SSD for the proposed architecture, and they achieved satisfactory results. Finally, Li et al. (P. Li, Chen, & Shen, 2019) proposed a stereo R-CNN-based 3D object detection for autonomous driving, wherein they outperformed the state-of-the-art stereo-based method by 30% Average Precision (AP).

### **2.4.3 Pedestrian and cyclist detection**

The suite of results shown on Scopus when searching the terms ‘pedestrian AND cyclist AND detection’ revealed that some papers used different methodologies in detecting, such as microwave detection, multi-layer laser scanners, etc. The forthcoming table (Table 2.2) will provide a consolidated overview of various research studies focused on the detection of pedestrians and cyclists. These studies employ object detection algorithms to address the challenges in identifying these vulnerable road users. Most research focused on evaluating a single object, for instance, pedestrians. (Tomè et al., 2015) carried out detection research where they focused on detecting pedestrians from the Caltech Pedestrian Dataset using the CNN algorithm. Additionally, (Gaspar et al., 2020) implemented a pattern detection to detect pedestrians; the proposed intelligent transport system (ITS-G5) is complex and requires further modification to be implemented, yet it can lead to increased safety of pedestrians. Faster RCNN is one of the most common object detectors for pedestrian and cyclist detection. (Z. Q. Zhao et al., 2017) used faster RCNN to detect pedestrians, (Saleh, Hossny, Hossny, & Nahavandi, 2018) applied the algorithm to detect cyclists, and (K. Wang & Zhou, 2019) used the same detection algorithm to detect both cyclists and pedestrians. Additionally, researchers often focus on the detection of one object across multiple datasets, as can be seen in Table 2.2, were (Benenson, Omran, Hosang, & Schiele, 2015) focused on the detection of pedestrians across several datasets (Caltech, INRIA, and KITTI datasets). (Brunetti, Buongiorno, Trotta, & Bevilacqua, 2018) carried a detection experiment where they detected pedestrians across INRIA and Caltech datasets.

It is to say that most research focused on training and testing on a single dataset, and few studies used more than one dataset to detect a single object. This thesis focuses on detecting cyclists, pedestrians, and other road objects across multiple

datasets using multiple object detection algorithms. In addition, detecting valuable road objects under different weather, light and driving conditions.

**Table 2.2** Summary of reviewed papers on pedestrian and cyclist detection

<i>Author</i>	<i>Detection method</i>	<i>Detection objects</i>	<i>Datasets</i>	<i>Object detection insights and techniques</i>	<i>Conclusion</i>
(Tomè et al., 2015)	CNN	Pedestrian	Caltech Pedestrian Dataset	Based on convolution neural networks, a system with a reasonable computational complexity is proposed that involves a combination of Locally De-correlated Channel Features (LDCF) as a region proposal algorithm and the fine-tuned deep convolutional neural network.	Early deep learning technique that achieved reasonable detection precision, however, requires modification for more accurate and complex detection.
(Z. Q. Zhao et al., 2017)	Fast RCNN	Pedestrian	INRIA (histograms of oriented gradients for human detection and ETH Depth and Appearance for Mobile Scene Analysis datasets)	RCNN is used to extract robust features of a pedestrian in a complicated environment. They imply the Edge-Boxes algorithm to generate effective region proposals from an image, as the quality of extracted region proposals can significantly affect the detection performance.	Fast RCNN achieve satisfactory results on the two trained datasets; nevertheless, a more complex detection algorithm has to be developed for detection of crowded and complex scenes.
(Brunetti et al., 2018)	Mixed methods	Pedestrian	INRIA and Caltech datasets	A survey that focuses on both 2D and 3D pedestrian detection highlights the different deep learning algorithms used in detecting and classifying pedestrian objects.	Modern detection approach will achieve more accurate detection and tracking results.
(Benenson et al., 2015)	Mixed methods	Pedestrian	Caltech, INRIA, and KITTI datasets	Reviewed the different algorithms used for benchmark datasets where it showed the difference in detection performance. The paper also suggested that deep networks and (boosted) decision forests achieved high-performance precision in pedestrian detection.	Detectors learned on one dataset may not necessarily transfer well to others.



(K. Wang & Zhou, 2019)	Fast RCNN	Pedestrian and Cyclist	Pedestrian and cyclist database established in Beijing's urban traffic environment	Paper provided a joint pedestrian and cyclist detection framework based on fast RCNN detection method.	Detecting small-sized objects are complex and challengeable.
(Saleh et al., 2018)	Faster RCNN	Cyclist	VGG16	A 3D LiDAR data with feed into a faster RCNN architecture for simultaneously detecting and localizing the cyclist's instance in image depth proposed.	Training and testing on the same dataset will result in high precision and accuracy.
(Bo Bo et al., 2020)	Tracking-by-detection	Road objects	Real-world traffic dataset captured by researchers	Multi-cameras used for tracking purposes, wherein data are fed into the YOLO object detector for classification and analysis. The algorithm succeeds in detecting, using trajectory in which the accuracy detection is 83% and the pedestrian accuracy is 93%.	YOLO is one of the most recent object detections able to achieve high accuracy.
(Al-Refai & Al-Refai, 2020)	Yolo and CNN	Pedestrians, Vehicles, Tracks, and Cyclists	KITTI	CNN and YOLO algorithms were adopted for the paper. A front-looking camera was used to collect data that were processed and classified using YOLO. However, the research manages to achieve the accuracy of cars; however, for small objects, the adoption produces higher false-negative results.	When mixing several detection methods it will lead to accurate detection results.
(L. Wang et al., 2020)	CNN and Feature fusion	Road object with a focus on the cyclist category	KITTI	Real-time object detection based on both CNN and feature fusion in smart cities where deep learning 3D object detection based on voxelization (the process of converting data structures that store geometric information in a continuous domain into a rasterized image), sparse convolution, and feature fusion was proposed.	Feature fusion network, better than pyramidal methods, and concatenation-based methods.

(Gaspar et al., 2020)	Pattern detection	Pedestrian	Data from motion sensors and the YOLO software (one or two cameras pointing to a crosswalk and with the standard YOLO trained dataset)	The paper aims to increase the safety of pedestrians using computer vision, in which an ITS-G5 was applied to notify drivers using a decentralized environmental notification message (DENM) to deliver.	The proposed intelligent transport system (ITS-G5) is complex; yet it can lead to increased safety of pedestrians.
(Dominguez, Sanguino, Veliz, & Gonzalez, 2020)	Mixed methods	Pedestrians, Vehicles, and Cyclists	-	Time-frequency algorithms based on Dopplers, are implied in which pre-processing is used for improving radar signal. A short-time Fourier transform is then applied to analyse and represent each signal received from the radar. The method proposed managed to detect pedestrians relying on their micro-Doppler signatures accurately. However, several back draws can detect vehicles and cyclists on the different vertical and horizontal lines.	Provides decision support system based on time-frequency pattern analysis for pedestrian recognition.

## 2.5 OBJECT DETECTION ALGORITHMS

Object detection is a key technology behind advanced driver assistance systems (ADAS) that enable cars to detect driving lanes or perform pedestrian/cyclist detection. These algorithms often identify a bounding box around the object and classify the object into one of the various classes present. Apart from localization and intra-class variations, the other challenges for improved detection include variations in viewpoint, resolution, aspect ratio, deformation, occlusion, background clutter, etc. that are present in the images.

Different deep learning detection algorithms are used to capture road information and produce an informative evaluation. For instance, Faster Region-based Convolutional Neural Network (Faster RCNN)(Girshick, 2015a) is defined as an object detection module that depends on region proposal algorithms to hypothesize object locations. Fully Convolution One-Stage object detection (FCOS)(Tian et al., 2019) is an anchor-free object detection that implies an Intersection Over Union (IOU) to detect the location and presence of objects. FCOS was used by Gao et al. (Gao et al., 2022) for vehicle detection in various remote sensing scenes, wherein they proposed a vehicle double FCOS vehicle detection model. RetinaNet (Lin, Goyal, et al., 2017) is a one-stage object detection model that has proven to work well with dense and small-scale objects. Cascade RCNN (Cai & Vasconcelos, 2017) was used for object detection that can be applied as part of Region-based Fully Convolutional Network (R-FCN) (Dai, Li, He, & Sun, 2016), thereby yielding more accurate detection results. Many other detection algorithms were proposed; they have showed improvements in detection (Carion et al., 2020; W. Liu et al., 2016; W. Liu, Liao, Ren, Hu, & Yu, 2020).

The algorithms discussed were generally tested on multiple benchmark/stock datasets. The datasets used in this thesis are listed here: COCO, KITTI, Cityscapes, and EuroCity Person. Many of the abovementioned algorithms were tested on COCO (Common Objects In Context) (Caesar et al., 2018).

Many other researchers used different datasets for training and testing purposes. However, it has not yet been investigated fully as to how a single algorithm performs on multiple datasets related to road object detection. Furthermore, the

performance of an algorithm when trained on a certain dataset and tested on another is yet to be understood. This understanding is key for real-world applications of any algorithm where the chosen algorithm cannot be trained a priori. In this study, we are comparing a set of chosen algorithms, Faster RCNN, Cascade RCNN, FCOS, and RetinaNet for their performance on three chosen datasets.

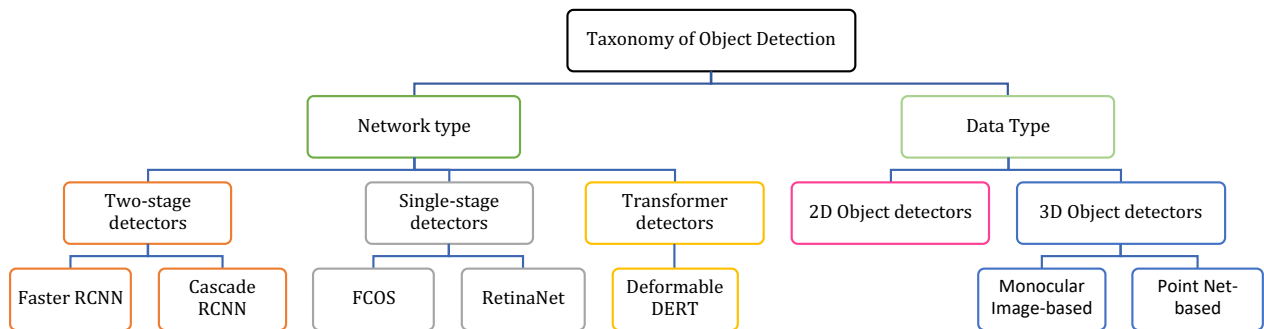
Models were trained and tested with KITTI (Geiger et al., 2013), Cityscape (Cordts et al., 2016), and EuroCity Persons (Braun, Krebs, Flohr, & Gavrila, 2019) public datasets. These public datasets carry traffic and road scenes under different conditions that are labelled for testing detection algorithms. Several different criteria and learning rate adjustment methods were evaluated to test the efficiency of the detection methodologies.

### **2.5.1 Overview of Object Detectors**

Most object detection consists of two main sub-tasks:

- Localization, which involves determining the location of an object in an image, and
- Classification, which involves assigning a class to the object (e.g., 'pedestrian', 'cyclist', 'car').

Figure 2.14 illustrates a taxonomy of the state-of-the-art deep learning-based object detectors. The taxonomy of these detectors will be discussed in the following sub-section.



**Figure 2.14** Taxonomy of object detectors

### 2.5.1.1 Network type

As indicated in Figure 2.14, there are several network types, of which two-stage and one-stage detectors are the most common types. However, transformer detectors are much recent—the first transformer-based object detector DETR (DEtection TRansformer) was developed by Facebook AI Research in 2020. DETR is a fully end-to-end object detection model that replaces the traditional two-stage object detection pipeline with a single transformer-based architecture.

Two-stage deep learning-based object detectors involve a two-stage process consisting of:

- 1) Region proposals stage, where the object detector proposes several Regions of Interest (ROIs) in an input image that have high likelihood of containing objects of interest.
- 2) Object classification stage, where the most promising ROIs are selected (with other ROIs being discarded) and objects within them are classified.

RCNN, Fast RCNN, and Faster RCNN are some examples of two-stage detectors. Single-stage object detectors use a single feed-forward neural network that creates bounding boxes and classifies objects in the same stage. These detectors have the advantage of fast detection though they are less accurate. SSD, FCOS, Retinanet, and YOLO are some of the single-stage detectors.

### **2.5.1.2 Data type**

The 2D object detector uses 2D image data for detection, and typically, images in those data are gathered from cameras. Recently, however, a sensor-fusion-based 2D object detection approach has been proposed, which combines data from camera and radar. 2D object detectors provide bounding boxes with four Degrees of Freedom (DOF). 2D object detection can only provide the position of the object on a 2D plane but does not provide information about the depth of the object. 3D object detectors use data from different sources such as camera, lidar, and radar to detect objects and generate 3D bounding boxes. These detectors provide bounding boxes and depth information.

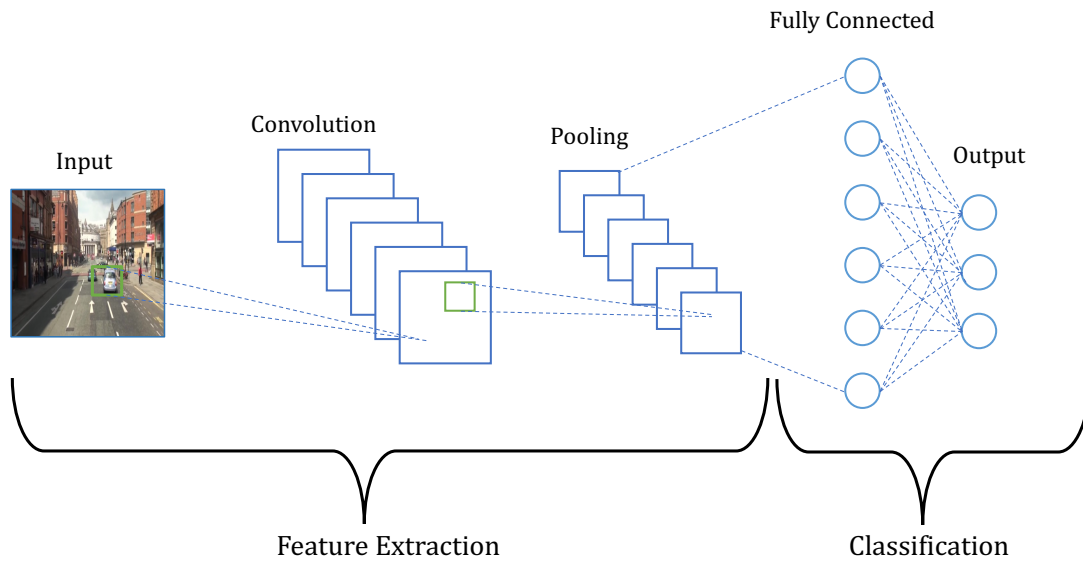
## **2.5.2 Convolutional Neural Network (CNN)**

Convolutional Neural Networks (CNN) are a deep learning algorithm that classifies an image by extracting features and image information. It can also differentiate images one from another. The feature extraction process is often seen as universal non-linear function approximators. Significantly, the pre-processing required in CNN is much less compared to other classification algorithms (Wu, 2017).

More details on the architecture of CNN will be discussed below.

### **2.5.2.1 CNN architecture**

The four main layers in CNN are convolutional layer, pooling layer, ReLU correction layer, and fully connected layer; Figure 2.15 illustrates the CNN model architecture (Purwono et al., 2022).



**Figure 2.15** CNN architecture

### **A) Convolution Layer**

The convolutional layer is the main component of the convolutional neural networks and the first layer of the network. The main purpose of this layer is the extraction of features, such as edges, colours, and corners, from the input. Additionally, as one penetrates the network, more complex features like shapes, digits, and face parts are identified by the network (Uchida, Tanaka, & Okutomi, 2018).

A convolution is a linear operation which involves a multiplication of a set of weights with the input, much like a traditional neural network. This technique was originally designed for two-dimensional inputs: the multiplication is performed between an array of input data and a two-dimensional array of weights, called a kernel or a filter.

The convolution filter is mostly smaller than the input data. A dot product multiplication is applied between a filter-sized patch of the input and the filter to filter it. The dot product multiplication produces a single value, and this operation is often called the 'scalar product' (T. Li et al., 2019).

In other words, a dot product between two matrices is one in which one matrix (kernel/filter) is the set of learnable material and the other matrix is the restricted portion of the image. For instance, for an RGB image, a filter is applied which will have smaller height and width compared to the image though it will have the same depth (height x width x 3) as the image.

Here, the convolution for a pixel in the next layer is calculated using the formula:

$$y_j = \sum w_{ij} * x + b_j$$

2.1

where  $y$  is the output,  $x$  is the input,  $w$  is the filter matrix,  $b$  is the bias, and  $*$  represents the convolution operation (Uchida et al., 2018).

Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data— left to right, top to bottom.

The systematic application of the filter across an image produces a ‘feature map’, which provides precise information about the features required to be identified in an image: the higher the value, the more the corresponding place in the image resembles the feature. Figure 2.16 illustrates the process of creating a feature map.

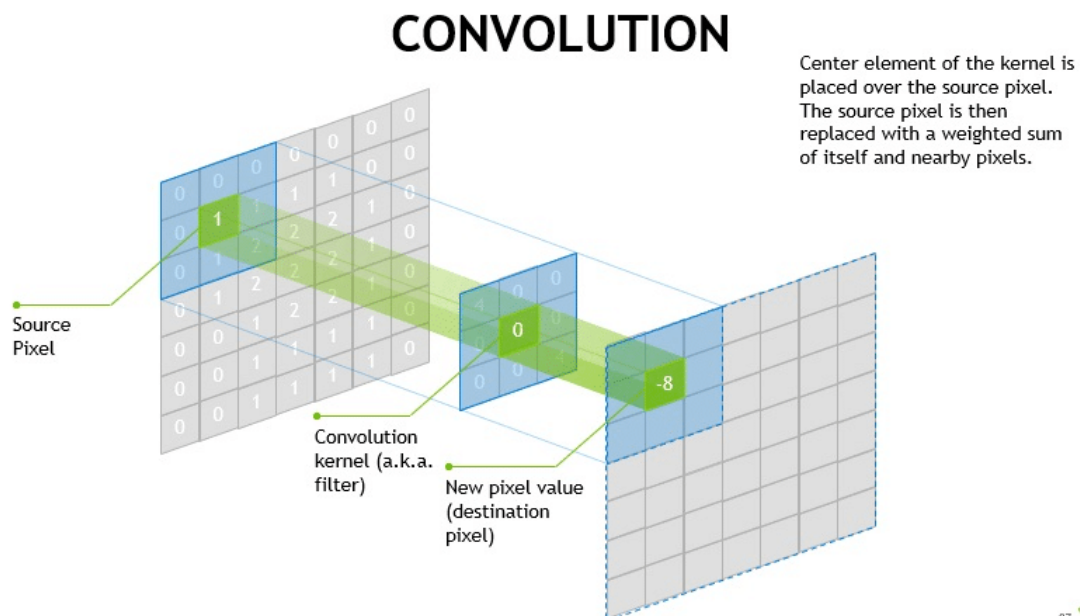


Figure 2.16 Convolution layer process(Martin, 2018)

In convolution, the features are not predefined, but learned by the network during the training. Filter kernels refer to the convolution layer weights. They are initialized and then updated by backpropagation using gradient descent.



With the feature map (feature matrix) created—which has fewer dimensions and clearer features than the actual image—the values are now passed through a nonlinearity layer for further processes, such as pooling and ReLU layers.

### B) Pooling layer

Pooling layer is a down sampling layer, wherein the main aim of pooling is to reduce the complexity for further layers. This can be achieved by reducing the size of the image while maintaining its important characteristics. Pooling is usually placed in between two layers of convolution; hence, it receives several feature maps and applies the pooling operation to each of them. One of the most common pooling methods is max-pooling. The image in max pooling is divided into squared sub-regions, and it returns the maximum values inside those sub-regions, as can be seen in Figure 2.17. The most common sizes used in max-pooling is 2x2 adjacent cells that do not overlap, or 3x3 cells, separated from each other by a step of 2 pixels in order to preclude overlapping (Jie & Wanda, 2020).

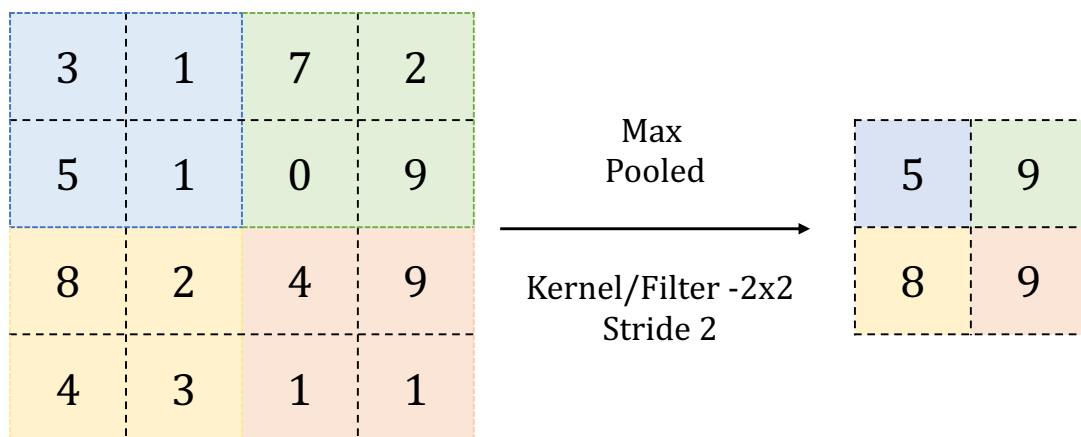


Figure 2.17 Pooling layer (2x2 max-pooling kernel)

#### i) ReLU correction layer

ReLU (Rectified Linear Units) refers to the real non-linear function defined by (Ali et al., 2020)

$$\text{ReLU}(x) = \max(0, x)$$

2.2

In other words

$$f(x) = \begin{cases} 1, & \text{if } x < 0 \\ 0, & \text{otherwise} \end{cases}$$

2.3

This layer replaces all negative values received as inputs by zeros, wherein it acts as an activation function. This can be depicted as below (Figure 2.18)



Figure 2.18 ReLU equation depiction

ii) Fully connected layer

The fully connected layer is usually the last layer of the neural network, convolution; or, in some cases, it is not included. This layer receives a vector input and outputs a new vector (K. Liu, Kang, Zhang, & Hou, 2018). A linear combination and then possibly an activation function are applied to the input values received in order for output to be produced, as can be seen in Figure 2.19.

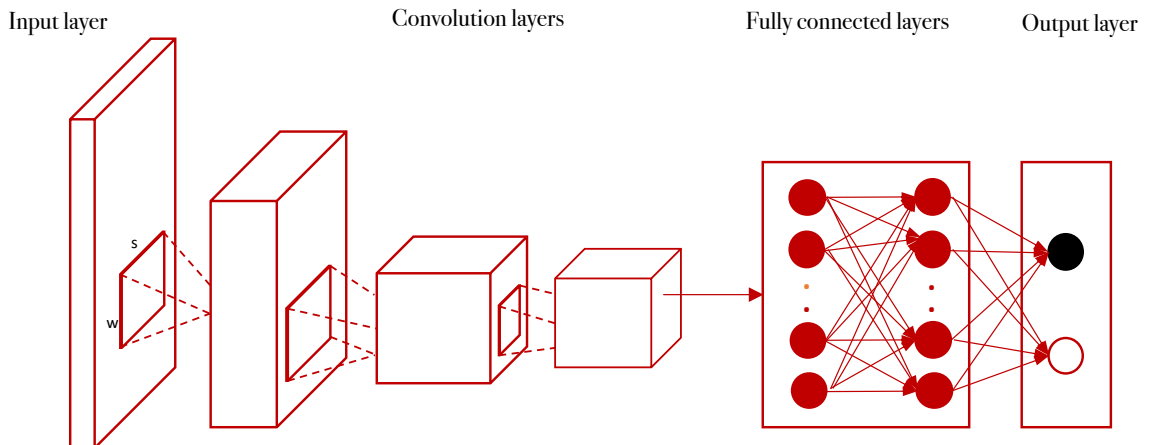


Figure 2.19 Fully connected layer

The fully connected layer returns a vector of  $N$  size, where  $N$  is the number of classes in the scope of the classification problem. Each element of the vector indicates the probability for the input image to belong to a class.

In order to evaluate the probabilities, the fully-connected layer, therefore, multiplies each input element by weight, makes the sum, and then applies an activation function (logistic if  $N=2$ , Softmax if  $N>2$ ). This is equivalent to multiplying the input vector by the matrix containing the weights. The fact that each input value is connected with all output values explains the term 'fully-connected'.

Moreover, one of the main functionalities of a fully-connected layer is to obtain the relationship between the position of features in the images and class. In this, since the input table is the result of the previous layer, it corresponds to a feature map for a given feature: **the high values indicate the location** (more or less precise, depending on the pooling) **of this feature in the image**. If the location of a feature at a certain point in the image is characteristic of a certain class, then the corresponding value in the table is given significant weight (K. Liu et al., 2018).

### iii) *Softmax*

Softmax regression layer or multi-class logistic regression is a form of logistic regression that assigns an input value into a vector of values which follows a probability distribution, the total of which sums up to 1. The output values are between the range  $[0,1]$ , which is nice, because one is able to avoid binary classification and accommodate as many classes or dimensions in our neural network model. Figure 2.20 illustrates the Softmax layer. The Softmax function is given as follows:

$$\rho(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

2.4

where  $\rho$  is the Softmax,  $\vec{z}$  is the input vector,  $e^{z_i}$  is the standard exponential function for input vector,  $K$  is the number of classes in the multi-class classifier, and  $e^{z_j}$  is the exponential function of the input vector (Dong, Zhu, & Gong, 2019).

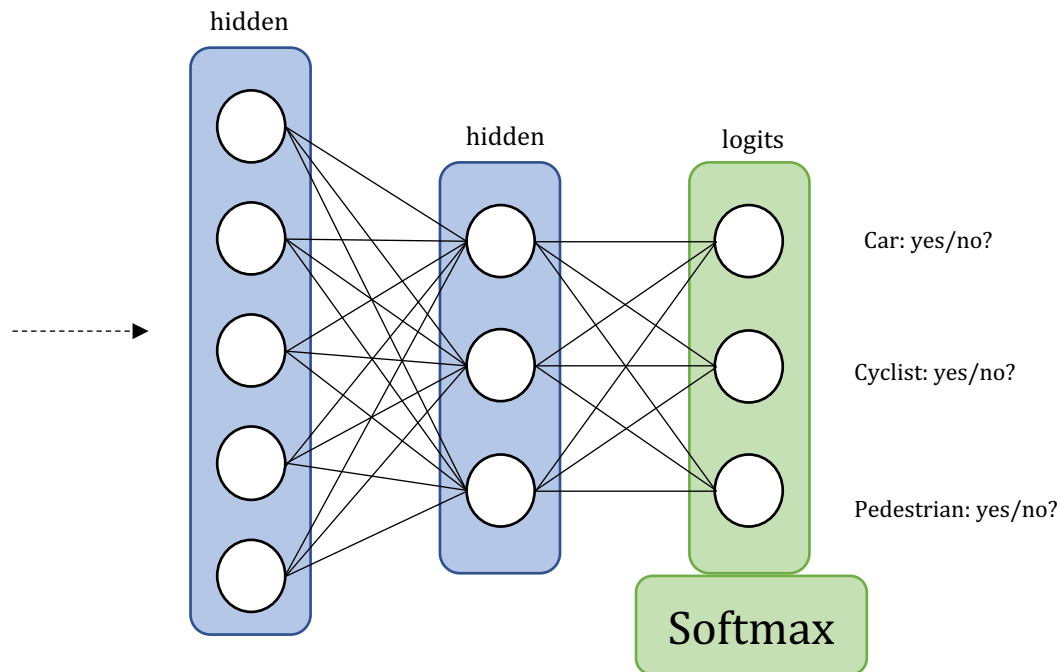


Figure 2.20 A Softmax layer within a neural network

### 2.5.2.2 CNN Parameters

For the convolution and pooling layer, there are hyperparameters that have to be predefined to obtain the size of the output feature maps. Each image (feature map) has dimensions of  $W \times H \times C$ . Here,  $W$  stands for the width in pixels,  $H$  is the height in pixels and  $C$  is the number of channels. For instance,  $C = 1$  refers to a grayscale image and  $C = 3$  refers to a coloured image (T. Zhang, Zhang, Shi, & Wei, 2019).

In a convolution layer,  $k$  is the number of filters,  $F$  represents the filter (where each filter has a dimension of  $F \times F \times C$  pixels), and  $S$  is the step of the window corresponding to the filter on image. For example, a step of  $S = 1$  means moving the window one step at a time. Finally comes  $P$ , which is the zero-padding. It is necessary to add a black contour of  $P$  pixels thickness to the input image of the layer so as to ensure that the exit dimensions are not smaller than the input.

Additionally, the more convolutional layers are stacked with  $P = 0$ , the smaller the input image of the network is; this results in the loss of a lot of information quickly, which makes the task of extracting features difficult (T. Zhang et al., 2019).

For each input image of size  $W \times H \times C$ , the convolution layer returns a matrix of dimensions  $W_c \times H_c \times C_c$  where:

$$W_c = \frac{W - F + 2P}{S} + 1 \quad 2.5$$

$$H_c = \frac{H - F + 2P}{S} + 1 \quad 2.6$$

$$D_c = K \quad 2.7$$

Having  $P = \frac{F-1}{2}$  and  $S = 1$  gives a feature map of the same width and height as those received in the input.

In a pooling layer the image is divided into square cells of size  $F \times F$  pixels, and cells are separated from each other's by  $S$  pixels. Hence, for each input image of size  $W \times H \times C$ , the convolution layer returns a matrix of dimensions  $W_p \times H_p \times C_p$  where:

$$W_p = \frac{W - F}{S} + 1 \quad 2.8$$

$$H_p = \frac{H - F}{S} + 1 \quad 2.9$$

$$D_p = D \quad 2.10$$

CNNs can capture different patterns as the depth of the network increases. For example, the layers at the beginning of the network capture the edges, while the deep layers will capture more complex features like the shape of the objects (leaves in trees, or tyres on a vehicle). On looking at the architecture of CNN, it can be seen that it is composed of two main blocks, which include four main components: convolutional layer, pooling layer, ReLU correction layer, and fully connected layer. As part of the neural network, the main functionality of the first block of CNN is to extract features; here, the first layer filters the image with several convolution

kernels and returns 'feature maps', which are then normalized and/or resized. This process can be repeated several times. The input of each filter is the result (output) of the previous features maps. On using new kernels, new features maps to normalize and resize will be obtained, and this process can be repeated. Finally, the values of the last feature maps are concatenated into a vector. This vector defines the output of the first block and the input of the second (T. Zhang et al., 2019).

The second block can be seen as the classifier block. The input of this block is the vector values obtained from the first block. The input values are transformed to return a new vector to the output. The output vector contains as many elements as there are classes: element  $i$  represents the probability that the image belongs to class  $i$ . Each element, therefore, lies between 0 and 1, and the sum of all is worth 1. These probabilities are calculated by the last layer of this block (network), which uses a logistic function (binary classification) or a Softmax function (multi-class classification) as an activation function.

Even though the state-of-art of CNN detection has achieved high results, several object detectors have been customized for different tasks, such as road object detection, cars detection, and detection of pedestrians, achieving impressive results (for more details on CNN, refer to (Gad & John, 2018; Z. Li, Liu, Yang, Peng, & Zhou, 2022)). Faster R-CNN (Girshick, 2015a), Cascade R-CNN (Cai & Vasconcelos, 2017), End-to-End Object Detection with Transformers (DETR) (Carion et al., 2020), Fully Convolutional One-Stage Object (FCOS) (Tian et al., 2019), You Only Look Once (YOLO) (Redmon et al., 2016), and Focal Loss for Dense Object Detection (RetinaNet) (Lin, Goyal, et al., 2017) are some examples of object detectors. The following section will highlight the four different detection algorithms used in this thesis, describing their performance, architecture, and the different building blocks.

### **2.5.3 DESCRIPTION OF ALGORITHMS**

Five different object detection algorithms were chosen and compared in this thesis, namely: Faster Region-based Convolutional Neural Networks (Faster RCNN)(Girshick, 2015a), Cascade Region-based Convolutional Neural Networks

(Cascade RCNN)(Cai & Vasconcelos, 2017), Focal Loss for Dense Object Detection (RetinaNet)(Lin, Goyal, et al., 2017), FCOS (Fully Convolutional One-Stage Object Detection)(Tian et al., 2019), and Deformed Transformers for End-to-End Object Detection (Deformed DETR)(Zhu et al., 2020). Following is a Table 2.3 that present the unique aspect of each algorithm and the reason it was chosen for evaluation throw out the thesis.

**Table 2.3** Chosen algorithms unique aspects and reason to be used for evaluation.

Algo	Unique aspect	Why
<b>Faster RCNN</b>	Introduce the Region Proposal Network (RPN)	It uses a region proposal network to generate potential object proposals, which are then classified and refined. This approach allows for more precise localization and better handling of overlapping objects.
<b>RetinaNet</b>	Introduces Feature Pyramid Network (FPN) and focal loss	Addresses the class imbalance in object detection by using feature pyramid network (FPN) to capture objects at different scales and a focal loss to give more weight to challenging examples.
<b>Cascade RCNN</b>	Implements a cascade structure, where multiple stages of detectors are used to progressively refine the object detection results.	Each stage focuses on refining the detection results from the previous stage, leading to improved precision and recall. This multi-stage architecture allows for better handling of challenging scenarios and improves the overall performance of the detector.
<b>FCOS</b>	Adopts a fully convolutional architecture for object detection. It eliminates the need for anchor	Adopts a fully convolutional architecture, which allows for efficient and parallel computation. It eliminates the need for anchor boxes, reducing complexity and improving speed.
<b>Deformable DETR</b>	Combines the power of transformer networks with object detection.	It incorporates deformable convolutional layers, which allow the model to adapt to object deformations and better capture spatial relationships. This helps improve the accuracy of object detection.

### ***2.5.3.1 Faster Region-based Convolutional Neural Networks (Faster RCNN)***

Regions with convolutional neural network (R-CNN) (Girshick et al., 2014) is a two-stage object detection algorithm. This network was proposed to solve the issue of locating an object in an image. The solution here was to slide a window over the whole image with rectangles of different sizes and search for the required objects. However, this creates an issue: there is a huge number of smaller images that have to be analysed. Therefore, regions with convolution neural network were developed. This combines rectangular region proposals with convolutional neural network features. This network is used for different applications like:

- Autonomous driving (Agnihotri, Saraf, & Rajesh Bapnad, 2019)
- Facial recognition (C. Zhang, Xu, & Tu, 2018)

- Smart surveillance systems (Babanne, Mahajan, Sharma, & Gargate, 2019)

The network is made of three modules. The first module generates category-independent region proposals. Where the proposals are present, the set of candidate detections available to the detector are defined. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear support vector machines (SVM)

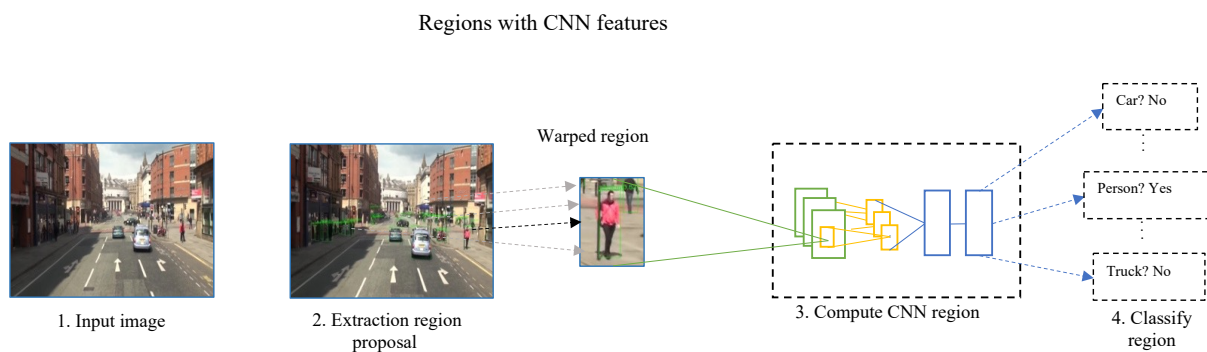
**Region proposal.** This method is implied for generating category-independent regions proposed for different detection purposes. For instance, objectness (Alexe, Deselaers, & Ferrari, 2012), which is a measurement that quantifies how likely it is for an image window to contain an object of any class, selective search (Uijlings, Van De Sande, Gevers, & Smeulders, 2013), and multi-scale combinatorial grouping (Arbeláez, Pont-Tuset, Barron, Marques, & Malik, 2014). R-CNN is agnostic to the particular region proposal method, where the selective search is used to enable a controlled comparison with earlier detection proposed methods.

**Feature extraction.** It is a dimensionality reduction process in which the initial set of raw data is reduced and combined into more manageable groups for processing purposes (Jia et al., 2014; Krizhevsky, Sutskever, & Hinton, 2012). The main characteristic of datasets that favours this process is a large number of variables that require a lot of computing resources. The extraction process selects and/or combines variables into features, effectively reducing the amount of data that must be processed while still wholly and accurately describing the original dataset. Feature extraction can reduce the redundant data and speed up the learning and generalization in the machine learning process.

**Support vector machine.** It is a supervised learning algorithm used for classification and regression problems. The algorithm creates the best line or decision boundary that can segregate n-dimensional spaces into classes so that the new data point can be placed in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help create the hyperplane. These extreme cases are called support vectors.



Figure 2.21 illustrates the RCNN network. In RCNN, the input image enters an extraction feature process wherein the selective search extracts around 2,000 region proposals. Each proposal is then wrapped and forward propagated through the CNN to compute features. Once there, it produces a 4,096-dimensional feature vector as output; these features are then fed into an SVM to classify the presence of the object within that candidate region proposal. An SVM is created and used for detection purposes for each object class. This means that for each feature vector,  $n$  outputs are created. Here,  $n$  is the number of the different objects that need to be detected. SVM outputs a confidence score. The confidence score is the score that shows how confidently this feature vector represents this particular class.



**Figure 2.21** Regions with convolutional neural network architecture

However, RCNN suffers from several drawbacks. These include the region proposal extraction, which requires the proposal of about 2,000 regions per image; this consumes a huge amount of time for training. In addition, the network fails to be implemented for real-time applications as it requires 47 seconds for each test image. Finally, the fixed selective search algorithm, wherein no learning takes place at this stage, can generate bad candidate region proposals. Therefore, Fast RCNN was developed to overcome some of the disadvantages of RCNN.

### **A) Fast RCNN**

Fast RCNN was developed by (Girshick, 2015a) in 2015, wherein he combined the different stages in RCNN into one so as to create a faster object detection algorithm compared to RCNN. Figure 2.22 illustrates the architecture of Fast RCNN. In fact,

RCNN and CNN process the whole image and produce a convolution feature map. From the convolution feature map, the region proposals are identified and warped into squares. Using the Region of Interest (RoI) pooling layer the region-proposals are reshaped to a fixed size. This will then be fed to a fully connected layer. Where an RoI pooling layer is used for utilizing a single feature map for all the proposals created by Region Proposal Network (RPN) in a single pass, this layer solves the issue of fixed image size requirement for object detection network. After this stage, a feature vector with the same size is outputted and used as the input of a Softmax layer. The Softmax layer is used to predict the class of object found from the RoI feature vector and the bounding box regressor, where the bounding box coordinates for each object are given.

Fast RCNN is faster than the original RCNN; however, the network remains slow and time-consuming, especially for real-time applications, as it uses selective search. Hence, it was necessary to develop this algorithm. A faster RCNN network was developed to overcome those issues (Ren, He, Girshick, & Sun, 2017).

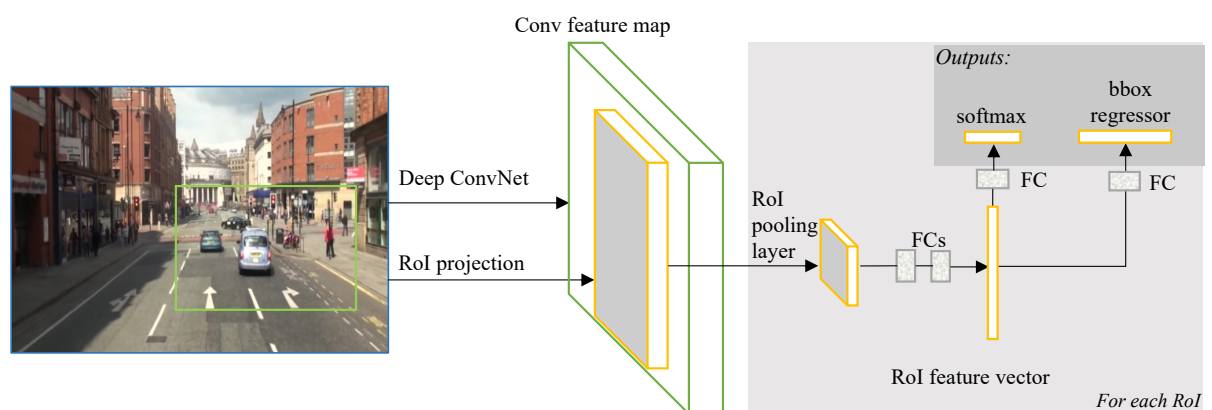


Figure 2.22 Fast RCNN architecture

### ***B) Fastest RCNN***

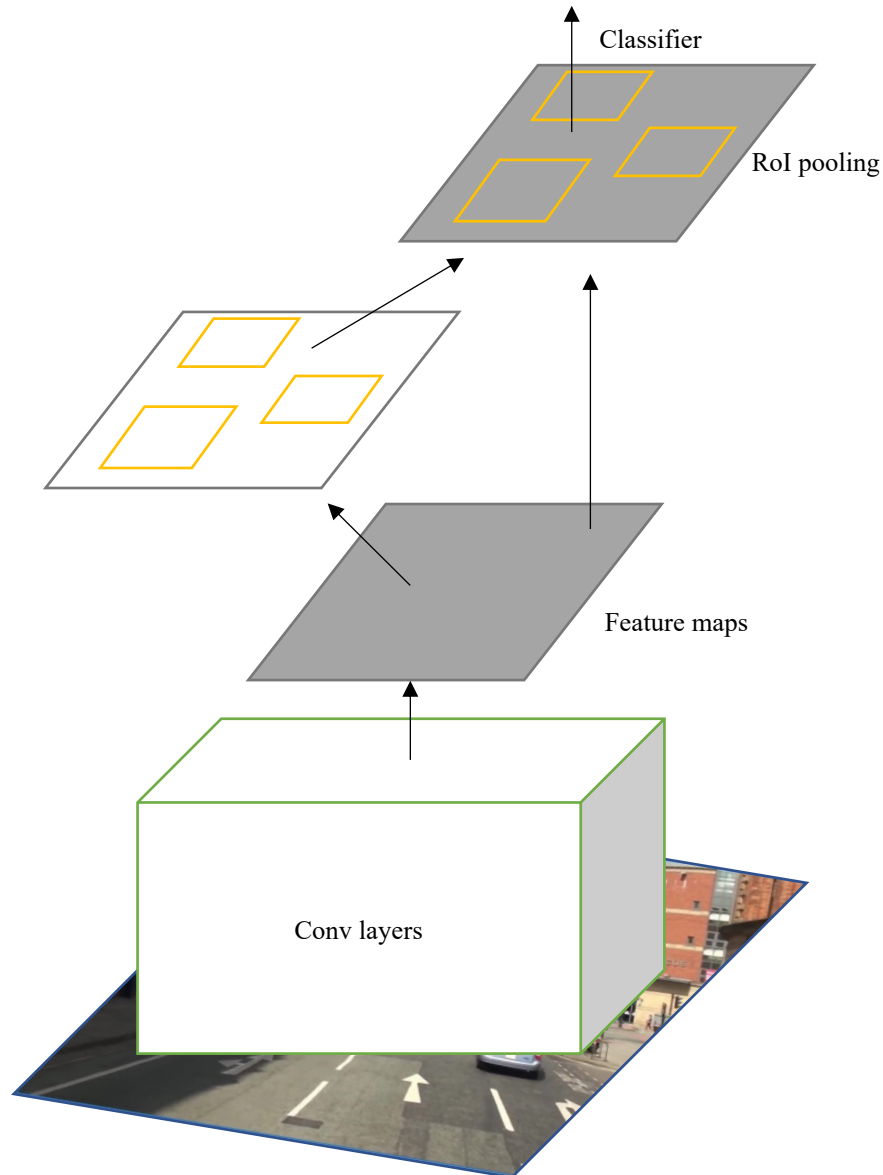
In 2016, Shaoqing Ren developed a Faster RCNN network, with Kaiming He, Ross Girshick, and Jian Sun (Ren, He, Girshick, & Sun, 2015a), to overcome the main drawback that affects the performance of the network, which is the selective search.

The detector is a modified version of Fast RCNN, in which it comes over the selective search. However, this proposed algorithm is made by a complex architect. It, therefore, comprises three parts, where an image undergoes through:

- i. Convolution layer: This layer extracts the appropriate features of the image.
- ii. Region Proposal Network (RPN): A small neural network sliding on the last feature map of the convolution layers, predicting whether there is an object or not; it also predicts the bounding box of those objects.
- iii. Classes and Bounding Boxes prediction: Fully connected neural networks that take as an input the regions proposed by the RPN and predict object class (classification) and bounding boxes (regression).

Figure 2.23 shows the architecture of faster RCNN where the image is represented as Height\*Width\*Depth. Then features are extracted using multidimensional arrays, passing through a pre-trained Convolutional Neural Network (CNN). This technique is used to transfer learning, which is used for training a classifier on a small dataset using the weights of a network trained on a bigger dataset.

Transfer learning is the use of previously acquired knowledge and skills in new learning or problem-solving situations (Steiner, 2001)



**Figure 2.23** Faster RCNN architecture

The main—and the hardest issue—with using Deep Learning (DL) for object detection of faces is the generation of a variable-length list of bounding boxes. To overcome this issue, anchors are introduced in the Region Proposal Network (RPN).

*i) Region Proposal Network (RPN)*

RPN takes an input image and outputs a set of rectangular object proposals, each with an objectness score. A fully convolutional network models this process. More specifically, a small network slides over the convolutional feature map outputted by the last shared convolutional layer. The small network takes an  $n \times n$  window as an

input of the convolutional feature map. Each sliding window is mapped to a lower-dimensional feature. The feature is fed into two sibling fully-connected layers (box-regression layer (*reg*) and a box-classification layer (*cls*)). Figure 2.24 illustrates the Region Proposal Network.

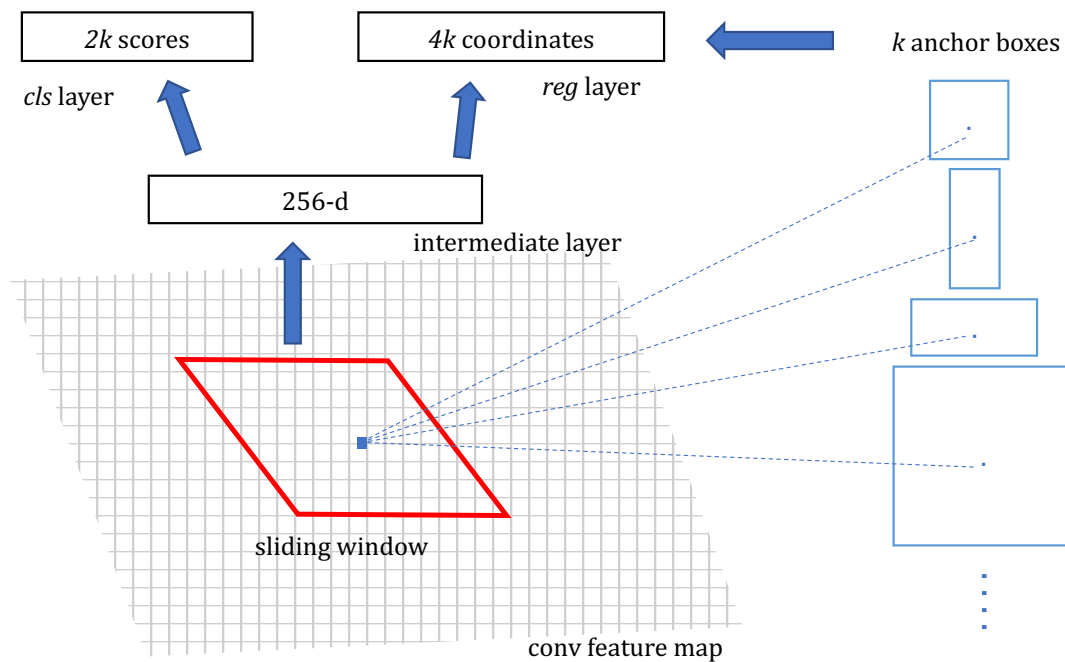


Figure 2.24 Region Proposal Network

ii) Anchors

Anchors are fixed reference bounding boxes placed uniformly throughout the original image. Instead of having to detect where objects are,  $k$  denotes the maximum possible proposals for each location. By default,  $k = 9$ , having three scales of  $(128 * 128, 256 * 256 \text{ and } 512 * 512)$  and three aspect ratios of  $(1:1, 1:2 \text{ and } 2:1)$  for each of the different sliding positions in the image. Hence, for a convolution feature map of  $W \times H$ , the total number of anchor boxes are  $WHk$ . This leads to the creation of a list of relevant objects and their location in the original images, and then detection becomes much more straightforward to solve.

### iii) Region of Interest pooling

The RoI pooling layer (Girshick, 2015a) is the layer deployed to utilize a single feature map for all the proposals generated by RPN in a single pass match. It is applied to match the detected features with the features extracted by CNN and the bounding boxes, creating a new tensor with relevant objects. Following is a forward pass diagram of RoI pooling layer (Figure 2.25).

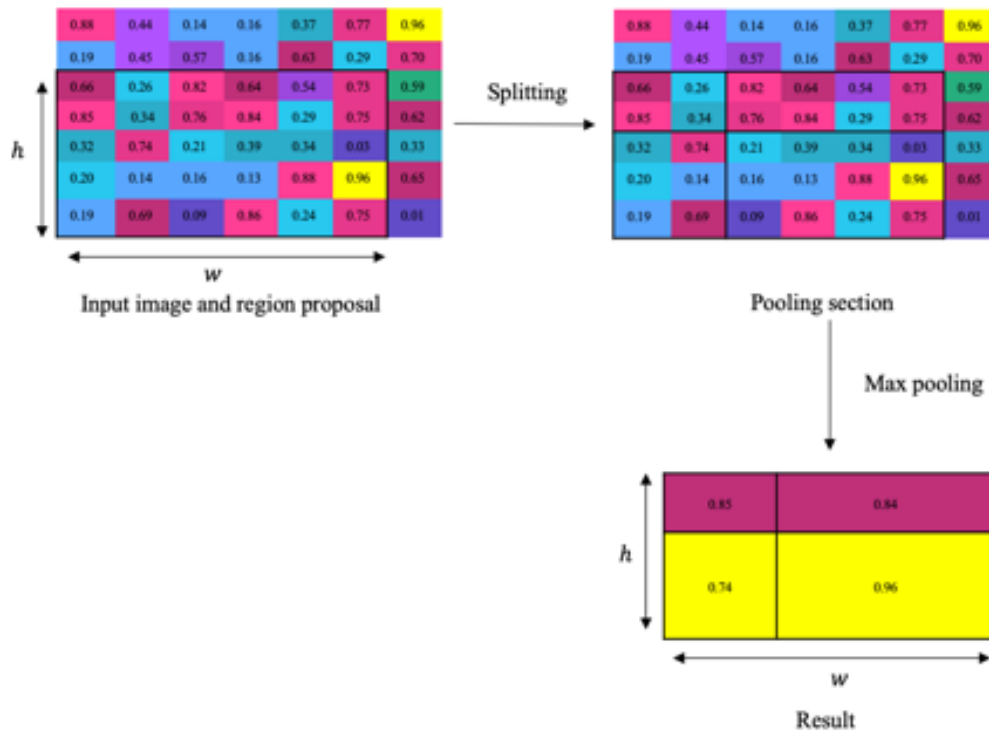


Figure 2.25 RoI pooling layer

RoI pooling produces fixed-size feature maps for non-uniform inputs applying max-pooling on the inputs. The number of output channels is equal to the number of input channels for this layer. This layer takes two inputs:

- Feature map obtained from the CNN after multiple convolutions and pooling layers.
- 'N' RoI or proposals from RPN. Each proposal has five values, the first value indicates the index, and the remaining four values are the proposal coordinates (the top-left and the bottom-right corner).

The output fixed dimension of RoI pooling for every RoI neither depends on the input feature map nor the proposal sizes; it depends solely on the layer parameters.

Finally, after extracting the relevant information, the R-CNN module comes into play, and it uses the information to:

- Classify the content in the bounding box, and
- Adjust the bounding box coordinates, so as, to better fit the objects.

For more details on Faster RCNN please refer to (Ren, He, Girshick, & Sun, 2015b).

### ***2.5.3.2 Cascade Region-based Convolutional Neural Networks (Cascade RCNN)***

#### ***A) Cascade RCNN***

Cascade R-CNN was developed in order to overcome some Faster RCNN issues. Cascade RCNN is an object detection architecture that addresses degrading performance problems with increased Intersection Over Union (IoU) thresholds. IoU threshold is used to define whether objects are positively or negatively detected.

To put it simply, the aim of developing Cascade RCNN is to overcome two main issues:

1. Overfitting during training caused by exponentially vanishing positive samples, i.e., many positive samples are gone when the IoU threshold increases.
2. The inference-time mismatch between the IoUs for which the detector is optimal and those of the input hypotheses. For instance, training at a higher (lower) IoU threshold but testing at a lower (higher) IoU threshold.

Cascade RCNN is a multi-stage extension of R-CNN, in which the detector stages deeper into the cascade are sequentially more selective against close false positives. The network was developed by Cai et al. (Cai & Vasconcelos, 2017) based on RCNN, in which they extended the two-stage faster CNN architecture by introducing object proposals for all images that enter to produce preliminary detection hypotheses. The network relies on a cascade of specialized regressors , defined as follows:

$$f(x, b) = f_T \circ f_{T-1} \circ \dots \circ f_1(x, b)$$

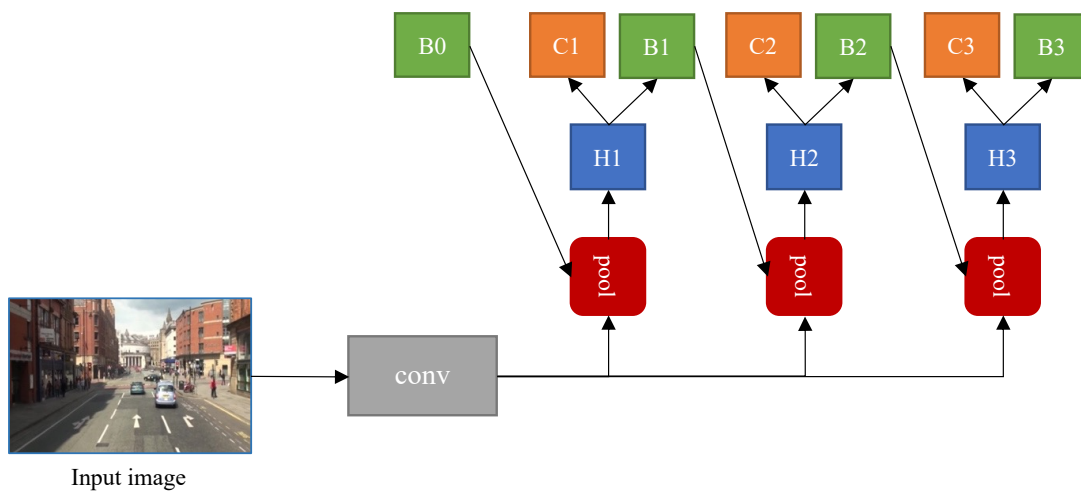
2.11

where T is the total of cascade stages—this cascade improves the hypothesis progressively as each regressor  $f_t$  in the cascade is optimized with respect to the sample distribution  $\{b_0\}$  arriving at the corresponding stage.

Cascade involve precise localization in which it considers iterative bounding boxes (bbox) architecture, but the main differences are:

- Different heads for each stage, i.e. H1, H2, H3, are used as shown in Figure 3.14, or  $f_T, f_{T-1}, \dots, f_1$  corresponding to the equation above.
- Each head is designed for one specific IoU threshold, from small to large.
- Cascaded regression is a resampling procedure, where iterative bbox uses a post-processing procedure to improve bounding boxes. This allows the cascade to pass good positive samples to the next stage.
- There is no discrepancy between training and inference since the architecture and IoU thresholds are the same during training and inference.

Figure 2.27 illustrates cascade architecture.



**Figure 2.26** Cascade RCNN architecture

*i) State-of-the-art network architecture*

Cascade RCNN is a regression problem involving combinations of different regression and face alignment tasks; Cascade RCNN was developed on the basis of the combination of several architectures. In this section, the architecture of each task will be described to attain the cascade architecture.

The development started from a Faster RCNN (Lin, Dollár, et al., 2017; X. Zhao et al., 2016) architecture, as shown in Figure 2.28; in the first stage, an input image goes through several convolutions, and the output of the convolutional features is then passed to RPN (Ren et al., 2015a). The proposal sub-network (H0) creates a preliminary detection hypothesis known as object proposals



(B0). Moving to the second stage, the hypotheses are then processed by a region of interest detection sub-network (H1), denoted as the detection head. Finally, each hypothesis gets a classification score (C1) and a bounding box (B1). However, B1 is inaccurate because an iterative bounding box at inference is applied.

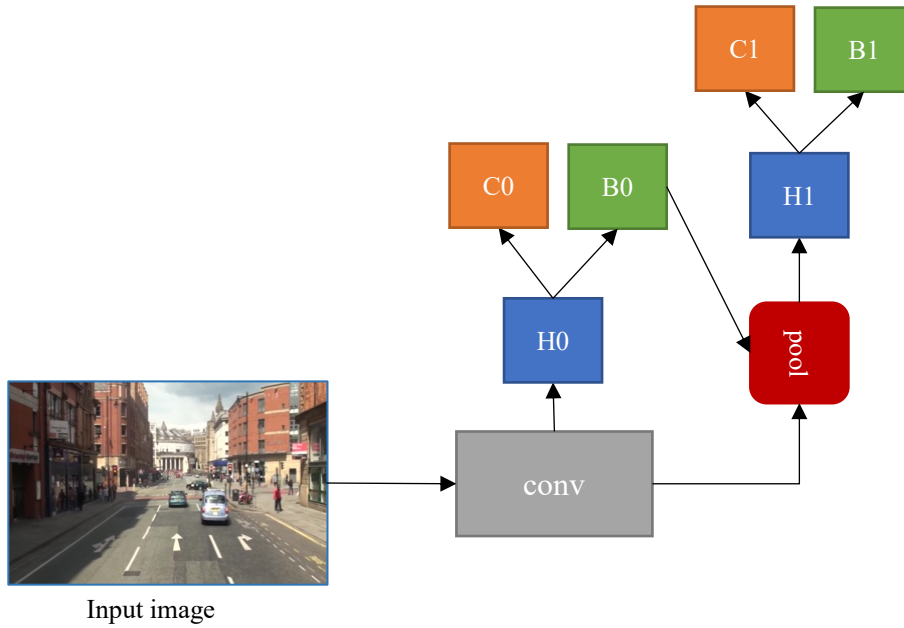


Figure 2.27 Faster RCNN architecture

ii) *Bounding box regression*

A bounding box regression is a task of regressing a candidate bounding box  $b$  into a target bounding box  $g$ , using a regressor  $f(x, b)$ . Here,  $b = (b_x, b_y, b_w, b_h)$  is the bounding box coordinates of an image patch  $x$ . Regression is learned from training sample  $\{g_i, b_i\}$  as below, to minimize the bounding box risk.

$$\mathcal{R}_{loc}|f| = \sum_{i=1}^N L_{loc}(f(x_i, b_i), g_i)$$

2.12

where  $L_{loc}$  is the loss function, and in order to encourage a regression invariant to scale and location, it operates on distance vector  $\Delta = (\delta_x, \delta_y, \delta_w, \delta_h)$  defined as:

$$\delta_x = \frac{(g_x - b_x)}{b_w}, \delta_y = \frac{(g_y - b_y)}{b_h}$$

$$\delta_w = \log\left(\frac{g_w}{b_w}\right), \delta_h = \log\left(\frac{g_h}{b_h}\right)$$

2.13

In order to improve the effectiveness of multi-task learning,  $\Delta$  is usually normalized by its mean and variance. Hence,  $\delta_x$  is replaced by  $\delta'_x = \frac{(\delta_x - \mu_x)}{\sigma_x}$ .

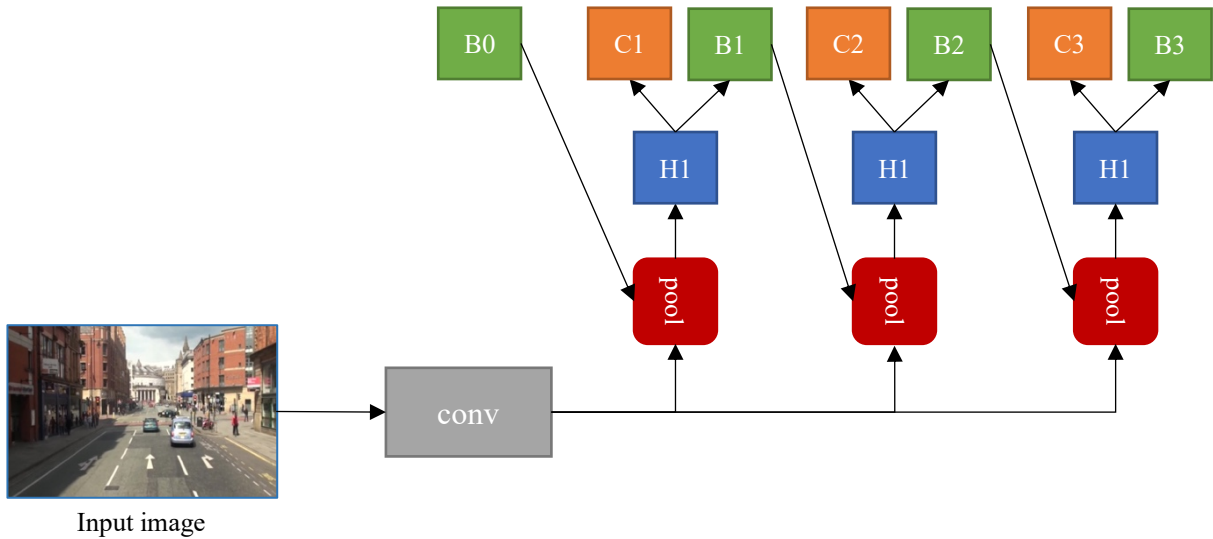
Figure 2.28 illustrates the iterative bounding box regression added on faster R-CNN, thereby adjusting the performance. Researchers such as (Gidaris & Komodakis, 2016) have argued that a single regression step for  $f$  is inadequate for accurate localization, hence,  $f$  is applied iteratively as a post-processing step to refine a bounding box  $b$  as follows:

$$f'(x, b) = f \circ f \circ \dots \circ f(x, b)$$

2.14

This is implied, as shown in Figure 2.28. B1 is inputted into the same H1 again to regress the bounding box to obtain B2, and so on. This iterative approach attempts to gradually fine-tune the bounding box to obtain a more accurate bounding box value. Not to mention that all heads (H1 in the figure and  $f$  in the equation) are the same. However, the performance improvement obtained from doing this is limited.

Again, while training (H1) is the only head available, however, during inference, multiple heads (H1) are present, which causes a mismatch between training and testing. Therefore, a loss classifier is presented. Integral loss is introduced in the next heading.



**Figure 2.28** Iterative BBox at Inference

iii) *Integral loss*

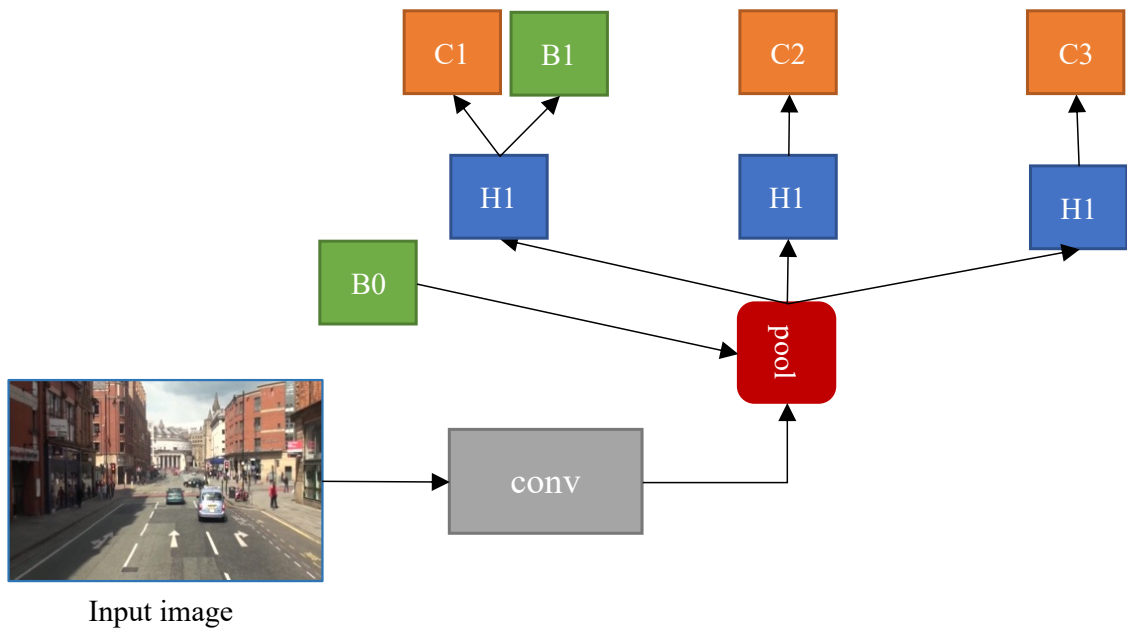
As mentioned, a mismatch occurs between training and testing on account of the presence of different heads (H); for this, an ensemble of classifiers is developed with the architecture presented in Figure 2.29, optimized with a loss that targets various quality levels.

$$L_{cls}(h(x), y) = \sum_{u \in U} L_{cls}(h_u(x), y_u)$$

2.15

Here  $L_{cls}$  is the classic cross-entropy loss,  $h(x)$  denotes the classifier function,  $U$  denotes the set of IoU thresholds, and  $u$  denotes the detector quality.

However, high-quality classifiers are prone to overfitting. The classifiers are also required to process proposals of overwhelming low quality at inference, for which they are not optimized (Cai & Vasconcelos, 2017).



**Figure 2.29** Integral Loss

Having explained these building blocks, Cascade RCNN is then formed, as shown in Figure 2.29. Cascade RCNN outperformed many object detection algorithms like Faster RCNN (Ren et al., 2015a), YOLO (Redmon & Farhadi, 2018), SSD (W. Liu et al., 2016), and R-FCN (Dai et al., 2016), in which it achieved a higher mean average performance. However, this method is slow, and incurs high computational cost. For more details on cascade RCNN, refer to (Cai & Vasconcelos, 2017).

### 2.5.3.3 Focal Loss for Dense Object Detection (RetinaNet)

RetinaNet is a one-stage object detection model that works well with dense and small-scale objects. The model is formed by improving two existing single-stage object detection:

- Feature Pyramid Network (FPN)(Lin et al., 2016)
- Focal Loss (Lin, Goyal, et al., 2017)

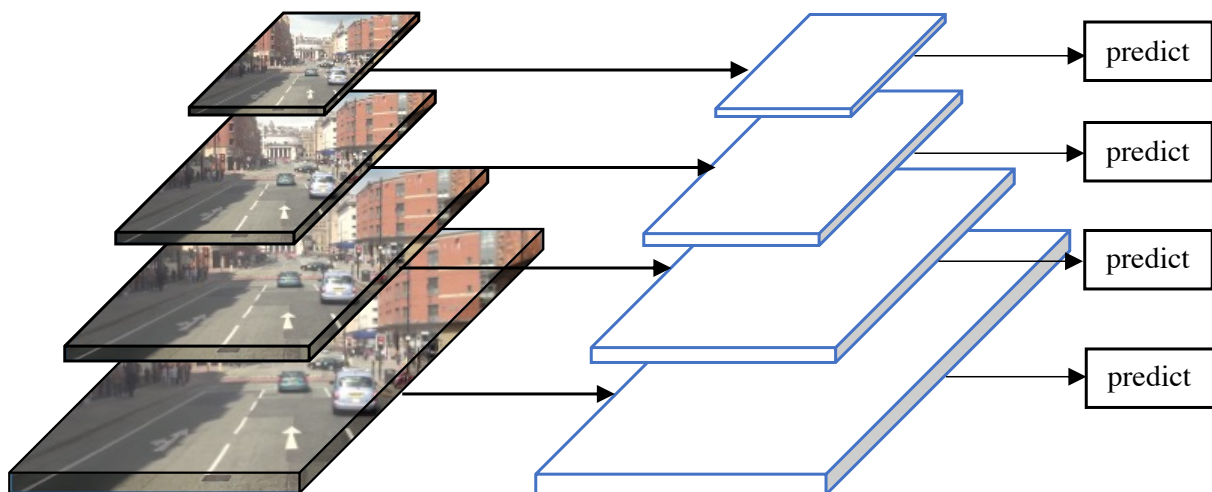
To define the network architecture, it is important to have a brief understanding of FPN.

### A) Feature Pyramid Network (FPN)

Traditionally, in computer vision, in order to detect objects, featurized image pyramids are used to detect the different scale objects in an image. Featurized image pyramids are feature pyramids built upon image pyramids. This means one would take an image and subsample it into lower resolution and smaller images, forming a pyramid. Hand-engineered features are extracted from each layer in the pyramid to detect the objects. This makes the pyramid scale invariant. However, this process is computational and memory-intensive.

With the development of deep learning, the hand-engineered features were replaced by CNN. The pyramid was formed by the inherent pyramidal hierarchical structure of CNN. The output size of feature maps in CNN decreases after each successive block of convolutional operations, and a pyramidal structure is formed.

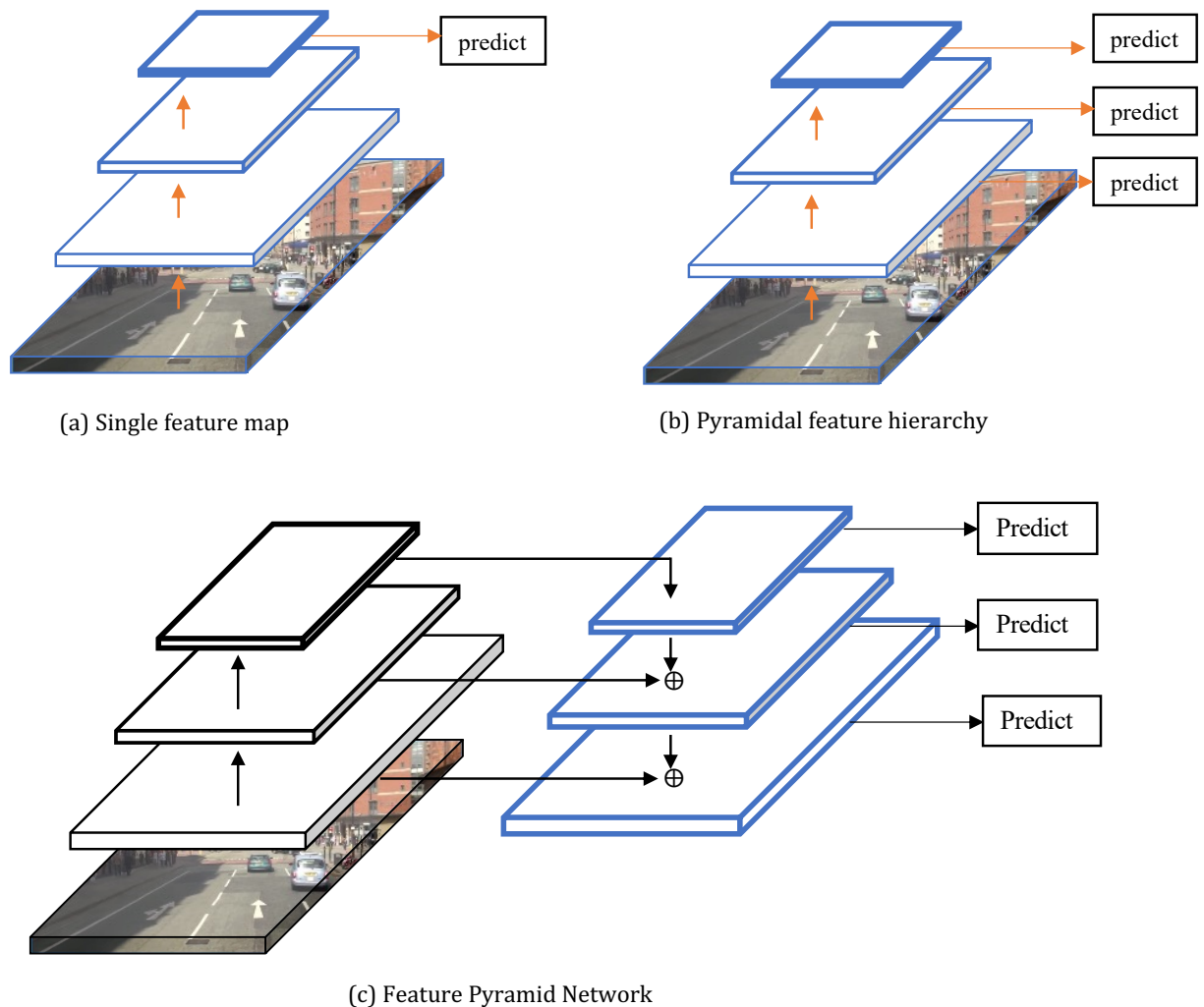
Figure 2.31 illustrates the various developments of the feature pyramid. Figure 2.30 illustrates the featurized pyramid in a shortened form; here it stands for feature pyramids built upon image pyramids. Pyramids are scale-invariant where, in this extraction technique, the scale of the objects is offset by shifting its level in the pyramid, i.e., as we move up the pyramid, the object detection scale changes; it, therefore, enables a model to detect objects across an extensive range of scales by scanning the model over both positions and pyramid levels.



**Figure 2.30** Featurized image pyramid

Next is a single feature map (Figure 2.31(a)). These maps were proposed for faster and more accurate detection. Research by (Lim, 2018) used an extended single

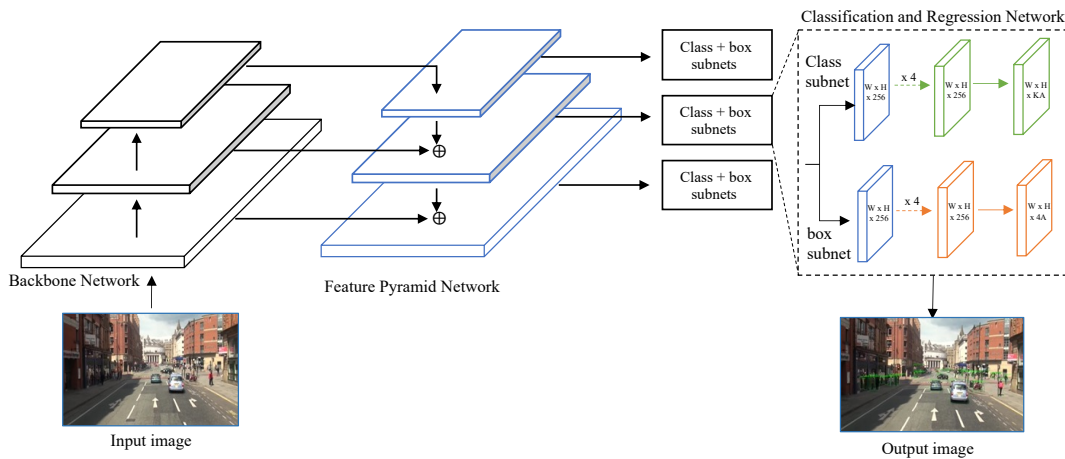
feature map based on object detection. The proposed method reduces the detection network, thus reducing memory and computational efficiency.



**Figure 2.31** Different type of pyramids architecture

Figure 2.31(b) is a pyramidal feature hierarchy utilized by several models, such as the Single Shot Detector (SSD) (W. Liu et al., 2016), but it does not reuse the multi-scale feature maps from different layers.

Figure 2.31(c) is an illustration of the Feature Pyramid Network (FPN). FPN is a feature extractor network, which takes a single-scale image of arbitrary size as input and outputs proportionally sized feature maps at multiple levels in a fully convolutional fashion (Lin et al., 2016). This is achieved by creating a top-down pathway with lateral connections to bottom-up convolutional layers. FPN uses a top-down pathway, bottom-up pathway, and lateral connections, which will be explained in detail in the following sub-section.



**Figure 2.32** RetinaNet network architecture

### ***B) Network architecture***

A RetinaNet network architecture contains four components (Figure 2.32):

- Bottom-up Pathway —The backbone network (e.g., ResNet) calculates the feature maps at different scales, irrespective of the input image size or the backbone.
- Top-down Pathway and Lateral Connections—The top-down pathway up-samples the spatially coarser feature maps from higher pyramid levels. The lateral connections merge the top-down and the bottom-up layers with the same spatial size.
- The classification subnetwork predicts the probability of an object being present at each spatial location for each anchor box and object class.
- The regression subnetwork regresses the offset for the bounding boxes from the anchor boxes for each ground-truth object.

### ***C) Focal Loss***

Focal Loss (FL) is a function designed to address the extreme imbalance between foreground and background classes during training for one-stage object detection. FL is an enhancement over Cross-Entropy Loss (CE) for binary classification. The imbalance problem of the single-stage detector is caused by the dense sampling of anchor boxes. In RetinaNet, there can be thousands of anchor boxes in each pyramid layer. However, only a few anchors will be assigned to a ground-truth object, while

most will be background class. To define focal loss, we have to start from the cross entropy (CE) loss for binary classification:

$$CE(p, y) = \begin{cases} -\log(p), & \text{if } y = 1 \\ -\log(1 - p), & \text{otherwise.} \end{cases} \quad 2.16$$

where  $y \in \{\pm 1\}$  specifies the ground-truth class and  $p \in [0, 1]$  is the estimated probability for the class with label  $y = 1$ .

For notation,  $p_t$  is defined as:

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise.} \end{cases} \quad 2.17$$

The cross entropy is rewritten as:

$$CE(p, y) = CE(p_t) = -\log(p_t). \quad 2.18$$

In order to address the class imbalance, a weighting factor  $\mu \in [0, 1]$  for Class 1 and  $1 - \mu$  for Class -1 is introduced. For notational convenience,  $\mu_t$  is defined analogously to the definition of  $p_t$  is defined. Introducing  $\mu$  - balance to CE:

$$CE(p_t) = -\mu \log(p_t) \quad 2.19$$

It is yet to acknowledge that CE is the baseline for focal loss.

For FL, a modulating factor  $(1 - p_t)^\gamma$  is added to the cross-entropy loss, with tuneable focusing parameter  $\gamma \geq 0$ . Therefore, FL is defined as:

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t). \quad 2.20$$

Focal Loss reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples. Further details on RetinaNet can be found at (Lin, Goyal, et al., 2017).

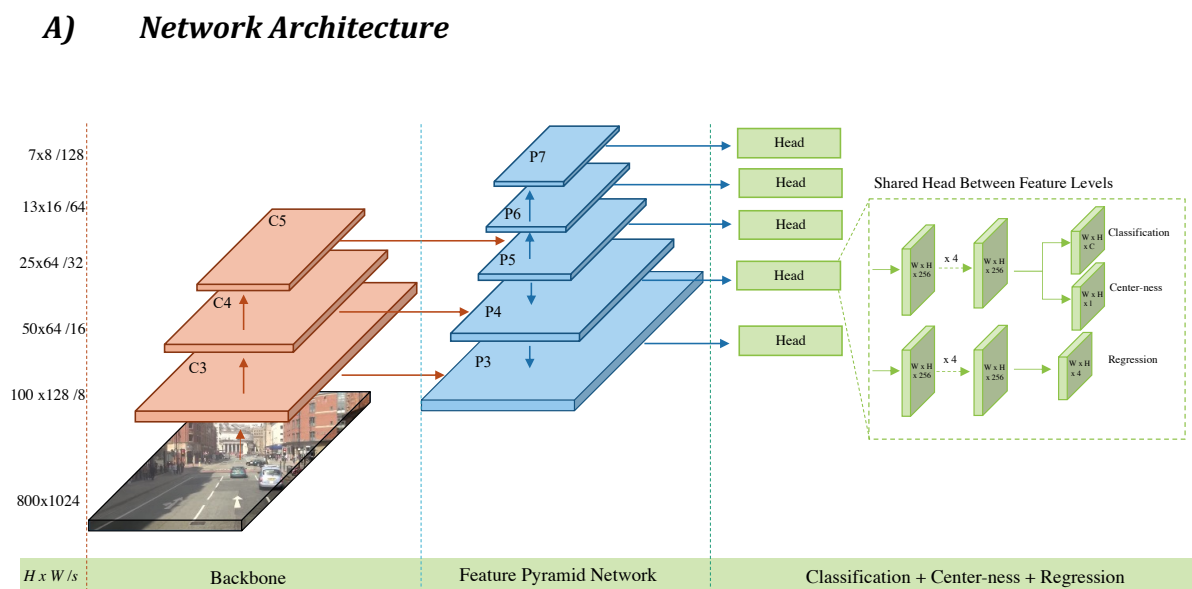
#### **2.5.3.4 Fully Convolutional One-Stage Object Detection (FCOS)**

A Fully Convolution One-Stage object detection—as the name indicates—is a one-stage object detection developed by the University of Adelaide (Tian et al., 2019). FCOS is anchor box-free, and, so, avoids all related hyperparameters and the



complicated computations related to anchor boxes, such as computing overlapping during training. In addition, it is a proposal-free detector. FOCUS uses the Intersection Over Union (IoU) between anchor boxes and ground-truth boxes to determine the label of an anchor box. If the value of IoU lies between  $[0.5,1]$ , it is a positive anchor.

FCOS redefines object detection in a per-pixel prediction fashion. It uses a multi-level prediction to improve the recall and resolve the ambiguity resulting from overlapped bounding boxes. Finally, a 'centre-ness' branch helps one overcome the low quality of the detected bounding boxes and improves the overall performance by a large margin.



**Figure 2.33** FCOSarchitecture

FCOS comprises the backbone, feature pyramid, and classification centre-ness regression section. Figure 2.33 shows the network architecture where  $H \times W$  is the height and the width of the feature map and  $'/s'$  is the down-sampling ratio of the feature maps at the level to the input image in the figure ( $s=8, 16, \dots, 128$ ). C3, C4, and C5 denote the feature map of the backbone network, and P3 to P7 denote the feature levels used for the final prediction. Finally, the classification is centre-ness regression, where each head includes the three predictions.  $'C'$  in the classification represents the total number of classes—for instance, 3 for the KITTI dataset.

We now move on to the technical and mathematical aspects of FCOS.

## B) Fully Convolutional One-Stage Object Detector

Let  $F_i \in \mathbb{R}^{H \times W \times C}$  be the feature maps at layer  $i$  of a backbone CNN and  $s$  the total stride till the layer. The ground truth bounding boxes for an input image are defined as  $\{B_i\}$ , where  $B_i = (x_0^{(i)}, y_0^{(i)}, x_1^{(i)}, y_1^{(i)}, c^{(i)}) \in \mathbb{R}^4 \times \{1, 2 \dots C\}$ .  $(x_0^{(i)}, y_0^{(i)})$  represents the coordinates of the left-top corner of the bounding box,  $(x_1^{(i)}, y_1^{(i)})$  represents the coordinates of the right-bottom corner of the bounding box, and  $c^{(i)}$  is the class that the object in the bounding box belongs to.  $C$ —as mentioned earlier—is the total number of classes.

In FCOS, a location with  $(x, \text{and } y)$  coordinates is considered a positive sample if it falls into any ground-truth box, and the class label  $c^*$  of the location is the class label of the ground-truth box. Otherwise, it is a negative sample and  $c^* = 0$ , where it represents a background class. Additionally, the network has a 4D real vector regression target for the location  $t^* = (l^*, t^*, r^*, b^*)$ . Here  $l^*$ ,  $t^*$ ,  $r^*$  and  $b^*$  are the distances from the location to the four sides of the bounding box. If a location  $(x, y)$  falls in multiple bounding boxes (ambiguous sample), the bounding box with the minimal area is considered the regression target.

The following are the mathematical equation used to calculate the location of the regression target for training when they have the location  $(x, y)$ , which is associated with  $B_i$  bounding box:

$$l^* = x - x_0^{(i)}, t^* = y - y_0^{(i)}, \tag{2.21}$$

$$r^* = x_1^{(i)} - x, b^* = y_1^{(i)} - y \tag{2.22}$$

It is worth noting that FCOS can leverage as many foreground samples as possible to train the regressor.

## C) Network output

Corresponding to the training targets, the final layer of the networks predicts an 80D vector  $p$  of classification labels and a 4D vector  $t = (l, t, r, b)$  bounding box coordinates.

In addition, four convolutional layers are added after the feature maps of the backbone networks for classification and regression branches.

#### **D) Loss function**

The loss function in FCOS is defined as follows:

$$L(\{p_{x,y}\}, \{t_{x,y}\}) = \frac{1}{N_{pos}} \sum_{x,y} L_{cls}(p_{x,y}, c_{x,y}^*) + \frac{\lambda}{N_{pos}} \sum_{x,y} \mathbb{1}_{\{c_{x,y}^* > 0\}} L_{reg}(t_{x,y}, t_{x,y}^*), \quad 2.23$$

where  $L_{cls}$  is focal loss (Lin, Goyal, et al., 2017),  $L_{reg}$  is the IoU loss (Zhou et al., 2017),  $N_{pos}$  is the number of positive samples, and  $\lambda$  is set to 1 to balance weight for  $L_{reg}$ . The summation is calculated over all locations on the feature maps  $F_i$ .  $\mathbb{1}_{\{c_{x,y}^* > 0\}}$  is the indicator function, being 1 if  $c_i^* > 0$  and 0 otherwise.

#### **E) Inference**

The image goes through the network and obtains the classification scores  $p_{x,y}$  and the regression prediction  $t_{x,y}$  for each location on the feature maps  $F_i$ . Moreover, to obtain the predicted bounding boxes, the location with  $p_{x,y} > 0.05$  is assumed to be a positive sample, and Eq.2.24 is inverted.

#### **F) Centre-ness for FCOS**

Centre-ness is the contribution proposed by (Tian et al., 2019). It is a simple yet effective strategy to suppress low-quality detected bounding boxes without introducing any hyper-parameters. The centre-ness depicts the normalized distance from the location to the centre of the object that the location is responsible for, and the target is defined as follows:

$$centerness^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}} \quad 2.24$$

The results of centre-ness range from 0 to 1, and the square root is employed to reduce the decay. Further details on FCOS can be found in (Tian et al., 2019).

### 2.5.3.5 Deformed Transformers for End-to-End Object Detection (Deformed DETR)

A deformed DETR is an extension of DETR. DETR or Detection Transformer was developed by Facebook in 2020 using transformers, as the name indicates, to detect objects. DETR is a set-based object detector using a Transformer on top of a convolutional backbone (Carion et al., 2020). In DETR, an image goes through a convolution Neural Network Encoder because CNN works best with images; the image features are then conserved. The feature map of the image thus produced is given to a transformer encoder-decoder, which outputs a set of box predictions. Each of these boxes consists of a tuple. The tuple will be a class and a bounding box. It is important to note that a class includes NULL or Nothing and its position as well. Having these classes causes a real problem, since in annotation, there is no object class annotated as nothing. To address this problem, they (Zhu et al., 2020) introduce a bipartite matching loss. This loss involves comparing each class and the bounding box associated with that class. It also includes a "none" class and a box that has no annotation. This way, ensure that the total number of boxes matches the value  $N$ . The assignment of the predicted to the actual is a one-to-one assignment, such that the total loss is minimized. The Hungarian (Kuhn, 1955) method is the method used to compute such minimum matching. Figure 2.34 illustrates the DETR framework.

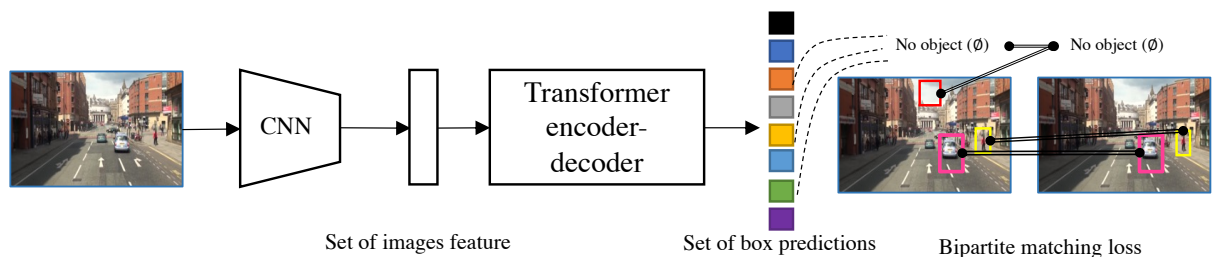


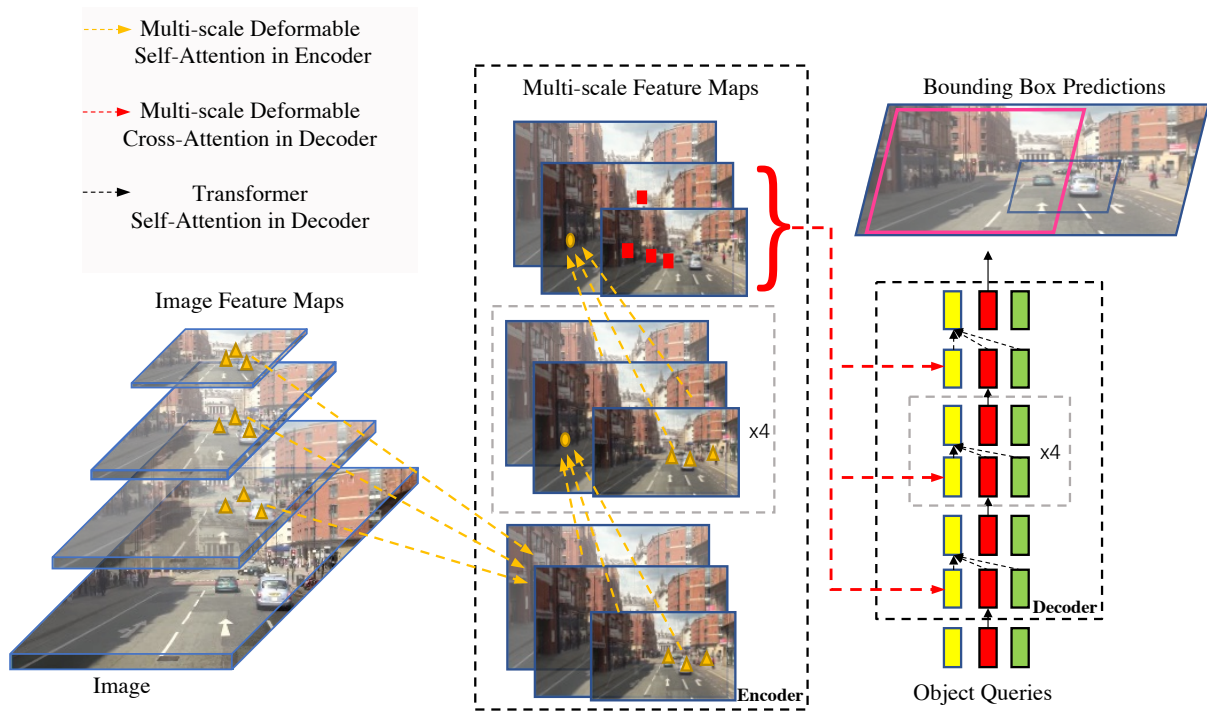
Figure 2.34 DETR framework

However, DETR suffers from slow convergence and limited feature spatial resolution on account of the limitation of Transformer attention modules.

### A) Deformable DETR

Deformed DETR (Zhu et al., 2020) was introduced to overcome these limitations. Deformed DETR aims to mitigate the slow convergence and high complexity issues of DETR. The detection network combines the advantages of deformable sparse spatial sampling and the relationship modelling capabilities of transformers. It considers the point that a small set of key sampling points of all feature map pixels is used as a pre-filter to highlight key elements. The module can be naturally extended to aggregate multi-scale features without the help of FPN; in deformable DETR, multi-scale is used. The deformable attention module replaces the transformer attention module that processes the feature map, as shown in the following figure, Figure 2.35.

#### i) Network Architecture



**Figure 2.35** Deformed DETR framework.

In deformable DETR, a multi-scale deformable attention module is utilized to replace the Transformer attention modules processing feature maps. Owing to its fast convergence its computational and memory efficiency, deformable DETR

provides the possibility to develop variants of end-to-end object detectors. Yet, the model explores a simple and effective iterative bounding box refinement mechanism to improve the detection performance. The discretion of the model is based on the work by Carion et al. (Carion et al., 2020).

## **2.5.4 DESCRIPTION OF DATASETS**

### **2.5.4.1 COCO**

COCO stands for Common Object in Context; it is one of the largest image recognition datasets. The datasets contain challenging and high-quality visual datasets for computer vision and was initially developed to push the state of the art in object detection forward. The dataset contains more than 330k images (>200K labelled images) with 80 object categories. This dataset has gone through several iterations—each one focuses on a different computer vision task, such as:

- Object detection
- Keypoint tracking
- Image captioning
- ‘Stuff’ detection

For this thesis, we based our work on the third iteration (COCO 2017). This dataset is used primarily for two cases:

- Training computer vision models, where the dataset provides a wide range of realistic images, showing disorganized scenes with various backgrounds, and overlapping objects. These enable training models on objects and people in realistic settings.
- Comparing AI models. In computer vision, to accurately compare performances, a model must be trained on a large-scale, standardized dataset, such as COCO.

The following are the sample images of the dataset (Figure 2.36).



Figure 2.36 Sample of COCO dataset images

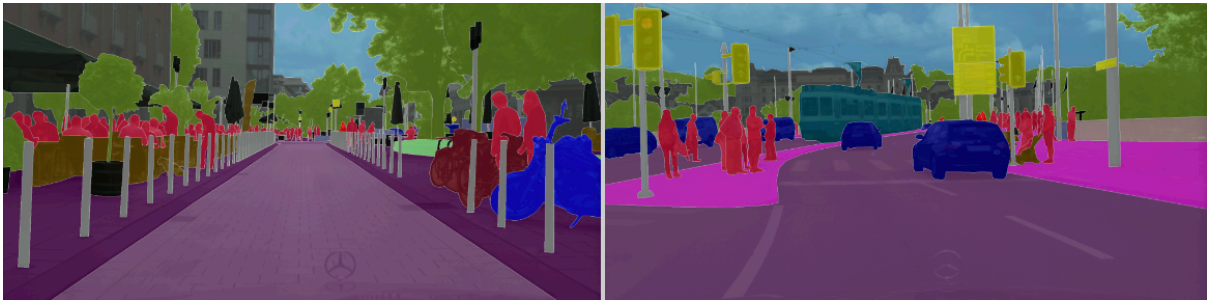
2.5.4.2

The Cityscapes dataset was created in 2016 by (Cordts et al., 2016) in order to understand the semantic of urban streets scenes. The data is comprised of a set of stereo video sequences recorded in streets from 50 different cities. In these, 5,000 of the images have high-quality pixel-level annotations but 20,000 additional images have coarse annotations. This dataset includes 30 classes related to road scenes like roads, sidewalks, parking, pedestrians, riders, cars, trucks, motorcycles, cycles, and other objects. Cityscapes specified a set for training which is 60% of the total data, set for validation which is 10% of the total data and a set for testing which is 30 % of the total data number.

The training set is typically separate from the test set, which is used to evaluate the performance of the trained model. The test set is also a subset of a dataset, typically separate from the training set used to train the model. The test set measures how well the model generalises to new, unseen data. The goal is to build a model that performs well on both the training and testing sets, indicating that the model has learned to recognise patterns in the data, rather than simply memorising the training set. A validation set is a subset of a data set that is used to evaluate the performance of a model during the training process. It is used to tune the model's hyperparameters, such as the learning rate or the number of layers in a neural network. The goal is to find the best set of hyperparameters that will result in a model that generalises well to new, unseen data. The validation set is usually separate from the training and test sets and is used to prevent overfitting, which occurs when a model learns to fit the training set too closely and performs poorly on new data.

Cityscape is used for deep learning training and testing purposes, especially by computer science and intelligent transport system researchers. (Hung, Tsai, Liou, Lin, & Yang, 2018) used Cityscape to train and test their proposed algorithm to improve semantic segmentation accuracy by coupling the adversarial loss with the standard cross entropy loss. Additionally, (Shu Liu, Qi, Qin, Shi, & Jia, 2018) tested their proposed network on this dataset; many other researchers also used the dataset for testing purposes (Kim & Kim, 2012; Songtao Liu, Huang, & Wang, 2020; Mao, Xiao, & Jiang, 2017; Han Wang, Li, & Wang, 2019) The following Figure 2.37 is an example of the high-quality image annotation.





**Figure 2.37** High quality cityscapes image annotation

Figure 2.38 illustrates the coarse annotations of the data, in which it is notable that data were not annotated accurately; the differences between the annotations can be seen.



**Figure 2.38** Coarse cityscape annotation

Figure 2.39 shows a sample of cityscapes images used for testing. It is important to notice that images in this dataset are saved in png format and annotation in json. Let us now move on to the second dataset used for training and testing algorithms, KITTI.



**Figure 2.39** Sample of Cityscapes dataset

### **2.5.4.3 KITTI**

Karlsruhe Institute of Technology and Toyota Technological Institute introduced a dataset called KITTI dataset in 2012 (Geiger, Lenz, & Urtasun, 2012), where it contains 7,481 training images and 7,518 test images, in a total of 80,256 labelled objects. The main purpose of developing and creating this dataset is for use as stereo, optical flow, visual odometry/SLAM, and 3D object detection tasks. Images for this dataset are saved in png and annotations in txt format. The dataset consists of 11 classes like cars, vans, trucks, pedestrians, cyclists, and trams. KITTI uses 50-0-50 split, where 50% of the data is specified as a training set and the remaining 50% is specified as the testing set. The KITTI dataset is used in many researches for detecting pedestrians, such as the autoregressive pedestrian detection (Brazil & Liu, 2020; Mao et al., 2017), in which the dataset is used to test the proposed aggregation features to boost the traditional pedestrian detection methods. (Al-Refai & Al-Refai, 2020) used the dataset to detect four classes: pedestrians, vehicles, trucks, and cyclists using You Only Look Once (YOLO) algorithm. Feng et al. proposed a system for fast and accurate object detection and localization based on binocular vision (Feng et al., 2020). KITTI dataset is also used for car detection, (Yebes, Bergasa, Arroyo, & Lazaro, 2014) carried out a discussion on the supervised learning of a car detector built as a Discriminative Part-based Model (DPM). Figure 2.40 depicts the diversity of objects that KITTI data include, where the images belong to different categories—city, residential road, campus, person from left to right.





**Figure 2.40** Sample of KITTI dataset

#### 2.5.4.4 EuroCity Persons

EuroCity Person (ECP) dataset—as the name indicates—is a dataset collected in different areas and cities around the European cities. The data was collected in 31 cities across 12 countries. The data contained over 238,200 person instances, manually labelled in over 47,300 images (Braun et al., 2019). This data focuses on annotating persons, wherein the annotation was divided into three overlapping data subsets, which were further defined after taking into consideration the ground-truth annotations:

- Reasonable: Persons with a bounding box height greater than 40 *px* which are occluded/truncated less than 40%
- Small: Persons with a height between 30 *px* and 60 *px* which are occluded/truncated less than 40%
- Occluded: Persons with a bounding box height greater than 40 *px* which are occluded between 40% and 80%

This dataset includes a day and night images, where images are saved in png format and annotations in json. ECP specified a set for training which is 60% of the total data, set for validation which is 10% of the total data and a set for testing which is 30 % of the total data number. Having the split explained, going onward in this thesis when mentioning training set means the training set as specified by the data itself.

Research such as Serial-to-Parallel Backbone Search for Object Detection used ECP for testing their desired backbone (Jiang, Xu, Zhang, Liang, & Li, 2021), Xie et al. (H. Xie, Zheng, Shin, & Proença, 2021) developed a novel pedestrian detector using a deformable attention-guided network and used ECP for evaluating the network. (Ren et al., 2017) (Hasan, Liao, Li, Ullah Akram, & Shao, 2021) (J. Zhang et al., 2020) were the researchers who used ECP to evaluate network purposes.

Figure 2.41 shows a sample image of an ECP dataset.





**Figure 2.41** Sample of ECP dataset



#### 2.5.4.5 Traffic field Dataset

A customized traffic field dataset for this, consisting of 300 images in five classes—cars, cyclists, pedestrians, buses, and motorcycles—were collected using a dash camera, with a 30 fps recording rate, and frame dimension of 1920 x 1080 pixels, and a 70° vertical field of view. The camera (Ring car cam) was located first at height  $H = 155\text{ cm}$  road surface and tilted at an angle  $\theta_c = 88.5^\circ$ . The camera used has two wide-angle, motion alerts, and a build in GPS. The images were collected in the United Arab Emirates in July, where the sky is sunny and clear and the wind is light. Images were then annotated manually using the Labellmg tool. A sample of the dataset is shown in Figure 2.42. Each dataset has its own resolution and annotation style; in order to use them for this thesis, resolutions and annotations were modified for training and testing purposes.



Figure 2.42 Sample of traffic field dataset

#### 2.5.4.6 Multiple Object Tracking Dataset

The Multiple Object Tracking (MOT) benchmark dataset is a challenging tracking dataset and has several variants, for this thesis MOT16 is used. The dataset contains fourteen challenging video sequences (7 training, 7 test). The videos are filmed in unconstrained environments using both static and moving cameras. All sequences have been annotated with high accuracy, strictly following a well-defined protocol (Milan et al., 2016).

#### **2.5.4.7 ImageNet Dataset**

ImageNet dataset (Deng et al., 2009) is an image database organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images. The data is widely used to build various architectures since it is large enough to create a generalized model (Huh, Agrawal, & Efros, 2016). The dataset is made of 14,197,122 images, 21841 synsets indexed, with an average of about 500 images per node (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, & Fei-Fei, 2015). Synsets index refer to the process of organizing and categorizing images based on their corresponding synset identifiers. Where synsets are groups of synonyms that represent a concept or idea. In ImageNet, each synset is assigned a unique identifier, which is used to index the images in that synset. ImageNet has played an important key role in advancing computer vision across applications such as object recognition, image classification, and object localization.

### **2.5.5 PERFORMANCE MEASURES**

Different evaluation metrics can be adopted in order to evaluate the performance of deep learning models. For instance, precision, recall, F1 score and IoU. Here, we shall implement a confusion matrix based on precision and recall results, Intersection Over Union, mean average precision, and (for model complexity) Floating Point Operation.

#### **2.5.5.1 Confusion Matrix**

A confusion matrix is a matrix that summarizes the performance of machine learning model on a set of test data. It is often used to measure the performance of



classification models, which aim to predict a categorical label for each input instance. Precision and recall are the performance metrics used in this thesis for creating the confusion matrix.

Two methods for counting classifiers in computer vision exist viz. the average precision and recall.

Precision refers to how well the classifier is able to correctly predict whether a character belongs to a certain category or not. If there is any doubt at all whether or not an image contains a cat, then it will most likely be labelled as 'not that'. This can also refer to classification tasks where each category has different levels, e.g., 1, 2, 3 and 4 are represented by 0/1/2/3 respectively. Recall refers to the ability of a model to find all the relevant cases within a dataset.

Precision measures how many of the predictions that the model made were correct values ranges from 0 to 1. It is calculated as follows:

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

2.25

*TP = True Positives (Predicted as positive as was correct)*

*FP = False Positives (Predicted as positive but was incorrect)*

Recall measures how well the model find all the positives values ranges from 0 to 1; it is calculated by:

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

2.26

*FN = False Negatives (Failed to predict an object that was there)*

Worth noticing is the fact that higher precision means that an algorithm returns more relevant results than irrelevant ones, and high recall means that an algorithm returns most of the relevant results. On having those values, a confusion matrix is then created. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.

### 2.5.5.2 F1 score

The F1 score is a metric commonly used in machine learning and statistics to evaluate the performance of a binary classification model. The F1 score is the harmonic mean of precision and recall. It combines both precision and recall into a single value, providing a balanced measure of a model's performance:

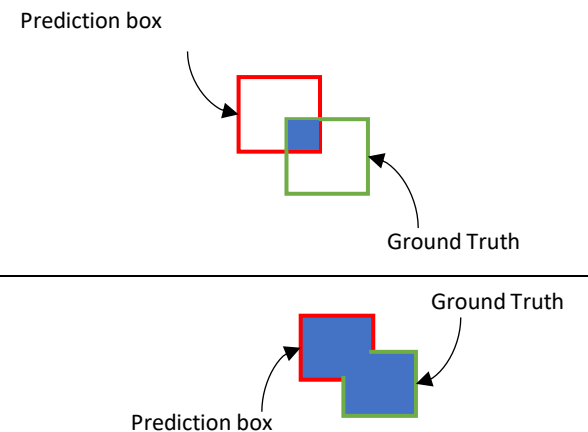
$$F1 = 2 \cdot \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.27

Although this metrics was discussed here it will not be used it is mentioned for completion purposes This evaluation metrics will not be used in this thesis.

### 2.5.5.3 Intersection Over Union

It is important to notice that for object detection systems, the predictions are in terms of a bounding box and a class label. For each bounding box, the overlap between the predicted bounding box and the ground truth bounding box needs to be measured. This is measured by IoU (Intersection Over Union).


$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of overlap} + \text{area of non-overlap}}$$

2.28

In addition, in tasks for object detection, Precision and Recall are calculated using the IoU value for a given IoU threshold. For example, if the IoU threshold is 0.5, and the IoU value for a prediction is 0.7, then we classify the prediction as True Positive (TP). On the other hand, if the IoU is 0.3, we classify it as False Positive (FP).

#### 2.5.5.4 Mean Average Precision

The mean Average Precision (mAP) score is a metric also used to evaluate the performance. It is calculated by taking the average precision scores for each query in a dataset and then averaging those scores across all classes and/or overall IoU threshold, depending on the different detection challenges that exist.

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

2.29

where the  $AP_i$  is the average precision (AP) of class  $i$ ,  $N$  is the number of classes.

Average Precision (AP) at  $i$  is the sum of the precision at  $i$  of the values of  $i$  divided by the total number of relevant items in the top  $i$  results.

mAP is a popular metric because it takes into account both Precision and Recall and provides a single number that summarizes the overall performance of a retrieval system.

#### 2.5.6 INFERENCE TIME MEASUREMENT

For real time application, inference time is a variable that has to be measured to optimize a neural network. Inference time can be defined as the time required for forward propagation. In other words, it is the amount of time it takes a machine learning model to process new data and make a prediction. This evaluation was implied mainly in Chapter 6, where Traffic RetinaNet was developed. To get the number of Frames per Second:

$$Frames\ per\ Second = \frac{1}{inference\ time}$$

2.30

To calculate the inference time of a model, FLOPs, and FLOPS must be considered.

### 2.5.6.1 Floating Point Operations (FLOPs)

A Floating Point Operation is any mathematical operation (such as addition, subtraction, multiplication, and division) or assignment that involves floating-point numbers (as opposed to binary integer operations). FLOPs give the complexity of the model. The total number of computations the model will have to perform is first calculated. For instance, to calculate the convolution when using a sliding window and on ignoring the nonlinear overhead computation, the FLOPs is calculated as follows (Hurtado 2022):

$$FLOPS = 2HW(C_{in} K^2 + 1)C_{out} \quad 2.31$$

Here  $H, W$  and  $C_{in}$  are the height, width, and number of channels in the input feature map.  $K$  is the size of kernel and  $C_{out}$  is the number of channels in the output.

### 2.5.6.2 Floating Point Operation per Second (FLOPS)

Floating Point Operation per second refers to the number of floating-point operations a computing entity can perform in one second. FLOPS is used to quantify the performance of hardware (Jeon 2021). The more operations per second can be done, the faster the inference will be it is calculated in multiplier-accumulator (Mac).

Finally, inference time is calculated as follows:

$$inference\ time = \frac{FLOPs}{FLOPS} \quad 2.32$$

## 2.6 Research Scope

The concept of automated driving is one of the most recent and attractive fields for research, especially researches into the intelligent transportation system, since it comes with a wide variety of possible applications. Owing to the timeframe and limited resources, a selection with regard to the scope of this research must be made. First of all, the motivation behind carrying out this research is to increase the safety of pedestrians and cyclists. A number of research gaps in the literature have been identified after conducting the literature review.

Several research have been carried to study the acceptance of AV however most studies where based on hypothetical assumptions. This research will focus on analysing the before/after perception of autonomous vehicles by utilizing a technology acceptance model. This will provide insight into both people's willingness to adopt technology and their associated concerns. Again, our results will help researchers while conducting further research in the field of AVs.

Additionally, this research addresses the detection of road objects, specifically pedestrians and cyclists, from the perspective of autonomous vehicles. The focus is on evaluating performance across multiple datasets using various algorithms, introducing a nuanced analysis where the literature lacks in such complexity analysis. This approach contributes to the advancement of knowledge in autonomous vehicle technology and enhances our understanding of intricate object detection scenarios.

Furthermore, this thesis develops an urban benchmark dataset meticulously designed to account for diverse weather, lighting, and driving conditions in the context of pedestrian and cyclist detection. Notably, there is a scarcity of urban datasets that specifically address the variability in weather, lighting, and driving conditions. The five algorithms performance—Faster RCNN, Cascade RCNN, RetinaNet, FCOS, and Deformable DETR—is then evaluated.

Additionally, for our purpose, we will use different methods to achieve the aim of the thesis: detection, tracking-by-detection, and estimation will be developed for autonomous vehicles use. However, we will not fully go into the details of the driving: the aim is to apply the methods studied to autonomous vehicles. Integrating these methods and algorithms into an autonomous vehicle is beyond the scope of this thesis.

In the next chapter, the thesis delves into a unique facet of the survey study, which centers around the acceptance of AV using the Technology Acceptance Model. Unlike many existing studies that rely on hypothesis-based evaluations, the approach offers a distinctive perspective. The study focus on participants who have had the opportunity to experience AV vehicles first-hand. By doing so, I aim to shed light on the practical acceptance of AV technology, drawing from real-world encounters and user experiences. The chapter explores the invaluable insights derived from this approach and its implications for the broader understanding of AV acceptance.

---

# Chapter 3

---

# 3 The Acceptance of Autonomous Vehicles in a Mixed Mode Environment

## 3.1 INTRODUCTION

The development of the transportation sector entails the introduction of more advanced modes of transportation like flying taxis, autonomous vehicles (AVs), hyperloops, and delivery drones. AVs are a ground-breaking concept, one that will significantly change how people travel in cars and how cars will behave within the transport network. The inclusion of AVs will impact the safety, control, convenience, and user perception of travel in cars and also generate fundamental changes as regards safety and behaviour on the roads where AVs are allowed to operate. These changes are extensive and need to be studied for their possible societal acceptance—or rejection—if implemented.

To better understand participant acceptance of AVs, a before-use and after-use survey of AVs was conducted (refer to Appendix A). The Technology Acceptance Model (Venkatesh & Davis, 2000) was used. Participants were asked to take the survey twice: once before using a self-driving vehicle and once after using it to find out how it affects their behaviour. The different questions and parts of the survey focused on analyses of the four different aspects that concerns users the most (safety, environment, contusion, and anxiety).

This chapter is divided into five sections. The first section highlights the theoretical concept of the Technology Acceptance Model (TAM); the second section deals with the reliability of the experimental details and a validity test analysis, followed by the demographics of the survey questionnaire. The second section is primarily about the safety aspect of autonomous vehicles and includes an analysis of the responses of the participants. The third section is about the data analysis of the environmental impact of autonomous vehicles while the fourth section examines the benefits of autonomous vehicles for users. The fifth section investigates anxiety related to autonomous vehicles while the last section connects the dots.

EuroCity Person. Many of the abovementioned algorithms were tested on COCO (Common Objects In Context) (Caesar et al., 2018).

### 3.2 TECHNOLOGY ACCEPTANCE MODEL (TAM)

The potential of technology to provide benefits has long motivated information systems (IS) management researchers to examine the willingness of the acceptance of innovative technology by individuals (Davis et al., 1989). In the 1980s, the adaptation of technology research became a primary research field, thanks to the growth in the use of personal computers. Several technological and organizational perspectives had aimed to advance IS-related research, such as (Benbasat, Dexter & Todd, 1986; Robey & Farrow, 1982; Franz & Robey, 1986), wherein they studied the acceptance of technology. These studies, however, used subjective performance perception scales and neglected the validation of the quality of those measures. As a result, the correlation of these subjective measures with actual use had not been sufficiently significant to confirm their internal and external validity (De Sanctis, 1983; Ginzberg, 1981; Schewe, 1976; Srinivasan, 1985). Hence, there was a need to develop reliable measures to investigate attitudinal factors. In 1989, Fred Davis (Davis et al., 1989) proposed a model called the Technology Acceptance Model (TAM), through which the possible use of a new technology/system can be studied and predicted by the motivation of the user and the features and capabilities of the system (Chuttur, 2009).

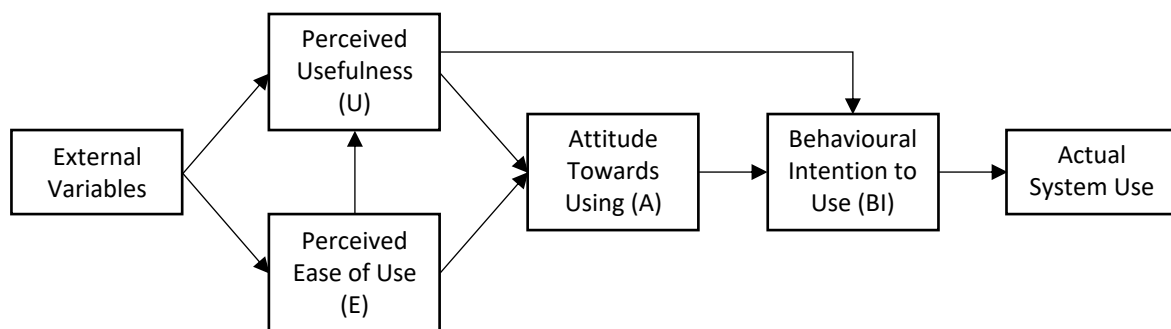
TAM was initially developed to highlight the process underpinning the acceptance of technology, to predict the behaviour of—and provide a theoretical explanation for—the successful implementation of technology. In addition, the objective of TAM is to inform practitioners about measures that they might take prior to the implementation of systems. For this, Davis embarked on the development of the model of technology acceptance by framing the processes mediating the relationship between IS characteristics (external factors) and actual system use. The model was based on Theory of Reasoned Action (Fishbein & Ajzen, 1977), which provided a psychological perspective on human behaviour that was missing in the IS literature at that time (Davis et al., 1989; Venkatesh et al., 2003).

The Technology Acceptance Model (TAM)—as shown in Figure 3.1—is a model with two main features, namely perceived ease of use and perceived usefulness. These two variables can be used to predict how innovations like autonomous vehicles are being accepted or rejected (Ma & Liu, 2005). Many research works have proven the effectiveness of TAM and many works still make use of TAM to



investigate the acceptance or rejection of an innovation (Holden & Rada, 2011; Lederer, Maupin, Sena, & Zhuang, 2000; Ma & Liu, 2005; Wintersberger, Frison, & Riener, 2018). Therefore, TAM has emerged as the most widely applied model to study user acceptance and use of information technology (Ma & Liu, 2005). In the case of autonomous vehicles (AVs), TAM can be used to predict how the participants in the survey accept or reject autonomous vehicles.

In the past few decades, the acceptance and adoption of innovation have received large attention to the extent that many interesting theoretical models have been developed to understand the end user's acceptance gestures. One of these models is the Technology Acceptance Model (TAM) which was developed by Davis (1989); this model has been applied in several fields to evaluate the acceptance of information technologies. It must be noted that TAM has been tested empirically, and is effective, parsimonious, predictive, and robust (Venkatesh & Davis, 2000). There are great benefits attached to the use of autonomous vehicles, but one would not know how users perceive the innovation till research is conducted on the topic. Considering the importance and relevance of this problem, the study of users' acceptance of innovation has been at the forefront of the research subject; however, despite the rich research that has been conducted to investigate some variables relating to individual, organization, and technology, it is still observed that there is inadequate information about the key factors that determine users' acceptance or rejection of innovation (Ma & Liu, 2005).



**Figure 3.1** Technology Acceptance Model (TAM).

### 3.2.1 Perceived usefulness and perceived ease of use

What motivates some people to embrace innovation and what prompts others to resist innovation? Answers to these questions have great potential to reveal the key factors that need to be considered when innovations like autonomous vehicles are to be adopted. According to the literature, many variables determine and influence system use. The two main factors—identified as exceptionally important in the study of the systems—are: (1) the factors that make people to use, or not use, an innovation is what is referred to as *perceived usefulness* and (2) the factor that deals with the view of people in respect of the complexity of innovation and the requisite knowledge to operate innovations like autonomous vehicles is what is referred to as *perceived ease of use* (Davis et al., 1989).

*Perceived usefulness* can be defined as the extent to which a user believes that by deploying information technology, his or her job performance would be improved (Davis et al., 1989). This definition stems from the definition of the word ‘usefulness’, which states that the user believes that by using a particular system of information technology, the accuracy, precision, and speed of his or her job would be influenced positively, thereby increasing his or her performance at the job position.

In contrast, *perceived ease of use* can be defined as the extent to which a user believes that deploying information technology at his or her workplace would be effort-free (Davis et al., 1989). This stems from the definition of the word ‘ease’, which means ‘freedom from difficulty or great effort’. Davis (1989) argued that the information technology with the lowest level of complexity is more likely to be received by users than the ones with the higher levels of complexity.

### 3.2.2 Mathematical theory

In order to predict the value of the dependent variable for individual information related to the explanatory variables that is available, or to estimate the effects if a regression analysis of the explanatory variables on the dependent variable is carried out, a regression analysis was conducted for this chapter. A regression analysis is used to identify the impact of variables on the topic of interest. This process determines the factor that matters most, the factor(s) that can be ignored, and how the factors influence each other. It is calculated as follows:

$$Y = a + bX + E$$

3.1

where  $Y$  is the dependent variable,  $X$  is the independent variable,  $a$  is the intercept,  $b$  is the slope, and  $E$  is the residual.

Moreover, the structural equation modelling (Kaplan, 2001), which is a multivariate statistical analysis technique used to analyse structural relationships, is the combination of factor analysis and multiple regression analysis. It is used to analyse the structural relationship between measured variables and latent constructs. The structural part is calculated as follows:

$$\eta = B\eta + \Gamma\xi + \zeta$$

3.2

where  $\eta$  is a vector of endogenous (criterion) latent variables,  $\xi$  is a vector of exogenous (predictor) latent variables,  $B$  is a matrix of regression coefficients relating the latent endogenous variables to each other,  $\Gamma$  is a matrix of regression coefficients relating endogenous variables to exogenous variables, and  $\zeta$  is a vector of disturbance terms.

Next, the latent variables have to be calculated. When the latent variables are linked to observable variables via measurement equations for the endogenous variables and exogenous variables, the equations are defined as follows (Kaplan, 2001):

$$y = \Lambda_y\eta + \varepsilon$$

3.3

$$x = \Lambda_x\xi + \delta$$

3.4

where  $\Lambda_y$  and  $\Lambda_x$  represent the matrices of factor loadings, and  $\varepsilon$  and  $\delta$  are the vectors of uniqueness. In addition, the general model specifies variances and covariances for  $\xi$ ,  $\zeta$ ,  $\varepsilon$  and  $\delta$  denoted  $\Phi$ ,  $\Psi$ ,  $\Theta_\varepsilon$  and  $\Theta_\delta$  respectively.

Structural equation modelling is a methodology designed to test substantive theories. As such, a theory might be sufficiently developed to suggest that certain constructs do not affect other constructs, that certain variables do not load on certain factors, and that certain disturbances and measurement errors do not covary. Yet, this indicates that some elements of  $B$ ,  $\Gamma$ ,  $\Lambda_y$ ,  $\Lambda_x$ ,  $\Phi$ ,  $\Psi$ ,  $\Theta_\varepsilon$  and  $\Theta_\delta$  are *fixed*

to zero by hypothesis, and the remaining parameters are *free* to be estimated. The pattern of *fixed* and *free* parameters implies a specific structure for the covariance matrix of the observed variables. Hence, structural equation modelling can be seen as a special case of a more general *covariance structure model* defined as follows:

$$\Sigma = \Sigma(\Omega) \tag{3.5}$$

where  $\Sigma$  represents the population covariance matrix, and  $\Sigma(\Omega)$  represents the matrix valued function of the *parameter vector*  $\Omega$ , where it contains all model parameters (Kaplan, 2001).

Moreover, in order to estimate BI, personal attitudes towards using system (A) and perceived usefulness (U) are joined, and then the following equation is used to calculate the estimated weight:

$$BI = A + U \tag{3.6}$$

### 3.3 EXPERIMENT AND DATA COLLECTION

The TAM model was built by collecting data after carrying out a survey through a group of volunteers. The survey was based on a questionnaire; participants responded to the same, or a similar, set of questions at two stages: once before using AVs, and a second time after using them.

#### 3.3.1 Survey Design

The survey was carried out to assess the role and acceptance of connected autonomous vehicles in a mixed-mode urban mobility environment; its aim is to measure the strengths and weaknesses of this fast-growing technology. It contains three sections. One section investigates the demographic information, wherein it provides background information on the survey participants. The next section includes an evaluation of autonomous vehicles, where open-ended questions were posed and analysed; finally comes the variables analysis section (safety, environment, conjunction, and anxiety), in which section participants were asked to rate AVs, based

on a 1–5 scale. The statistical analysis was carried out, wherein participants were asked to fill this section twice—once before using AVs, and the second time after using it. For this section, it was ensured that questions belong to different categories viz. safety, environment, conjunction, and anxiety. Questions for each of the factors can be seen below:

**Table 3.1 Questions asked for TAM before and after analysis**

<b>Perceived usefulness:</b>	(i) Self-driving vehicles generate fewer accidents.
<b>Perceived Ease of Use:</b>	(i) Self-driving vehicles offer more personal freedom and independence.
<b>Attitude Towards Use:</b>	(i) Before/after using AVs, I was afraid an emergency would arise as the vehicle might malfunction.  (ii) Before/after using AVs, I was worried if all my journeys with AVs would be successful.
<b>Behavioural Intention to Use:</b>	(i) I was afraid I would not be able to react in case of an emergency.

The full survey can be found in Appendix A and all before and after questions are clearly mentioned.

### 3.3.2 Administration Method

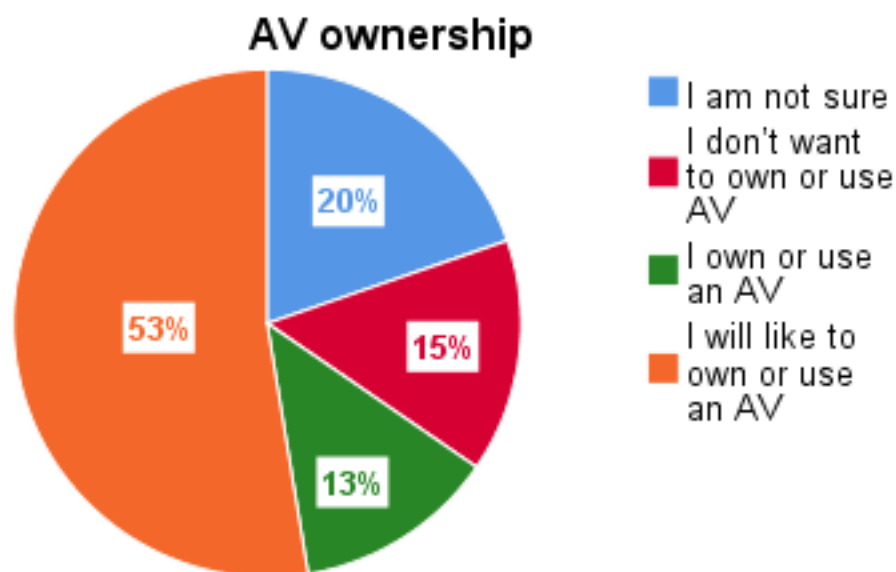
The survey is a self-administered questionnaires that was delivered online, using the SurveyMonkey tool. A self-administered questionnaire is a questionnaire that is designed to be completed by a respondent without the physical intervention of the researcher. The before-survey was out in March 2021 and remained open for three months, after which participants were then asked to take the after-survey in September 2021. This survey remained open till December 2021. A total of 350 participants from different backgrounds completed the survey. All participant responses are treated with the utmost confidentiality, and personal information is anonymized to eliminate any potential identification. The data was collected and stored in a secure and confidential manner where it is accessible only to authorized personnel directly involved in the research, and individuals' privacy and consent

were upheld throughout the study. Additionally, any potential biases were identified and addressed in the analysis of the data. In the following sections, the responses will be analysed and conclusions drawn (Wolf, 2008).

### 3.4 DATA DESCRIPTION AND MEASUREMENTS

#### 3.4.1 Demography data analysis

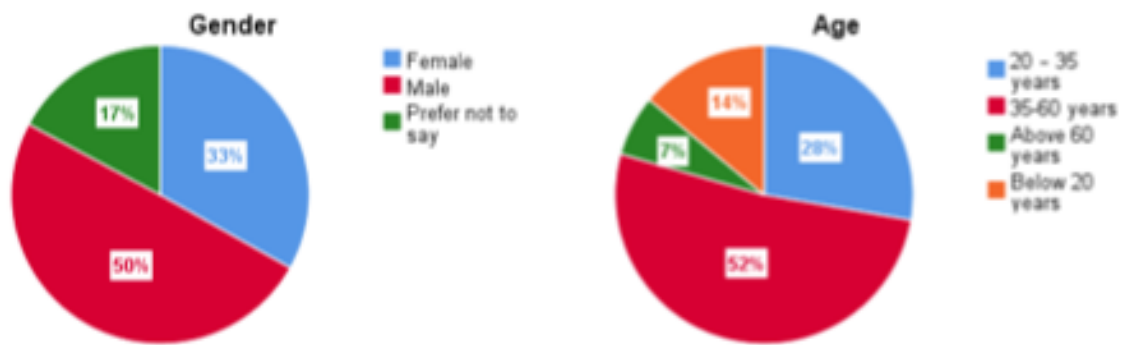
In this study, 350 survey questionnaires were administered to different categories of people, ranging from those working in the automotive industry, the technology sector, regulatory agencies, and so on. The demographic details included: ownership of autonomous vehicles, gender, age, education, employment sector, and years of experience.



**Figure 3.2** Which category describes you the best?

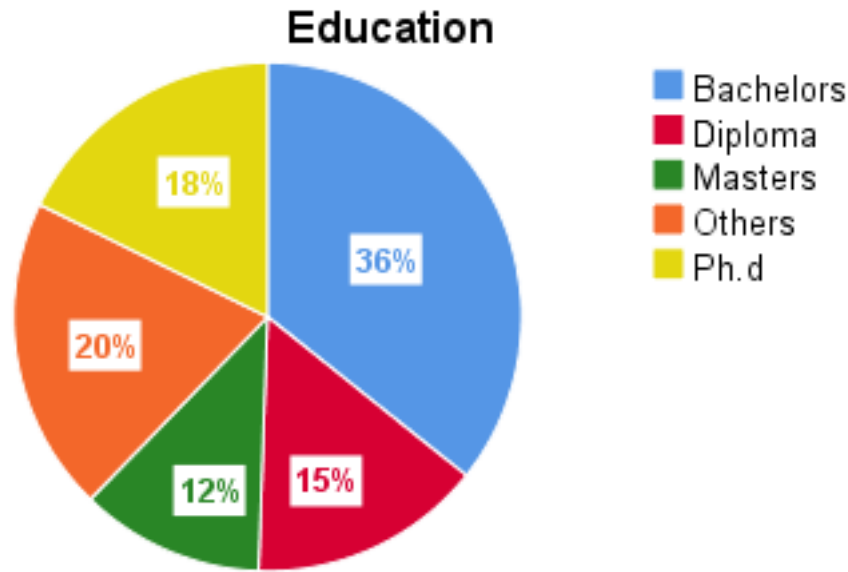
Figure 3.2 show the statistics of the category of the participants with regard to the possession of autonomous vehicles (AVs). From these statistics, it can be seen that 'I will like to own or use an AV' has the highest percentage of 53%, followed by 'I am not sure' (with a percentage of 20%); 'I don't want to own or use AV' logs a percentage of 15%, and lastly 'I own or use an AV' records a percentage of 13%. It is evident from these statistics that the largest percentage of the participants did not have an

autonomous vehicle (AV) at the time they responded to the survey (see the percentage of those saying ‘I will like to own or use an AV’). About 15% of the participants declared that they did not want to own or use an AV. Notably, the lowest percentage (13%) of the participants own or use an autonomous vehicle (AV).



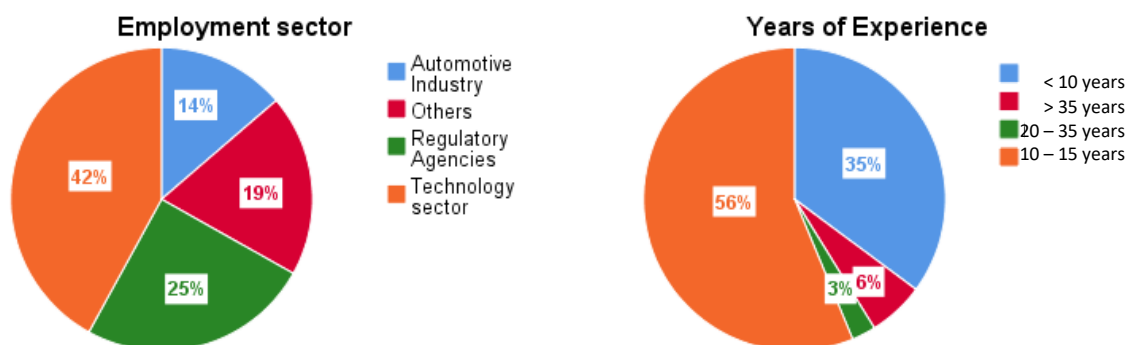
**Figure 3.3** Gender & Age

One of the important aspects of the demographic data is the gender of the participants. In this work, three categories of gender were made available in the survey questionnaire: Male, Female, and prefer not to say. From Figure 3.3, it can be observed that ‘Male’ has the highest percentage of 50%, followed by Female, with a percentage of 33%, while the ‘Prefer not to say’ category has a percentage of 17%. Therefore, we can conclude that about half the participants in the survey are Male and 33% are Female. Again, Figure 3.3—which show the percentage of the ages of the participants—the age categories are classified as follows: Below 20 years, 20–35 years, 35–60 years, above 60 years. ‘35–60 years’ logs a percentage of 52%, ‘20–35 years’ has a percentage of 28%, ‘Below 20 years’ has a percentage of 14%, and ‘Above 60 years’ has a percentage of 7%. It can, therefore, be deduced that the largest percentage of participants in the survey is within the age range of 35–60 years, and the lowest percentage of the participants is in the age group above 60 years. A good number (49) of participants also are below 20 years.



**Figure 3.4** Education levels.

The educational levels of the participants were also found useful and were included in the survey. The education levels were classified as: Ph.D., Master’s, Bachelor’s, Diploma, and Others. From Figure 3.4, Bachelor’s degree holders logged a percentage of 36%, ‘Others’ had a percentage of 20%, ‘Ph.D.’ holders recorded a percentage of 18%, ‘Diploma’ holders had a percentage of 15%, and ‘Master’s’ degree holders had a percentage of 12%. Therefore, the majority of the participants in the survey are Bachelor’s degree holders, followed by the participants with education levels not included in the survey. A very good number (62) of the participants are Ph.D. holders.



**Figure 3.5** Employment sector and years of experience.

The employment sectors in which the participants worked were also included in the survey questionnaire. The employment sectors considered in the survey are primarily ‘Automotive Industry’, ‘Technology Sector’, and ‘Regulatory Agencies’. ‘Others’ cover



those employment sectors not listed in the survey. From Figure 3.5 (left), 'Technology Sector' has the highest percentage of 42%, 'Regulatory Agencies' has a percentage of 25%, 'Others' accounts for a percentage of 19%, and, lastly, 'Automotive Industry' shows a percentage of 14%. From these statistics, it can be deduced that the largest percentage of the participants work in the 'Technology Sector', followed by the participants working at 'Regulatory Agencies', while the lowest percentage of the participants work in the 'Automotive Industry'. The years of experience of the participants at their respective workplaces were also examined. The years of experience were divided into four categories: '< 10 years'; '10-15 years'; '20-35 years'; '>35 years'. From Figure 3.5 (right), '10-15 years' of experience has the highest percentage of 56%, '<10 years' of experience has the percentage of 35%, '>35 years' of experience has a percentage of 6%, and '20-35 years' of experience shows a percentage of 3%. Therefore, it can be observed that the participants with working experience of '10-15 years' dominated the survey response. This was followed by the participants with less than 10 years of working experience. The participants with 20-35 years of working experience recorded the lowest percentage.

#### **3.4.2 Reliability and Validation of the Measurement Model**

To ensure the accuracy of data, reliability and validity tests were conducted. Reliability refers to the degree to which any measuring tool controls for random error. In general, the higher the reliability coefficient (Cronbach's alpha), the more repeatable or reliable the test. When the Cronbach's alpha coefficient is 0.6 or higher, the reliability is considered high enough (Nunnally et al., 1967). Validity refers to the extent in which a concept is accurately measured in a quantitative study (Heale & Twycross, 2015). To evaluate the validity construct in this study, convergent validity was utilized. The results of the reliability and validity tests were reported in the following table (Table 3.2):

**Table 3.2** Reliability and Validity Tests

Variables	Reliability	Validity			
	Cronbach's alpha	KMO	P-value Bartlett's Test	AVE	Composite Reliability
Threshold value	>0.7	>0.5	Sig. <0.05	>0.5	>0.6
<b>Independent Variables</b>					
Safety	0.829	0.714	<0.000	0.371	0.646
Environment	0.848	0.706	<0.000	0.423	0.747
Conjunction	0.555	0.601	<0.000	0.244	0.436
Anxiety	0.859	0.857	<0.000	0.119	0.435
All variables	0.939	0.904	<0.000	0.245	0.828

Table 3.2 shows the reliability and validity test results, which include Cronbach's alpha, Kaiser–Meyer–Olkin (KMO) test, P-value Bartlett's Test, Average Variance Extracted (AVE), and Composite Reliability. The test was carried out on each of the independent variables, viz., safety, environment, conjunction, and anxiety. Each of the independent variables was connected to questions, from which the responses of the participants were recorded. In addition, each test was carried out on the aggregate variable, i.e., the sum of the independent variables. From these statistics, considering the fact that Cronbach's alpha value is above the threshold value (0.7) for all the independent variables, except 'conjunction' (with 0.555), it can be concluded that the survey is reliable. This also reveals that the responses of the participants are consistent across the variables. For the validity of the survey, out of the four (4) tests carried out, three (3) proved the validity of the survey. Even though the AVE test results were below the threshold value, the survey can still be rendered valid as the results of the other tests appear to be strong (Fornell & Larcker, 1981).

Descriptive statistics shows the basic details about the responses of the participants. The number of the responses (N) vary for each measured independent variable (V) on account of some missing data; constructs were measured through multiple items on a 5-point Likert scale, ranging from 1 (strongly agree) to 5 (strongly disagree);

mean values have their whole number to be 2 though—when approximated—some of the responses are neutral on average. On an average, the responses of the participants range from ‘Agree’ to ‘Neutral’ (the statistical representation can be found in Appendix B Table 0.1).

## **3.5 ANALYSIS AND RESULTS**

### **3.5.1 Frequency and Comparative Analysis**

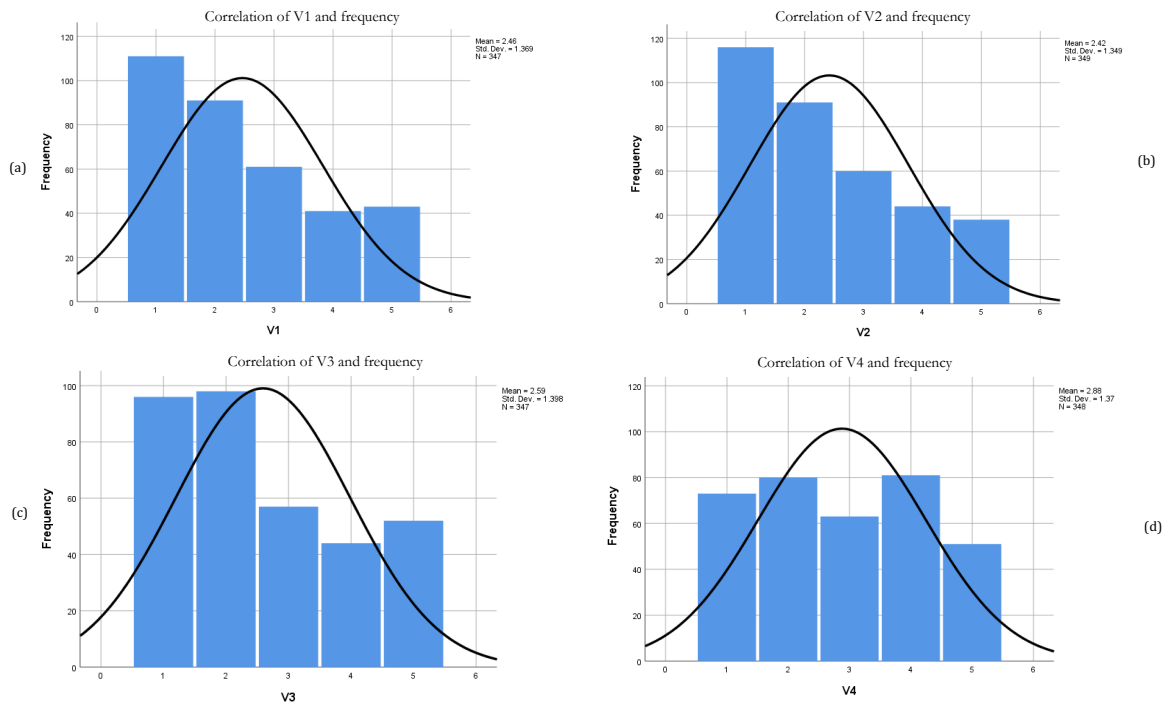
In this section, the responses of the participants are analysed by illustrating the responses to each of the sections. Again, the responses for each of the independent variables are now compared with one another to extract useful information and test how the decisions of the participants in one variable relate to the other.

In the following, each independent variable (hypothesis) will be evaluated and an analysis based on the relevant sections conducted.

#### ***A) Safety***

The safety aspect of AVs was evaluated based on four hypotheses:

- V1: Self-driving vehicles generate fewer accidents.
- V2: Self-driving vehicles decrease traffic congestion.
- V3: Unlike ordinary vehicles, which may well be operated by drunk or distracted drivers, self-driving vehicles will reduce risky driving behaviours.
- V4: Self-driving vehicles outperform humans in detecting dangerous situations.



**Figure 3.6** Safety statistics: (a) Statistical representation of the notion that self-driving vehicles generate fewer accidents, (b) Statistical representation of the claim that self-driving vehicles decrease traffic congestion, (c) Statistical representation of the idea that self-driving vehicles will reduce risky driving behaviours, (d) Statistical representation of the assertion that self-driving vehicles outperform humans in detecting dangerous situations.

The statistics related to the responses of the participants to the notion that self-driving vehicles generate fewer accidents were analysed. From these statistics, it can be observed that ‘Agree’ and ‘Strongly Agree’ dominated the responses to the survey, implying that the majority of the participants believed that self-driving vehicles generate fewer accidents as Figure 3.6(a) presents.

The responses of the participants to the notion that self-driving vehicles decrease traffic congestion are similar to the previous notion: the majority of the participants also chose ‘Agree’ and ‘Strongly Agree’ as their responses to the notion. Hence, it can be said that the participants believed that self-driving vehicles decrease traffic congestion ( Figure 3.6 (b)).

The notion ‘Since there would be no drunk or distracted drivers, self-driving vehicles will reduce risky driving behaviours’ received responses of the participants as shown in (Figure 3.6(c)). From these statistics, it can be seen that combining the responses for ‘Agree’ and ‘Strongly Agree’, the majority of the participants believed that since there would be no drunk or distracted drivers, self-driving vehicles would reduce

risky driving behaviours. This is similar to the patterns observed in connection with the previous two notions.

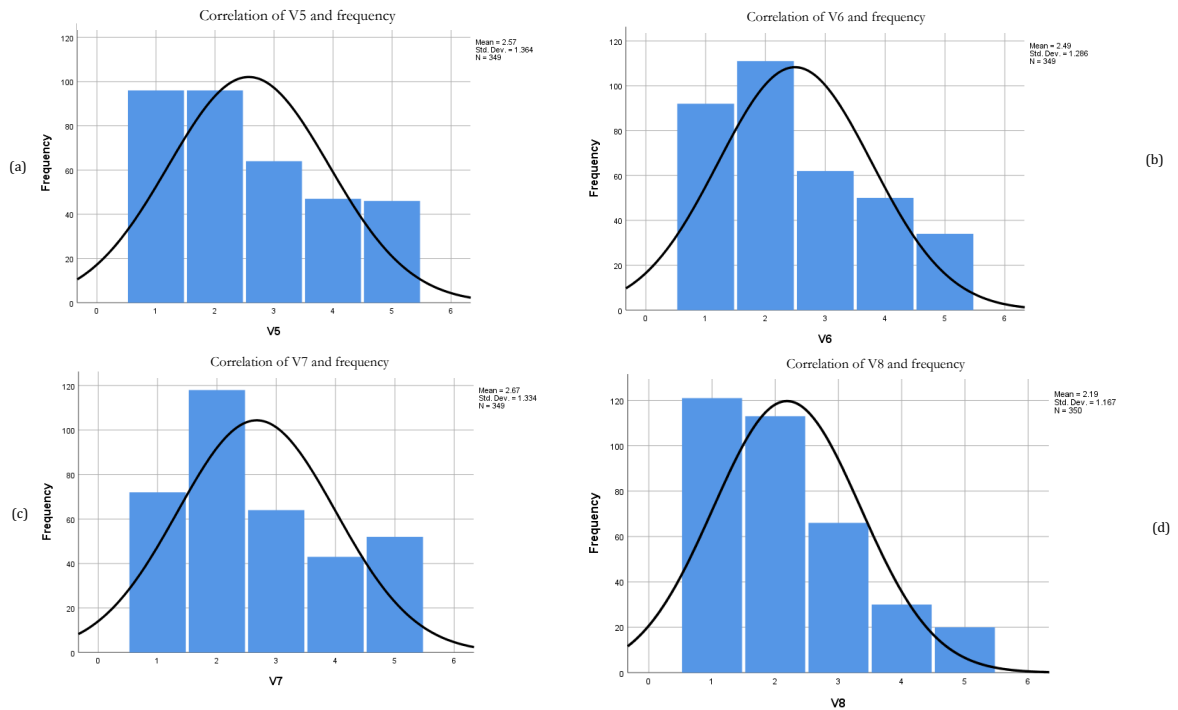
Again, from Figure 3.6(d), it can be observed from the responses of the participants to the survey that both 'Agree' and 'Disagree' had similar percentages (of approximately 23%). Combining the percentages of 'Agree' and 'Strongly Agree', we have 44%, while those of 'Disagree' and 'Strongly Disagree' yielded 38%. Hence, it can be deduced that the majority of the participants believed that self-driving vehicles outperform humans in detecting dangerous situations.

In comparison with the safety section of the evaluation of autonomous vehicles, the majority of the participants in the survey believed that self-driving vehicles generate fewer accidents and self-driving vehicles decrease traffic congestion, about half of the participants believed that as there would be no drunk or distracted drivers, self-driving vehicles will reduce risky driving behaviours, and about 44% of the participants believed that self-driving vehicles outperform humans in detecting dangerous situations. On evaluating the safety aspects based on the survey, most participants agree that autonomous vehicles appear to be safe, and that they would lead to a safer transport fleet.

## ***B) Environment***

The environmental aspect of AVs was evaluated on the basis of the following five hypotheses:

- V5: Self-driving vehicles will reduce energy consumption.
- V6: Self-driving vehicles are environmentally friendly.
- V7: Self-driving vehicles cause increases in car use and emissions.
- V8: Self-driving vehicles increase the number of miles people travel, thereby increasing pollution.
- V9: Self-driving vehicles will free up public spaces and promote clean air.



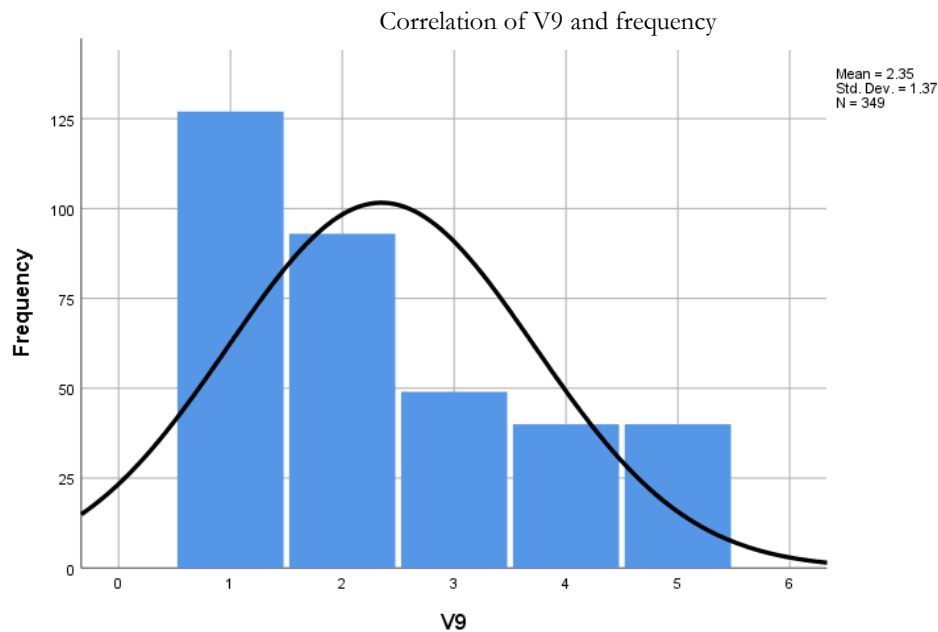
**Figure 3.7** Environmental statistics: (a) Statistical representation of the notion that self-driving vehicles will reduce energy consumption, (b) Statistical representation of the idea that self-driving vehicles are environmentally friendly, (c) Statistical representation of the notion that self-driving vehicles cause an increase in car use and emissions, (d) Statistical representation of the idea that self-driving vehicles will increase the number of miles people travel.

Under the environmental evaluation of autonomous vehicles, the notion ‘self-driving vehicles will reduce energy consumption’ received responses as shown in Figure 3.7(a). From these statistics, it can be seen that ‘Agree’ and ‘Strongly Agree’ recorded the maximum percentages. Hence, it can be inferred that a substantial number of the participants believed that self-driving vehicles would reduce energy consumption.

The notion that ‘self-driving vehicles are environmentally friendly’ received responses as shown in Figure 3.7(b). The ‘Agree’ level has the highest percentage of about 32%, followed by ‘Strongly Agree’. A larger number of the participants evidently believed that self-driving vehicles are environmentally friendly. This pattern is similar to the pattern of the responses for the notion ‘self-driving vehicles will reduce energy consumption’.

One of the notions stated that ‘self-driving vehicles cause an increase in car use and emissions’. The responses of the participants—as shown in Figure 3.7(c)—revealed that the largest percentage of the participants agreed, since the ‘Agree’ level recorded the highest frequency. ‘Agree’ plus ‘Strongly Agree’ levels yielded 55% while those of

'Disagree' plus 'Strongly Disagree' logged 27%. Therefore, the majority of the participants believed that self-driving vehicles cause car use and emissions to rise. Figure 3.7(d) shows the responses of the participants to the notion that self-driving vehicles will increase the number of miles people travel, thereby increasing pollution. From these statistics, it can be seen that 'Agree' and 'Strongly Agree' recorded a combined figure of 67% while 'Disagree' and 'Strongly Disagree' logged just 15%. Hence, the majority of the participants believed that self-driving vehicles will increase the number of miles people travel, thereby increasing pollution. This is consonant with the previous notion, since the majority of the participants believed that self-driving vehicles cause increases in car use and emissions.



**Figure 3.8** Environmental statistics: Statistical representation of the notion that self-driving vehicles will free up public spaces and promote clean air.

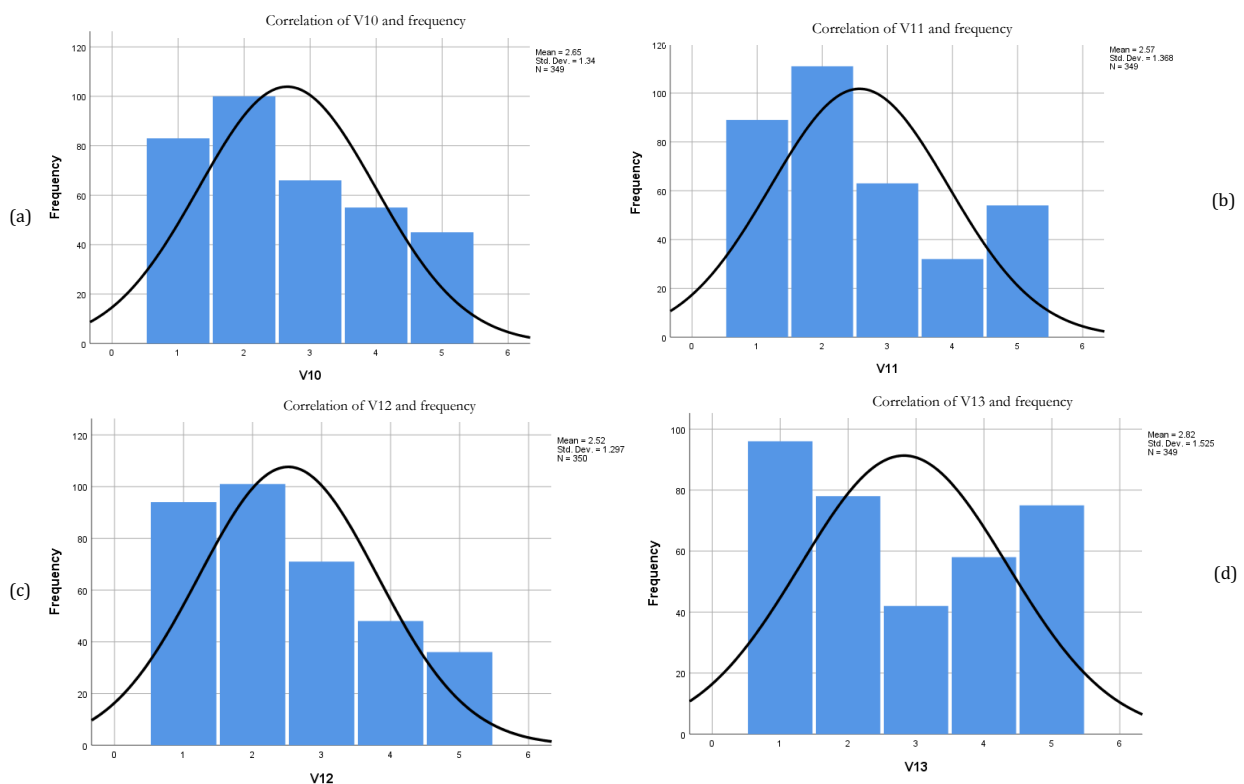
The statistics of the participants' responses to the notion that 'self-driving vehicles will free up public spaces and promote clean air' can be seen in Figure 3.8. Combining the 'Agree' and 'Strongly Agree' levels, we had a figure of 63%, while those of 'Disagree' and 'Strongly Disagree' logged 22%. Therefore, we can infer that the majority of the participants believed that self-driving vehicles will free up public spaces and promote clean air.

In comparison, in the environmental segment of the evaluation of autonomous vehicles, it can be inferred that autonomous vehicles are environmentally friendly, as the majority of the participants have positive perceptions. However, on looking at the results, we can say that most participants agreed that autonomous vehicles have a deleterious effect on the atmosphere, which may contribute to air pollution.

### C) Conjunction

Under the conjunction aspect, the following hypotheses were evaluated:

- V10: Self-driving vehicles offer more convenience and productivity.
- V11: Self-driving vehicles offer more personal freedom and independence.
- V12: Mobility is more affordable with self-driving vehicles through ridesharing.
- V13: Self-driving vehicles are expected to reduce the efforts of driving.



**Figure 3.9** Conjunction statistics:(a) Statistical representation of the idea that self-driving vehicles offer more convenience and productivity, (b) Statistical representation of the point that self-driving vehicles offer more personal freedom and independence, (c) Statistical representation of the idea that mobility is more affordable with self-driving vehicles through ridesharing, (d) Statistical representation of the assertion that self-driving vehicles will reduce driving efforts.



The statistics of the notion that self-driving vehicles offer more convenience and productivity are shown in Figure 3.9(a). From these, 'Agree' recorded the highest percentage followed by 'Strongly Agree'. Combining the levels of 'Agree' and 'Strongly Agree', we can see that 53% of the participants accounted for these, while those of 'Disagree' and 'Strongly Disagree' were the viewpoints of 29% of the participants. Hence, it is evident that the number of participants who agreed were more than the ones who disagreed; hence, the majority of the participants believe that self-driving vehicles offer more convenience and productivity.

Figure 3.9(b): the 'Agree' percentage plus 'Strongly Agree' percentage gave a figure of 57% while those of 'Disagree' and 'Strongly Disagree' yielded 24%. Therefore, it can be inferred that the majority of the participants believed that self-driving vehicles offer more personal freedom and independence. This is in consonance with the previous notion, which asserts that self-driving vehicles offer more convenience and productivity.

Now, Figure 3.9(c)—combining the percentages of 'Agree' and 'Strongly Agree'—came up with 53%, while the combination of 'Disagree' and 'Strongly Disagree' yielded 24%. It would be reasonable to conclude—based on the facts above—that most of the participants agreed that mobility is more affordable with self-driving vehicles through ridesharing.

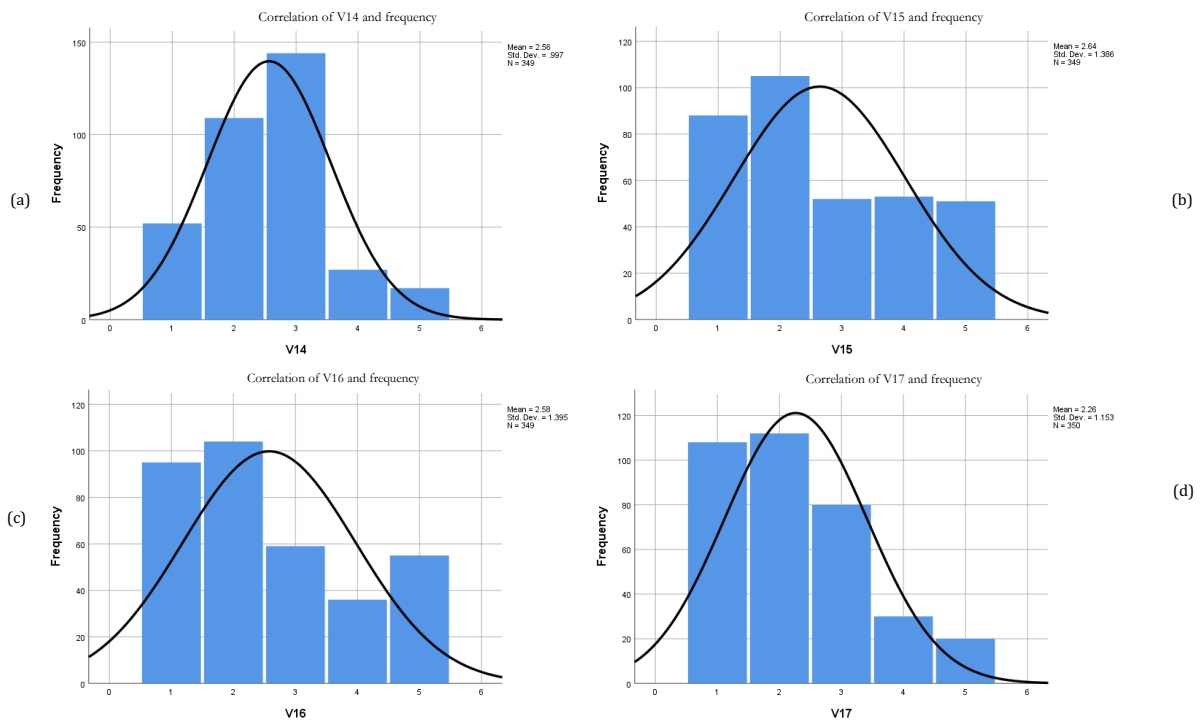
The responses of participants to the notion that 'self-driving vehicles are expected to reduce the efforts of driving' are recorded in Figure 3.9(d). Participants who 'Agree' and 'Strongly Agree' accounted for a percentage of 49% while those who 'Disagree' and 'Strongly Disagree' accounted for a percentage of 38% (which is smaller). It is noteworthy that the participants who believe that self-driving vehicles will reduce the efforts of driving exceeded the ones that believed otherwise. Nevertheless, 38% is a substantial percentage, implying that a good number of the participants also believed that self-driving vehicles will not reduce the efforts of driving.

Under the conjunction of the evaluation of autonomous vehicles, through a comparison, it can be deduced that there appear to be more advantages associated with autonomous vehicles than disadvantages: the majority of the participants responded positively to most of the notions considered in the survey.

#### ***D) Anxiety***

Anxiety can be caused by different life factors, and driving can be considered one of those factors. In some cases, it requires help from a psychologist or other mental health specialist. In this section, we examined anxiety based on the following hypotheses:

- V14: Before using AVs, I doubted if I would be able to control the vehicle if and when an ethically complicated situation arises.
- V15: After using AVs, I became confident that all my journeys with AVs would be successful.
- V16: Before using AVs, I was worried that interacting with the vehicle would require much mental effort.
- V17: After using AVs, I became confident that an emergency would hardly arise because of malfunctioning.
- V18: After using AVs, I became confident that all my journeys with AVs would be successful.
- V19: Interacting with the AVs did not require a lot of mental effort.
- V20: I would trust such a vehicle.
- V21: I am afraid I would not be able to react in case an emergency occurs.
- V22: I was afraid an emergency would arise as the vehicle might malfunction.



**Figure 3.10** Anxiety statistics: (a) Statistical representation of the statement that ‘before using an AV, I doubted if I would be able to control the vehicle if an ethically complicated situation arises’, (b) Statistical representation of the statement that ‘after using an AV, I became confident that all my journeys with AVs would be successful’, (c) Statistical representation of the statement that ‘before using an AV, I was worried that interacting with the vehicle would require much mental effort’, (d) Statistical representation of the statement that ‘after using an AV, I became confident that all my journeys with AVs would be successful’.

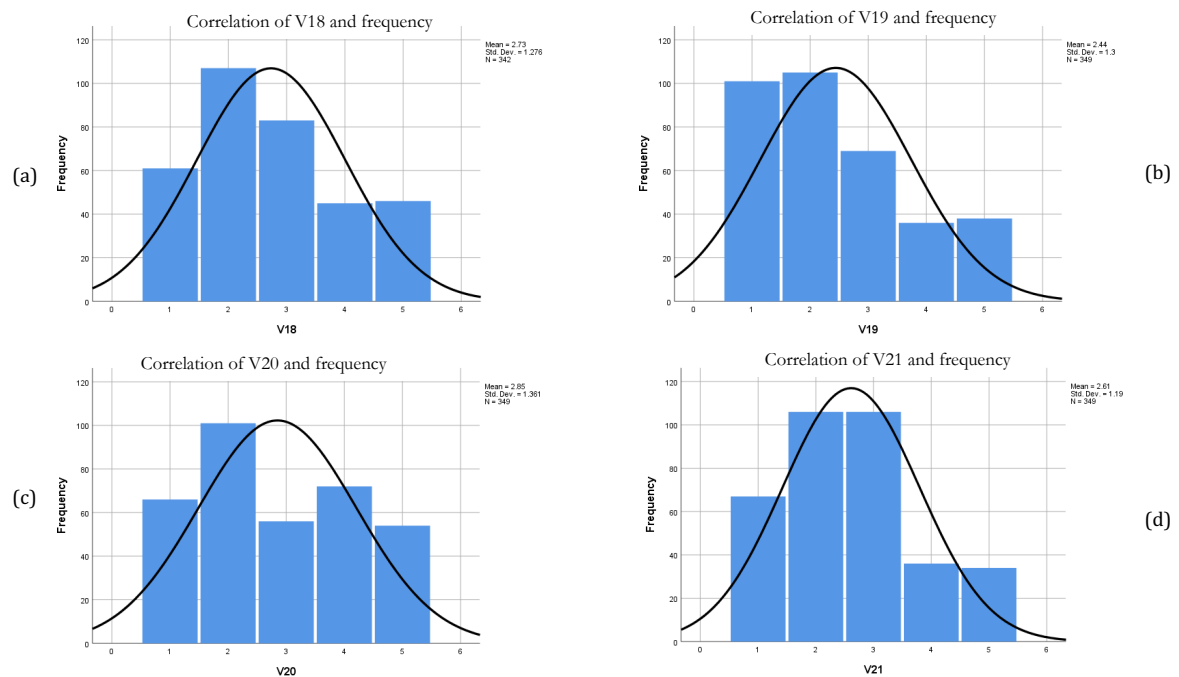
In the interviews with different individuals, conducted to understand their concerns regarding autonomous vehicles, anxiety was the concern that most of them raised. On analysing and evaluating the different hypothesis, we find:

‘Before using AVs, I doubted I may not be able to control the vehicle if an ethically complicated situation arises’ logged the responses listed in Figure 3.10(a). It can be seen that the majority of the participants were neutral to the notion and combining the percentage of ‘Agree’ and ‘Strongly Agree’—which gave a figure of 46%—it implied that a very good number of the participants also agreed that before using AVs, they doubted they would not be able to control the vehicle if an ethically complicated situation arose.

About 55% of the participants claimed that after using AVs, they became confident that all their journeys with AVs would be successful, while some 30% claimed that after using AVs, they were not confident that all their journeys with AVs would be successful (Figure 3.10(b)).

Moreover, some 57% of the participants claimed that before using AVs, they had been worried that interacting with the vehicles would require much mental effort, but about 26% disagreed with the notion that before they started using AVs, they were worried that interacting with the vehicles would require much mental effort as shown in Figure 3.10(c).

On evaluating the responses obtained from Hypothesis V17 (Figure 3.10(d)), when combining the levels of 'Agree' and 'Strongly Agree' (which gave us 63%) and the levels of 'Disagree' and 'Strongly Disagree' (which gave us 15%), it can be inferred that the majority of the participants agreed that after using AVs, they became confident that an emergency would hardly ever arise on account of the vehicle malfunctioning.



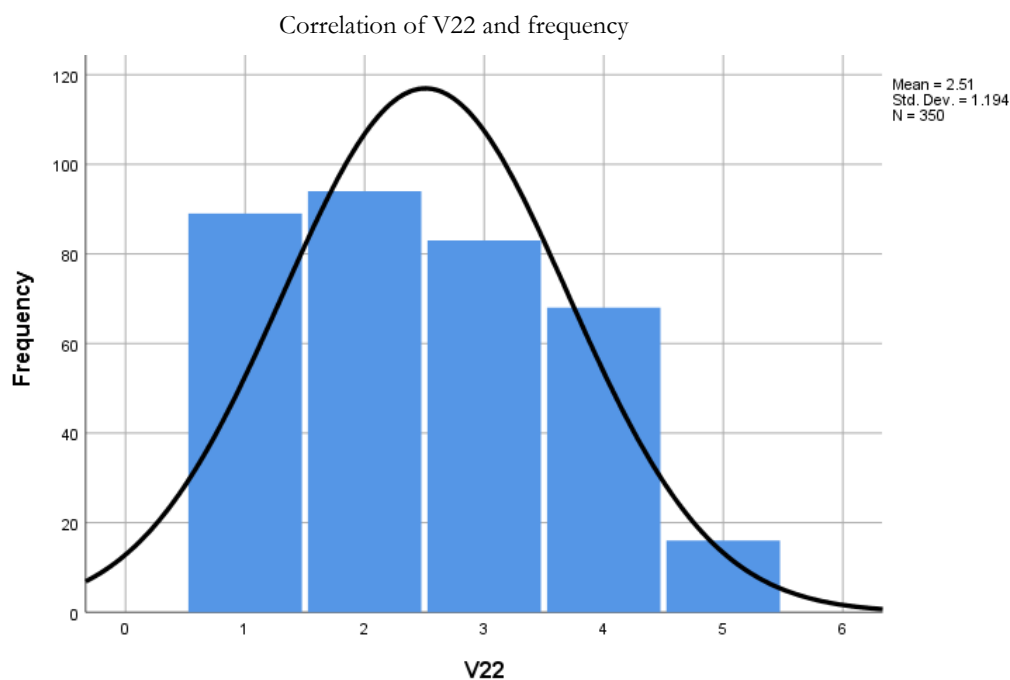
**Figure 3.11** Anxiety statistics: (a) Statistical representation of the statement that 'After using AV, I became confident that all of my journeys with AVs will be successful', (b) Statistical representation of the statement that 'Interacting with the AV did not require a lot of mental effort', (c) Statistical representation of the statement that 'I would trust the vehicle', (d) Statistical representation of the statement that 'I am afraid that I won't be able to react in case an emergency occurs'.

The statistical representation of the notion that after using AVs, the participants were confident that all their journeys with AVs would be successful, shows that about 26% of the participants disagreed with the notion that after using AVs, they gained confidence that all their journeys with AVs would be successful. Some 48% asserted

that after using AVs, they became confident that all their journeys with AVs would be successful (as shown in Figure 3.11(a)).

Considering the statistics obtained from evaluating the notion ‘interacting with the AV did not require a lot of mental effort’, it can be observed that ‘Agree’ logged the highest percentage (30%), followed by ‘Strongly Agree’ (with a percentage of 29%). ‘Agree’ plus ‘Strongly Agree’ gave us a figure of 59% while ‘Disagree’ plus ‘Strongly Disagree’ yielded 21%. Therefore, most of the participants believed that interacting with the AVs require a lot of mental effort. The statistical evaluation can be found in Figure 3.11(b).

The participants’ responses to the notion ‘I would trust the vehicle’ show that when a combination of ‘Agree’ and ‘Strongly Agree’ yielded 48%, those of ‘Disagree’ and ‘Strongly Disagree’ gave 36%. There were more participants who would, therefore, trust the autonomous vehicle that the ones who would not as shown in Figure 3.11(c). Again, 30% of the participants were neutral to the notion that ‘I am afraid that I would not be able to react in case an emergency occurs’. After combining the percentages of ‘Agree’ and ‘Strongly Agree’, 49% claimed that they were afraid that they would not be able to react in case an emergency occurs. Some 20% of the participants disagreed with the idea that they are afraid that they would not be able to react in case an emergency occurs as presented in Figure 3.11(d).



**Figure 3.12** Statistical representation of the notion ‘Before using AV, I was afraid an emergency would arise because the vehicle would malfunction’.

The idea that ‘Before using AVs, I was afraid an emergency would arise because the vehicle would malfunction’ recorded responses as shown in Figure 3.12. The largest percentage of the participants agreed to the notion (with a percentage of 27%). The combination of ‘Agree’ and ‘Strongly Agree’ logged a percentage of 52% while that of ‘Disagree’ and ‘Strongly Disagree’ led to a percentage of 24%. Therefore, the responses of the participants inclined to the claim that before using AVs, they were afraid an emergency would arise on account of the malfunctioning of the vehicle.

### 3.5.2 Applying TAM TAM to autonomous vehicles

Scrutinizing the questions in the survey under the safety and environment variables of autonomous vehicles, we can deduce that the questions that were posed examined the use of autonomous vehicles, while the categories ‘conjunction’ and ‘anxiety’ examined the ease of use of autonomous vehicles. The responses of the participants will be used to determine their perceptions for each of the determinants. See the figure below for details.

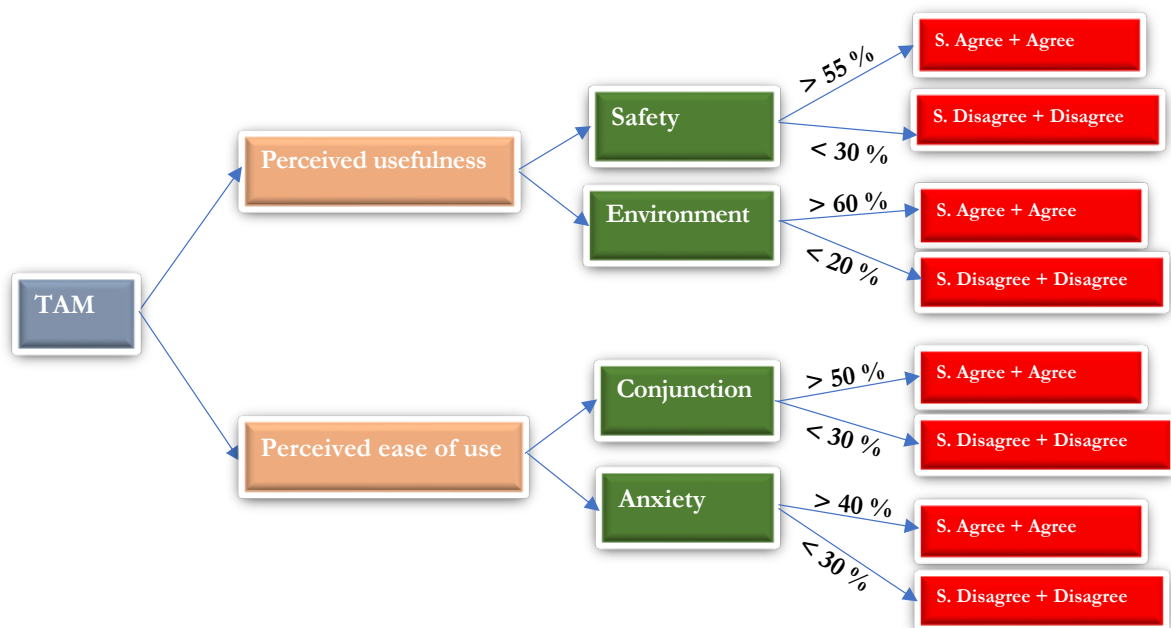


Figure 3.13 Results of TAM, determinants, variables, and perceptions.

From the analysis, as far as the evaluation of the autonomous vehicles with respect to their safety aspect is concerned, the majority of the participants in the survey believed that self-driving vehicles cause fewer accidents and decrease traffic congestion. More

than half the participants believed that unlike ordinary vehicles, which may be operated by drunk or distracted drivers, self-driving vehicles will reduce risky driving behaviours. Then again, a larger number of the participants disagreed with the notion that self-driving vehicles outperform humans in detecting dangerous situations. From these statistics (see Figure 3.13), it can be deduced that the participants in the survey (more than 55%) had a positive perception of autonomous vehicles, i.e., they perceived autonomous vehicles to be useful to them. Less than 30% had a negative perception, i.e., they were of the view that autonomous vehicles were not useful. As regards the environmental usefulness of autonomous vehicles, more than 60% of the participants had a positive perception that autonomous vehicles would be environmentally useful.

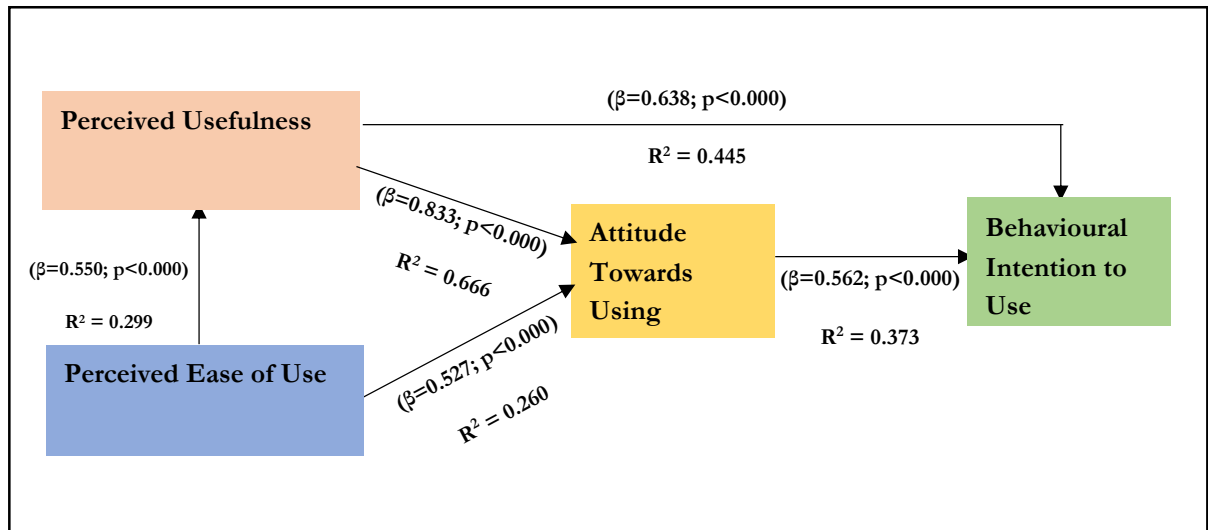
Under the perceived ease of use in the technology acceptance model (TAM), it has been observed that the largest percentage of the participants believed that self-driving vehicles offer more convenience and productivity. The majority of the participants believed that self-driving vehicles offer more personal freedom and independence. But most of the participants neither agreed nor disagreed with the claim that mobility is more affordable with self-driving vehicles through ridesharing. Again, very few of the participants disagreed with the claim that self-driving vehicles will reduce driving efforts.

From Figure 3.13, it can be seen that on an aggregate, more than 50% of the participants had the positive perception that autonomous vehicles would be easy to use, i.e., the level of skills required to operate autonomous vehicles is minimal. Less than 30% of the participants had a negative perception, as they perceived the ease-of-use attribute of autonomous vehicles to be a complex system.

Also, while considering the view of the participants as regards the anxiety they may harbour in respect of autonomous vehicles, it was observed that at least around 40% showed that they had some iota of fear as far as autonomous vehicles are concerned. For instance, about 45% (when the percentages of 'Agree' and 'Strongly Agree' are added together) of the participants said that before using AVs, they doubted if they would be able to control the vehicle if an ethically complicated situation arises. But less than 30% of the participants stated that they had no fear about using autonomous vehicles, i.e., they perceived the ease of use of autonomous vehicles to be high.

Therefore, by applying TAM, it has been observed that both perceived usefulness and perceived ease of use of autonomous vehicles have generated positive perceptions among the participants. This implies that autonomous vehicles have found acceptance among the majority of the participants.

### A) The before analysis



**Figure 3.14** Regression analysis results for before using AVs.

The results obtained from the regression can be seen clearly in Figure 3.14. The intention of creating Figure 3.14 stems from the interest to study the relationship between some factors in the TAM model, as stated by Masron (2007). Masron (2007) studied the relationship between Perceived Usefulness, Perceived Ease of Use, Attitude Towards Use, and Behavioural Intention to Use, and found that perceived usefulness had a significant effect on perceived ease of use. However, while perceived usefulness had a significant effect on the intention to use, attitudes towards using it did not. In this research, Figure 3.14 was obtained by selectively identifying questions closely related to the factors. Regression analysis was carried out on the questions: one was used to predict the other, and the results are shown in Figure 3.14. All the factors are significantly related to each other (as shown by p-value). If  $R^2 = 0.3$  is taken as the threshold value to justify the correlation between the factors, then it can be deduced that the effect on one factor can be used to predict the other. Remarkably, it has been observed in this work that perceived usefulness and attitude towards use have the strongest correlation; this can be linked to the fact that many of the



participants with adequate knowledge of the benefits of AVs were willing to acquire such vehicles.

### B) The after-analysis

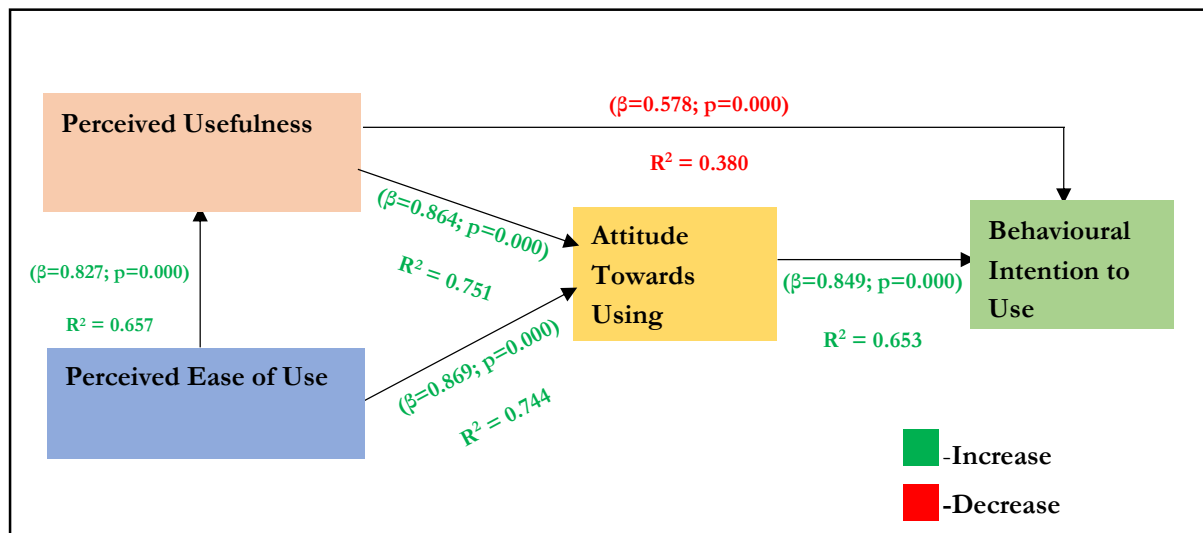


Figure 3.15 Regression analysis results for after using AVs.

Figure 3.15 shows the regression analysis of the responses of the participants to the questions indicating their views after using AVs. Starting with the relationship between the ‘Perceived Ease of Use’ and ‘Perceived Usefulness’,  $\beta = 0.827$  and  $R^2 = 0.657$ , whereas the values for these parameters before using AVs are:  $\beta = 0.550$  and  $R^2 = 0.299$ . From these statistics, it can be inferred that there has been an improvement in the perceptions of the participants after using AVs; in other words, the mindset of the participants related to ‘Perceived Usefulness’ towards AVs improved probably because of the increase in their ‘Perceived Ease of Use’. For instance, a participant believed that the operation of AVs would require a high level of technical knowledge before using these, but after using AVs, he/she discovered that his/her perception was wrong. Such a participant, therefore, would have an improved ‘Perceived Ease of Use’, and, hence, ‘Perceived Usefulness’. On the other hand, it was observed that a decrease in the relationship between ‘Perceived Usefulness’ and ‘Behavioural Intention to Use’. This can be linked to the fact that before using AVs, there might have been a high level of curiosity to experience the use of AVs and after having such an experience, the behavioural intention to use declined. In general, an increase in the positive perceptions of the participants has been observed from the

regression analysis, as the values of  $\beta$  and  $R^2$  increased significantly. This suggests that there is a high probability that members of the public will eventually admire and cherish AVs when they acquire one each. Having their own AVs would satisfy their curiosity and dispel the perceptions of uncertainties initially associated with AVs.

### **3.6 DISCUSSION AND CONCLUSION**

This chapter analysed the responses of participants regarding their perceptions of AVs using the TAM model to evaluate the acceptance of AVs. The responses were grouped into four independent variables: safety, environmental impact, conjunction, and anxiety.

As regards safety, the report analysed four different statements related to the safety of autonomous vehicles. The participants then had to rate their levels of agreement or disagreement with each of them. The four statements were:

- Self-driving vehicles generate fewer accidents.
- Self-driving vehicles decrease traffic congestion.
- Unlike ordinary vehicles, which may be operated by drunk or distracted drivers, self-driving vehicles are expected to reduce risky driving behaviours.
- Self-driving vehicles outperform humans in detecting dangerous situations.

The analysis showed that a significant majority of the participants believed that self-driving vehicles generate fewer accidents and decrease traffic congestion. Additionally, the majority of participants believed that self-driving vehicles will reduce risky driving behaviours, and about half of the participants believed that self-driving vehicles outperform humans in detecting dangerous situations.

These results are in line with the general perception of AVs as being safer than human-driven vehicles. Studies have shown that most accidents are caused by human errors, such as distraction, speeding, and driving under the influence of drugs or alcohol. AVs, on the other hand, do not suffer from these limitations; hence, they can potentially reduce accidents significantly.

Based on these findings, self-driving vehicles offer more personal freedom and independence: self-driving vehicles can provide more personal freedom and independence for people who are unable to drive on account of physical disabilities, age, or other reasons. They can also help reduce the need for car ownership and the associated expenses, giving people more freedom to choose how they travel.

The results showed that self-driving vehicles will reduce the efforts of driving: self-driving vehicles can reduce the physical and mental efforts required for driving, which can reduce stress and fatigue for drivers. Yet, the TAM results showed the willingness of people to adapt to AVs, where the after-study showed that the attitude towards using AVs improved after trying the technology.

The results of this survey can be useful for policymakers, researchers, and manufacturers of autonomous vehicles. Policymakers can use these results to better understand the perceptions of the public and tailor their policies and regulations accordingly. Researchers can use these results to design better autonomous vehicles that address the concerns of the public. Finally, manufacturers can use these results to improve their marketing strategies and educate the public about the safety benefits of autonomous vehicles.

Overall, the results of this survey provide valuable insights into the public perceptions of the safety aspect of AVs. As AVs continue to gain popularity and become more prevalent on our roads, understanding and addressing the concerns of the public about their safety will be critical to their success.

In the next chapter, we will delve into the training and testing phases of the five chosen algorithms—Faster R-CNN, Cascade R-CNN, RetinaNet, FCOS, and Deformable DETR of the datasets. Where the intricacies of their performance will be presented and provide insightful interpretations of the results.

---

# Chapter 4

---

# 4 Comparative Evaluation of Algorithms

Motivated by the expected benefits of automated driving in terms of increased safety, comfort, and traffic efficiency, research efforts are focused on object recognition for the automation of the driving task. As explained in the introduction and literature review, the development of automated driving requires the involvement of computer vision. Therefore, a comprehensive evaluation of the different detection methods is performed on the different datasets discussed in Chapter 2. This chapter first presents the experimental setup (Section 4.1). The next section (Section 4.2) is the data organisation, where the data flow, pre-processing, data comparison and training of the different object detection algorithms are discussed. Section 4.3 is the evaluation of the training on each individual dataset using the different algorithms. A cross analysis is then discussed (Section 4.4). The knowledge gained allows Section 4.5 to conclude with a discussion of the performance of detection on the datasets.

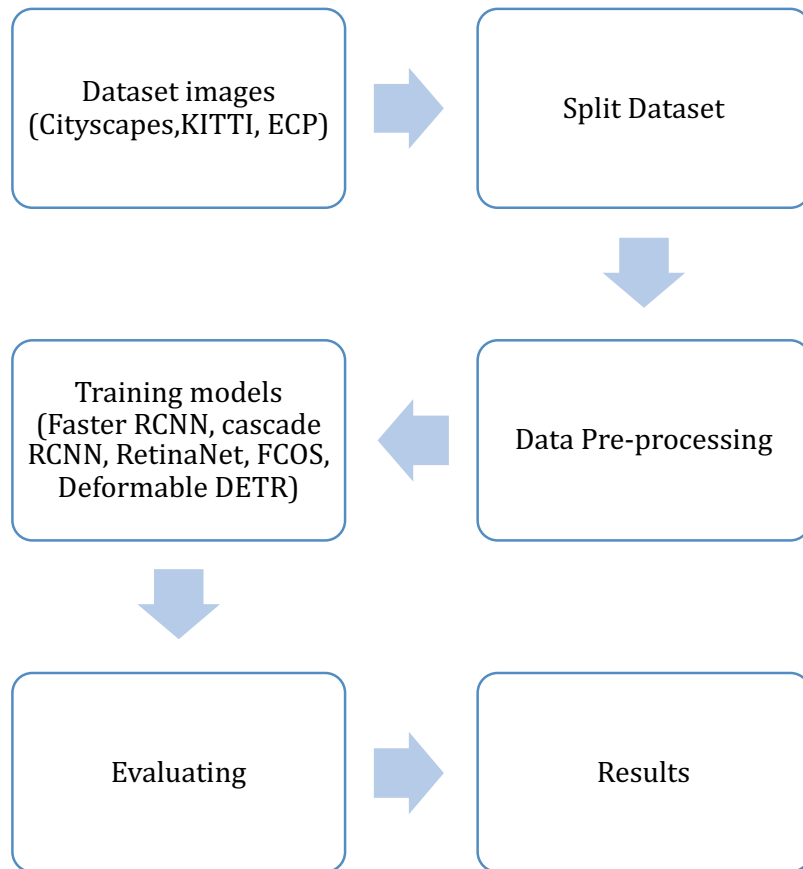
## 4.1 DESIGN OF EXPERIMENT

For this work, MMDetection is the platform from which all experiments were performed (<https://github.com/open-mmlab>). MMDetection is a Python toolbox built as a codebase exclusively for object detection and instance segmentation tasks. Developed in a modular way with PyTorch implementation, it is part of OpenMMLab, an open-source project for academic and industrial research and implementation. This lab covers several computer vision research topics such as classification, detection, segmentation, and super-resolution. The tool allows fast training and high-quality inference.

MMDetection acts as a benchmark with the flexibility to re-implement the existing methods or to develop a new detector with the available modules. As a result, the detection framework can be broken down into different components making it easy to customise and personalise an object detection framework just by combining different modules.

As a first step of this work, an environment was initiated on anaconda. Once the environment was activated and all prerequisites were downloaded, datasets were added, and configuration files were modified (Appendix C contains more details on

structuring the dataset and modifying the configuration). A flowchart of the experiment is shown in Figure 4.1.



**Figure 4.1** Experiment framework

#### 4.1.1 Hardware Requirements

The hardware used to train deep learning models refers to the physical computing resources used to perform the computations required to train a neural network. The choice of hardware depends on the size and complexity of the neural network being trained, as well as the amount of data used for training. GPUs are often used for deep learning because they can perform many calculations in parallel, which can significantly speed up training. A 2080Ti GPU was used for the experiments in this chapter, but a 3090 GPU was used for Chapter 5 because the GPU capacity was exhausted by the experiments in that chapter.

### **4.1.2 Software Requirements**

Before we actually start the model building phase, we need to make sure that the right libraries and frameworks are installed. MMDetection works on different operating platforms (OP) Linux, Microsoft Windows, or MacOS. In this thesis, Linux is used as the OP. Python 3.7+, CUDA 9.2+ and PyTorch 1.6+ are required libraries:

- matplotlib
- numpy
- opencv-python
- sklearn

Most of the above-mentioned libraries will already be present on machine when installing Anaconda. However, it is necessary to ensure that they are available.

## **4.2 DATASET ORGANISATION AND TRAINING**

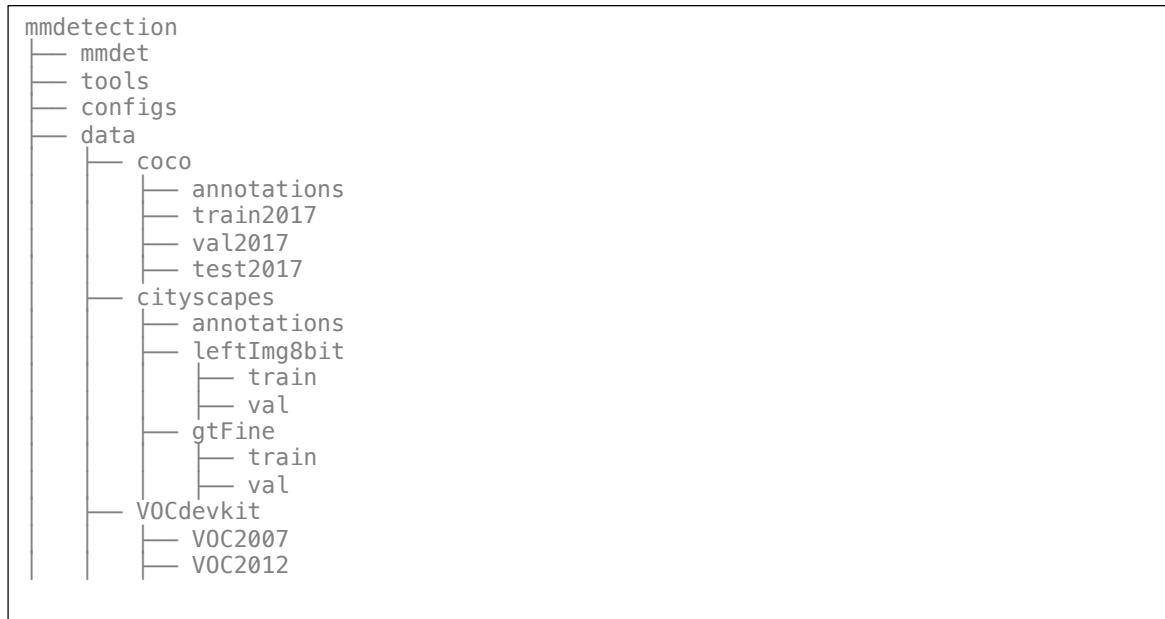
### **4.2.1 Experiment Summary**

The experiment is conducted using the four selected datasets, of which, approximately 3,000 1333\*800 RGB road images are randomly selected from each dataset. Since the resolution was too large, direct training could cause GPU memory overflow, so a random cropping method was used to convert the image sizes to 1333\*800 pixels. A total of 50,900 images were used for training. The images in ECP and Cityscapes contain photographs from different seasons and include clouds, rain, and different weather conditions. However, this is not the case for the KITTI dataset, where the data was collected in clear, clean weather conditions without fog or other climatic factors affecting the clarity of the dataset.

### **4.2.2 Data Flow**

For this experiment, data was gathered from different urban road datasets available as shown in figure 4.1. To start applying MMDetection to the datasets, the

structure of the data must be consistent, in which the images and the labels should follow a certain format as in Figure 4.2:



**Figure 4.2** Dataset structure

If the data needs to be organized according to Figure 4.2, but still use the modular design of mmdetection, once the dataset has been prepared, it is time to customise the configuration, pipeline, models, and runtime setting losses, and fine-tune the models. Configuration file is the file that specifies the parameters, options, settings, and preferences applied to operating systems, infrastructure devices (where infrastructure is the framework that supports a system) and applications. Pipeline is the end-to-end construct that orchestrates the flow of data into and out of a machine learning model (or set of models). Runtime describes the software/instructions that are executed while your program is running. Fine-tuning (Gunawan, Lau, and Lindawati 2011) in computer science is the process of taking a model that has already been trained for a given task and making it perform a second similar task using transfer learning.

For the first experiment, the split provided by each dataset was used for training and testing (i.e. the Cityscape dataset uses 60-10-30 while KITTI uses 50-0-50). For the second experiment (Chapter 5), the data followed the 60-10-30 split. Data pre-processing, which includes data resizing and cropping. Finally, each object detection algorithm is trained and compared.



### 4.2.3 Data Pre-processing

Data pre-processing refers to the cleaning, transformation, and preparation of data for analysis/modeling. This includes removing missing or invalid data, normalising, scaling data to ensure consistency, and converting data types to ensure compatibility. In addition, data pre-processing involves splitting the data into training and test sets to ensure that models are validated on independent data. Yet data is pre-processed by rescaling each image into an image size of 1333\*800 pixels while maintaining the aspect ratio.

### 4.2.4 Parametrization of Object Detection Algorithms and Training Details

For object detection, models need to be trained using a set of input images and their associated ground truth boxes for each of their classes. In our approach, all the models used are pre-trained on the COCO dataset before moving on to the phase of training on the used dataset. To compare the different object detection algorithms using real-time images, the main aspects that we compare each of the models consist of their precision, recall, and mAP values are the aspect used to compare the performance of the model on the different datasets. However, confusion matrix with precision and recall was carried for the ECP dataset, so it is the most challenging dataset and used as the pre-trained data for benchmark dataset in Chapter 5. In this experiment model uses PyTorch framework, model were trained for 12 epochs Stochastic Gradient Descent (SGD) optimiser.

For training, the following table (Table 4.1) shows the hyperparameter settings used for each algorithm.

**Table 4.1** Hyperparameter Settings

Hyperparameter	
Batch Size	16
Optimizer Momentum	0.9
Weight Decay	0.001
Learning Rate Scheduler	Step /AdamW
Base learning rate	0.01
Weight loss	1
Gamma loss	2
Alpha loss	0.25

#### 4.2.5 Comparison of Performance Across Datasets

The modules were trained and tested on different datasets due to the uniqueness of each dataset and challenges, where each dataset uses its labelling, resolution and format. Table 4.2 shows the uniqueness of each selected dataset. The COCO dataset is not included in the comparison as it is a multipurpose object detection dataset. For example, the ECP dataset covers 12 European countries in 4 different seasons, whereas the KITTI dataset was collected in one country in one season. This has an impact on the performance of the modules. In addition, the Cityscapes dataset was collected over three seasons. However, the weather was dry compared to the ECP dataset where the weather was both dry and wet in some cases.

In this section, we will compare the performance obtained from training and testing the different networks on the desired datasets. We will first evaluate the performance on individual datasets and then perform a cross evaluation. However, it should be noted that the ECP dataset has received extensive attention as it is the data that will be used as the pre-trained model for the desired benchmark dataset.

**Table 4.2** Comparison of datasets

	<b>Cityscapes</b>	<b>KITTI</b>	<b>EuroCity Persons</b>
<b>Countries</b>	3	1	12
<b>Cities</b>	27	1	31
<b>Seasons</b>	3	1	4
<b>Images</b>	25000	14999	47335
<b>Pedestrians (day / night)</b>	31514/ -	9400/ -	183004/ 35309
<b>Riders (day / night)</b>	3502/ -	3300/ -	18216/ 1564
<b>Ignored regions (day / night)</b>	13172	22600	75673 / 20032
<b>Resolution</b>	2048 x 1024	1240 x 376	1920 x 1024
<b>Weather</b>	Dry	Dry	Dry / Wet
<b>Train - val - test split (%)</b>	60 - 10 - 30	50 - 0 - 50	60 - 10 - 30

Since the resolution was too large for Cityscapes and EuroCity Person, direct training could cause GPU memory overflow, so a random cropping method was used to convert the size of the image to 1333\*800-pixels.

#### 4.2.6 Training of ODAs

All datasets were well labelled, where labels were used in deep learning algorithms to learn and evaluate the performance of the network. The experimental training neural network was implemented using synchronised stochastic gradient descent (SGD) over single using the PyTorch framework in Ubuntu 20.04, 3.6GHz Core i7 10,700K CPU, DDR4 3,200Mhz 2X16-32GB quad-channel memory, and RTX2080Ti dual-card GPU, with a total of 2 images per minibatch. The initial learning rate was set to 0.0025, weight decay to 0.0001, and momentum to 0.9. The datasets were split as follows:

- COCO dataset: 70% for training, 20% for validation, and 10% for testing.
- Cityscapes dataset: 70% for training, 20% for validation, and 10% for testing.
- KITTI dataset: 50% for training and 50% for validation and testing
- EuroCity Person: 70% for training, 20% for validation, and 10% for testing.

Moreover, for training these datasets, modification on mmdet folder is a must.

'class\_name' for example need to be modified so that it matches dataset classes. For

further information regarding modification carried for this work refer to (Appendix B).

### **A) Training and evaluation of RCNN**

For this thesis, Faster RCNN and Cascade RCNN networks are used. Both networks are two-stage object detection as mentioned in the previous chapter. Faster RCNN has been further developed by Cai and Vasconcelos (2017) to Cascade RCNN. In order to use these networks on the selected datasets, modifications have to be made to the configuration file. After the modifications mentioned in 4.2.2, the configuration file needs to be modified.

For Faster RCNN, the number of classes in `bbox_head` need to be modified. Using KITTI as an example, the following changes are made:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 28 2021

@author: afnan
"""

_base_ = [
    '../_base_/models/faster_rcnn_r50_fpn.py', '../_base_/datasets/kitti_detection.py',
    '../_base_/schedules/schedule_1x.py', '../_base_/default_runtime.py'
]
model = dict(
    bbox_head=dict(
        num_classes=3, #change num_class to 3 to match detection classes
    ))
```

For Cascade RCNN, we are training using object detection via bounding boxes, and not segmentation. Therefore, modifying the `bbox` head is necessary refer to Appandix C.

### **B) Training and evaluation of RetinaNet**

RetinaNet, as mentioned in Chapter 2, is a one-stage object detection model, where the network consists of a backbone network and two task-specific subnetworks (Lin et al. (2017)). In order for the network to be used on the different datasets, it is necessary to make several changes to the original configuration file. First is to create a customised dataset file that includes the dataset type, the data root (the path where the data is stored), the image normalisation configuration, the pipeline where it contains the training and testing pipeline, and finally the evaluation matrix.

The next step is to add the created dataset to “\_init\_.py” to initiate the dataset. The last step is to edit the configuration file. For RetinaNet, the boundary box head (bbox\_head) needs to be modified, changing the number of classes to match the classes identified for the dataset (refer to Appendix C).

### ***C) Training and evaluation of FCOS***

The fully convolutional one-stage object detector (FCOS) is a one-stage object detection model. As mentioned in Chapter 2, it is an is-anchor box free, as well as proposal free model. The network consists of a ResNet 50 backbone, a FPN backbone, and an FCOS head. To use it on a customised dataset, the number of classes (num\_classes) must be changed (refer to Appendix C).

### ***D) Training and evaluation of DETR***

As mentioned above, DETR is one of the most widely used object detection framework. It is the first object detection framework to successfully integrate transformers as a central building block in the detection pipeline, and the network achieved high average performance percentages when tested on the COCO dataset. MMDetection has integrated this network into their tool, however, when training on a custom dataset, the network failed to run correctly where AP=0. This may be because the network is new and more testing is needed to overcome network bugs. Deformable DETR (Carion et al., 2020) was used for training and evaluation. Deformable DETR was proposed to overcome the drawbacks of DETR (low convergence and limited spatial resolution of features), where it showed its ability to achieve better performance than DETR with 10 times fewer training epochs. However, it is important to modify the num\_classes in bbox\_head to suit the customised dataset (refer to Appendix C). For this network, images were resized to avoid overflow of data and due to limited GPU space.

The following subsection will delve into a comprehensive examination of the training and testing processes employed for the five object detection algorithms. Table 4.3 presents the algorithms and datasets used for this chapter analysis.

**Table 4.3** Datasets and algorithms used for training and testing in this chapter.

Process	Dataset	Algorithms	Remark
Pre training	COCO	<ul style="list-style-type: none"> <li>Faster RCNN</li> <li>Cascade RCNN</li> <li>RetinaNet</li> <li>FCOS</li> <li>Deformable DETR</li> </ul>	<ul style="list-style-type: none"> <li>Each algorithm was pre trained used COCO</li> </ul>
Transfer learning \ Finetune	ImageNet	<ul style="list-style-type: none"> <li>Faster RCNN</li> <li>Cascade RCNN</li> <li>RetinaNet</li> <li>FCOS</li> <li>Deformable DETR</li> </ul>	<ul style="list-style-type: none"> <li>Used to initialize and speed the learning proses.</li> <li>Each algorithm was finetuned using ImageNet</li> </ul>
Training and Testing Detection Algorithms	Cityscapes ECP Kitti	<ul style="list-style-type: none"> <li>Faster RCNN</li> <li>Cascade RCNN</li> <li>RetinaNet</li> <li>FCOS</li> <li>Deformable DETR</li> </ul>	<ul style="list-style-type: none"> <li>Trained and tested on each algorithm (i.e. trained and tested on Cityscapes dataset)</li> </ul>

### 4.3 TRAINING AND EVALUATION ON INDIVIDUAL DATASETS

#### 4.3.1 Cityscapes Dataset

The Cityscapes dataset was trained on four different architectures, one-stage detection (RetinaNet), two-stage detection (Faster RCNN), cascade detection (Cascade RCNN), and transformer detection (Deformed DETR). The mean average precision (mAP) results are shown in Table 4.4.

**Table 4.4** Results of different ODAs on Cityscapes

Method	AP	AP <sub>50</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>	Training time
Cascade RCNN	.40	.657	.194	.401	.610	21h22min
Faster RCNN	.392	.667	.189	.393	.565	17h11min
FCOS	~.001	~.001	~.001	.002	.011	16 days 17h
RetinaNet	.358	.582	.143	.353	.549	19h29min
Cascade RCNN with ResNext backbone	.42	.667	.196	.426	.617	21h22min
ResNeXt	.367	.606	.155	.358	.559	19h1min

AP detection results ranged from 35.8 with single-step RetinaNet object detection to 42% with Cascade RCNN. The average precision at IoU = 50% (AP50) ranges from 58.2% to 66.7%. Comparing these results with the literature, Table 4.4 shows that the results obtained are in the upper range of cityscape detection results. In this application, small stands for objects with a pixel area of <322 pixels. Medium stands for objects with an area range of 322<area<962 pixels. Large represents objects with

an area of 962<area pixels. AP was evaluated using IoU (Intersection over Union of boundary boxes) = 0.5 : 0.05 : 0.95 with AP: MaxDets = 100 (given 100 detections 100 image). For the road senses, we focus on large and medium objects, as these are the most important objects for predicting the movement of the environment. Looking at the training results, the models achieved 60% accuracy in predicting objects. However, for safer traffic systems, detection should be more accurate, a modification was applied using a different backbone one stage detection as it is the cheapest solution. ResNeXt (S. Xie, Girshick, Dollár, Tu, & He, 2017) detection is used as the detection model and the result is shown in Table 4.4. ResNeXt is a homogeneous neural network that reduces the number of hyperparameters required by the conventional ResNet (He, Zhang, Ren, & Sun, 2016). It was also used as a backbone for Cascade RCNN and the results are given in Table 4.4.

When comparing the results obtained from the tests using the MMDetection tool, with the results on the same dataset summarised in Table 4.5, the tool achieved better results than most papers.

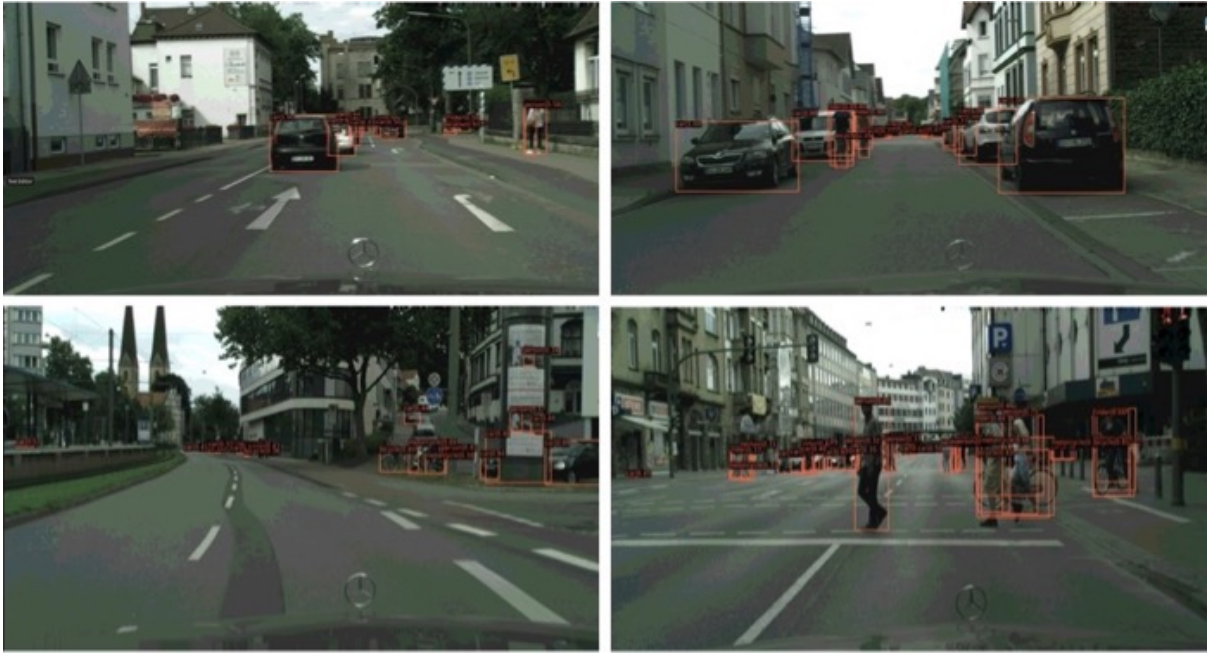
**Table 4.5** Results of detection on cityscape provided in literature.

Reference	AP(%)	AP <sub>50</sub> (%)
(Cordts et al., 2016)	4.6	12.9
(Watanabe & Wolf, 2019)	7.7	14.9
(Santarossa et al., 2021)	8.5	19.3
(Uhrig, Cordts, Franke, & Brox, 2016)	8.9	21.1
(Mazzini & Schettini, 2019)	9.2	16.8
(Ortelt, Herrmann, Willersinn, & Beyerer, 2018)	12.5	25.2
(X. Xu, Chiu, Huang, & Shi, 2020)	27.5	48
(He, Gkioxari, Dollár, & Girshick, 2017)	26.2	49.9

(Shu Liu, Jia, & Fidler, 2017)	25	44.9
(Porzi, Rotabui, Colovic, Kontschieder, & Research, 2019)	22.1	39.4
(Kendall, Gal, & Cipolla, 2018)	21.6	39
(De Brabandere & Neven, 2017)	17.5	35.9
(Hayder & He, 2017)	17.4	36.7
(Shu Liu et al., 2018)	36.4	63.1
(Cheng et al., 2019)	34.6	57.3
(Huiyu Wang et al., 2020)	34.0	55.9
(Homayounfar, Xiong, Liang, Ma, & Urtasun, 2020)	33.3	58.2
(Peng et al., 2020)	31.7	58.4
(L.-C. Chen, Wang, & Qiao, 2020)	43.4	68.7
(L.-C. Chen, Lopes, et al., 2020)	42.6	67.6

For illustrative purposes, a specific set of scenes showcasing the detection of all object classes was chosen. The algorithm successfully detected different traffic object classes in the same scene when trained and tested on the cityscape dataset, as shown in the images. Figure 4.3 shows a sample of training and testing images from the cityscape dataset.





**Figure 4.3** Cityscapes ODA where the images are obtained from testing using faster RCNN, RetinaNet and cascade RCNN.

#### 4.3.2 KITTI Dataset

The KITTI dataset was trained and tested using different detection algorithms and the results are then recorded and presented in Table 4.6. The KITTI dataset overall achieved very high mean average precision (mAP) values. As mentioned earlier, this dataset was collected from a city with high resolution and dry weather. This fact affected the results when compared to the other datasets. The two-stage object detection using resNeXt backbone achieved the highest mAP value of 88.94%, whereas the one-stage RetinaNet achieved 85.47 % mAP with a lower computational cost.

In addition, the results using a transformer algorithm appear to be the lowest, as the dataset is divided by two, leaving the GPU with insufficient memory.

**Table 4.6** KITTI dataset training results

<b>Algorithm</b>	<b>mAP</b>	<b>AP<sub>50</sub></b>	<b>Training time</b>
<b>Cascade RCNN</b>	<b>.8894</b>	<b>.889</b>	5h38min
<b>Deformable DETR*</b>	.7697	.797	8h56min
<b>Faster RCNN</b>	.8810	.8807	4h7min
<b>FCOS</b>	.8815	.822	10h27min
<b>RetinaNet</b>	.8547	.855	1h51min
<b>ResNext</b>	.8816	.882	3h18min

Figure 4.4 shows a selected set of test scenes where all object classes were detected based on Faster RCNN, for illustrative purposes. The algorithm successfully detected different traffic object classes in the same scene when trained and tested on the KITTI dataset, as shown in the images.



Figure 4.4 KITTI ODA where the images are obtained from testing using faster RCNN.

It is also worth noting that the mAP obtained from testing and trained on the data itself achieved a high detection percentage. A deeper view of Cascade RCNN performance at IoU 50 is shown below (Table 4.7):

**Table 4.7** Ground truths (GTs), Detection(Dets), AP and Recall values for the classes

<b>Class</b>	<b>GTs</b>	<b>Dets</b>	<b>Recall</b>	<b>AP</b>
<b>Car</b>	3319	5393	0.958	0.941
<b>Person</b>	404	1220	0.851	0.759
<b>Cyclist</b>	133	424	0.910	0.822
<b>mAP</b>	0.841			

The overall AP<sub>50</sub> obtained from the classes is 84.1% and the mAP is 84.078%.

FCOS is another example of a much recent detection algorithm performance on KITTI. It was chosen for deeper detection performance evaluation as it is one of the newest detection algorithms and there has been little evaluation of the model, following Table 4.8 is a detailed evaluation of the FCOS.

**Table 4.8** Ground truths (GTs), Detection(Dets), AP and Recall values for the classes

<b>Class</b>	<b>GTs</b>	<b>Dets</b>	<b>Recall</b>	<b>AP</b>
<b>Car</b>	3319	13070	0.973	0.942
<b>Person</b>	404	5013	0.876	0.750
<b>Cyclist</b>	133	3692	0.940	0.849
<b>mAP</b>	0.847			

The overall AP<sub>50</sub> obtained from the classes is 84.7% and the mAP is 84.714%.

When comparing results to literature following Table 4.9 illustrate results

**Table 4.9** KITTI literature testing result (mAP)

<b>Reference</b>	<b>Detection</b>	<b>mAP(%)</b>
(Al-Refai & Al-Refai, 2020)	Darknet-53 CNN- YOLOv3	98.7
(Brazil & Liu, 2020)	RRC	75.33
	MS-CNN	73.70
	SDS-RCNN	63.05
	GDFL	68.62
	AR-Ped	73.44
(Feng et al., 2020)	SSD (W. Liu et al., 2016)	61.29
	YOLOv3 (Redmon et al., 2016)	80.52
	MobileNet-YOLOv3	78.32
(Wei et al., 2020)	Faster RCNN (Ren et al., 2015b)	79.11
	multiple scale CNN network model for	89.64

In the evaluation of the KITTI dataset, the performance results obtained align closely with the findings reported in the existing literature. My analysis and experiments on the dataset consistently demonstrate comparable outcomes to those documented in related studies. This alignment not only underscores the robustness and reliability of the dataset but also reinforces the validity of my research methods. These congruent results in the context of the KITTI dataset corroborate the consistency and repeatability of the findings and contribute to the broader body of knowledge in the field.

Now, we move on to the third dataset, EuroCity Persons (ECP), the most complex of the selected datasets.

### 4.3.3 EuroCity Person dataset

The ECP dataset has been trained and tested using different detection algorithms, the results are then recorded and displayed in Table 4.10.

**Table 4.10** ECP performance results when trained on 2080Ti GPU

Method	AP	AP <sub>50</sub>	Ap <sub>75</sub>	Ap <sub>1</sub>	Training time
Cascade RCNN	0.529	0.822	0.579	0.565	5day 7h36min
Faster RCNN	<b>0.550</b>	<b>0.866</b>	<b>0.606</b>	<b>0.598</b>	3day 5h7min
FCOS	0.347	0.694	0.3	0.406	3day 9h3min

When trained on the 2080Ti GPU, Faster RCNN achieved an average accuracy of 86.6% at IoU=0.5 and 82.2% when trained and tested using the Cascade RCNN model. FCOS achieved the lowest average accuracy of 69.4% at IoU=0.5 compared to the two-stage object detection models. However, when comparing this result with the result obtained by (Tian et al., 2019) when training on COCO dataset, they achieved an AP of 54.9%.

However, owing to low GPU space due to the large datasets used for this experiment, the training and testing of this data was carried out again using a 3090Ti GPU and MMDetection 2.28.2. This was adopted as the previous version was not compatible, the results were then recorded in the following table (Table 4.11). All experiments in this subsection were performed on the same GPU (3090 TI GPU).

**Table 4.11** ECP performance results when trained on 3090Ti GPU

Method	AP	AP <sub>50</sub>	Ap <sub>75</sub>	Ap <sub>1</sub>	Training time
Cascade RCNN	<b>0.181</b>	<b>0.31</b>	<b>0.184</b>	<b>0.273</b>	20h36min
Deformable DETR	0.0804	0.174	0.066	0.142	16h 26min
Faster RCNN	0.115	0.23	0.102	0.177	15h 47min
FCOS	0.104	0.222	0.085	0.18	18h 55min
RetinaNet	0.1012	0.206	0.086	0.167	14h 45min

The differences in graphics cards (2080Ti, 3090Ti) introduced variations in performance results due to variations in computational power, memory capacity, architecture, parallel processing capabilities, and numerical precision, among other factors.

Furthermore, when looking at the literature to compare the results obtained during training and testing on the ECP dataset, it is found that the ECP evaluation is mostly based on the *log average miss-rate* (LAMR), which is an evaluation metric developed by (Braun et al., 2019). To evaluate the performance of the dataset, it is calculated as follows:

$$LAMR = \exp\left(\frac{1}{9} \sum_f \log\left(\underset{fppi(c)}{\operatorname{argmax}} mr\left(\underset{fppi(c)}{\operatorname{argmax}} fppi(c)\right)\right)\right), \quad 4.1$$

where,  $mr$  is the *miss rate* for confidence value  $c$  such that only detections are taken into account with a confidence value greater or equal than  $c$ , which is calculated as shown in equation (4.2),  $fppi$  is the *false positives per image*, and its calculation is shown in equation (4.3).

$$mr = \frac{fn(c)}{tp(c) + fn(c)}, \quad 4.2$$

$fn(c)$  is the number of false negatives,  $tp(c)$  is the number of true positive.

$$fppi(c) = \frac{fp(c)}{\#img}, \quad 4.3$$

$fp(c)$  is the number of false positive,  $\#img$  is the total number of images. It is important to note that the lower the LAMR, the better is the performance of the detector. For LAMR evaluation you can refer to appendix C.

However, for the purpose of this thesis evaluating using LAMR is not necessary and wont achieve thesis comparison purpose and therefore mAP evaluation of the dataset was then captured and recorded in the following Table 4.12.



**Table 4.12**ECP literature testing results (mAP )

Reference	Detection method	mAP(%)
(Jiang et al., 2021)	HRNet-W18	88.7
	SPNet w cascade	89.6
	ResNet50 w FPN	88.4
	ResNet101 w FPN	87.8
	HRNet-W40	88.1

The ECP dataset has been extensively studied in this thesis, where, in addition to the different scoring matrices, a confusion metrics has been created, which will be used as a pre-trained model for the collected dataset in Chapter 5.

Starting with the single stage FCOS and RetinaNet detection, the different FCOS and RetinaNet scores are discussed in the following section.

### **A) FCOS Evaluation**

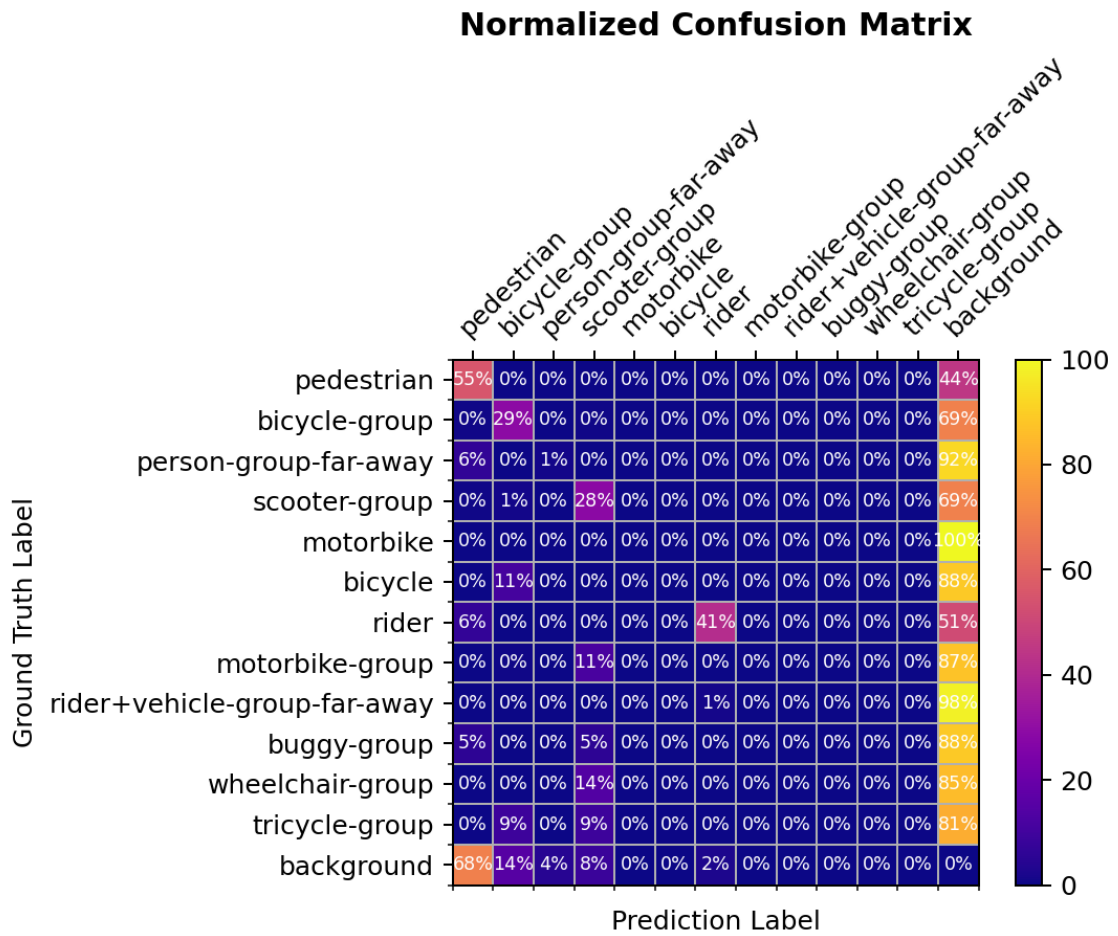
This dataset has 12 classes (pedestrian, bicycle-group, person-group-far-away, scooter-group, motorbike, bicycle, rider, motorbike-group, rider& vehicle-group-far-away, buggy-group, wheelchair-group, tricycle-group) that focus on evaluating the detection of the three main categories: cyclist, pedestrian, and cars. Table (4.13) provides an objective evaluation of the FCOS (Tian et al., 2019). Recall is also captured and presented in Table 4.13, where recall values estimate the ability of a classifier to label all positive objects.

**Table 4.13** Average percentage and average recall of FCOS

<b>Average Precision</b>					
All	All @50	All @75	Small	Medium	Large
0.104	0.222	0.085	0.037	0.099	0.180
<b>Average Recall</b>					
All	All @50	All @75	Small	Medium	Large
0.198	0.198	0.198	0.133	0.192	0.302

Following (Figure 4.5) is a confusion metrics; it is used as an evaluation matric where it helps in understanding the classes that are being confused by model as other class.





**Figure 4.5** ECP FCOS confusion matrix

Moving to the subjective evaluation of the network. Figure 4.6 present detection results where the red boxes show the detected objects.



**Figure 4.6** FCOS detection results on ECP dataset

Looking at the objective results obtained, it can be said that the FCOS model is yet to become capable for real time application.

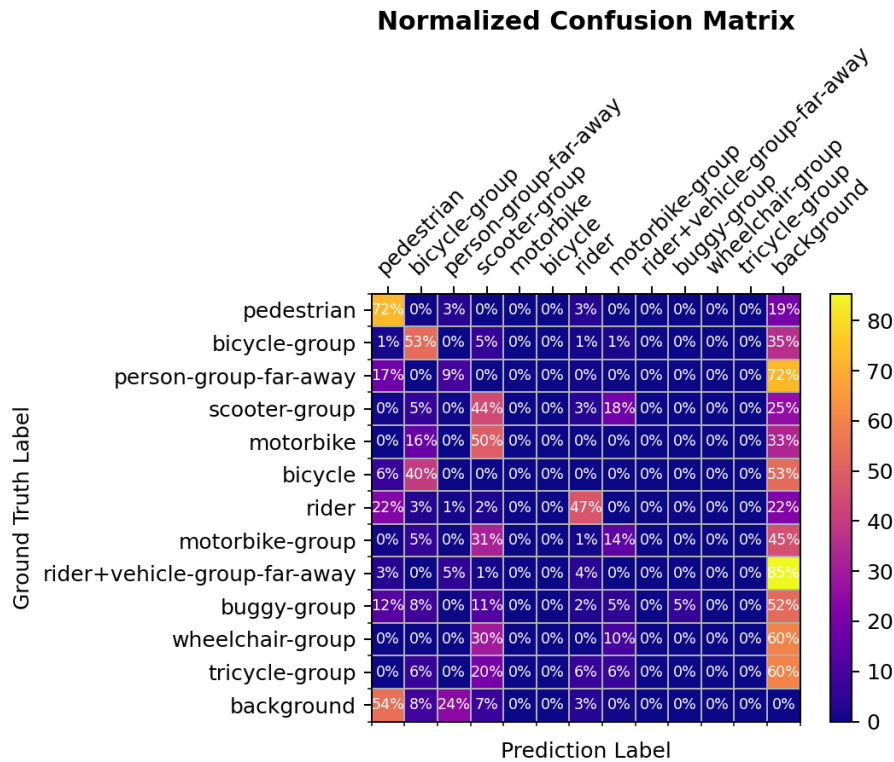
### B) RetinaNet Evaluation

When evaluating the subjective aspect of RetinaNet (Lin, Goyal, et al., 2017) using the 12 classes, the results showed better performance than FCOS as shown below.

**Table 4.14** Average percentage and average recall of RetinaNet

Average Precision					
All	All @50	All @75	Small	Medium	Large
0.1012	0.206	0.086	0.035	0.105	0.167
Average Recall					
All	All @50	All @75	Small	Medium	Large
0.203	0.203	0.203	0.118	0.205	0.284

A confusion matrix is then created for this network (Figure 4.7)



**Figure 4.7** ECP RetinaNet confusion matrix

Looking at the objective evaluation, the network was able to successfully detect the different objects from different classes, as shown in Figure 4.8. The blue box indicates the detected objects where the same images are used for compression purposes.





**Figure 4.8** RetinaNet detection results on ECP

FCOS performed better when compared to RetinaNet, although most objects were detected, however, owing to occlusion, a large number of bboxes were identified in the images as multiple objects and backgrounds, yet the network was able to identify and detect most objects in the image frame. Moving on to the two-step object

detection for ECP evaluation, the subjective and objective evaluation of Faster RCNN is analysed.

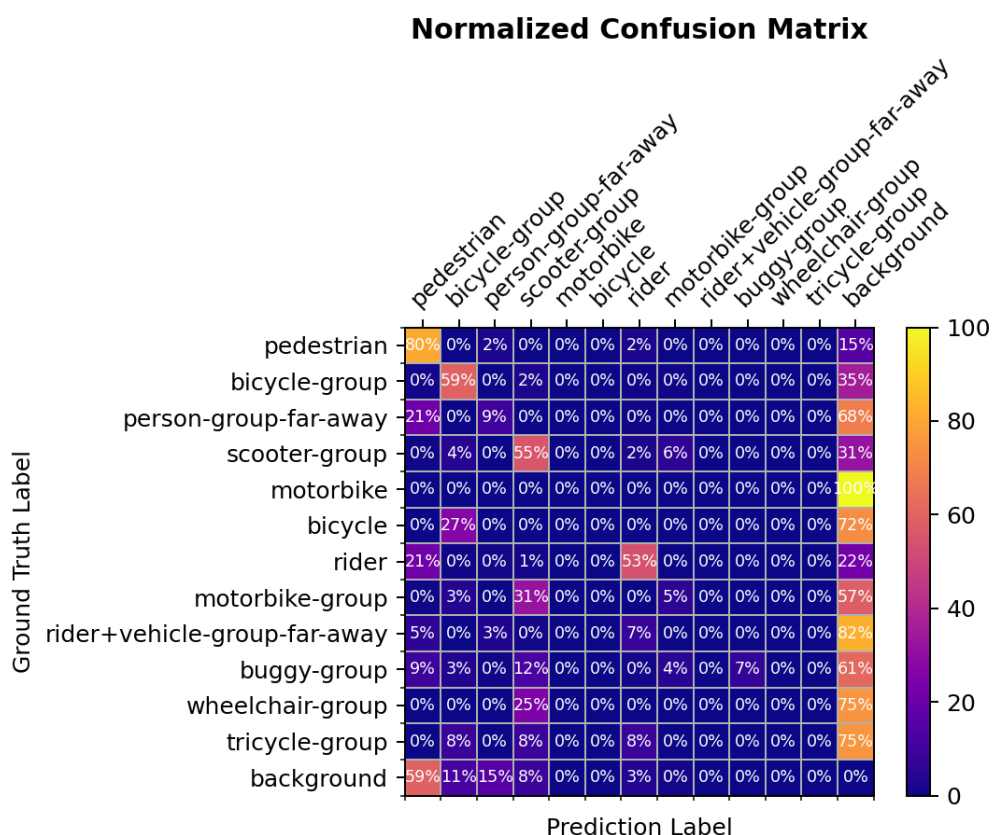
### C) Faster RCNN Evaluation

When training and testing dataset on Faster RCNN the following results are obtained (Table 4.15):

**Table 4.15** Average percentage and average recall of Faster RCNN

Average Precision					
All	All @50	All @75	Small	Medium	Large
0.115	0.23	0.102	0.066	0.115	0.177
Average Recall					
All	All @50	All @75	Small	Medium	Large
0.205	0.205	0.205	0.153	0.202	0.278

Plotting a confusion matrix for detection (Figure 4.9)



**Figure 4.9** Faster RCNN ECP confusion matrix

The results showed a better performance than the single level detection where the network was able to correctly detect pedestrians (80%) and riders (53%) in an occluded traffic.

Figure 4.10 shows a sample of the detection results, where the blue boxes show the detected objects and the scores and classes are given.



**Figure 4.10** Faster RCNN detection results on ECP

Moving to Cascade RCNN (Cai & Vasconcelos, 2017) detection, which, as mentioned earlier (Chapter 2), is a multi-stage extension of the two-stage RCNN object detection framework. It aims to achieve better object detection results.



#### ***D) Cascade RCNN Evaluation***

The following table (Table 4.16) illustrates the results obtained when using the cascade network.

**Table 4.16** Average percentage and average recall of Cascade RCNN

<b>Average Precision</b>					
All	All @50	All @75	Small	Medium	Large
0.181	0.31	0.184	0.129	0.171	0.273
<b>Average Recall</b>					
All	All @50	All @75	Small	Medium	Large
0.259	0.259	0.259	0.223	0.340	0.453

A confusion matrix was then computed (Figure 4.11), which shows that the network correctly detected more objects than the previous networks. More than 60% of the cyclists were correctly identified as cyclists, 80% of the pedestrians were correctly identified and classified, and 42% of the bicycles were correctly classified.



### Normalized Confusion Matrix

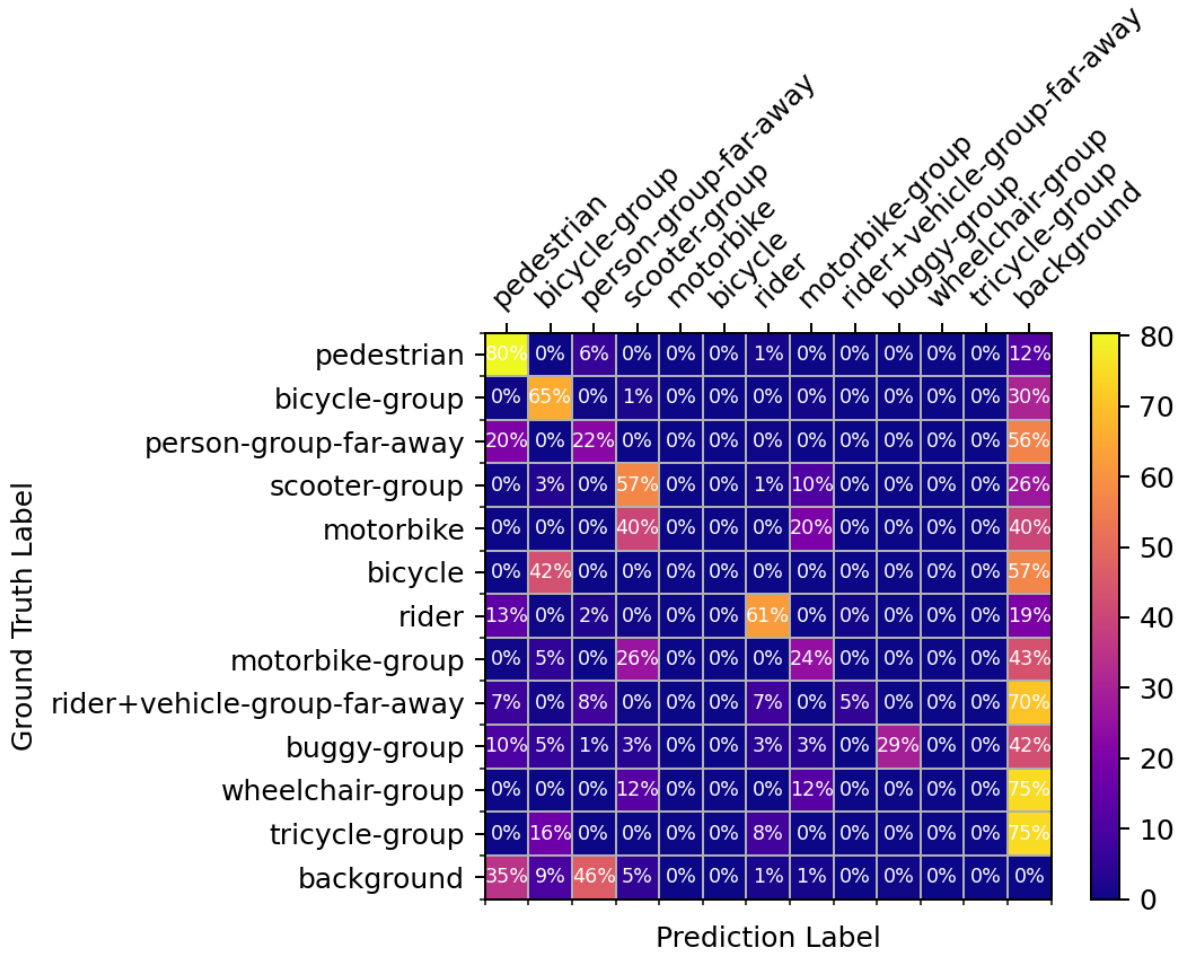


Figure 4.11 Cascade RCNN ECP confusion matrix

When the results are analysed visually, Figure 4.12 shows an example of detection results, where the blue box is the bbox, showing the classes and the detection score.



**Figure 4.12** Cascade RCNN results on ECP

Finally, we come to the deformable DETR, one of the latest object detection methods that aims to alleviate the slow convergence and high complexity issues witnessed in DETR as mentioned in Chapter 2.

### ***E) Deformable DETR Evaluation***

This network has not received much attention as it is a new algorithm. In this thesis, we examined its performance on the three datasets. The detection result on ECP is shown in the following table (Table 4.17).

**Table 4.17** Average percentage and average recall of Deformable DETR

<b>Average Precision</b>					
All	All @50	All @75	Small	Medium	Large
0.0804	0.174	0.066	0.036	0.075	0.142
<b>Average Recall</b>					
All	All @50	All @75	Small	Medium	Large
0.189	0.189	0.189	0.124	0.183	0.268

These results are visually presented in Figure 4.13 below. The detection results are low compared to the other detection models, but it needs further modifications to be used for real-time applications.



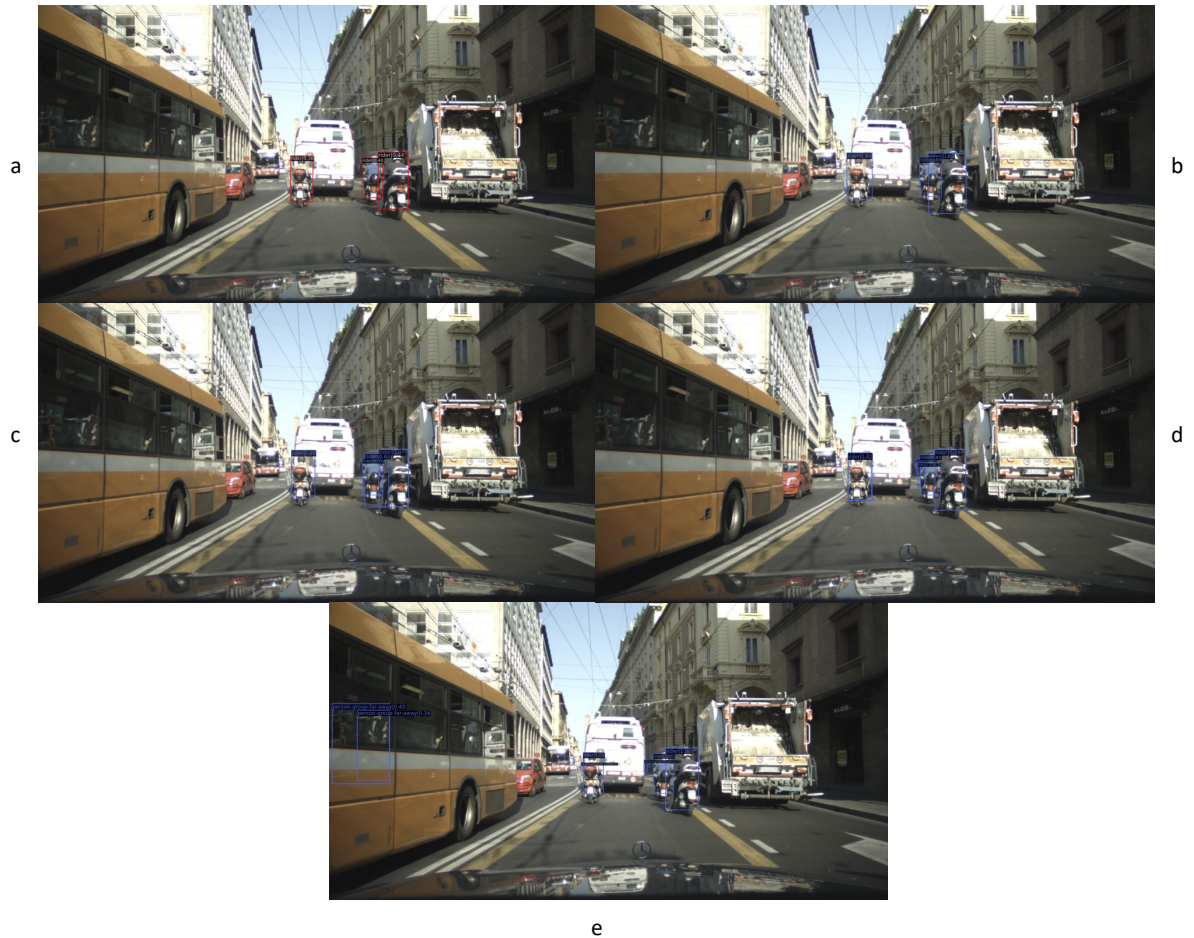
**Figure 4.13** Deformable DETR results on ECP.

Finally, comparing the performance of the five models on the ECP dataset, Cascade RCNN had the best performance accuracy of all the models. Figure 4.14 shows a visual comparison of object recognition, where the recognition scores are given next to the



class name. Taking the horse rider as an example, FCOS was able to detect the object with 38% accuracy (Figure 4.14(a)), RetinaNet detected the object with 84% accuracy (Figure 4.14(b)), Faster RCNN detected the same object with 99% accuracy (Figure 4.14(c)), deformable DETR managed to detect the object with a score of 42% (Figure 4.14(d)), and Cascade RCNN achieved a detection accuracy score of 100% in identifying the object as shown in (Figure 4.14(e)), but the network was also able to correctly categorise the other detected object with an average of 42%.

In an overall evaluation of ECP, the performance on this dataset can be considered poor due to the low performance percentage obtained. It is important to point out the different states of the sensor, which has a direct effect on the performance, the different classes involved, i.e. when focusing on the detection of one class (pedestrian, for example) the model achieved an average performance of 76%; this shows that reducing the number of classes detected will result in better performance, however in a real-time application it is not ideal to have one class and therefore the different classes related to cyclists and pedestrians are considered.



**Figure 4.14** Comparing detection result obtained on ECP, (a) FCOS detection results, (b) RetinaNet detection results, (c) Faster RCNN detection result, (d) Deformable DETR detection result, (e) Cascade RCNN detection result.

Results of testing on different datasets differ, which could be due to:

- Different image resolution.
- Overfitting of testing images.
- Flip and rotation of images.

For further evaluation of models, cross-validation is performed. Cross-validation is a technique used in machine learning to evaluate a model by training several models on subsets of the available input data and evaluating them on the complementary subset of the data. It is used to detect overfitting, which gives a better indication of how well a model performs on unseen data. This is useful in real-time applications.

## 4.4 DISCUSSION

The two-stage object detection generally showed a better detection result compared to the one-stage detection. From the above experimental results, it can be observed that the one-stage detection algorithm RetinaNet has excellent performance in terms of speed and achieved high accuracy compared to other detection algorithms for road object detection. Cascade RCNN achieved the highest accuracy in all datasets, but it requires a long computation time. The use of ResNext backbone improved the accuracy of detection and classification. Furthermore, using the proposed TRN detector showed a better result in the test, as shown in Table 4.3. The one-stage RetinaNet detector achieved lower results compared to the other one-stage detection models, however it has a strong ability to multiscale because of FPN and focal loss; therefore, it achieves higher accuracy results for  $AP_{\text{large}}$  and  $AP_{\text{medium}}$  as shown in ECP and Cityscapes results. This model can be considered as a balanced model as it maintains the same high level of precision and recall in all category levels for the targeted road objects. The results shown for both ECP and Cityscapes are lower than the results obtained using the KITTI dataset, due to the conditions in which the dataset was recorded, where the KITTI dataset is less demanding than the other two as it was collected in a city with clear weather. The number of classes that both ECP and Cityscapes cover for detection, the type of sensors used, and the location of the objects (i.e. the more distant the objects, the less likely they are to be detected). We considered testing on three datasets to analyse the detection performance and compare the performance of the proposed detection model under different complexity of road scenes.

## 4.5 CONCLUSION

This chapter compares the results of training and testing different datasets using five detection algorithms. The algorithms were trained on one dataset and tested on several datasets to check the dataset-independent performance of the algorithms. The Cascade RCNN algorithm performed best across the board, but performance varied significantly for each different dataset. In general, the algorithms performed best when tested on the KITTI dataset. This can be attributed to the consistency of the

conditions under which the data were collected. In general, the algorithms performed reasonably well when trained on the ECP dataset. Training times were long, and the dataset was collected in more variable situations. The algorithms performed best when trained on the KITTI dataset, which may be due to the better lighting and weather conditions, i.e., images were captured in dry and clear weather. An important implication of this study is that better performance results may not be obtained when tested on a larger training dataset; the quality of the training images proved to be a key factor influencing the performance of the algorithms. Furthermore, using the original image resolution proved to be better for algorithm performance, whereas changing the resolution to improve training time or computational cost proved to be worse.

The performance of none of the algorithms was consistent across the datasets, indicating that none of the algorithms are ready for real-time application with guaranteed accuracy. Furthermore, applications in complex scenes with cloudy and rainy weather conditions may not be consistent with the performance of the algorithms when tested on images taken in good visibility conditions. Further research is needed to improve the performance of the algorithms by minimising computational time and improving the consistency of detection accuracy prior to real-world application.

The inference time required that was obtained from training different models gives an insight into the suitability of a model for real-time application.

While detectors learned on one dataset do not necessarily transfer well to others, their ranking is stable across datasets, suggesting that insights can be learned from well-performing methods regardless of the benchmark.

We move on to Chapter 5, where all the insights gained in this chapter are considered and applied. The chapter will focus on the development of a benchmark dataset meticulously designed to evaluate the algorithms under various weather, lighting, and driving conditions. We delve into the creation process and conduct a thorough evaluation, setting the stage for a nuanced understanding of algorithmic adaptability in diverse real-world scenarios.



---

# Chapter 5

---

# 5 Benchmarking Framework for Training & Evaluation

This chapter focuses on the construction of an aggregated benchmark dataset for urban traffic scenes using dashboard camera images. The development of driver assistance systems and self-driving vehicles means that the detection of road users such as cyclists and pedestrians should be accurate and efficient to ensure road safety in a connected, automated environment. Research shows that the performance of attentive pedestrian detection lags by an order of magnitude, and this is one of the problems encountered by some detection methods.

This chapter starts with a cross-validation analysis of the different datasets used for the detection comparison (KITTI, Cityscapes and ECP). Section 5.2 introduces the benchmarking evaluation dataset, and Section 5.3 explains the details of the unified dataset. This is followed by the experimental design (Section 5.4) and finally the conclusion (Section 5.5).

## 5.1 TRAINING AND EVALUATION ACROSS DATASETS

We perform a cross-dataset evaluation to test how well the algorithms perform on unseen data from another dataset. This helps with generalisation and real-time application. Cross-validation is used in machine learning for:

- Better use of data: Cross-validation allows better use of data by using it for training and testing. Splitting the data into multiple folds and using each fold for testing gives a more accurate estimate of how well the model will perform on new, unseen data.
- Reduced risk of overfitting: Cross-validation can help reduce the risk of overfitting, which occurs when a model is too complex and fits the training data too well. By evaluating the model on multiple test sets, a better estimate can be made of how well the model will generalise to new data.
- Model selection: Cross-validation can help select the best model for a dataset. By comparing the performance of different models on the same data, the model that performs best and is most likely to generalise well to new data can be selected.

- **Robustness:** Cross-validation helps to assess how robust the model is. By using multiple test sets, a better estimate can be made of how well the model will perform on new data that may be different from the training data.

In this dissertation, we compared the performance of the different models on the datasets (Cityscapes, KITTI, ECP) on which this dissertation focuses. The algorithms were trained on a subset of each dataset and then tested on the test set for the KITTI dataset (see 2.4.4.3) and the validation set for both Cityscapes (see 2.4.4.2) and ECP (see 2.4.4.4).

Performing a cross-evaluation on three different datasets with different resolutions, annotation styles, class imbalance and data size is challenging. However, the main challenge of training on one dataset and testing on another is that the model may not perform as well on the test dataset as it did on the training dataset. This is because the testing dataset may contain different types of data or different distributions than the training dataset, which may cause the model to generalise poorly. In addition, if the test dataset is too small, it may not be representative of the larger population, which can also lead to poor generalisation. Nevertheless, experimentation is necessary to reach a conclusion. To perform cross-validation, several changes are required for each dataset. The following matrix illustrates the changes made.

**Table 5.1** Matrix of changes involved in unifying datasets.

	<b>Cityscapes</b>	<b>KITTI</b>	<b>ECP</b>
<b>Cascade RCNN</b>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>
<b>Faster RCNN</b>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>
<b>RetinaNet</b>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>
<b>FCOS</b>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>
<b>Deformable DETR</b>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Image size</li> </ul>	<ul style="list-style-type: none"> <li>• Annotation</li> <li>• Classes</li> <li>• Image size</li> </ul>

Annotation is the practice of adding explanatory notes to the images; the different datasets use different annotation styles. For example, the cityscapes annotation is designed for object detection and semantic segmentation of urban street scenes. The annotation of this dataset is considered as the most detailed annotation among the selected datasets. Therefore, it is necessary to modify the annotation of all the datasets for the cross-validation task.

Classes is the category of the object to be recognised. Each dataset has a different number of classes, so changing the number of classes is one of the steps considered in the validation—three main classes used for validation: Cars, Pedestrians and Cyclists. In the ECP dataset, the different categories related to pedestrians were combined into one class; this was done for the different classes in the dataset to eventually form three classes, similarly for the Cityscapes dataset.

Image size can affect recognition, as smaller images may not contain enough detail for accurate recognition, while larger images may require more processing power and time to analyse. Images were resized to avoid computational time issues, keeping the size in the range where image detail remains accurate for recognition.

### 5.1.1 Cross-Validation Results

The cross-validation results are shown in Table 5.1, where mAP (equation 2.33) is used to calculate the performance. The results show that Cascade RCNN performs better on most datasets, but takes longer to compute (see to Table 5.2).

However, some methods failed to detect when tested on a different dataset (i.e. training on cityscapes and testing on KITTI, the results shown in the table are almost 0. The reason may be due to resolution mismatch, different weather and lighting conditions); this gives an insight into the ability of the model to generalise. Cascade RCNN, Faster RCNN, FCOS, and RetinaNet were evaluated in this cross-dataset evaluation.

**Table 5.2** Cross dataset evaluation on Cityscapes, KITTI and EuroCity Person (mAP). A→B, refer to training on A and testing on B.

Method	Cityscapes → KITTI	Cityscapes → ECP	KITTI → Cityscapes	KITTI → ECP	ECP → Cityscapes	ECP → KITTI
RetinaNet	9.2	12.1	24.3	14.5	13.4	7.5
FCOS	4.5	11.3	16.9	12.4	7.7	6.7
Faster RCNN	7.9	17.8	27.5	19.1	15.7	8.8
Cascade RCNN	8.9	10.9	26.4	20.2	17.4	10.3

The results of the cross-evaluation showed that the models are not yet ready for real-time use, as they performed poorly when tested on unseen data. This may be due to:

- The model may be unable to generalize well due to the mismatch of data size.
- Overfit of data, in the case of training on KITTI and testing on other datasets. Nevertheless, generally, it is better to have more training data.
- A class mismatch will cause the model to not generalise the test data.

To overcome this problem, some techniques can be used, such as transfer learning (Torrey & Shavlik, 2010) or data augmentation (Shorten & Khoshgoftaar, 2019), which will help to improve performance. In this experiment, although the model was pre-trained using COCO, the model results were significantly low. Furthermore, if the classes in the training and test data do not match, the model may not be able to generalise to the test data. In this case, the classes were adjusted, and the model learned to classify the training data according to its classes, but it will not have seen the new classes in the test data. This could lead to poor performance on the test data. Reducing the image size could be seen as a reason for the poor scores, where some details of the images are lost, and the model may not be able to identify objects accurately. However, this step was taken firstly to have all images of the same size and secondly to speed up the training process and reduce the amount of memory required to store the images.

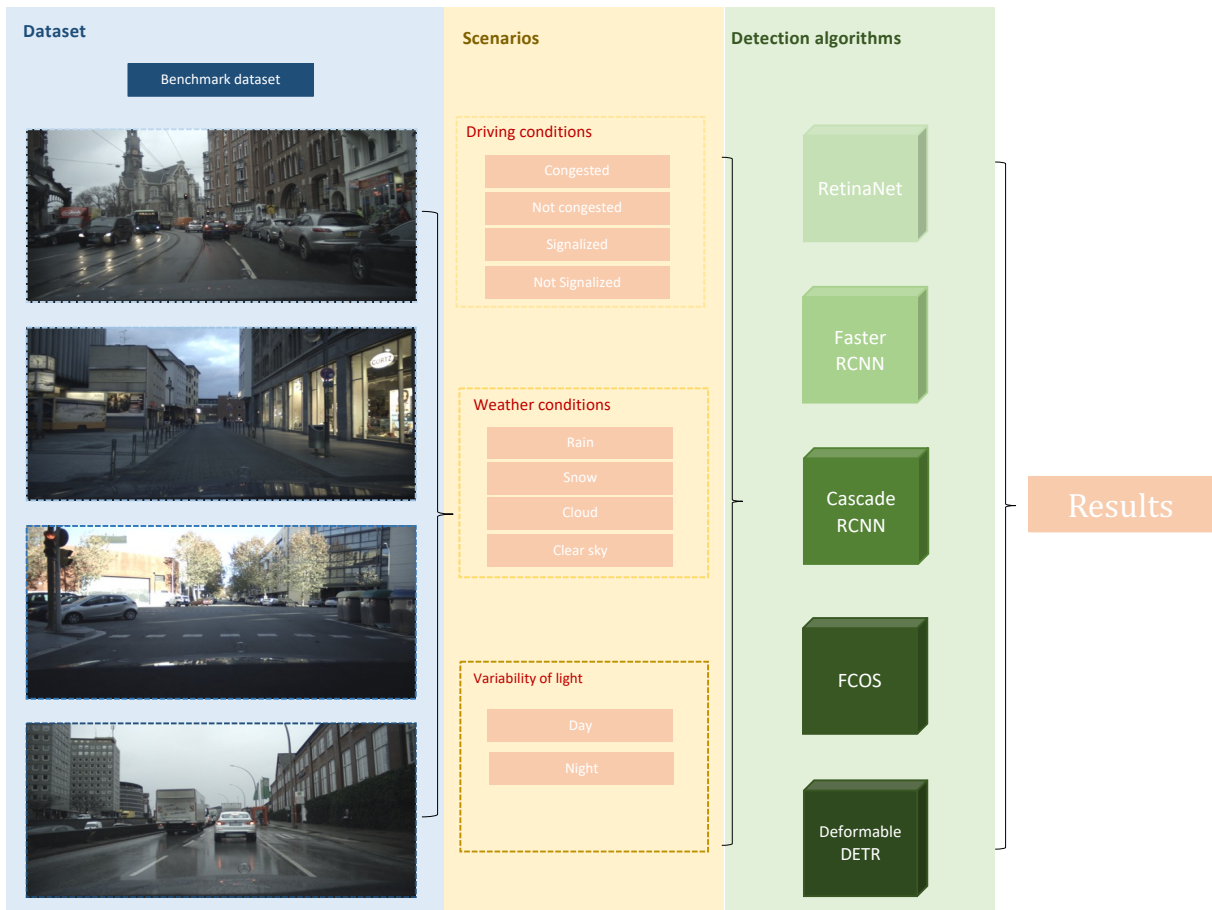
Moreover, on the cross evaluation of Deformable DETR results were considerably lower and exhibited a level of statistical insignificance when compared to the baseline algorithms and the proposed approach. Therefore, results were omitted from being presented in table 5.2.

## 5.2 DEVELOPMENT OF A UNIFIED BENCHMARKING EVALUATION DATASET

Datasets are critical entities in computer vision. In computer vision, a dataset is the source of information that a machine uses to train itself to learn and predict future outcomes based on the patterns found in the dataset.

A Unified Benchmarking Evaluation Dataset (UBED) is designed to ensure that state-of-the-art traffic object detection algorithms are evaluated using a standardised framework, with a particular focus on performance variations due to natural variations in real-world scenes. The benchmark dataset aims to advance the state of the art in traffic object detection under the different weather and driving conditions by placing the issue of object detection in the context of the broader issue of scene understanding. The UBED dataset contains images of complex everyday urban traffic scenes containing the two most important traffic object classes in the context of this research, namely pedestrians and cyclists. Put simply, UBED contains photographs that would be easily recognisable by a 4-year-old child, with a total of 8,000 images. All images were taken on urban roads and come from the previously presented Cityscapes, KITTI, and ECP datasets. The four detection algorithms used for testing are FCOS (Tian et al., 2019), RetinaNet (Lin, Goyal, et al., 2017), Faster RCNN(Ren et al., 2015b), Cascade RCNN(Cai & Vasconcelos, 2017) and Deformable DETR (Carion et al., 2020).

The UBED dataset specifically focuses on images containing pedestrians and cyclists, and Figure 5.1 summarises the framework of its formulation.



**Figure 5.1** Framework

Images in UBED are classified on the basis of different weather, visibility, and driving conditions. The reason for focusing on these three variables is that they contribute massively to the safety of road users. In addition, it is important to consider driving conditions when building a dataset for driving-related applications, such as autonomous driving or driver assistance systems. Different driving conditions, such as weather, traffic, and road conditions, can affect driving behaviour and the performance of these systems. For example, autonomous vehicles may need to adjust their speed or following distance in response to changing traffic or weather conditions. In addition, driver assistance systems, such as lane departure warning or adaptive cruise control, may need to be calibrated differently for different driving conditions. By including a variety of driving conditions in the data set, developers can ensure that their systems are robust and effective in a wide range of real-world situations.

A number of studies have been conducted to investigate the impact of daylight on collisions, for example (James, 2022; Sarma & Carey, 2017; Uttley & Fotios, 2017). For AV deployment, the different traffic conditions need to be analysed.

Traffic scenarios with the different road objects in the scope of this research/study will be evaluated using the different evaluation metrics discussed in the previous chapters.

### **5.2.1 Collation of datasets**

Combining multiple datasets involves several necessary steps. First, ensuring that the datasets are compatible means that they have the same data types and structure. This may involve converting data types or reformatting data. The next step is to identify common variables or fields that can be used to link the datasets. This involves normalising the data to ensure consistency between the datasets. For this work, changes were made accordingly, as mentioned earlier in Table 5.1. Each dataset underwent several changes before being combined. It is worth noting that the data were structured based on the COCO format and the annotation also followed the COCO format. A script developed by Hasan et al. (Hasan et al., 2021) to convert annotation to COCO format was used to combine the different annotation files for each subset (training, testing and validation) and generate a single annotation file.

#### **i) Dataset Size:**

- Total Number of Images: 21,274

#### **ii) Data Split:**

The dataset is divided into the following splits for training, validation, and testing purposes:

- Training: 60% (12,744 images)
- Validation: 30% (6,382 images)
- Testing: 10% (2,148 images)



### iii) Dataset Composition:

1. **ECP Dataset:**
  - Number of Images: 15896
2. **Cityscapes Dataset:**
  - Number of Images: 4878
3. **Kitti Dataset:**
  - Number of Images: 300
4. **Traffic field Dataset:**
  - Number of Images: 200

#### 5.2.2 Variation in Scenes

The main factors that were considered for variations in scenes:

1. Signalisation: a) Signalised or b) non-signalised
2. Congestion: a) Congested or b) uncongested
3. Light/Visibility: a) Daylight or b) Night
4. Weather Conditions: a) Dry, b) Rainy, c) Cloudy, and d) Snow

Various combinations of these factors were studied. Some key areas that were considered can be listed:

- Signalized Daylight Snow
- Signalized Night Rainy
- No signal Daylight
- No signal Daylight Cloudy
- Congested Night Rainy
- Uncongested Day Rainy
- Sideroad Daylight cloudy signalized
- Sideroad Daylight rain signalized.

The performance of the algorithms under these variable conditions is presented later in Section 5.4. Yet it is important to note that not all possible combinations were exhaustively tested. To ensure a focused and informative analysis, we deliberately selected a subset of scenarios that yielded meaningful and discernible results. The chosen scenarios were carefully curated to present a representative spectrum of challenges and variations in real-world traffic conditions.

### 5.3 TRAINING, TESTING AND VALIDATION OF UBED

The training, testing and validation experiment of UBED follows a similar experimental framework as presented in Chapter 5 (Figure 5.1). The data follows the same path, starting with the data split. The data follows a 70-10-20 split where 5,600 images are used for the training subset, 800 images for the validation subset, and 1,600 images for the testing subset. Next, the data is pre-processed, followed by training with the different models (Faster RCNN, Cascade RCNN, RetinaNet, FCOS, and Deformable DETR). After training, model performance is evaluated under different conditions.

The following subsection will delve into a comprehensive examination of the training and testing processes employed for the five object detection algorithms. Table 5.3 below presents the algorithms and datasets used for this chapter analysis.

**Table 5.3** Summary of dataset and algorithms used in this chapter.

Process	Dataset	Algorithms	Remark
Pre training	ECP	<ul style="list-style-type: none"> <li>• Faster RCNN</li> <li>• Cascade RCNN</li> <li>• RetinaNet</li> <li>• FCOS</li> <li>• Deformable DETR</li> </ul>	<ul style="list-style-type: none"> <li>• Each algorithm was pre trained used ECP</li> </ul>
Transfer learning \ Finetune	ImageNet	<ul style="list-style-type: none"> <li>• Faster RCNN</li> <li>• Cascade RCNN</li> <li>• RetinaNet</li> <li>• FCOS</li> <li>• Deformable DETR</li> </ul>	<ul style="list-style-type: none"> <li>• Used to initialize and speed the learning proses.</li> <li>• Each algorithm was finetuned using ImageNet</li> </ul>
Training and Testing Detection Algorithms	UBED	<ul style="list-style-type: none"> <li>• Faster RCNN</li> <li>• Cascade RCNN</li> <li>• RetinaNet</li> <li>• FCOS</li> <li>• Deformable DETR</li> </ul>	<ul style="list-style-type: none"> <li>• Trained on the entire training set and tested on a random traffic scenario (i.e. urban signalized road)</li> </ul>

### 5.3.1 Training of UBED

Before training the UBED, the data had to be pre-processed. Data pre-processing refers to the cleaning, transformation and preparation of data for analysis/modelling as mentioned in Chapter 4(4.2.3). Typically, the ImageNet dataset is used for fine tuning and transfer learning; however, for this task, ECP was used for transfer learning and fine tuning.

The experimental training neural network was implemented on images with a resolution of 1333\*800 using the PyTorch framework in Ubuntu 20.04, a 3.6GHz Core i7 10700K CPU, DDR4 3200Mhz 2X16-32GB quad-channel memory, and NVIDIA GeForce RTX 3090 Ti dual-card GPU. The hyperparameters used are listed in Table 5.1.

When evaluating the performance under each condition, the results obtained for the average accuracy were lower than values obtained in previous chapter (chapter 4) for several reasons, such as insufficient data and overfitting. Therefore, the algorithm was retrained using the entire UBED training.

### 5.3.2 Testing and Validation

Precision and recall at different thresholds are evaluated to validate and test the different models on the dataset. Precision measures the percentage of true positives among all positive predictions (equation 2.29) and recall measures the percentage of true positives among all actual positive samples (equation 2.30).

Precision was evaluated at  $AP_{50}$ ,  $AP_{75}$ ,  $AP_{small}$ ,  $AP_{medium}$ , and  $AP_{large}$ . Where the subscript indicates the IoU threshold (see 2.5.5.3). In addition, the small subscript represents an object with a pixel area  $<32^2$  pixels. Medium subscript represents objects with an area of  $32^2 < \text{area} < 96^2$  pixels. A large subscript indicates an object with an area of  $96^2 < \text{area}$  pixels. Note that the higher the AP, the better the performance.

Recall was also evaluated at  $AR_{100}$ ,  $AR_{1000}$ ,  $AP_{small}$ ,  $AP_{medium}$ , and  $AP_{large}$ .  $AR_{100}$ , for example, stands for Average Recall at 100, which measures the average percentage of objects correctly detected by the model when the number of detections per image is varied from 1 to 100. This also applies to  $AR_{1000}$ . However,  $AR_{small}$ ,

$AP_{\text{medium}}$ , and  $AP_{\text{large}}$  evaluate the recall when the number of detections per image varies between 1 and 1000 under the previously defined pixel area. The higher the AR score, the better the performance in terms of detecting more objects in an image.

## 5.4 PERFORMANCE EVALUATION

This section discusses an evaluation of the models for the scenarios.

### 5.4.1 Faster RCNN

When evaluating the performance under Faster RCNN with the data trained on the UDEB training set and tested on the different condition subset, results are recorded in the following table (Table 5.4).

**Table 5.4** Evaluation of UDEB trained and tested using Faster RCNN

<b>Average Precision</b>					
All	All @50	All @75	Small	Medium	Large
0.197	0.348	0.199	0.122	0.209	0.256
<b>Average Recall</b>					
All	All @50	All @75	Small	Medium	Large
0.309	0.309	0.309	0.205	0.319	0.359

The performance evaluation on the different condition subset results are shown in the following table (Table 5.5)

**Table 5.5** Result obtained from testing UBED on Faster RCNN

Scene types		AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>small</sub>	AP <sub>Medium</sub>	AP <sub>Large</sub>	AR	AR <sub>small</sub>	AR <sub>Medium</sub>	AR <sub>Large</sub>
Day	Sideroad cloudy signalised	0.162	0.301	0.159	0.092	0.177	0.209	0.285	0.208	0.289	0.322
	Sideroad non- signalised	0.150	0.290	0.138	0.082	0.157	0.209	0.259	0.169	0.261	0.314
	Urban non- signalised	0.144	0.282	0.129	0.095	0.153	0.182	0.256	0.195	0.265	0.278
	Urban signalised	0.148	0.122	<b>0.209</b>	<b>0.256</b>	<b>0.309</b>	0.205	<b>0.319</b>	<b>0.359</b>	<b>0.348</b>	0.122
	Urban non- signalised rainy	0.162	0.301	0.159	0.092	0.177	0.209	0.285	0.208	0.289	0.322
	Urban Snow signalised	0.191	<b>0.339</b>	0.191	0.117	0.204	<b>0.249</b>	0.303	0.207	0.314	<b>0.349</b>
Night	Sideroad non-signalised	0.125	0.259	0.105	0.074	0.131	0.169	0.230	0.169	0.233	0.263
	Urban rain signalised	<b>0.232</b>	0.054	0.092	0.054	0.119	0.152	0.207	0.150	0.214	0.246
	Urban signalised	0.139	0.270	0.128	0.074	0.144	0.197	0.268	0.181	0.276	0.313

The complexity of the model was also evaluated. The results are recorded as follows:

FLOPs: 206.72 GMAC	Parameters: 41.18 M
Input shape: (3, 1280, 800)	

#### 5.4.2 Cascade RCNN

The UDEB cascade RCNN evaluation, is presented in Table 5.6.

**Table 5.6** Evaluation of UDEB trained using Cascade RCNN

Average Precision					
All	All @50	All @75	Small	Medium	Large
0.193	0.325	0.207	0.107	0.208	0.245
Average Recall					
All	All @50	All @75	Small	Medium	Large
0.281	0.281	0.281	0.192	0.283	0.321

Next Table 5.7 present is the evaluation of the different traffic scene.

**Table 5.7** Result obtained from testing UBED on Cascade RCNN

Scene types		AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>small</sub>	AP <sub>Medium</sub>	AP <sub>Large</sub>	AR	AR <sub>small</sub>	AR <sub>Medium</sub>	AR <sub>large</sub>
Day	Sideroad cloud signalised	0.161	0.314	0.175	0.094	0.191	0.233	0.285	0.2	0.3	0.332
	Sideroad non- signalised	0.173	0.308	0.173	0.097	0.178	0.241	0.278	0.189	0.279	0.337
	Urban non- signalised	0.185	0.32	0.193	0.101	0.194	0.252	<b>0.292</b>	<b>0.213</b>	0.301	0.328
	Urban signalised	0.162	0.290	0.162	0.092	0.169	0.222	0.242	0.149	0.246	0.298
	Urban non- signalised rain	0.181	0.316	0.185	0.098	0.193	0.243	0.295	0.203	0.301	<b>0.342</b>
	Urban Snow signalised	<b>0.193</b>	<b>0.332</b>	<b>0.203</b>	<b>0.116</b>	<b>0.20</b>	<b>0.249</b>	0.29	0.203	<b>0.303</b>	<b>0.342</b>
Night	Sideroad non-signalised	0.15	0.28	0.146	0.086	0.153	0.214	0.245	0.171	0.241	0.316
	Urban rain signalised	0.125	0.241	0.116	0.059	0.128	0.182	0.202	0.141	0.2	0.253
	Urban signalised	0.168	0.298	0.169	0.092	0.175	0.223	0.265	0.178	0.270	0.301

The complexity of the model is evaluated as follows:

FLOPs: 234.5 GMAC	Parameters: 68.96 M
Input shape: (3, 1280, 800)	

### 5.4.3 RetinaNet

The training evaluation of RetinaNet on UDEB is shown in Table 5.8.

**Table 5.8** Evaluation of UDEB trained using RetinaNet

Average Precision					
All	All @50	All @75	Small	Medium	Large
0.116	0.226	0.109	0.041	0.129	0.175
Average Recall					
All	All @50	All @75	Small	Medium	Large
0.266	0.266	0.266	0.131	0.297	0.319

Next is the evaluation on the different scenarios Table 5.9.

**Table 5.9** Result obtained from testing UBED on RetinaNet

	Scene types	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>small</sub>	AP <sub>Medium</sub>	AP <sub>Large</sub>	AR	AR <sub>small</sub>	AR <sub>Medium</sub>	AR <sub>large</sub>
Day	Sideroad cloud signalised	0.075	0.171	0.053	0.028	0.089	0.110	0.202	0.1	0.21	0.247
	Sideroad non- signalised	0.054	0.127	0.037	0.022	0.062	0.085	0.181	0.088	0.207	0.227
	Urban non- signalised	0.092	0.19	0.077	<b>0.039</b>	0.104	0.135	0.239	0.122	0.264	0.292
	Urban signalised	0.066	0.147	0.049	0.022	0.075	0.1	0.2	0.103	0.203	0.273
	Urban non- signalised rain	0.075	0.163	0.057	0.027	0.084	0.115	0.202	0.099	0.216	0.257
	Urban Snow signalised	<b>0.1</b>	<b>0.215</b>	<b>0.101</b>	0.038	<b>0.123</b>	<b>0.166</b>	<b>0.254</b>	<b>0.125</b>	<b>0.285</b>	<b>0.305</b>
Night	Sideroad non-signalised	0.028	0.073	0.015	0.012	0.035	0.038	0.112	0.05	0.117	0.155
	Urban rain signalised	0.014	0.043	0.005	0.007	0.019	0.018	0.068	0.024	0.09	0.072
	Urban signalised	0.032	0.074	0.021	0.016	0.039	0.036	0.108	0.066	0.106	0.143

The efficiency of the model is then calculated and recorded as follows:

FLOPs: 195.23 GMAC	Parameters: 39.83 M
Input shape: (3, 1280, 800)	

#### 5.4.4 FCOS

The evaluation of training a more recent one-stage detection FCOS on UDEB can be seen in Table 5.10.

**Table 5.10** Evaluation of UDEB trained and tested using Faster RCNN

Average Precision					
All	All @50	All @75	Small	Medium	Large
0.154	0.301	0.144	0.05	0.166	0.257
Average Recall					
All	All @50	All @75	Small	Medium	Large
0.274	0.274	0.274	0.151	0.287	0.484

Traffic sense evaluation is given in the following Table 5.11:

**Table 5.11** Result obtained from testing UBED on FCOS

	Scene types	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>small</sub>	AP <sub>Medium</sub>	AP <sub>Large</sub>	AR	AR <sub>small</sub>	AR <sub>Medium</sub>	AR <sub>large</sub>
Day	Sideroad cloudy signalised	0.102	0.248	0.057	0.026	0.117	0.148	0.197	0.084	0.214	0.282
	Sideroad non- signalised	0.092	0.237	0.052	0.025	0.103	0.163	0.203	0.066	0.233	0.398
	Urban non- signalised	0.11	0.264	0.064	0.039	0.120	0.176	0.228	0.131	0.241	0.392
	Urban signalised	0.091	0.243	0.04	0.031	0.096	0.162	0.195	0.095	0.202	0.293
	Urban non- signalised rainy	0.1	0.236	0.047	0.024	0.112	0.133	0.182	0.075	0.204	0.271
	Urban snow signalised	<b>0.13</b>	<b>0.279</b>	<b>0.104</b>	<b>0.044</b>	<b>0.139</b>	<b>0.212</b>	<b>0.256</b>	<b>0.135</b>	<b>0.276</b>	<b>0.471</b>
Night	Sideroad non-signalised	0.082	0.213	0.037	0.014	0.085	0.149	0.179	0.073	0.183	0.367
	Urban rain signalised	0.058	0.159	0.025	0.013	0.067	0.099	0.155	0.061	0.175	0.22
	Urban signalised	0.086	0.238	0.028	0.024	0.093	0.137	0.175	0.075	0.186	0.251

Computing the FLOPs of the network resulted in the following:

FLOPs: 435.25 GMAC	Parameters: 89.63 M
Input shape: (3, 1280, 800)	

#### 5.4.5 Deformable DETR

Deformable DETR for this experiment uses an AdamW optimiser, because AdamW uses a combination of adaptive learning rates and weight decay to update the weights of the model. The adaptive learning rates help to ensure that the model converges more quickly, while the weight decay helps to prevent overfitting. The results of evaluating deformable DETR performance in the different scenarios are shown in Table 5.12.

**Table 5.12** Evaluation of UDEB trained and tested using Deformable DETR

Average Precision					
All	All @50	All @75	Small	Medium	Large
0.041	0.112	0.019	0.008	0.046	0.082
Average Recall					
All	All @50	All @75	Small	Medium	Large
0.139	0.139	0.139	0.046	0.142	0.213

On evaluating the different conditions results obtained as follow (Table 5.13):



**Table 5.13** Result obtained from testing UBED on Deformable DETR

Scene types		AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>small</sub>	AP <sub>Medium</sub>	AP <sub>Large</sub>	AR	AR <sub>small</sub>	AR <sub>Medium</sub>	AR <sub>large</sub>
Day	Sideroad cloudy signalised	0.03	0.091	0.014	<b>0.009</b>	<b>0.034</b>	0.062	0.114	0.038	0.119	0.19
	Sideroad non- signalised	0.015	0.054	0.002	0.001	0.015	0.034	0.068	0.011	0.068	0.135
	Urban non- signalised	0.027	0.086	0.01	0.007	0.030	0.054	0.102	0.022	0.115	0.161
	Urban signalised	0.02	0.061	0.008	0.005	0.02	0.048	0.075	0.018	0.089	0.120
	Urban non- signalised rainy	0.028	0.086	<b>0.012</b>	0.005	0.031	0.059	0.111	0.035	0.113	0.185
	Urban Snow signalised	<b>0.031</b>	<b>0.093</b>	<b>0.012</b>	0.005	0.029	<b>0.072</b>	<b>0.122</b>	<b>0.05</b>	<b>0.12</b>	<b>0.198</b>
Night	Sideroad non-signalised	0.009	0.037	0.001	0.002	0.009	0.025	0.055	0.010	0.055	0.111
	Urban rainy signalised	0.007	0.027	0.001	0.004	0.008	0.019	0.047	0.012	0.051	0.088
	Urban signalised	0.013	0.046	0.003	0.001	0.012	0.038	0.073	0.009	0.061	0.171

Looking at the FLOPs results below, it can be seen that it achieved the best results apart from RetinaNet. However, it requires further modifications to suit traffic detection.

FLOPs: 195.23 GMAC	Parameters: 39.83 M
Input shape: (3, 1280, 800)	

## 5.5 DISCUSSION & CONCLUSION

This chapter presented an evaluation of different weather conditions using five different deep learning algorithms on detection models for autonomous vehicles. The objective was to assess the performance of the detection models under various weather scenarios, providing insights into their suitability for real-world applications.

Through the experimentation on the benchmark dataset, it was observed that the detection models exhibited varying levels of performance across different weather conditions. The results demonstrated that certain weather conditions, such as heavy rain or snow, posed significant challenges for the detection algorithms, leading to decreased accuracy and reliability. On the other hand, clear weather conditions generally yielded better detection results.

Among the five deep learning algorithms tested, it was found that certain algorithms performed better under specific weather conditions. For example,

cascade RCNN demonstrated superior performance in detecting objects during sunny weather, while FCOS excelled in night driving conditions. These findings highlight the importance of selecting appropriate algorithms based on the prevailing weather conditions to optimize the detection capabilities of autonomous vehicles.

Additionally, it became evident that the limited availability of annotated data for specific weather conditions posed a challenge in training and evaluating the detection models. The lack of diverse and comprehensive datasets for all weather scenarios limited the generalizability of the results. Therefore, future research should focus on expanding and diversifying benchmark datasets to encompass a wide range of weather conditions and driving scenarios.

Moreover, the evaluation of the detection models highlighted the importance of considering both detection accuracy and computational efficiency in real-time applications. While some algorithms exhibited higher accuracy rates such as Cascade RCNN, they were computationally intensive and required significant processing power and time. Balancing accuracy and efficiency is crucial to ensure that the detection models can be effectively deployed in real-world autonomous vehicle systems.

It should be noted that this evaluation focused solely on the detection aspect of autonomous vehicles under different weather conditions. Future research should consider the integration of detection with other modules, such as tracking and decision-making, to assess the overall performance of autonomous vehicles in adverse weather situations.

In conclusion, the evaluation of different weather conditions using five deep learning algorithms on detection models provides valuable insights into the strengths and limitations of these algorithms in challenging weather scenarios. The findings emphasize the need for continued research and development to improve the accuracy, and computational efficiency of detection models for autonomous vehicles operating in diverse weather conditions. By addressing these challenges, we can advance the capabilities of autonomous vehicles and pave the way for safer and more reliable autonomous transportation systems in the future.

The next chapter present the development of a unique detection, tracking, and estimation algorithm. This innovative approach aims to address specific challenges

posed by real-time video scenarios. Based on Chapters 4 and 5 and the FLOPs evaluation, the single-stage object detection RetinaNet is modified and used for detection in addition to the tracking and estimation algorithms.

---

# Chapter 6

---

# 6 Detection, Estimation & Tracking Road Objects for Assisting Driving

This study presented in this chapter focuses on extending the scope of image analysis algorithms from object detection to object tracking using dash-cam imagery. A combined detection-tracking-estimation (DTE) algorithm of road objects is developed and validated using images from existing datasets and real-world images. It is organized as follows: Section 6.1 covers the methodology used and the framework developed. Section 6.2 highlight the data used for this paper. Section 6.3 presents the analysis and the results of the application. Finally, section 6.4 discusses the applications of the proposed method, and draws some conclusions.

## 6.1 PROPOSED METHODOLOGY OF DTE ALGORITHM

This study focuses on developing methodologies for detecting, estimating, and tracking the flow, speed, and distance of pedestrians, cyclists, and vehicles from an autonomous vehicle utilizing only visual information. The study uses videos captured from a single dash-cam of a human-driven car in the absence of access to an autonomous vehicle or appropriate video footage acquired from an autonomous vehicle. The complete proposed framework for detecting moving objects from a moving vehicle in an urban transport network is described in this section, as shown in Figure 6.1. The framework consists of three main modules: detection, estimation, and tracking. The detection module consists of a RetinaNet detector, which contains the ResNeXt backbone network (S. Xie et al., 2017), Feature Pyramid Network (FPN)(Welch & Bishop, 1995), and class/ bbox subnets (Lin et al., 2016). The detection network finally outputs an image with a boundary box around the detected object, the relevant class, and a confidence score. The confidence score shows the probability of the object being detected correctly by the algorithm is fed to the next modules. Both estimation and tracking use this score to process. In estimation, a triangle, the distance, volume, flow rate per second and the pedestrian speed are calculated, and the model will give an accurate distance and flow of surrounding objects. The output of the two models' detection and estimation will be tracked, which is achieved by utilizing an improved Simple Online and Real-time

tracking algorithm. The algorithm used Kalman prediction (Welch & Bishop, 1995), object association, buffer module for miss detection, and tracking information update. Figure 6.1 provide a schematic of the whole process. Table 6.1 presents the different datasets and algorithms used for this chapter.

**Table 6.1** Datasets and algorithms used for training and testing in this Chapter.

<b>Process</b>	<b>Dataset</b>	<b>Algorithms</b>	<b>Remark</b>
Transfer learning \ Finetune	ImageNet	<ul style="list-style-type: none"> <li>Traffic RetinaNet</li> </ul>	<ul style="list-style-type: none"> <li>Used to initialize and speed the learning proses.</li> <li>Each algorithm was finetuned using ImageNet</li> </ul>
Pre training	COCO	<ul style="list-style-type: none"> <li>Traffic RetinaNet</li> </ul>	<ul style="list-style-type: none"> <li>COCO is used to pre train the network</li> </ul>
Training and Testing Algorithms	COCO Cityscapes Kitti Traffic Field Dataset MOT 16	<ul style="list-style-type: none"> <li>Traffic RetinaNet</li> <li>Distance Estimation</li> <li>Improved SORT</li> </ul>	<ul style="list-style-type: none"> <li>For trained the different datasets were used as there is no one dataset that considers all different functions yet for testing the network it was tested on a single video collected in the UAE and is part of the complex videos provided in the Traffic Field Dataset</li> </ul>

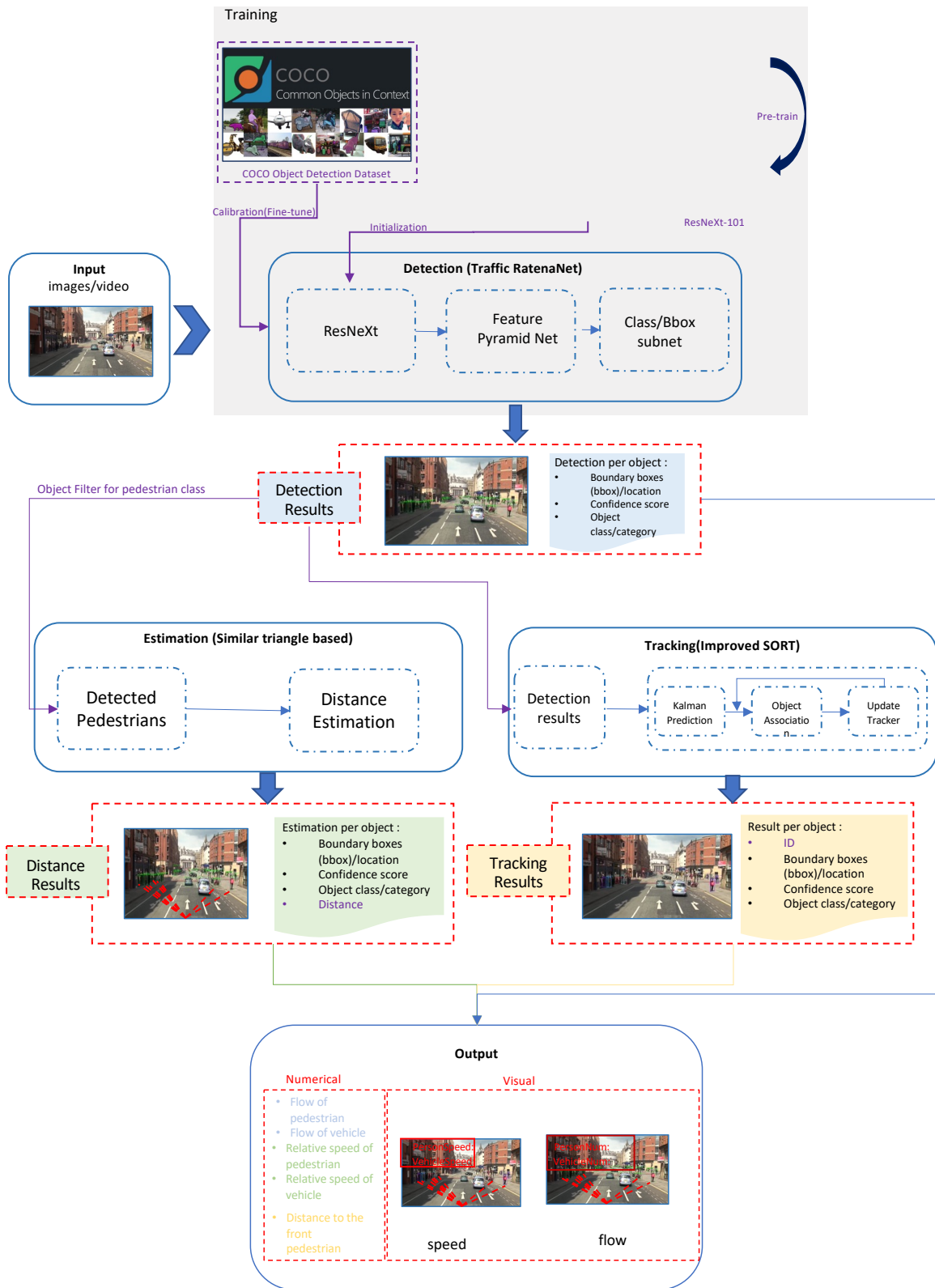


Figure 6.1 Framework of detection, tracking and estimation.

## 6.2 DESCRIPTION OF DETECTION, ESTIMATION, AND TRACKING ALGORITHMS

### 6.2.1 Object Detection Algorithm: Traffic RetinaNet

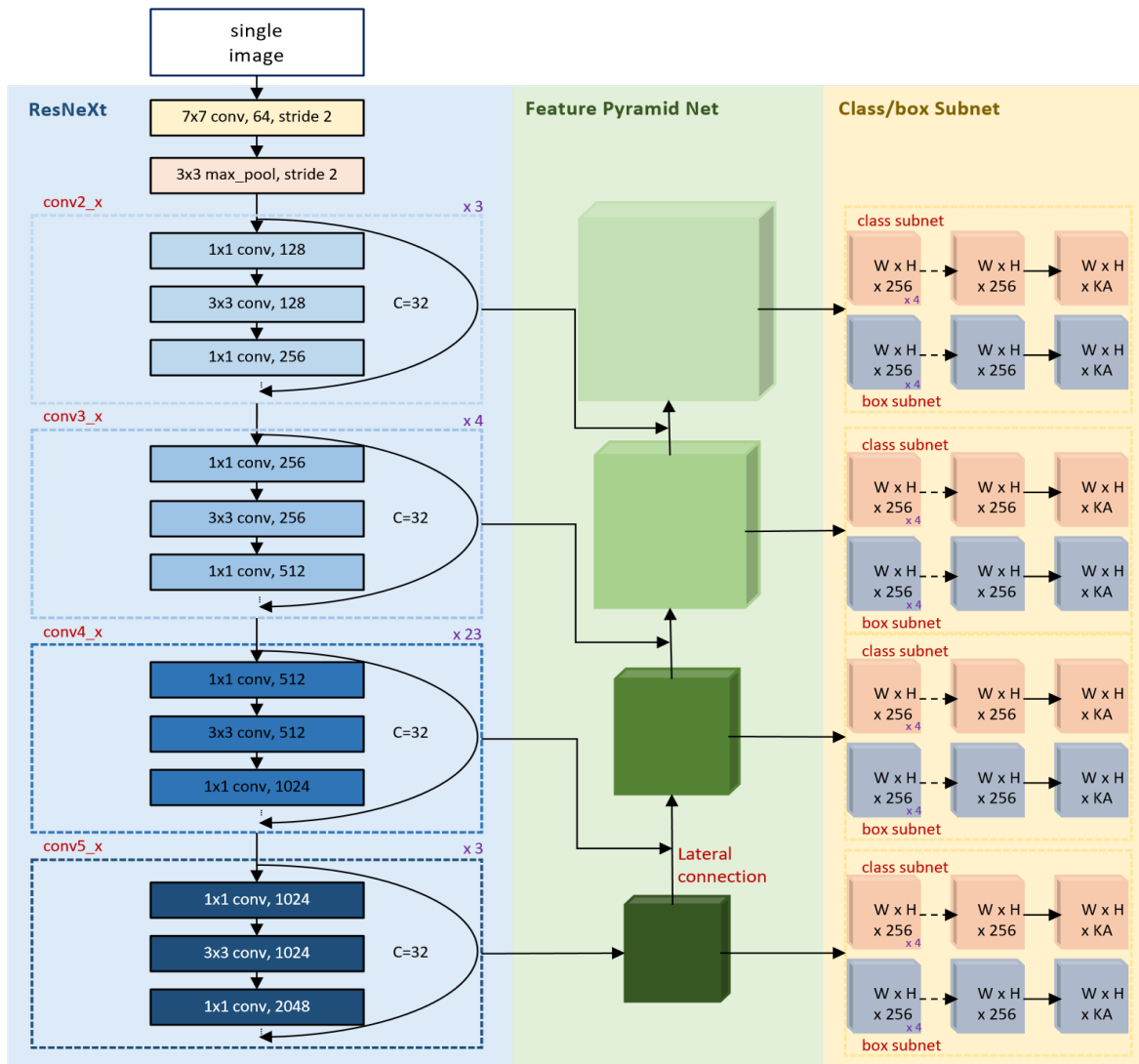
This study introduces 'Traffic RetinaNet,' a novel algorithm that combines elements from established object detection algorithms, specifically RetinaNet and the ResNext residual block. By integrating the RetinaNet algorithm with a ResNext backbone, this work presents a unique approach to object detection in traffic scenarios, distinct from existing methods in the literature. Additionally, transfer learning was employed by initializing the network with a pre-trained model. Transfer learning is a method used in machine learning where a model developed for a task is reused as a starting point for a model in another task (Yosinski, Clune, Bengio, & Lipson, 2014). In other words, a model is trained on a large dataset, and knowledge is transferred to a smaller dataset. Thus, we can apply the weight and architecture obtained to the problem statement. Finetuning is the process of precisely adjusting model parameters to fit specific observations (Gunawan, Lau, & Lindawati, 2011). ImageNet dataset (refer to 2.6.4.7) is used for transferring and finetuning.

The problem statement is to train a model to locate and classify regions into five different categories, as mentioned earlier. The pre-trained model shows a strong ability to generalize images outside the dataset (ImageNet) by transfer learning. This was used on the different datasets such as COCO (refer to 2.5.4.1), Cityscapes (refer to 2.5.4.2) and KITTI (refer to 2.5.4.3) datasets in order for modifications to take place thus to suit our model.

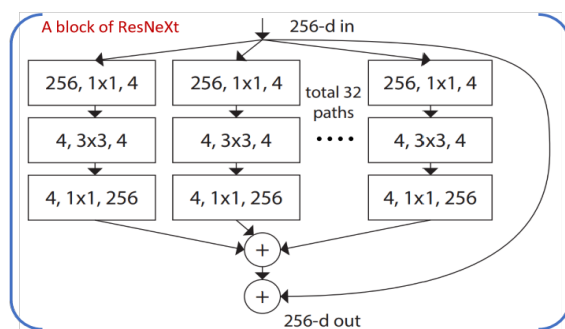
#### *A) ResNext*

In order to achieve more efficient detection, ResNeXt (S. Xie et al., 2017) is utilized as the backbone of RetinaNet as shown in Figure 6.2.

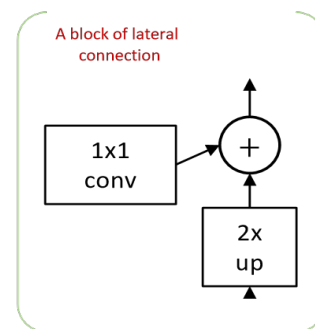




(a)



(b)



(c)

**Figure 6.2** Framework of RetinaNet with ResNeXt backbone (Traffic RetinaNet).

The initialization product and finetuning output obtained from the previous steps are now fed into ResNeXt. In image classification, ResNeXt architecture is characterized as a simple, complex and efficient modularized network. The network is made of a set of repeated building blocks that aggregates through several transformations with the

same topology. ResNeXt is a homogeneous, multi-branch architecture with only a few hyperparameters to set. As shown in Figure 2(a), the input image passes through a set of lower-dimensional embeddings (by  $1 \times 1$  convolutions), followed by a set of specialized filters ( $3 \times 3$ ,  $5 \times 5$ , etc). Figure 2(b) shows a block of ResNext with 32 cardinalities with similar complexity in which aggregation of residual transformation is performed (S. Xie et al., 2017). The output of each convolution is then passed to the feature pyramid net (refer to Chapter 5 for more details on FPN). Figure 2(a) illustrates the pathway, and Figure 2(c) shows the lateral connection details. After this stage, outputs are passed to the last subnet. The object classification and box regression subnets are shown in Figure 2(right), in which the object's presence is predicted at each location, and each anchor offset is registered and matches the nearest ground truth. Focal losses are also calculated at this stage.

The Traffic RetinaNet was deployed on Ubuntu16.04 with pyTorch 1.2 environment. The backbone net was initialized according to ResNeXt (Hoang, Nguyen, Truong, Lee, & Park, 2019) and pre-trained on ImageNet (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, Fei-Fei, et al., 2015). The rest of conv layers except class/bbox subnet are initialized with bias,  $b = 0$  and a Gaussian weight with standard deviation,  $\sigma = 0.01$ . For class/bbox subnet, the bias is initialized as  $b = -\log((1 - \tau)\tau)$ , where  $\tau = 0.01$ . The model was trained with synchronized Stochastic gradient descent (SGD) over single GTX2080Ti GPUs with a total of 2 images per minibatch. The initial learning rate of 0.0025, weight decay of 0.0001 and momentum of 0.9 were used. The dataset was split as follows 7000 images for training, 2000 for validation and 1000 for testing.

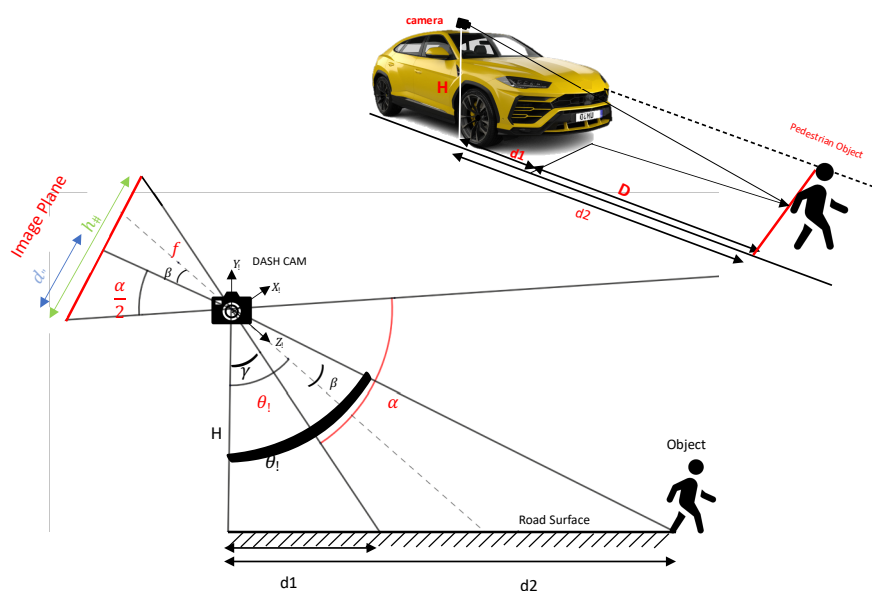
The output obtained from the Traffic RetinaNet algorithm (boundary box that illustrates the location of the object, configuration score and the object class) is used as an input for both estimation and tracking.

## 6.2.2 Object Distance Estimation

Distance estimation has received extensive attention from transportation research, where many researches were carried out to estimate the distance between road

objects and vehicles using monocular cameras, stereo cameras, sensors and lidars. (Diaz-Cabrera, Cerri, & Medici, 2015) carried a study to estimate the distance between traffic lights and vehicles. They developed a driver assistance system that uses distance estimation and tracking.

After completing detection using Traffic RetinaNet, an object filter is applied to filter pedestrians and cyclists from the other detected classes. A similar triangle-based distance estimation algorithm is used to estimate the distance of pedestrians and cyclists. The centre point of the boundary box's bottom line is considered the input for this stage.



**Figure 6.3** Distance estimation in 2D image plane.

The method for estimating the distance of objects from images captured using a dash-cam is described here. First, the camera was located toward the car's rear mirror, looking forward at a height  $H$  above the road surface, and an angle  $\alpha$ , the camera was located at an angle  $\theta_c$  in  $X_c Y_c Z_c$  coordinates as shown in Figure 6.3. Supposing the detected object on the road sense, located on an unknown position  $(X_w, Y_w, Z_w)$ .  $\theta_v$  is the angle of the projected ray (from the camera) pointing to the intersection of the planar of the detected object with the road surface planar as shown in Figure 6.3 (the top figure). The distance  $D$  is the actual distance between the vehicle and the object

that is equal to  $d_2-d_1$  and can be calculated using the following equation (Rezaei, Terauchi, & Klette, 2015; Schaffalitzky, Zisserman, Hartley, & Torr, 2000):

$$D = H * \tan(\theta_v) - H * \tan(\gamma) = H * \left[ \tan(\theta_c + \beta) - \tan\left(\theta_c - \frac{\alpha}{2}\right) \right] \quad 6.1$$

To compute  $D$ ,  $\beta$  must be calculated as both  $\theta_c$  and  $\alpha$  are known.(Nienaber, Kroon, & Booyesen, 2015; Rezaei et al., 2015):

$$\tan(\beta) = \frac{\left(\frac{h_i}{2} - d_p\right)}{f} \quad 6.2$$

where  $h_i$  is the height captured image plane (in pixel),  $d_p$  is the distance from the bottom side of the detected vehicle to the bottom of the image plane (in pixel), and  $f$  is the focal length of the camera. Where (Rezaei et al., 2015):

$$f = \frac{h_i}{2 \tan\left(\frac{\alpha}{2}\right)} \quad 6.3$$

Substituting all the parameters back to evaluate  $D$ , the following equation is obtained (Rezaei et al., 2015):

$$D = H * \left[ \tan\left(\theta_c + \tan^{-1}\left(\frac{\frac{h_i}{2} - d_p}{\frac{h_i}{2 \tan\left(\frac{\alpha}{2}\right)}}\right)\right) - \tan\left(\theta_c - \frac{\alpha}{2}\right) \right] \quad 6.4$$

Figure 6.3 (the bottom figure) illustrated distance measurement technique used.

### 6.2.3 Object Tracking Algorithm: Simple, Online and Realtime Tracking (SORT)

Simple Online and Realtime Tracking (SORT) is a method used for online and real-time tracking of objects. It tracks multiple objects simply and efficiently. SORT algorithm combines both the Kalman filter (Welch & Bishop, 1995) and the Hungarian(Kuhn, 1955) method thus to handle motion prediction and the data

association components (Bewley, Ge, Ott, Ramos, & Upcroft, 2016). In this study, the original SORT algorithm was enhanced by introducing a novel buffering mechanism that leverages the Hungarian algorithm. This approach dynamically reassigns unmatched detections (unmatched det) as new tracks and retains unmatched tracks (unmatched trks) for a specified number of iterations ( $k$ ). This innovative enhancement, not previously explored in existing literature, significantly augments the tracking capabilities of the SORT algorithm, offering a novel perspective on multi-object tracking.

In the presented work following both detection and estimation, tracking by detection using visuals only is introduced. The state of each detected object is modelled as:

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T$$

6.5

where  $u$  and  $v$  represents the vertical pixel location of the targeted object, and the other two variables  $s$  and  $r$  corresponds to scale (area) and the aspect ratio of the targeted object bounding box respectively (Bewley et al., 2016).

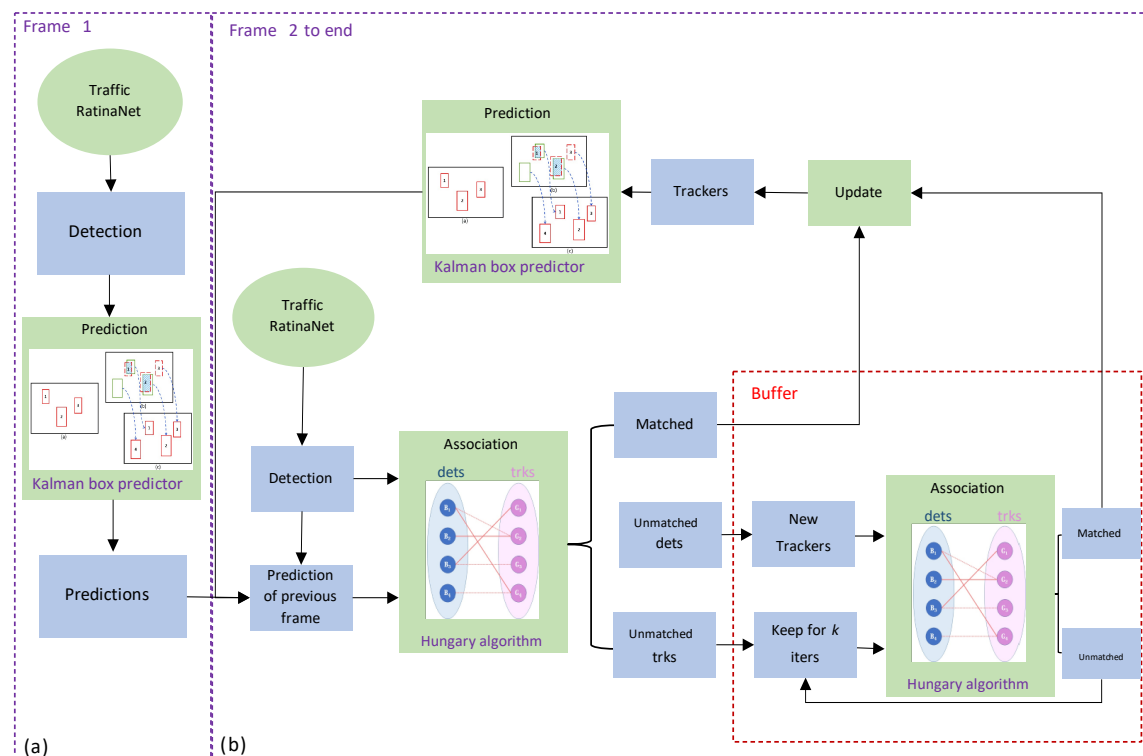
Compare the detection results with prediction results and calculate the IoU to determine whether the objects were matched. Then using the Hungarian algorithm, associates detections with tracking. Finally, update the tracker and prepare for the next frame (Ristani, Solera, Zou, Cucchiara, & Tomasi, 2016). Table 6.2 summarise the tracking processes pointing to the three classes of the associated obtained results: matched, unmatched detections, and unmatched track-let (track-let is a short track that corresponds to the motion of an object in a short period of time, typically a few frames.).

The overlapping of objects in traffic scenes can result in missing some of the overlapped objects. The original SORT algorithm failed to track these objects. To solve this issue, a simple yet powerful buffer module is proposed. The buffer is introduced after the unmatched track-lets, as shown in Figure 6.4. The buffer mainly captures the unmatched detections, which will be assigned as the new trackers, and the unmatched tracks are kept for  $k$  number of iterations these go through Hungary algorithm. This resulted in matched and unmatched objects. The matched frames update the trackers that will be fed into the Kalman box predictor, and the unmatched objects are stored

for  $k = 3$  iterations in our case before being discarded, thus checking for match objects in the next upcoming frames.

**Table 6.2** Association Status

Status	Details
Matched	Detection in frame $i$ is matched to the track-let in frame $i - 1$ .
Unmatched dets	Detection in frame $i$ can not be matched with the track-let in frame $i - 1$ .
Detection	New track-let will be created.
Unmatched trks	The track-let in frame $i - 1$ can not be matched with existing Detection.
Track-lets	This track-let will be removed.



**Figure 6.4** Framework of improved SORT

### 6.2.4 Count and Speed estimation

The number of objects in key classes and their approximate speed are estimated based on the average difference of location in 5 consecutive frames. The flow was estimated and computed using the detection results (boundary box and class) where we considered only the detected objects of both cyclist and pedestrians' class. Speed estimation carried out using tracking, where objects speed was estimated by

computing the average difference of consecutive frames, as shown in the following equation (Kumar & Kushwaha, 2016):

$$S = \delta \frac{1}{mn} \sum_D \sum_n df$$

6.6

where  $d$  is the difference in distance between consecutive frames in meter,  $f$  is the frame rate in frames/second,  $n$  is the number of tracking object (pedestrian vehicle) per frame,  $m$  is the frame pair, and  $\delta$  is a parameter introduced to convert units to (meters/second).

## 6.3 DATA

This section highlights the different datasets used for training and testing purpose and the implementation requirements.

### 6.3.1 Datasets used for training

Each module was trained separately before combining the algorithm. This was done to evaluate how well our model performs on new, unseen data. By training on a dataset, we can teach the model to recognize patterns and make accurate predictions. By testing on a different dataset, we can evaluate how well the model generalizes to new, unseen data and assess its overall performance.

#### *A) Detection*

For the detection, COCO dataset was used for training validation and testing. As mentioned in 2.5.4.1, COCO contains 80 classes, yet to be used for traffic scenes with the categories in the scope of the study. The class IDs that correspond to the objects (pedestrian, cyclist, cars, buses and tracks) to be detected is selected. The COCO dataset documentation gives the list of class IDs and their corresponding class names

(Lin et al., 2014). Once the class IDs are selected, the dataset will be filtered, and only the images and annotations corresponding to the classes of interest will be extracted. Additionally, it was trained on two traffic-related dataset Cityscapes dataset and KITTI dataset (refer to 2.5.4.2 and 2.5.4.3 for more details related to the dataset).

### ***B) Estimation***

For estimation, the field dataset (refer to 2.5.4.5) was used for training and testing, as the images captured for this dataset match the setting requirements for the estimation algorithm, and the dataset contains examples representative of the problem the model is trying to solve. For example, the model is built to estimate pedestrian and cyclist flow, and the dataset contains images of both categories. Therefore, choosing this dataset ensures that the model is trained on examples similar to the ones it will encounter in the real world, which can improve its performance. The parameters of the dash-cam were used to estimate distance using Matlab (equation 6.4) and to compare the obtained values to the ground truth values. Additionally, MOT16 was also used for evaluating the estimation algorithm.

### ***C) Tracking***

For tracking unsupervised learning (refer to Figure 6.4) method is introduced where MOT16 dataset was used for our purposes. As mentioned in 2.5.4.6, the data contains challenging scenarios, such as multiple people walking, running, and crossing paths in a crowded environment. The dataset was chosen because it is representative of the types of scenarios that tracking algorithms are likely to encounter in the real world, and it provides a standardized benchmark for evaluating tracking performance.

## **6.3.2 Dataset used for testing**

The field dataset is the dataset used for testing the whole DET algorithm; the main reason for this is that the dataset was collected and annotated for this work which contains challenging videos and images (refer to 2.5.4.5).



Analysis and evaluation of all algorithms introduced and developed are evaluated and discussed in the next section.

## 6.4 EVALUATION OF DTE

### 6.4.1 Evaluation of Traffic RetinaNet

The simple dense detector RetinaNet can work with different backbone encoders such as ResNet (He et al., 2016), ResNeXt(Hoang et al., 2019) (Hoang et al. (2019)), and DenseNet (Lin, Goyal, et al., 2017). ReNeXt was used as the backbone for Traffic RetinaNet algorithm. By applying this, we managed to increase the detection's average precision compared to the original RetinaNet, which includes a ResNet backbone (Alshkeili, Ghosh, & Qiu, 2019). Traffic RetinaNet as mentioned earlier was trained on different dataset before been tested on the field dataset. For evaluating the detection performance, AP (equation 2.29) an AR (equation 2.30) are used. Tables 6.3 and 6.5 show the results obtained by applying the proposed algorithm on COCO2017 dataset, Table 6.4 is Cityscapes dataset results, and Table 6.5 is KITTI dataset results.

**Table 6.3** Compare traffic RetinaNet to the original RetinaNet (COCO dataset)

	Traffic RetinaNet		RetinaNet	
	Average Precision	Average Recall	Average Precision	Average Recall
All	<b>0.43</b>	0.263	0.422	<b>0.264</b>
Small	<b>0.235</b>	<b>0.374</b>	0.221	0.341
Medium	0.442	<b>0.606</b>	<b>0.449</b>	0.582
Large	<b>0.614</b>	<b>0.753</b>	0.612	0.726

**Table 6.4** Compare traffic RetinaNet to the original RetinaNet (Cityscapes dataset)

	Traffic RetinaNet		RetinaNet	
	Average Precision	Average Recall	Average Precision	Average Recall
All	<b>0.367</b>	<b>0.456</b>	0.358	0.455
Small	<b>0.155</b>	<b>0.213</b>	0.143	0.210
Medium	<b>0.358</b>	<b>0.441</b>	0.353	0.438
Large	<b>0.559</b>	<b>0.67</b>	0.549	0.60

For this application, small stands for objects had a pixel area of  $< 32^2$  pixels. Medium stands for objects with an area range of  $32^2 < area < 96^2$  pixels. Large stands for objects with an area of  $96^2 < area$  pixels. The object detection results showed that both the AP

and AR values improved for Traffic RetinaNet except in medium objects. AP and AR were evaluated under IoU = 0.5 : 0.05 : 0.95 with AP: MaxDets = 100 (given 100 detection 100 image), AR: MaxDets = 1 (given 1 detection per image). IoU is the Intersection over Union of the boundary boxes.

**Table 6.5** Compare traffic RetinaNet to the original RetinaNet (KITTI dataset)

	Traffic RetinaNet		RetinaNet	
	Average Precision	Average Recall	Average Precision	Average Recall
Car	<b>0.962</b>	0.983	0.902	<b>0.984</b>
Pedestrian	<b>0.820</b>	<b>0.928</b>	0.630	0.859
Cyclist	<b>0.863</b>	<b>0.962</b>	0.555	0.827
mAP	<b>0.882</b>		0.696	

As illustrated in Tables 6.3 and 6.4, the Traffic RetinaNet, which focuses more on traffic-related objects, surpasses the original RetinaNet both in precision and recall. Especially for recall, Traffic RetinaNet achieved higher performance, which means Traffic RetinaNet can detect more traffic objects, for instance, in Table 6.5 where KITTI dataset is used for training and testing purposes, because it is a traffic based dataset, proved the efficiency and reliability of the proposed Traffic RetinaNet algorithm for road object detection that is significant for self-driving cars.

Table 6.6 shows the precision and recall for each category of objects detected. Traffic RetinaNet achieves good results on pedestrians and buses, reasonable for cars and motorcycles but limited for cyclists. The main reason for this is the imbalance of training data in the COCO2017 dataset, as it is not designed for this specific purpose of object detection from moving vehicles. However, due to the lack of an appropriately labelled dataset for evaluating these algorithms, it was prudent to use a well-known stock video such as COCO2017.

**Table 6.6** Average percentage and average recall of the different categories

Detected objects	All	Average Precision		
		Small	Medium	Large
Pedestrian	0.513	0.335	0.593	0.689
Cyclist	0.263	0.155	0.322	0.506
Car	0.393	0.301	0.549	0.563
Motorcycle	0.384	0.215	0.34	0.55
Bus	0.595	0.17	0.4	0.763
Detected objects		Average Recall		

	All	Small	Medium	Large
Pedestrian	0.185	0.475	0.685	0.776
Cyclist	0.223	0.268	0.5	0.0718
Car	0.172	0.459	0.687	0.756
Motorcycle	0.251	0.338	0.509	0.675
Bus	0.481	0.331	0.649	0.843

The detection model was also tested on MOT16 dataset, in Figure 6.5, a chosen set of scenes detecting all object types for illustrative purposes. The algorithm successfully detected different traffic object classes in the same scene, as shown in the images. Traffic RetinaNet successfully classifies the different objects from different angles and distances, which is crucial for analyzing dash-cam footage from a moving vehicle where the angles and distance are uncontrollable. The objects were detected both in shadows and in illuminated areas of the same scene. Additionally, occlusion effects were minimized as a large number of bbox were identified in images in multiple objects.



**Figure 6.5** Detection of multiple classes of objects in MOT16 dataset.

### 6.4.2 Evaluation of Tracking

The tracking element was vital in estimating traffic parameters from the video. The number of pedestrians, vehicles, and their average speed relative to the moving vehicle was estimated using this algorithm.

In traffic scenes, the object's scale is changing, so scale-insensitivity is crucial for the tracker. We chose SORT as it tracks objects only depending on the IoU region of objects, robust to object size. Table 6.7 shows the results of the improved SORT described earlier. A variety of evaluation matrices is widely used in MOT to evaluate the improved algorithm. The specific evaluation indicators are shown in the Table 6.8 (Welch & Bishop, 1995).

**Table 6.7** Comparing results of the original SORT to the improved.

	IDF1	IDP	IDR	GT	MT	PT	ML	MOT A	MOTP
SORT	44.10 %	58.20 %	35.50 %	50 0	11 2	22 4	16 4	32.9	73. 7
Improve d	<b>47.10</b> %	56.90 %	<b>40.20</b> %	50 0	<b>118</b>	<b>236</b>	<b>146</b>	<b>39.8</b>	72. 8

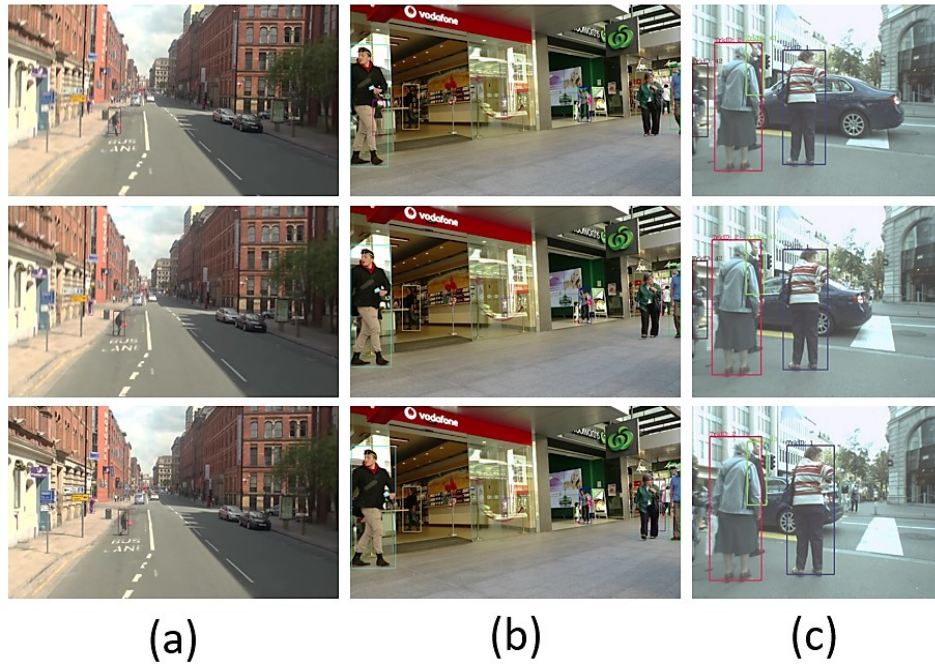
Additionally, statistical estimation of the videos used presented in the following Table 6.9. Figure 6.6 shows the result of tracking algorithm, where the distance and robustness of tracking vary between a) and c); thus, it shows the accuracy of the model.

**Table 6.8** MOT evaluation matrix

Metric	Description	Note
IDF1	the ratio of correctly identified detections over the average number of ground-truth and computed detections	↑
IDP	identification precision	↑
IDR	identification recall	↑
GT	Total Number	-
MT	Number of objects tracked for at least 80 percent of lifespan.	↑
PT	Number of objects tracked between 20 and 80 percent of lifespan.	↑
ML	Number of objects tracked less than 20 percent of lifespan.	↓
MOTA	Multiple object tracker accuracy.	↑
MOTP	Multiple object tracker precision.	↑

**Table 6.9** Statistical Estimation Results of Videos

Scene	(a)	(b)	(c)
Length (frames)	750	525	837
FPS	25	30	14
Av. Number Pedsetrian	14.7	14	8.7
Vehicle	9.8	0	0.7
Av. Speed Pedsetrian	3.48 m/s	1.7 m/s	1.87 m/s
Vehicle	12.5 km/h	0 km/h	3 km/h



**Figure 6.6** Tracking results in multiple scenes over three consecutive frames.

It is essential to estimate any traffic object-detection algorithms' computational costs to establish whether real-time detection is plausible. FLOPs (equation 2.36) is how fast the microprocessor operates; it has a performance unit of the multiplier-accumulator (Mac). Table 6.10 shows the performance parameters and FLOPs of the framework.

**Table 6.10** Computational performance of the framework

FLOPs:286.83 GigaMAC	Parameters: 54.86 Million	
	Training	Testing
Detection using Traffic RetinaNet	3 days 18 h	9 Frames/Sec
Tracking	-	263 Frames/Sec

The detection is at the rate of 9 fps, which is slower than a video rate of 30 fps. However, this rate can be considered real-time for vehicles and pedestrians on urban signalised roads from a relative movement point of view, due to several reasons such as:

- **Relative movement speed:** In urban traffic scenarios, the movement speed of vehicles and pedestrians is relatively slow compared to the frame rate of 9 fps.

This means that even with a slightly lower detection rate, the system can effectively capture and process the relative movement of objects, allowing it to track and respond to potential hazards in a timely manner. The average speed of a 3 m vehicle in urban area is 30-50 km/h, pedestrian travels at an average speed of 4 km/h and cyclist travels at an average speed of 15km/h. Therefore the movement of at 30 fps is: for vehicle 0.5m/s, pedestrians 0.04m/s, and cyclist 0.14m/s. Where the movement at 9 fps is: for vehicles 1.5m/s, pedestrians 0.1m/s and cyclist 0.5m/s. From a pedestrian and cyclist perspective the movement is negligible.

- Safety and decision time: 9 fps detection rate provides sufficient time for decision-making and taking corrective actions to ensure safety on signalized urban roads. Real-time decisions can be made to avoid accidents or conflicts for pedestrians and cyclist.

The tracking rate is much higher than the video rate and is compatible with the advanced alarm of collision avoidance requirements for an autonomous vehicle.

#### **6.4.3 Distance, Speed and Flow estimation**

Figure 6.7 present the result obtained when evaluating the speed distance and flow estimation on MOT16 dataset for illustrative purposes. Where we considered three different frames, the top left image (Figure 6.7 (a)) shows the estimated number of pedestrians ( $Q_p$ ) presented in that frame and the estimated number of vehicles ( $Q_v$ ), whereas the top right image (Figure 6.7 (a)) present the average flow of pedestrian ( $V_p$ ) and vehicles ( $V_v$ ). The two other frames, (b) and (c), present the same information. However, it can be noticed that in Figure 6.7(b), no vehicle is presented, and the algorithm was successful in detecting and set the estimation of the number of the vehicle presented and flow to 0, which mean no vehicle was found.

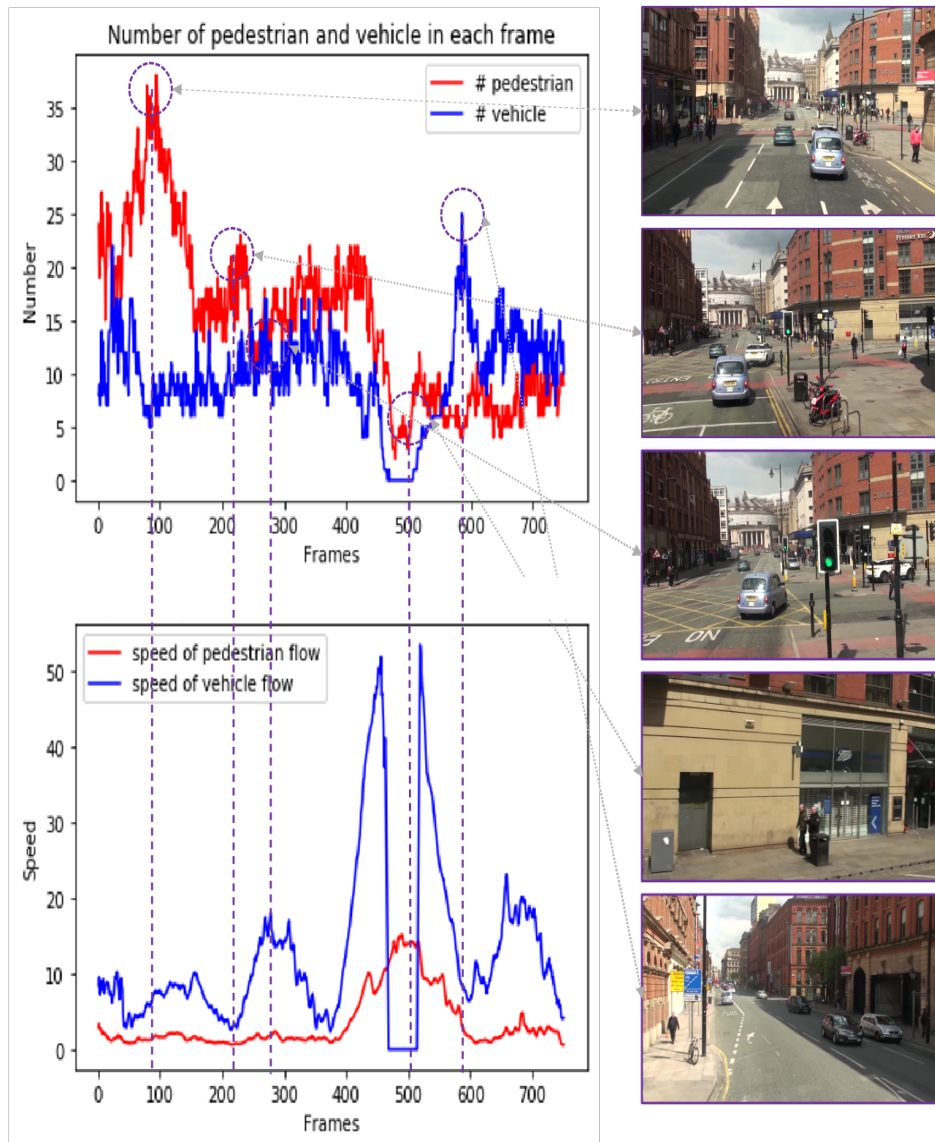




**Figure 6.7** Flow and Speed estimation.

Furthermore, Figure 6.8 shows the flow of pedestrians and vehicles per frame. The curves' tendency illustrates the object speed note that there is a gap around frame 480, as shown by the blue curves; no vehicles exist in that period. Additionally, the two peaks shown in the figure illustrates the highest recorded number of either vehicles or pedestrian during the recorded period. The first frame on the right illustrates the traffic density where a peak of pedestrian is detected at frame 97 with more than 35 pedestrians. In frame 596 the blue curve illustrates a maximum number of vehicles where it recorded 24 automobiles using the road at that specific frame, the fourth figure is the visual representation.



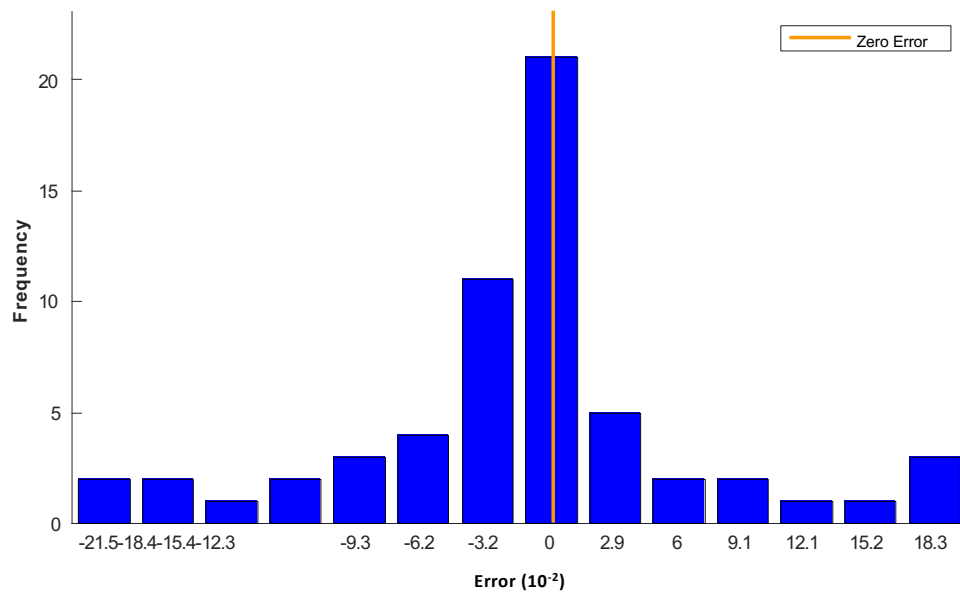


**Figure 6.8** Number and Speed of Pedestrian and Vehicle in Scene (a).

#### 6.4.4 Evaluating DET on the field data

Starting with the distance estimation, where an error histogram illustrated in the following Figure 6.9 where the distance to vehicle errors, defined by comparing with ground truth represented by the red line. Objects are at a distance of 2 to 25m to the dash camera. We considered a confidence interval of  $\pm 20cm$  for ground truth measurement. The error level lies between  $\pm 20\%$ , due to lack of availability of reported uncertainty error values in literature we were unable to compare it with

the state of art. Considering the movement of pedestrians and cyclist which is less than 1m between frames this error uncertainty can be considered as acceptable.



**Figure 6.9** Percentage error of distance estimation.

Moving to the result obtained from combining the different algorithms to finally produce the DTE, where Figure 6.12 presents an evaluation of the enter algorithm applied to our data. (a) illustrates the detection and distance estimation results, (b) shows both vehicle and pedestrian estimated flow.

(a)



(b)



Estimated speed of person flow:  $\sim 3.15$  m/s  
Estimated relative speed of vehicle flow:  $\sim 3.85$  km/h

Figure 6.10 Detection, Tracking & Estimation result.

## 6.5 DISCUSSION & CONCLUSION

In this chapter, traffic detection, distance/parameter estimation, and tracking framework are presented utilizing vision-based analysis for autonomous driving. The framework considered a new object detection algorithm named Traffic RetinaNet, which simultaneously focused on five different moving object classes, including pedestrians, cyclists, cars, buses, and motorcycles. The detection precision of Traffic-RetinaNet was higher compared to the traditional RetinaNet when applied to the MOT16 dataset. The second part of the framework was pedestrians' distance estimating, in which the estimation error was less than 20%. Tracking was also successful, in which results obtained showed a track improvement compared to the original SROT method. The final part of the framework was to combine the success estimate of the distance and parameters tracked by the detected objects to estimate traffic parameters such as volume and speed. The framework's performance succeeded in different light conditions, changes of scenes due to the moving frame of reference, angles and relative distances, and crowded environments (occlusion). The framework proved that combining Traffic RetinaNet and the improved SORT will provide an advanced information provision for self-driving vehicles at a reduced cost due to using a single-camera source.

---

# Chapter 7

---

# 7 Conclusion

The main research objective of this thesis was to increase the safety of pedestrians and cyclists using object detection algorithm. This has been achieved by better understanding peoples concerns regarding AV, analysing detection performance of the different algorithms across datasets, developing benchmark dataset that focuses on detecting objects under different weather, light and driving conditions, and developing a detection tracking estimation algorithm that aims to enhance real-time object detection estimation and tracking.

This chapter provides a summary of the primary contributions made in this thesis. It critically evaluates the conducted research, and it suggests potential directions for future research.

## 7.1 RESEARCH CONTRIBUTIONS

The main contributions of this thesis can be summarized into the following: exploring the impact of AV and their abilities from a computer-vision-based perspective. The abilities of the AVs to recognize road objects, such as pedestrians, cyclists, vehicles etc., this was explored using a computer-vision-based approach to sensing. The impacts of the presence of these vehicles in the traffic flow were explored. As outlined in the introduction, autonomous vehicles were integrated into the transportation fleet for their safety features, as they contribute to accident reduction by minimizing human errors. In order to implement the use of AVs across the transportation fleet, user persecution was analysed using before and after TAM. Following the TAM analysis, technical evaluation and analysis of the state-of-the-art algorithm was carried out, and cross-validation was conducted to examine those models. A condition-based dataset was then developed and validated to investigate the impact of different weather, visibility and traffic conditions. Finally, a combined detection, tracking and estimation algorithm was proposed for real-time application.

The experimental chapters started by examining the acceptance of Autonomous Vehicles through the implementation of a TAM, aligning with the primary objective outlined in the thesis. Chapter 3 conducted a unique TAM (Technology Acceptance

Model) study to analyze participant acceptance of autonomous vehicles both before and after interacting with the technology. The online survey received more than 300 responses. While numerous studies explore AV adaptation, the specific examination of attitudes toward technology use before and after interaction remains unexplored in the literature. This study significantly contributes to enhancing our understanding of participant perspectives on using AVs. The research evaluated the relationship between “Perceived Ease of Use” and “Perceived Usefulness” before and after interacting with AVs, revealing a positive improvement in participant perceptions post-interaction. Moreover, participants expressed increased confidence and a heightened sense of safety while commuting with AVs. However, concerns about the safety of road users persist. In relation to our research objective, this survey provides valuable insights for researchers and developers, offering a deeper understanding of user concerns and apprehensions in the adoption and use of AVs.

Based on Chapter 3 results, the subsequent development of Chapter 4 is intricately tied to the second thesis objective outlined in Chapter 1 (section 1.3). This chapter contributes novel insights to the existing research field by evaluating different algorithms across multiple datasets. The comparative analysis of performance not only sheds light on the algorithms' generalization capabilities but also assesses their readiness for real-time applications, adding depth to the understanding of their practical applicability. The five different object detection namely —RetinaNet, FCOS, Faster RCNN, cascade RCNN and deformable DETR—were chosen for image-based object detection based on their uniqueness and performance. A comprehensive evaluation of the models used on the different datasets (Cityperson, KITTI and ECP) was presented. Each dataset has a unique structure and complexity level, as mentioned earlier (Chapter 2 and Chapter 4). Algorithms performed the best on KITTI dataset achieving the highest accuracy across the five algorithms. This is due to the fact that KITTI dataset was captured in one season, with balanced class distribution and high-quality annotation. However, during cross-validation, while KITTI achieved a higher average precision value, it failed to attain high accuracy when tested on unseen datasets. Issues such as class mismatch, overfitting, and a lack of robustness contributed to the inability to achieve high detection values.

Training and testing machine learning models on large datasets is advisable to prevent overfitting and enhance accuracy. Overfitting, stemming from a model being

overly complex and closely tailored to training data, can lead to poor performance on new data. Larger datasets facilitate capturing underlying patterns, improving prediction accuracy, and aiding in the identification of rare or crucial cases. Nevertheless, in some situations, for instance, in the case of this thesis, testing on a smaller dataset resulted in better accuracy. Additionally, there are situations where training and testing on small datasets may be necessary or desirable. For example, if the available data is limited or working with a specialized domain where data is scarce, then working with a small dataset is needed. In these cases, it's important to use techniques like cross-validation to make the most of the data and to evaluate the model's performance carefully. This conclusion is also implied by the collated benchmark dataset in Chapter 5.

Chapter 5 is dedicated to the development of a benchmark dataset, specifically designed to account for various weather, lighting, and driving conditions, aligning with the third research objective. The uniqueness of the benchmark dataset used for testing and training in Chapter 5 is that it combines four different datasets. Each was captured and annotated differently. For this, it required many modifications to create one unified dataset that focuses on the different weather conditions that affect driving behaviours. The experiment concluded that one-stage detection algorithm requires less computational time and lower complexity. Therefore, it can be combined with different algorithms to achieve higher system accuracy and serve the purpose of real-time application.

In Chapter 6, the fourth thesis objective is addressed, focusing on the utilization of a unified end-to-end algorithm for detection, tracking, and estimation. This thesis introduces a novel algorithm designed for real-time implementation. The chapter details the creation of a combined detection and tracking estimation algorithm, using ResNext as the backbone of RetinaNet. This modification to the detection algorithm yielded heightened detection accuracy. The developed algorithm, referred to as TRN, underwent testing across various models and datasets, demonstrating promising usability. While the model showed high accuracy on multiple datasets, further modification is required for enhancing tracking precision for even better results. The proposed Improved SORT algorithm significantly elevated tracking performance,



though further refinement is needed for improved accuracy. Overall, development and testing of similar algorithms will be essential for effective real time implementation. Combining the three algorithms' results showed that the proposed algorithm successfully provided an advanced information provision for self-driving vehicles at a reduced cost due to using a single-camera source.

## **7.2 RESEARCH LIMITATIONS**

During the last two-year Covid-19 went viral and spread very fast; countries were locked for more than nine months, which affected the research process. The survey was initially planned to be carried out in workshops yet to gather an interactive view and to understand people's acceptance of AV better; however, due to the lockdown, it was completed online. In addition, the GPU capacity, where some tests were reformulated to match the available capacity. The limited number of data collected for the different categories affected the performance. Moreover, the limitations when training on several deep learning models can be summarized as follow:

- **Computational resources:** As mentioned earlier, training deep learning models can be computationally expensive, especially if large datasets are used. This can limit the number of models that can be trained and the number of hyperparameters that can be tuned.
- **Time constraints:** Training the different deep learning models can also be time-consuming, especially if large datasets are used; for instance, training ECP on the different models requires two weeks. This can limit the number of models that can be trained and the amount of time that can be spent tuning hyperparameters.
- **Overfitting:** When training deep learning models, avoiding overfitting the training data is important. This can be challenging, especially if the models are complex or the training data is not diverse enough.
- **Data quality:** The quality of the data used to train deep learning models can also be a limitation. The robustness of a dataset can affect the efficiency of model learning.

- Bias: Deep learning models can also be biased, which can result in unfair or inaccurate predictions. This can be a limitation, especially if the models are used in sensitive applications.

### **7.3 FUTURE RESEARCH DIRECTION**

This thesis investigates individuals' adoption of AV, the different detection algorithm in the autonomous vehicle context and a combined detection tracking estimation method that enable AV to move safely and freely in the surrounding. This thesis also developed a benchmark dataset that investigates the different driving weather conditions. For future research, the benchmark dataset needs some modification for the dataset to be ready for urban traffic detection problems.

With respect to the findings in Chapter 4, future research should prioritize increasing the participation survey to evaluate perceptions comprehensively. A more diverse and extensive participant base will provide nuanced insights into various perspectives on AV adoption, ensuring a holistic understanding of user attitudes and concerns.

Additionally, there is a need to expand the algorithm testing framework to include 3D bounding boxes. While traditional detection algorithms predominantly operate in two-dimensional space, the adoption of 3D bounding boxes introduces an added layer of sophistication that aligns more closely with real-world scenarios. This enhancement holds particular significance in the context of urban traffic, where the dynamic and multi-dimensional nature of the environment necessitates a more nuanced perception system for autonomous vehicles (AVs). The incorporation of 3D bounding boxes in detection algorithms enables a more accurate representation of objects in the AV's surroundings. This is especially crucial in urban settings, where vehicles, pedestrians, and other obstacles can occupy multiple planes simultaneously. By extending the algorithmic evaluation to include 3D bounding boxes, researchers can better simulate and address the intricacies of real-world traffic scenarios, ensuring that detection algorithms are equipped to navigate complex spatial relationships and provide more reliable information to AV systems.

Moreover, the proposed combined detection-tracking-estimation (DTE) method can be further tested in real-time scenarios. Real-world testing will validate the effectiveness and reliability of the DTE algorithm in dynamic and unpredictable traffic conditions, contributing to its practical implementation in AV systems.

For the benchmark dataset to be a global asset, that is by expanded and made openly accessible. This inclusivity fosters collaboration and allows experts from different regions to contribute their insights and expertise, enriching the dataset with a multitude of perspectives. Open access also promotes transparency in research methodologies and data collection processes, fostering a collective understanding and trust within the research community.

Enabling worldwide training and testing on regional data is a pivotal step toward creating adaptable and effective autonomous vehicle (AV) models tailored to specific geographic nuances. AVs operating in diverse regions encounter unique challenges, such as varying traffic patterns, road infrastructure, and cultural factors. By allowing researchers and developers worldwide to train and test their models on regional data from the benchmark dataset, the resulting AV algorithms can be fine-tuned to better navigate the intricacies of specific locations.

The future research trajectory focuses on refining and expanding the benchmark dataset, incorporating more participants for a comprehensive perception evaluation, testing algorithms for 3D bounding boxes, and conducting real-time assessments of the proposed combined DTE method. These endeavours will collectively contribute to advancing the understanding and implementation of autonomous vehicles in diverse and dynamic traffic environments.

# Bibliography

- Aber, J. S., Marzolf, I., Ries, J. B., & Aber, S. E. W. (2019). Digital Image Processing and Analysis. In *Small-Format Aerial Photography and UAS Imagery* (pp. 191–221). <https://doi.org/10.1016/b978-0-12-812942-5.00011-2>
- Acheampong, R. A., & Cugurullo, F. (2019). Capturing the behavioural determinants behind the adoption of autonomous vehicles: Conceptual frameworks and measurement models to predict public transport, sharing and ownership trends of self-driving cars. *Transportation Research Part F: Traffic Psychology and Behaviour*, *62*, 349–375. <https://doi.org/10.1016/j.trf.2019.01.009>
- Ackar, H., Almisreb, A. A., & Saleh, M. A. (2019). A Review on Image Enhancement Techniques. *Southeast Europe Journal of Soft Computing*, *8*(1). <https://doi.org/10.21533/scjournal.v8i1.175>
- Agnihotri, A., Saraf, P., & Rajesh Bapnad, K. (2019). A Convolutional Neural Network Approach Towards Self-Driving Cars. *2019 IEEE 16th India Council International Conference (INDICON)*, 1–4.
- Al-Refai, G., & Al-Refai, M. (2020). Road Object Detection using Yolov3 and Kitti Dataset. *IJACSA) International Journal of Advanced Computer Science and Applications*, *11*(8). Retrieved from [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org)
- Alexe, B., Deselaers, T., & Ferrari, V. (2012). Measuring the Objectness of Image Windows. *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, *34*, 2189–2202. <https://doi.org/10.1109/TPAMI.2012.28>
- Ali, M. S., Bin Iqbal, M. T., Lee, K. H., Muqet, A., Lee, S., Kim, L., & Bae, S. H. (2020). ErDNN: Error-resilient deep neural networks with a new error correction layer and piece-wise rectified linear unit. *IEEE Access*, *8*, 158702–158711. <https://doi.org/10.1109/ACCESS.2020.3017211>
- Alshkeili, A., Ghosh, B., & Qiu, W. (2019). *Cyclist and Pedestrian in Autonomous Vehicles ' View*.
- Amara, D. K., Karthika, R., & Soman, K. P. (2020). DeepTrackNet: Camera Based End to End Deep Learning Framework for Real Time Detection, Localization and Tracking for Autonomous Vehicles. *Advances in Intelligent Systems and Computing*, *1039*, 299–307. [https://doi.org/10.1007/978-3-030-30465-2\\_34](https://doi.org/10.1007/978-3-030-30465-2_34)
- Aneesh, A. N., Shine, L., Pradeep, R., & Sajith, V. (2019). Real-time Traffic Light

- Detection and Recognition based on Deep RetinaNet for Self Driving Cars. *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 1554–1557.  
<https://doi.org/10.1109/ICICICT46008.2019.8993293>
- Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., & Malik, J. (2014). Multiscale combinatorial grouping. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 328–335.  
<https://doi.org/10.1109/CVPR.2014.49>
- Babak, S. J., Hussain, S. A., Karakas, B., & Cetin, S. (2017). Control of autonomous ground vehicles: A brief technical review. *IOP Conference Series: Materials Science and Engineering*, 224(1), 012029. <https://doi.org/10.1088/1757-899X/224/1/012029>
- Babanne, V., Mahajan, N. S., Sharma, R. L., & Gargate, P. P. (2019). Machine learning based Smart Surveillance System; Machine learning based Smart Surveillance System. *2019 Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 84–86.
- Banham, M. R., & Katsaggelos, A. K. (1997). Digital image restoration. *IEEE Signal Processing Magazine*, 14(2), 24–41. <https://doi.org/10.1109/79.581363>
- Bansal, P., & Kockelman, K. M. (2017). Forecasting Americans' long-term adoption of connected and autonomous vehicle technologies. *Transportation Research Part A: Policy and Practice*, 95, 49–63. <https://doi.org/10.1016/j.tra.2016.10.013>
- Bansal, P., Kockelman, K. M., & Singh, A. (2016). Assessing public opinions of and interest in new vehicle technologies: An Austin perspective. *Transportation Research Part C: Emerging Technologies*, 67, 1–14.  
<https://doi.org/10.1016/j.trc.2016.01.019>
- Barrett, G., Rageade, J. P., Wallis, D., & Weil, H. (2021). The future of legal Europe: Will we trust in it? In *The Future of Legal Europe: Will We Trust in It?: Liber Amicorum in Honour of Wolfgang Heusel*. <https://doi.org/10.1007/978-3-030-68253-8/COVER>
- Benenson, R., Omran, M., Hosang, J., & Schiele, B. (2015). Ten years of pedestrian detection, what have we learned? *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8926, 613–627. <https://doi.org/10.1007/978-3-319-16181->

- Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. *Proceedings - International Conference on Image Processing, ICIP, 2016-August*, 3464–3468. <https://doi.org/10.1109/ICIP.2016.7533003>
- Bianco Levrin Giovanna Larini, F. (2017). *D3.2 Report on the state of the art of connected and automated driving in Europe (Final)*.
- Blaschke, T. (2010, January 1). Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 65, pp. 2–16. <https://doi.org/10.1016/j.isprsjprs.2009.06.004>
- Bo Bo, N., Slembrouck, M., Veelaert, P., & Philips, W. (2020). Distributed Multi-class Road User Tracking in Multi-camera Network For Smart Traffic Applications. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12002 LNCS, 517–528. [https://doi.org/10.1007/978-3-030-40605-9\\_44](https://doi.org/10.1007/978-3-030-40605-9_44)
- Braun, M., Krebs, S., Flohr, F., & Gavrila, D. M. (2019). The EuroCity Persons Dataset: A Novel Benchmark for Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41, 1844–1861. <https://doi.org/10.1109/TPAMI.2019.2897684>
- Brazil, G., & Liu, X. (2020). *Pedestrian Detection With Autoregressive Network Phases*. 7224–7233. <https://doi.org/10.1109/cvpr.2019.00740>
- Brell, T., Philipsen, R., & Ziefle, M. (2019). sCARY! Risk Perceptions in Autonomous Driving: The Influence of Experience on Perceived Benefits and Barriers. *Risk Analysis*, 39(2), 342–357. <https://doi.org/10.1111/risa.13190>
- Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300, 17–33. <https://doi.org/10.1016/j.neucom.2018.01.092>
- Bubeníková, E., Muzikářová, L., & Halgaš, J. (2012). Application of image processing in intelligent transport systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 11(PART 1), 53–56. <https://doi.org/10.3182/20120523-3-cz-3015.00012>
- Caesar, H., Uijlings, J., & Ferrari, V. (2018). COCO-Stuff: Thing and Stuff Classes in Context. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1209–1218. Retrieved from <http://calvin.inf.ed.ac>.
- Cai, Z., & Vasconcelos, N. (2017). Cascade R-CNN: Delving into High Quality Object

- Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6154–6162. Retrieved from <https://arxiv.org/abs/1712.00726v1>
- Campbell, M., Egerstedt, M., How, J. P., & Murray, R. M. (2010). Autonomous driving in urban environments: approaches, lessons and challenges. *Trans. R. Soc. A*, 368, 4649–4672. <https://doi.org/10.1098/rsta.2010.0110>
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12346 LNCS, 213–229. Retrieved from <http://arxiv.org/abs/2005.12872>
- Chaitanya, K., Karani, N., Baumgartner, C. F., Erdil, E., Becker, A., Donati, O., & Konukoglu, E. (2020). Semi-supervised Task-driven Data Augmentation for Medical Image Segmentation. *Medical Image Analysis*, 68. Retrieved from <http://arxiv.org/abs/2007.05363>
- Chen, K., Lui, L. M., & Modersitzki, J. (2019). Image and surface registration. In *Handbook of Numerical Analysis* (Vol. 20, pp. 579–611). <https://doi.org/10.1016/bs.hna.2019.07.001>
- Chen, L.-C., Lopes, R. G., Cheng, B., Collins, M. D., Cubuk, E. D., Zoph, B., ... Shlens, J. (2020). Naive-Student: Leveraging Semi-Supervised Learning in Video Sequences for Urban Scene Segmentation. *Computer Vision--ECCV 2020: 16th European Conference, Glasgow, UK, August 23--28, 2020, Proceedings, Part IX 16*, 695–714.
- Chen, L.-C., Wang, H., & Qiao, S. (2020). Scaling Wide Residual Networks for Panoptic Segmentation. *ArXiv Preprint ArXiv:2011.11675*.
- Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H., & Chen, L.-C. (2019). *Panoptic-DeepLab*.
- Chuttur, M. (2009). Overview of the Technology Acceptance Model: Origins, Developments and Future Directions. *All Sprouts Content*, 9(37). Retrieved from [https://aisel.aisnet.org/sprouts\\_all/290](https://aisel.aisnet.org/sprouts_all/290)
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... Schiele, B. (2016). *The Cityscapes Dataset for Semantic Urban Scene Understanding*. Retrieved from <http://arxiv.org/abs/1604.01685>

- Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. *Cognitive Technologies*, 21–49. [https://doi.org/10.1007/978-3-540-75171-7\\_2](https://doi.org/10.1007/978-3-540-75171-7_2)
- da Silva, E. A. B., & Mendonca, G. V. (2005). Digital Image Processing. In *The Electrical Engineering Handbook* (pp. 891–910). <https://doi.org/10.1016/B978-012170960-0/50064-5>
- Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. *Advances in Neural Information Processing Systems*, 29. Retrieved from <https://github.com/daijifeng001/r-fcn>.
- Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, I*, 886–893. <https://doi.org/10.1109/CVPR.2005.177>
- Davis, F. D., Bagozzi, R., & Warshaw, P. (1989). *User Acceptance of Computer Technology: A Comparison of Two Theoretical Models Reshoring from a demand-side perspective View project household food waste minimization View project*. 35(8), 982–1003. <https://doi.org/10.1287/mnsc.35.8.982>
- De Brabandere, B., & Neven, D. (2017). Semantic Instance Segmentation with a Discriminative Loss Function. *ArXiv Preprint ArXiv:1708.02551*.
- Deb, S., Strawderman, L., Carruth, D. W., DuBien, J., Smith, B., & Garrison, T. M. (2017). Development and validation of a questionnaire to assess pedestrian receptivity toward fully autonomous vehicles. *Transportation Research Part C: Emerging Technologies*, 84, 178–195. <https://doi.org/10.1016/j.trc.2017.08.029>
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. Retrieved from <http://www.image-net.org>.
- Descombes, X. (2018). Markov models and MCMC algorithms in image processing. In *Academic Press Library in Signal Processing: Image and Video Processing and Analysis and Computer Vision* (Vol. 6, pp. 305–344). <https://doi.org/10.1016/B978-0-12-811889-4.00008-7>
- Diaz-Cabrera, M., Cerri, P., & Medici, P. (2015). Robust real-time traffic light detection and distance estimation using a single camera. *Expert Systems with Applications*, 42(8), 3911–3923. <https://doi.org/10.1016/J.ESWA.2014.12.037>



- Dollár, P., Wojek, C., Schiele, B., & Perona, P. (n.d.). *Pedestrian Detection: A Benchmark*. Retrieved from [www.vision.caltech.edu/Image](http://www.vision.caltech.edu/Image)
- Dominguez, J. M. L., Sanguino, T. D. J. M., Veliz, D. M., & Gonzalez, I. J. F. D. V. (2020). Multi-Objective Decision Support System for Intelligent Road Signaling. *Iberian Conference on Information Systems and Technologies, CISTI, 2020-June*.  
<https://doi.org/10.23919/CISTI49556.2020.9141083>
- Dong, Q., Zhu, X., & Gong, S. (2019). Single-label multi-class image classification by deep logistic regression. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, (2), 3486–3493. <https://doi.org/10.1609/aaai.v33i01.33013486>
- Du, F., Zhu, A., Liu, J., & Yang, L. (2020). Predictive mapping with small field sample data using semi-supervised machine learning. *Transactions in GIS*, 24(2), 315–331. <https://doi.org/10.1111/tgis.12598>
- Feng, M., Liu, Y., Jiang, P., & Wang, J. (2020). Object Detection and Localization Based on Binocular Vision for Autonomous Vehicles. *Journal of Physics: Conference Series*, 1544, 12134. <https://doi.org/10.1088/1742-6596/1544/1/012134>
- Figueiredo, M. A. T., & Jain, A. K. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3), 381–396. <https://doi.org/10.1109/34.990138>
- Fishbein, M., & Ajzen, I. (1977). Attitude-behavior relations: A theoretical analysis and review of empirical research. *Psychological Bulletin*, 84, 888.
- Force, E. T. (2015). Automated Driving Roadmap Status: final for publication. *European Road Transport Research Advisory Council: Brussel, Belgium*.
- Gad, A. F., & John, S. (2018). Practical Computer Vision Applications Using Deep Learning with CNNs. In *Springer*. <https://doi.org/10.1007/978-1-4842-4167-7>
- Galatsanos, N. P., Wernick, M. N., Katsaggelos, A. K., & Molina, R. (2005). Multichannel Image Recovery. In *Handbook of Image and Video Processing* (pp. 203–217). <https://doi.org/10.1016/B978-012119792-6/50076-0>
- Gao, P., Tian, T., Zhao, T., Li, L., Zhang, N., & Tian, J. (2022). Double FCOS: A Two-Stage Model Utilizing FCOS for Vehicle Detection in Various Remote Sensing Scenes. *IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING*, 15, 4730–4743.

- <https://doi.org/10.1109/JSTARS.2022.3181594>
- Gaspar, F., Guerreiro, V., Loureiro, P., Costa, P., Mendes, S., & Rabadão, C. (2020). Prototype to Increase Crosswalk Safety by Integrating Computer Vision with ITS-G5 Technologies. *Information*, 11(11), 503.  
<https://doi.org/10.3390/info11110503>
- Gavrila, D. M. (2000). *Pedestrian Detection from a Moving Vehicle*. 37–49. Dublin: Proc. of the European Conference on Computer Vision.
- Gavulová, A., Pirník, R., & Hudec, R. (2011). Technical support of traffic control system of Slovak agglomerations in NaTIS project. *Communications in Computer and Information Science*, 239 CCIS, 382–391. [https://doi.org/10.1007/978-3-642-24660-9\\_44](https://doi.org/10.1007/978-3-642-24660-9_44)
- Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research*, 32(11), 1231–1237. Retrieved from <http://www.cvlibs.net/datasets/kitti>.
- Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3354–3361.  
<https://doi.org/10.1109/CVPR.2012.6248074>
- Gerónimo, D., López, A. M., Sappa, A. D., & Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7), 1239–1258.  
<https://doi.org/10.1109/TPAMI.2009.122>
- Gidaris, S., & Komodakis, N. (2016). Attend Refine Repeat: Active Box Proposal Generation via In-Out Localization. *ArXiv Preprint ArXiv:1606.04446*.
- Girshick, R. (2015a). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 1440–1448. <https://doi.org/10.1109/ICCV.2015.169>
- Girshick, R. (2015b). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440–1448. <https://doi.org/10.1109/ICCV.2015.169>
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 580–587. <https://doi.org/10.1109/CVPR.2014.81>
- Goda, K., & Jalali, B. (2013, February). Dispersive Fourier transformation for fast

- continuous single-shot measurements. *Nature Photonics*, Vol. 7, pp. 102–112.  
<https://doi.org/10.1038/nphoton.2012.359>
- Goldberg, X. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6, 1–116.  
<https://doi.org/10.2200/S00196ED1V01Y200906AIM006>
- Graves, A. (2012). *Supervised Sequence Labelling*. [https://doi.org/10.1007/978-3-642-24797-2\\_2](https://doi.org/10.1007/978-3-642-24797-2_2)
- Gunawan, A., Lau, H. C., & Lindawati. (2011). Fine-tuning algorithm parameters using the design of experiments approach. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6683 LNCS, 278–292. [https://doi.org/10.1007/978-3-642-25566-3\\_21](https://doi.org/10.1007/978-3-642-25566-3_21)
- Häne, C., Zach, C., Cohen, A., Angst, R., & Pollefeys, M. (2013). *Joint 3D Scene Reconstruction and Class Segmentation*. 97–104.  
<https://doi.org/10.1109/CVPR.2013.20>
- Haralick, R. M., & Shapiro, L. G. (1985). Image segmentation techniques. *Computer Vision, Graphics, & Image Processing*, 29(1), 100–132.  
[https://doi.org/10.1016/S0734-189X\(85\)90153-7](https://doi.org/10.1016/S0734-189X(85)90153-7)
- Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems*, 3323–3331. Retrieved from <https://arxiv.org/abs/1610.02413v1>
- Hasan, I., Liao, S., Li, J., Ullah Akram, S., & Shao, L. (2021). Generalizable Pedestrian Detection: The Elephant In The Room. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11328--11337.
- Hayder, Z., & He, X. (2017). Boundary-aware Instance Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5696–5704.
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2961–2969. Retrieved from <https://github.com/>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. Retrieved from <http://image->

- net.org/challenges/LSVRC/2015/
- Heale, R., & Twycross, A. (2015). Validity and reliability in quantitative studies. *Evidence-Based Nursing, 18*(3), 66–67. <https://doi.org/10.1136/eb-2015-102129>
- Hoang, T. M., Nguyen, P. H., Truong, N. Q., Lee, Y. W., & Park, K. R. (2019). Deep retinonet-based detection and classification of road markings by visible light camera sensors. *Sensors (Switzerland), 19*(2). <https://doi.org/10.3390/s19020281>
- Hógye-Nagy, Á., Kovács, G., & Kurucz, G. (2023). Acceptance of self-driving cars among the university community: Effects of gender, previous experience, technology adoption propensity, and attitudes toward autonomous vehicles. *Transportation Research Part F: Traffic Psychology and Behaviour, 94*, 353–361. <https://doi.org/10.1016/J.TRF.2023.03.005>
- Holden, H., & Rada, R. (2011). Journal of Research on Technology in Education | 343 Understanding the Influence of Perceived Usability and Technology Self-Efficacy on Teachers. In *Technology Acceptance JRTE* / (Vol. 43).
- Homayounfar, N., Xiong, Y., Liang, J., Ma, W. C., & Urtasun, R. (2020). LevelSet R-CNN: A Deep Variational Method for Instance Segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12368 LNCS*, 555–571. [https://doi.org/10.1007/978-3-030-58592-1\\_33/COVER](https://doi.org/10.1007/978-3-030-58592-1_33/COVER)
- Huang, Y., & Qian, L. (2021). Understanding the potential adoption of autonomous vehicles in China: The perspective of behavioral reasoning theory. *Psychology & Marketing, 38*(4), 669–690. <https://doi.org/10.1002/mar.21465>
- Huck, F. O., & Fales, C. L. (2004). Imaging: Information Theory in Imaging. In *Encyclopedia of Modern Optics, Five-Volume Set* (pp. 107–117). <https://doi.org/10.1016/B0-12-369395-0/00708-9>
- Huh, M., Agrawal, P., & Efros, A. A. (2016). What makes ImageNet good for transfer learning? *ArXiv Preprint ArXiv:1608.08614*.
- Hung, W.-C., Tsai, Y.-H., Liou, Y.-T., Lin, Y.-Y., & Yang, M.-H. (2018). Adversarial Learning for Semi-Supervised Semantic Segmentation. *ArXiv Preprint ArXiv:1802.07934*.
- Ignatious, H. A., Sayed, H. El, & Khan, M. (2021). An overview of sensors in

- Autonomous Vehicles. *Procedia Computer Science*, 198, 736–741.  
<https://doi.org/10.1016/J.PROCS.2021.12.315>
- Image Processing and Image Analysis. (2008). In *Electron Microscopy of Polymers* (pp. 161–171). [https://doi.org/10.1007/978-3-540-36352-1\\_8](https://doi.org/10.1007/978-3-540-36352-1_8)
- James, J. (2022). Let there be light: Daylight saving time and road traffic collisions. *Economic Inquiry*, 1–23. <https://doi.org/10.1111/ECIN.13130>
- Janai, J., Güney, F., Behl, A., & Geiger, A. (2020). Computer vision for autonomous vehicles. *Foundations and Trends in Computer Graphics and Vision*, 12(1–3), 1–308. <https://doi.org/10.1561/06000000079>
- Jarraya, S. K. (2018). A Method for Tracking Road Objects. *The International Journal of Multimedia & Its Applications (IJMA)*, 10(4).  
<https://doi.org/10.5121/ijma.2018.10501>
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding \*. *Proceedings of the 22nd ACM International Conference on Multimedia*, 675–678. Retrieved from <http://demo.caffe.berkeleyvision.org/>
- Jiang, C., Xu, H., Zhang, W., Liang, X., & Li, Z. (2021). SP-NAS: Serial-to-Parallel Backbone Search for Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11863–11872.
- Jiao, L., Zhang, F., Liu, F., Member, S., Yang, S., Li, L., ... Qu, R. (2019). A Survey of Deep Learning-based Object Detection. *IEEE Access*, 7(IEEE), 128837--128868.
- Jie, H. J., & Wanda, P. (2020). Runpool: A dynamic pooling layer for convolution neural network. *International Journal of Computational Intelligence Systems*, 13(1), 66–76. <https://doi.org/10.2991/ijcis.d.200120.002>
- Johari, A., & Swami, P. D. (2020). Comparison of autonomy and study of deep learning tools for object detection in autonomous self driving vehicles. *2nd International Conference on Data, Engineering and Applications, IDEA 2020*.  
<https://doi.org/10.1109/IDEA49133.2020.9170659>
- Kammel, S., Ziegler, J., Pitzer, B., Werling, M., Gindele, T., Jagzent, D., ... Stiller, C. (2008). Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9), 615–639.  
<https://doi.org/10.1002/ROB.20252>
- Kaplan, D. (2001). Structural Equation Modeling. *International Encyclopedia of the*

- Social & Behavioral Sciences*, 15215–15222. <https://doi.org/10.1016/B0-08-043076-7/00776-2>
- Kaur, R., Chawla, M., Khiva, N. K., & Ansari, M. D. (2017). On Contrast Enhancement Techniques for Medical Images with Edge Detection: A Comparative Analysis. *Journal of Telecommunication*, 9, 35–40.
- Keller, J. M., & Wang, X. (1995). Comparison of spatial relation definitions in computer vision. *Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS*, 679–684.  
<https://doi.org/10.1109/isuma.1995.527776>
- Kendall, A., Gal, Y., & Cipolla, R. (2018). *Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics; Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics*.  
<https://doi.org/10.1109/CVPR.2018.00781>
- Kim, I., & Kim, D. (2012). Optimal design of extended slot allocation for multiuser relay networks. *2012 1st IEEE International Conference on Communications in China, ICC China 2012*, 475–480. <https://doi.org/10.1109/ICCChina.2012.6356930>
- Koopman, P., & Wagner, M. (2016). Challenges in Autonomous Vehicle Testing and Validation. *SAE International Journal of Transportation Safety*, 4(1), 15–24.  
<https://doi.org/10.4271/2016-01-0128>
- Krapp, H. G., & Hengstenberg, R. (1996). Estimation of self-motion by optic flow processing in single visual interneurons. *Nature*, 384(6608), 463–466.  
<https://doi.org/10.1038/384463a0>
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25. Retrieved from <http://code.google.com/p/cuda-convnet/>
- Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2), 83–97.  
<https://doi.org/10.1002/nav.3800020109>
- Kumar Kukkala, V., Tunnell, J., Pasricha, S., & Bradley, T. (2018). Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles; Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles. *IEEE Consumer Electronics Magazine*, 7. <https://doi.org/10.1109/MCE.2018.2828440>

- Kumar, T., & Kushwaha, D. S. (2016). An Efficient Approach for Detection and Speed Estimation of Moving Vehicles. *Procedia Computer Science*, 89, 726–731.  
<https://doi.org/10.1016/j.procs.2016.06.045>
- Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., ... Ferrari, V. (n.d.). *The Open Images Dataset V4 Unified image classification, object detection, and visual relationship detection at scale*. Retrieved from  
<https://storage.googleapis.com/openimages/web/2018->
- Kyriakidis, M., Happee, R., & De Winter, J. C. F. (2015). Public opinion on automated driving: Results of an international questionnaire among 5000 respondents. *Transportation Research Part F: Traffic Psychology and Behaviour*, 32, 127–140.  
<https://doi.org/10.1016/j.trf.2015.04.014>
- Lavasani, M., Jin, X., & Du, Y. (2016). Market penetration model for autonomous vehicles on the basis of earlier technology adoption experience. *Transportation Research Record*, 2597, 67–74. <https://doi.org/10.3141/2597-09>
- Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., ... Ng, A. Y. (2011). Building high-level features using large scale unsupervised learning. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 8595–8598. Retrieved from <http://arxiv.org/abs/1112.6209>
- Lederer, A. L., Maupin, D. J., Sena, M. P., & Zhuang, Y. (2000). The technology acceptance model and the World Wide Web. *Decision Support Systems*, 29(3), 269–282. [https://doi.org/10.1016/S0167-9236\(00\)00076-2](https://doi.org/10.1016/S0167-9236(00)00076-2)
- Lee, P. H., Chiu, T. H., Lin, Y. L., & Hung, Y. P. (2009). Real-time pedestrian and vehicle detection in video using 3D cues. *Proceedings - 2009 IEEE International Conference on Multimedia and Expo, ICME 2009*, 614–617.  
<https://doi.org/10.1109/ICME.2009.5202571>
- Li, P., Chen, X., & Shen, S. (2019). Stereo R-CNN based 3D object detection for autonomous driving. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June*, 7636–7644.  
<https://doi.org/10.1109/CVPR.2019.00783>
- Li, T., Wu, B., Yang, Y., Fan, Y., Zhang, Y., & Liu, W. (2019). Compressing convolutional neural networks via factorized convolutional filters. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June*, 3972–3981. <https://doi.org/10.1109/CVPR.2019.00410>

- Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999–7019.  
<https://doi.org/10.1109/TNNLS.2021.3084827>
- Liao, Q., & Poggio, T. (2016). *Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex*. Retrieved from  
<http://arxiv.org/abs/1604.03640>
- Liljamo, T., Liimatainen, H., & Pöllänen, M. (2018). Attitudes and concerns on automated vehicles. *Transportation Research Part F: Traffic Psychology and Behaviour*, 59, 24–44. <https://doi.org/10.1016/j.trf.2018.08.010>
- Lim, Y.-C. (2018). Object Detection Using a Single Extended Feature Map; Object Detection Using a Single Extended Feature Map. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. [https://doi.org/10.0/Linux-x86\\_64](https://doi.org/10.0/Linux-x86_64)
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2016). *Feature Pyramid Networks for Object Detection*.
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2117–2125.
- Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). *Focal Loss for Dense Object Detection*.
- Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., ... Dollár, P. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision--ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755.
- Liu, K., Kang, G., Zhang, N., & Hou, B. (2018). Breast Cancer Classification Based on Fully-Connected Layer First Convolutional Neural Networks. *IEEE Access*, 6, 23722–23732. <https://doi.org/10.1109/ACCESS.2018.2817593>
- Liu, P., Guo, Q., Ren, F., Wang, L., & Xu, Z. (2019). Willingness to pay for self-driving vehicles: Influences of demographic and psychological factors. *Transportation Research Part C: Emerging Technologies*, 100, 306–317.  
<https://doi.org/10.1016/j.trc.2019.01.022>
- Liu, P., Yang, R., & Xu, Z. (2019). Public Acceptance of Fully Automated Driving: Effects of Social Trust and Risk/Benefit Perceptions. *Risk Analysis*, 39(2), 326–



341. <https://doi.org/10.1111/risa.13143>
- Liu, Shu, Jia, J., & Fidler, S. (2017). *SGN: Sequential Grouping Networks for Instance Segmentation*. <https://doi.org/10.1109/ICCV.2017.378>
- Liu, Shu, Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8759–8768. Retrieved from <https://github.com/>
- Liu, Songtao, Huang, D., & Wang, Y. (2020). *Adaptive NMS: Refining Pedestrian Detection in a Crowd*. 6452–6461. <https://doi.org/10.1109/cvpr.2019.00662>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS, 21–37. [https://doi.org/10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2)
- Liu, W., Liao, S., Ren, W., Hu, W., & Yu, Y. (2020). High-Level Semantic Feature Detection: A New Perspective for Pedestrian Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5182–5191. <https://doi.org/10.1109/cvpr.2019.00533>
- Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the IEEE International Conference on Computer Vision*, 2, 1150–1157. <https://doi.org/10.1109/iccv.1999.790410>
- Ma, Q., & Liu, L. (2005). The Role of Internet Self-Efficacy in the Acceptance of Web-Based Electronic Medical Records. *Journal of Organizational and End User Computing*, 17(1), 38–57. <https://doi.org/10.4018/JOEUC.2005010103>
- Maini, R., & Aggarwal, H. (2010). *A Comprehensive Review of Image Enhancement Techniques* (Vol. 2).
- Mao, J., Xiao, T., & Jiang, Y. (2017). What Can Help Pedestrian Detection? *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3127–3136.
- Martin, S. (2018). What's the Difference Between a CNN and an RNN? - Edge AI and Vision Alliance. Retrieved May 29, 2020, from <https://www.edge-ai-vision.com/2018/09/whats-the-difference-between-a-cnn-and-an-rnn/>
- Martínez-Díaz, M., & Soriguera, F. (2018). Autonomous vehicles: Theoretical and practical challenges. *Transportation Research Procedia*, 33, 275–282. <https://doi.org/10.1016/j.trpro.2018.10.103>

- Mazzini, D., & Schettini, R. (2019). Spatial Sampling Network for Fast Scene Understanding. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. <https://doi.org/10.1109/CVPRW.2019.00168>
- Meyer, J., Becker, H., Bösch, P. M., & Axhausen, K. W. (2017). Autonomous vehicles: The next jump in accessibilities? *Research in Transportation Economics*, 62, 80–91. <https://doi.org/10.1016/J.RETREC.2017.03.005>
- Milan, A., Leal-Taixé, L., Taixé, T., Reid, I., Roth, S., & Schindler, K. (2016). MOT16: A Benchmark for Multi-Object Tracking. *ArXiv Preprint ArXiv:1603.00831*. Retrieved from <http://www.motchallenge.net/>
- National Highway Traffic Safety Administration and others. (2017). Federal motor vehicle safety standards; V2V communications. *Federal Register*, 82, 3854–4019. Retrieved from <https://www.federalregister.gov/documents/2017/01/12/2016-31059/federal-motor-vehicle-safety-standards-v2v-communications>
- Nienaber, S., Kroon, R. S., & Booyen, M. J. (2015). A comparison of low-cost monocular vision techniques for pothole distance estimation. *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, 419–426. <https://doi.org/10.1109/SSCI.2015.69>
- Nieuwenhuijsen, J. A. H. (2015). *Diffusion of Automated Vehicles: A quantitative method to model the diffusion of automated vehicles with system dynamics*. Retrieved from <https://repository.tudelft.nl/islandora/object/uuid%3A0f3f5155-88ec-4f66-bb3b-4a60fec61863>
- Nordhoff, S., De Winter, J., Kyriakidis, M., Van Arem, B., & Happee, R. (2018). *Acceptance of Driverless Vehicles: Results from a Large Cross-National Questionnaire Study*. <https://doi.org/10.1155/2018/5382192>
- Oliver, J., Baxter, R. A., & Wallace, C. (1996). Unsupervised Learning Using MML. *Undefined*.
- Ortelt, B., Herrmann, C., Willersinn, D., & Beyerer, J. (2018). *Foveal Vision for Instance Segmentation of Road Images*. <https://doi.org/10.5220/0006616103710378>
- Papadoulis, A., Quddus, M., & Imprialou, M. (2019). Evaluating the safety impact of connected and autonomous vehicles on motorways. *Accident Analysis and Prevention*, 124, 12–22. <https://doi.org/10.1016/j.aap.2018.12.019>

- Papageorgiou, C. P., Oren, M., & Poggio, T. (1998). General framework for object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 555–562. <https://doi.org/10.1109/iccv.1998.710772>
- Park, J. E., Byun, W., Kim, Y., Ahn, H., & Shin, D. K. (2021). *The Impact of Automated Vehicles on Traffic Flow and Road Capacity on Urban Road Networks*. <https://doi.org/10.1155/2021/8404951>
- Peng, S., Jiang, W., Pi, H., Li, X., Bao, H., & Zhou, X. (2020). Deep Snake for Real-Time Instance Segmentation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8530–8539. Retrieved from <https://github.com/zju3dv/snake/>.
- Pillath, S. (2016). *Briefing European Parliamentary Research Service*.
- Porzi, L., Rotabuì, S., Colovic, A., Kontschieder, P., & Research, M. (2019). Seamless Scene Segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8277–8286. Retrieved from <http://cocodataset.org/workshop/coco->
- Purwono, Ma'arif, A., Rahmani, W., Fathurrahman, H. I. K., Frisky, A. Z. K., & Haq, Q. M. U. (2022). Understanding of Convolutional Neural Network (CNN): A Review. *International Journal of Robotics and Control Systems*, 2(4), 739–748. <https://doi.org/10.31763/ijrcs.v2i4.888>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*. Retrieved from <http://pjreddie.com/yolo/>
- Redmon, J., & Farhadi, A. (2017). *YOLO9000: Better, Faster, Stronger*. Retrieved from <http://pjreddie.com/yolo9000/>
- Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv*. Retrieved from <http://arxiv.org/abs/1804.02767>
- Reeves, S. J. (2014). *Image Restoration: Fundamentals of Image Restoration*. <https://doi.org/10.1016/b978-0-12-396501-1.00006-6>
- Rejali, S., Aghabayk, K., Esmaeli, S., & Shiwakoti, N. (2023). Comparison of technology acceptance model, theory of planned behavior, and unified theory of acceptance and use of technology to assess a priori acceptance of fully automated vehicles. *Transportation Research Part A: Policy and Practice*, 168, 103565. <https://doi.org/10.1016/j.TRA.2022.103565>
- Ren, S., He, K., Girshick, R., & Sun, J. (2015a). Faster R-CNN: Towards Real-Time

- Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28. Retrieved from <http://image-net.org/challenges/LSVRC/2015/results>
- Ren, S., He, K., Girshick, R., & Sun, J. (2015b). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28. Retrieved from <https://github.com/>
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149. <https://doi.org/10.1109/TPAMI.2016.2577031>
- Rezaei, M., Terauchi, M., & Klette, R. (2015). Robust Vehicle Detection and Distance Estimation Under Challenging Lighting Conditions. *IEEE Transactions on Intelligent Transportation Systems*, 16(5), 2723–2743. <https://doi.org/10.1109/TITS.2015.2421482>
- Ristani, E., Solera, F., Zou, R., Cucchiara, R., & Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9914 LNCS(c), 17–35. [https://doi.org/10.1007/978-3-319-48881-3\\_2](https://doi.org/10.1007/978-3-319-48881-3_2)
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. <https://doi.org/10.1038/323533a0>
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115, 211–252. Retrieved from <http://image-net.org/challenges/LSVRC/>
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3), 211–252. <https://doi.org/10.1007/s11263-015-0816-y>
- Sahu, P., Gupta, N., & Sharma, N. (2014). A Survey on Underwater Image Enhancement Techniques. *International Journal of Computer Applications*, 87(13), 19–23.

- Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann Machines. *E Twelfth International Conference on Artificial Intelligence and Statistics*, 448–455. Retrieved from <http://proceedings.mlr.press/v5/salakhutdinov09a.html>
- Saleh, K., Hossny, M., Hossny, A., & Nahavandi, S. (2018). Cyclist detection in LIDAR scans using faster R-CNN and synthetic depth images. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, 2018-March*, 1–6. <https://doi.org/10.1109/ITSC.2017.8317599>
- Sanbonmatsu, D. M., Strayer, D. L., Yu, Z., Biondi, F., & Cooper, J. M. (2018). Cognitive underpinnings of beliefs and confidence in beliefs about fully automated vehicles. *Transportation Research Part F: Traffic Psychology and Behaviour*, 55, 114–122. <https://doi.org/10.1016/j.trf.2018.02.029>
- Santarossa, M., Schneider, L., Zelenka, C., Schmarje, L., Koch, R., & Franke, U. (2021). *Learning Stixel-based Instance Segmentation*. <https://doi.org/10.1109/IV48863.2021.9575565>
- Sarma, K. M., & Carey, R. N. (2017). Impact of daylight saving time on road traffic collision risk: a systematic review. *BMJ Open*, 7, e014319. Retrieved from <https://bmjopen.bmj.com/content/bmjopen/7/6/e014319.full.pdf>
- Schaffalitzky, F., Zisserman, A., Hartley, R. I., & Torr, P. H. S. (2000). A six point solution for structure and motion. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1842, 632–648. [https://doi.org/10.1007/3-540-45054-8\\_41](https://doi.org/10.1007/3-540-45054-8_41)
- Scherer, R., Siddiq, F., & Tondeur, J. (2019). The technology acceptance model (TAM): A meta-analytic structural equation modeling approach to explaining teachers' adoption of digital technology in education. *Computers and Education*, 128, 13–35. <https://doi.org/10.1016/j.compedu.2018.09.009>
- Schulz, T. J. (2005). Multi-Frame Image Restoration. In *Handbook of Image and Video Processing* (pp. 219–233). <https://doi.org/10.1016/B978-012119792-6/50077-2>
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), 1–48. <https://doi.org/10.1186/s40537-019-0197-0>
- Shubbak, M. H. (2017). Self-Driving Cars; Legal, Social, and Ethical Aspects. *Social, and Ethical Aspects (March 13, 2017)*.

- <https://doi.org/https://doi.org/10.2139/ssrn.2931847>
- Standard, S. (2018). J3016B: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International. Retrieved March 31, 2021, from [https://www.sae.org/standards/content/j3016\\_201806/](https://www.sae.org/standards/content/j3016_201806/)
- Steiner, G. (2001). Transfer of Learning, Cognitive Psychology of. *International Encyclopedia of the Social & Behavioral Sciences*, 15845–15851.  
<https://doi.org/10.1016/B0-08-043076-7/01481-9>
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications* - Richard Szeliski - Google Books. Retrieved from [https://books.google.ie/books?hl=en&lr=&id=bXzAlkODwa8C&oi=fnd&pg=PR4&dq=computer+vision&ots=g-Y4a4kGFB&sig=H0ML8kv9Svyc6yUMwIXmMrCBHAg&redir\\_esc=y#v=onepage&q=computer vision&f=false](https://books.google.ie/books?hl=en&lr=&id=bXzAlkODwa8C&oi=fnd&pg=PR4&dq=computer+vision&ots=g-Y4a4kGFB&sig=H0ML8kv9Svyc6yUMwIXmMrCBHAg&redir_esc=y#v=onepage&q=computer%20vision&f=false)
- Tarpley, P., & Jesma, S. D. (2016). Autonomous vehicles: The legal landscape in the US. *Norton Rose Fulbright*. Retrieved from <https://www.nortonrosefulbright.com/en/knowledge/publications/2951f5ce/autonomous-vehicles-the-legal-landscape-in-the-us>
- Thoma, M. (2016). *A Survey of Semantic Segmentation*. Retrieved from <http://arxiv.org/abs/1602.06541>
- Thrun, S. (2010). Toward robotic cars. *Communications of the ACM*, 53(4), 99–106.  
<https://doi.org/10.1145/1721654.1721679>
- Tian, Z., Shen, C., Chen, H., & He, T. (2019). FCOS: Fully Convolutional One-Stage Object Detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9627–9636.
- Tomè, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., & Tubaro, S. (2015). Deep convolutional neural networks for pedestrian detection. *Signal Processing: Image Communication*, 47, 482–489.  
<https://doi.org/10.1016/j.image.2016.05.007>
- Torralba, A., & Oliva, A. (2002). Depth estimation from image structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9), 1226–1238.  
<https://doi.org/10.1109/TPAMI.2002.1033214>
- Torrey, L., & Shavlik, J. (2010). Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. In *Transfer*

- learning* (pp. 242–264). <https://doi.org/10.4018/978-1-60566-766-9.CH011>
- Uchida, K., Tanaka, M., & Okutomi, M. (2018). Coupled convolution layer for convolutional neural network. *Neural Networks*, *105*, 197–205. <https://doi.org/10.1016/j.neunet.2018.05.002>
- Uhrig, J., Cordts, M., Franke, U., & Brox, T. (2016). Pixel-level encoding and depth layering for instance-level semantic labeling. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9796 LNCS*, 14–25. [https://doi.org/10.1007/978-3-319-45886-1\\_2/COVER](https://doi.org/10.1007/978-3-319-45886-1_2/COVER)
- Uijlings, J. R. R., Van De Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, *104*(2), 154–171. <https://doi.org/10.1007/S11263-013-0620-5/FIGURES/9>
- Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Dolan, J., ... Taylor, M. (2007). Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge. *Defense*, *94*(4), 386–387. <https://doi.org/10.1002/rob.20251>
- Uttley, J., & Fotios, S. (2017). The effect of ambient light condition on road traffic collisions involving pedestrians on pedestrian crossings. *Accident Analysis and Prevention*, *108*, 189–200. <https://doi.org/10.1016/J.AAP.2017.09.005>
- Van Witsen, M. (2016). Zelfrijdende auto's te ambitieus - Automatisch wegverkeer alleen in afgeschermd gebied. In *De Ingenieur*. Retrieved from <https://www.rijksoverheid.nl/onderwerpen/mobiliteit-nu-en-in-de-toekomst/vraag-en-antwoord/zelfrijdende-auto-op-openbare-weg>
- Venkatesh, Morris, Davis, & Davis. (2003). User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly*, *27*(3), 425. <https://doi.org/10.2307/30036540>
- Venkatesh, V., & Bala, H. (2008). Technology Acceptance Model 3 and a Research Agenda on Interventions. *Decision Sciences*, *39*(2), 273–315. <https://doi.org/10.1111/j.1540-5915.2008.00192.x>
- Venkatesh, V., & Davis, F. D. (2000). A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science*, *46*(2), 186–204. <https://doi.org/10.1287/mnsc.46.2.186.11926>
- Viola, P., & Jones, M. J. (2004). Robust Real-time Object Detection. *International Journal of Computer Vision*, *57*, 137–154.

- Wang, Han, Li, Y., & Wang, S. (2019). Fast Pedestrian Detection With Attention-Enhanced Multi-Scale RPN and Soft-Cascaded Decision Trees. *IEEE Transactions on Intelligent Transportation Systems*, 1–8.  
<https://doi.org/10.1109/tits.2019.2948398>
- Wang, Huiyu, Zhu, Y., Green, B., Adam, H., Yuille, A., & Chen, L. C. (2020). Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12349 LNCS, 108–126.  
[https://doi.org/10.1007/978-3-030-58548-8\\_7/COVER](https://doi.org/10.1007/978-3-030-58548-8_7/COVER)
- Wang, K., & Zhou, W. (2019). Pedestrian and cyclist detection based on deep neural network fast R-CNN. *International Journal of Advanced Robotic Systems*, 16(2).  
<https://doi.org/10.1177/1729881419829651>
- Wang, L., Fan, X., Chen, J., Cheng, J., Tan, J., & Ma, X. (2020). 3D object detection based on sparse convolution neural network and feature fusion for autonomous driving in smart cities. *Sustainable Cities and Society*, 54, 102002.  
<https://doi.org/10.1016/j.scs.2019.102002>
- Watanabe, T., & Wolf, D. F. (2019). Instance Segmentation as Image Segmentation Annotation; Instance Segmentation as Image Segmentation Annotation. *2019 IEEE Intelligent Vehicles Symposium (IV)*. [https://doi.org/10.0/Linux-x86\\_64](https://doi.org/10.0/Linux-x86_64)
- Waterfall, R. C., & Dickinson, K. W. (1984). IMAGE PROCESSING APPLIED TO TRAFFIC. *Traffic Engineering & Control*, 25. Retrieved from  
<https://trid.trb.org/view/211437>
- Wei, J., He, J., Zhou, Y., Chen, K., Tang, Z., & Xiong, Z. (2020). Enhanced Object Detection With Deep Convolutional Neural Networks for Advanced Driving Assistance. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 21(4). <https://doi.org/10.1109/TITS.2019.2910643>
- Welch, G., & Bishop, G. (1995). *An Introduction to the Kalman Filter*. Retrieved from  
<http://www.cs.unc.edu/~gb>
- Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* (Harvard University). Retrieved from  
<https://www.bibsonomy.org/person/14165e2708a0468e89f8305f21ee2c711/author/0>
- Wintersberger, P., Frison, A. K., & Riener, A. (2018). Man vs. machine: Comparing a



- fully automated bus shuttle with a manually driven group taxi in a field study. *Adjunct Proceedings - 10th International ACM Conference on Automotive User Interfaces and Interactive Vehicular Applications, AutomotiveUI 2018*, 215–220. <https://doi.org/10.1145/3239092.3265969>
- Wolf, J. (2008). Self-Administered Questionnaire. *Encyclopedia of Survey Research Methods*. <https://doi.org/10.4135/9781412963947>
- Woods, J. W. (2012). Image Estimation and Restoration. In *Multidimensional Signal, Image, and Video Processing and Coding* (pp. 257–327). <https://doi.org/10.1016/b978-0-12-381420-3.00008-4>
- Wu, J. (2017). *Introduction to Convolutional Neural Networks*.
- Xie, H., Zheng, W., Shin, H., & Proença, P. (2021). *Occluded Pedestrian Detection Techniques by Deformable Attention-Guided Network (DAGN)*. <https://doi.org/10.3390/app11136025>
- Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January*, 5987–5995. <https://doi.org/10.1109/CVPR.2017.634>
- Xu, C., Ding, Z., Wang, C., & Li, Z. (2019). Statistical analysis of the patterns and characteristics of connected and autonomous vehicle involved crashes. *Journal of Safety Research*, 71, 41–47. <https://doi.org/10.1016/J.JSR.2019.09.001>
- Xu, X., Chiu, M. T., Huang, T. S., & Shi, H. (2020). Deep Affinity Net: Instance Segmentation via Affinity. *ArXiv Preprint ArXiv:2003.06849*.
- Yebes, J. J., Bergasa, L. M., Arroyo, R., & Lazaro, A. (2014). Supervised learning and evaluation of KITTI's cars detector with DPM. *IEEE Intelligent Vehicles Symposium, Proceedings*, 768–773. <https://doi.org/10.1109/IVS.2014.6856452>
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, 4(January), 3320–3328.
- Yuen, K. F., Cai, L., Qi, G., & Wang, X. (2021). Factors influencing autonomous vehicle adoption: an application of the technology acceptance model and innovation diffusion theory. *Technology Analysis and Strategic Management*, 33(5), 505–519. <https://doi.org/10.1080/09537325.2020.1826423>
- Zhang, C., Xu, X., & Tu, D. (2018). Face Detection Using Improved Faster RCNN. *ArXiv*

*Preprint ArXiv:1802.02142.*

- Zhang, J., Lin, L., Li, Y., Chen, Y.-C., Hu, Y., & Hoi, S. C. H. (2020). Attribute-aware Pedestrian Detection in a Crowd. *IEEE Transactions on Multimedia*, 23(IEEE), 3085--3097.
- Zhang, T., Zhang, X., Shi, J., & Wei, S. (2019). Depthwise separable convolution neural network for high-speed SAR ship detection. *Remote Sensing*, 11(21).  
<https://doi.org/10.3390/rs11212483>
- Zhao, X., Li, W., Zhang, Y., Gulliver, T. A., Chang, S., & Feng, Z. (2016). A faster RCNN-based pedestrian detection system. *IEEE Vehicular Technology Conference*, 0.  
<https://doi.org/10.1109/VTCFall.2016.7880852>
- Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2018). *Object Detection with Deep Learning: A Review*.
- Zhao, Z. Q., Bian, H., Hu, D., Cheng, W., & Glotin, H. (2017). Pedestrian detection based on fast R-CNN and batch normalization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10361 LNCS, 735–746. [https://doi.org/10.1007/978-3-319-63309-1\\_65](https://doi.org/10.1007/978-3-319-63309-1_65)
- Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). EAST: An Efficient and Accurate Scene Text Detector. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5551–5560.
- Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J., & Research, S. (2020). DEFORMABLE DETR: DEFORMABLE TRANSFORMERS FOR END-TO-END OBJECT DETECTION. *ArXiv Preprint ArXiv:2010.04159*. Retrieved from <https://github>.

## Appendix A

This Appendix records the survey noted in Chapter 3.

### Section A—Demographic Information

Please answer the following questions. Tick  $\checkmark$  the appropriate boxes for your answer.

1. Gender

Male       Female       Prefer not to say

2. Age

Below 20 years    20–35 years    35–60 years    Above 60 years

3. Education

Ph.D.       Master's       Bachelor's       Diploma       Others

4. Employment Sector

Automotive industry       Technology sector       Regulatory Agencies

5. Years of Experience

<10 years       0–15 years       20–35 years       >35 years

### Section B—Evaluation of Autonomous Vehicles

Please answer the following questions.

#### Safety

- i. What would be the greatest advantage(s) be in terms of safety to the users of autonomous vehicles (AVs)? (a) fewer accidents (b) reduction in risky driving behaviours of human drivers (c) controlled decisions in term of accelerating, decelerating, and changing lanes
- ii. How do autonomous vehicles (AVs) minimize car crashes on the road? (a) by minimizing the involvement of human operations (b) through technology such as side view assist, adaptive headlights, forward collision, and lane departure warning system
- iii. What makes the number of accidents in autonomous vehicles fewer when compared to traditional vehicles? (a) controlled accelerating and decelerating (b) better fleet management by lowering peak speeds (c) higher effective speed (d) reduction in travel time (e) lighter design
- iv. How do autonomous vehicles avoid accidents involving other road users?  
(a) manageable parking arrangements (b) effective hazardous behaviour detection and prediction mechanism (c) false alarm mechanism

## **Environment**

- i. How does the environment benefit from the use of autonomous vehicles? (a) Improvement in fuel consumption reduces carbon emissions in the environment (b) As some AVs are electrically powered, as such there are no carbon emissions
- ii. How do autonomous vehicles contribute to the environmental sustainability drive to reduce the impact of climate change? (a) reduction in traffic congestion, which reduces carbon emissions in the environment (b) automated acceleration and braking by AVs (known as eco-driving), which reduces fuel consumption
- iii. Since AVs mean more vehicles will be used, leading to increased pollution, what makes it different from the traditional vehicle in pollution control? (a) it makes travelling from one point to another easier, thereby reducing the vehicle miles travelled (b) On-demand mobility and carsharing made possible by autonomous technology significantly reduces GHG (c) Platooning (a method for driving a group of vehicles together.)
- iv. What measure is put in place for the user of autonomous vehicles to account for greenhouse gas emission of the vehicle? (a) government legislation (b) emission costs

## **Conjunction**

- i. What features in autonomous vehicle reduce human effort in their operations? (a) sensors (b) lane departure warning (c) night vision (d) adaptive cruise control
- ii. How do autonomous vehicles contribute to the productivity of the user? (a) since it reduces traffic congestion, less time is spent in traffic (b) it drives itself, and so the user can use that driving time to engage in some other productive activities without having to worry about the driving effort
- iii. What features make autonomous vehicle more convenient than traditional vehicles? (a) it reduces travelling time (b) there is an opportunity to rideshare and carshare (c) all-time carpooling availability
- iv. How is freedom achieved by the autonomous vehicles? (a) AVs are designed to be more spacious, affording the user the freedom to do other things (b) being a self-driving vehicle, users can care less about driving efforts, and use the time to attend to other beneficial things.

## **Anxiety**

- i. Since machines cannot be 100% efficient, what features can make the users of autonomous vehicles have full confidence in the efficiency of such vehicles? (a) cruise control (b) controlled decisions in term of accelerating, decelerating, and changing lanes (c) effective hazardous behaviour detection and prediction mechanism.
- ii. How do autonomous vehicles ensure that its users do not activate the panic mode during emergencies? (a) false alarm mechanism (b) manageable parking arrangement (c) effective hazardous behaviour detection and prediction mechanism.
- iii. What technologies are put in place in autonomous vehicles to reinforce users' trust while using the vehicle? Answers the same as in the previous (ii)
- iv. What features in the autonomous vehicle make interactions possible to reduce user anxiety? (a) action controller (b) actuator (c) interconnections between systems.

### Section C—Analysis of Variables

*Please tick the appropriate boxes with respect to the following variables.*

#### Safety

S / N	Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1.	Self-driving vehicles generate fewer accidents					
2.	Self-driving vehicles decrease traffic congestion					
3.	Unlike ordinary vehicles, which are often operated by drunk or distracted drivers,					

	self-driving vehicles can be expected to reduce risky driving behaviours					
4.	Self-driving vehicles outperform humans in detecting dangerous situations					

**Environment**

<b>S / N</b>	<b>Question</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Neutral</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
1.	Self-driving vehicles reduce energy consumption					
2.	Self-driving vehicles are environmentally friendly					
3.	Self-driving vehicles cause increases in car use and emissions					
4.	Self-driving vehicles will increase the number of miles people travel, thereby					

	increasing pollution					
5.	Self-driving vehicles will free up public spaces and promote clean air					

**Conjunction**

S/N	Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1.	Self-driving vehicles offer more convenience and productivity					
2.	Self-driving vehicles offer more personal freedom and independence					
3.	Mobility is more affordable with self-driving vehicles through ridesharing					
4.	Self-driving vehicles will reduce driving efforts					

**Anxiety Before Using AV**

S/N	Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree

1.	Before using an AV, I doubted if I would be able to control the vehicle if an ethically complicated situation arises					
2.	Before using an AV, I was afraid an emergency will arise if the vehicle malfunctions					
3.	Before using AVs, I was worried if all my journeys with AVs would be successful					
4	Before using AVs, I was worried that interacting with such a vehicle would require much mental effort					

**Anxiety After Using AV**

S/ N	Question	Strong ly Agree	Agr ee	Neutr al	Disagr ee	Strong ly
---------	----------	-----------------------	-----------	-------------	--------------	--------------



						<b>Disagree</b>
1.	After using an AV, my trust level in AVs has increased					
2.	After using an AV, I became confident that an emergency would hardly arise caused by the vehicle malfunctioning					
3.	After using an AV, I became confident that all my journeys with AVs would be successful					
4.	Interacting with the AV did not require a lot of mental effort					

**Anxiety of AV**

<b>S/N</b>	<b>Question</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Neutral</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
1.	Interacting with the vehicle does not require a lot of mental effort					

2.	I would trust such a vehicle					
3.	I am afraid I would not be able to react in case an emergency occurs					

## Appendix B

This Appendix includes all statistical figures of all the different hypotheses evaluated in Chapter 3

**Table 0.1** Descriptive Statistics.

<b>Descriptive Statistics</b>						
	N	Minimu m	Maximu m	Sum	Mean	Std. Deviation
V1: Generate fewer accidents	347	1	5	855	2.46	1.369
V2: Decrease traffic congestion	349	1	5	844	2.42	1.349
V3: Reduce risky driving behaviours for distracted drivers	347	1	5	899	2.59	1.398
V4: Outperform humans in detecting dangerous situations	348	1	5	1001	2.88	1.370
V5: Reduce energy consumption	349	1	5	898	2.57	1.364
V6: Vehicles are environmentally friendly	349	1	5	870	2.49	1.286
V7: Increase in car use and emissions	349	1	5	932	2.67	1.334
V8: Increase the number of miles people travel, hence increasing pollution	350	1	5	765	2.19	1.167
V9: Free up public spaces and promote clean air	349	1	5	820	2.35	1.370
V10: Offer more convenience and productivity.	349	1	5	926	2.65	1.340
V11: Offer more personal freedom and independence	349	1	5	898	2.57	1.368
V12: Mobility is more affordable through ridesharing	350	1	5	881	2.52	1.297
V13: Reduce the efforts of driving	349	1	5	985	2.82	1.525

V14: Before using AV, I doubted if I may not be able to control the vehicle if an ethically complicated situation arises	349	1	5	895	2.56	.997
V15: Before using AV, I was afraid a case of emergency will arise due to malfunctioning of the vehicle	349	1	5	921	2.64	1.386
V16: Before using AV, I was worried if all my journeys with AVs will be successful	349	1	5	899	2.58	1.395
V17: Before using AV, I was worried that interacting with the vehicle would require much mental effort	350	1	5	792	2.26	1.153
V18: After using AV, my trust level for AV increased	342	1	5	934	2.73	1.276
V19: After using AV, I became confident that a case of emergency will hardly arise due to malfunctioning of the vehicle.	349	1	5	852	2.44	1.300
V20: After using AV, I became confident that all my journeys with AVs will be successful.	349	1	5	994	2.85	1.361
V21: Interacting with the AV did not require a lot of mental effort.	349	1	5	911	2.61	1.190
V22: Interacting with the vehicle does not require a lot of my mental effort	350	1	5	878	2.51	1.194
V23: I would trust the vehicle.	349	1	5	888	2.54	1.298
V24: I am afraid that I won't be able to react in case an emergency occurs.	349	1	5	852	2.44	1.275
Valid N (listwise)	337					

## Appendix C

This Appendix illustrates changes done for evaluation, the evaluation of ECP using LAMR, different challenges encountered when computing the different algorithms in chapter 4.

For Cascade RCNN, we are training using object detection via bounding boxes, and not segmentation. Therefore, modifying the bbox head is necessary. Using KITTI as an example, here are the modifications:

```

model = dict(
  bbox_head=[
    dict(
      num_classes=3, #change num_class to 3 to match detection classes
      type='Shared2FCBBoxHead',
      in_channels=256,
      fc_out_channels=1024,
      roi_feat_size=7,
      num_classes=3, #change num_class to 3 to match detection classes
      bbox_coder=dict(
        type='DeltaXYWHBBoxCoder',
        target_means=[0., 0., 0., 0.],
        target_stds=[0.1, 0.1, 0.2, 0.2]),
      reg_class_agnostic=True,
      loss_cls=dict(
        type='CrossEntropyLoss',
        use_sigmoid=False,
        loss_weight=1.0),
      loss_bbox=dict(type='SmoothL1Loss', beta=1.0,
        loss_weight=1.0)),
    dict(
      type='Shared2FCBBoxHead',
      in_channels=256,
      fc_out_channels=1024,
      roi_feat_size=7,
      num_classes=3, #change num_class to 3 to match detection classes
      bbox_coder=dict(
        type='DeltaXYWHBBoxCoder',
        target_means=[0., 0., 0., 0.],
        target_stds=[0.05, 0.05, 0.1, 0.1]),
      reg_class_agnostic=True,
      loss_cls=dict(
        type='CrossEntropyLoss',
        use_sigmoid=False,
        loss_weight=1.0),
      loss_bbox=dict(type='SmoothL1Loss', beta=1.0,
        loss_weight=1.0)),
    dict(
      type='Shared2FCBBoxHead',
      in_channels=256,
      fc_out_channels=1024,
      roi_feat_size=7,
      num_classes=3, #change num_class to 3 to match detection classes
      bbox_coder=dict(
        type='DeltaXYWHBBoxCoder',
        target_means=[0., 0., 0., 0.],
        target_stds=[0.033, 0.033, 0.067, 0.067]),
      reg_class_agnostic=True,
      loss_cls=dict(
        type='CrossEntropyLoss',
        use_sigmoid=False,
        loss_weight=1.0),
      loss_bbox=dict(type='SmoothL1Loss', beta=1.0, loss_weight=1.0))
  ])

```

For RetinaNet to be trained modifications to `bbox_head` is to be done. As an example, for the KITTI dataset, the configuration file has been modified as follows:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 28 15:15:19 2021

@author: afnan
"""

_base_ = [
    '../_base_/models/retinanet_r50_fpn.py', '../_base_/datasets/kitti_detection.py',
    '../_base_/schedules/schedule_1x.py', '../_base_/default_runtime.py'
]
model = dict(
    pretrained=None,
    bbox_head=dict(
        num_classes=3, # change num_class to 3 to match detection classes
    ))
# optimizer
optimizer = dict(type='SGD', lr=0.0025, momentum=0.9, weight_decay=0.0001)
# We can use the pre-trained retinanet model to obtain higher performance
load_from = 'checkpoints/retinanet_r101_fpn_1x_coco_20200130-7a93545f.pth'
```

To use FCOS it on a customised dataset, the number of classes (`num_classes`) must be changed. Using KITTI as an example, here are the modifications that need to be considered:

```
model = dict(
    bbox_head=dict(
        type='FCOSHead',
        num_classes=3, #change num_class to 3 to match detection classes
        in_channels=256,
        stacked_convs=4,
        feat_channels=256,
        strides=[8, 16, 32, 64, 128],
```

For Deformable DETR it is important to modify the `num_classes` in `bbox_head` to suit the customised dataset. Again using KITTI as an example:

```
model = dict(
    bbox_head=dict(
        type='DeformableDETRHead',
        num_query=300,
        num_classes=3, #change num_class to 3 to match detection classes
```

The ECP team evaluated the performance of their dataset using modules such as SSD, YOLO, Faster RCNN and R-FCN. Based on their evaluation matrix, the dataset achieved the following results (Table 0.1), results are reported for completion purposes.

**Table 0.1** ECP literature testing evaluation (LAMR)

Reference	Detection method	LAMR			
		Reasonable	Small	Occluded	All
(Jiang, Xu, Zhang, Liang, & Li, 2021)	HRNet-W18	0.067	0.132	0.248	0.187
	SPNet w cascade	0.054	0.11	0.252	0.139
	ResNet50 w FPN	0.088	0.193	0.337	0.228
	ResNet101 w FPN	0.089	0.194	0.340	0.226
	HRNet-W40	0.067	0.132	0.272	0.181

## Challenges

### A) Challenges

Many challenges and errors occur during training and testing, even when the correct steps are followed; these errors may be due to errors related to data location, mmcv version, naming errors, missing package, file not found, and so on.

1. ModuleNotFoundError: No module named 'mmdet'
2. TypeError: 'numpy.float64' object cannot be interpreted as integral
3. 'list' object has no attribute 'astype'
4. Model 'pycococreatortools' has no attribute 'create\_image\_info'
5. ModuleNotFoundError: No module named 'pycococreatortools'
6. AttributeError: model 'pycococreatortools.pycococreatortools' has no attribute 'create\_annotation\_info'
7. AttributeError: 'ConfigDict' object has no attribute 'log\_level'
8. TypeError: bbox\_head={'num\_classes': 1} in child config cannot inherit from base because bbox\_head is a dict in the child config but is of type <class 'list'> in base configuration
9. '\pretrained\': None, '\bbox\_head\': {\num\_classes\': 18}}\n{\train\_cfg\': None, \test\_cfg\': None
10. return \_VF.meshgrid(tensors, \*\*kwargs) # type: ignore[attr-defined] Killed
11. AssertionError: CascadeRCNN: ResNeXt: init\_cfg and pretrained cannot be specified at the same time
12. ImportError: No module named 'skimage' when pip install scikit-image
13. ValueError: need at least one array to concatenate
14. TypeError: FCOS: FPN: \_\_init\_\_() got an unexpected keyword argument 'extra\_convs\_on\_inputs'
15. TypeError: CocoDataset: \_\_init\_\_() got an unexpected keyword argument 'times'
16. KeyError: 'xxxDataset is not in the dataset registry' when test

17. AssertionError: MMCV==1.3.0 is used but incompatible. Please install mmcv>=1.2.6, <=1.3.
18. ModuleNotFoundError: No module named 'seaborn'
19. KeyError: 'KittiDataset is not in the dataset registry'
20. FileNotFoundError: CocoDataset: [Errno 2] No such file or directory: '/home/afnan/Desktop/mmdetection/data/Fussed/Fussedday\_val.json'
21. AssertionError: The `num\_classes` (18) in RetinaHead of MMDataParallel does not matches the length of `CLASSES` (80) in CocoDataset
22. TypeError: CocoDataset: \_\_init\_\_() got an unexpected keyword argument 'times'
23. FileNotFoundError: [Errno 2] No such file or directory: '/tmp/tmp2kazkgh4/tmpw6c614ao.py'
24. TypeError: CocoDataset: MultiScaleFlipAug: \_\_init\_\_() missing 1 required positional argument: 'transforms'
25. OSError: [Errno 28] No space left on device

## B) Overcoming challenges

Each error has been resolved individually. For example, the 'ModuleNotFoundError'. In fact, all models and libraries were initially downloaded, but after a problem occurred, reinstalling the library was the action that was taken. In addition, some problems were caused by the MMDetection version, where the detection models had many bugs; therefore, installing the latest version was the easiest solution, but this was not the case. Modifying the different building blocks was the solution, followed by not installing the latest version because it would cause a mismatch in training performance.



## Appendix D

This Appendix shows the different steps and files to be modified to implement MMDetection for the different dataset used for evaluation in both chapter 4 and 5.

### Files to modify

#### *Add collected.py to mmdet /datasets*

The created custom dataset needs to be located in mmdet file and initialized as follow:

```
import os.path as osp
import mmcv
import numpy as np

from mmdet.datasets.builder import DATASETS
from mmdet.datasets.custom import CustomDataset

@DATASETS.register_module()
class EuropCityDataset(CustomDataset):

    CLASSES = ('pedestrian', 'bicycle-group', 'person-group-far-away', 'scooter-group', 'co-rider',
'scooter', \
        'motorbike', 'bicycle', 'rider', 'motorbike-group', 'rider+vehicle-group-far-away', None, \
        'buggy-group', 'wheelchair-group', 'tricycle-group', 'buggy', 'wheelchair', 'tricycle')

    def load_annotations(self, ann_file):
        cat2label = {k: i for i, k in enumerate(self.CLASSES)}
        # load image list from file
        image_list = mmcv.list_from_file(self.ann_file)

        data_infos = []
        # convert annotations to middle format
        for image_id in image_list:
            filename = f'{self.img_prefix}/{image_id}.png'
            image = mmcv.imread(filename)
            height, width = image.shape[:2]

            data_info = dict(filename=f'{image_id}.png', width=width, height=height)

            # load annotations
            label_prefix = self.img_prefix.replace('image', 'label')
            lines = mmcv.list_from_file(osp.join(label_prefix, f'{image_id}.json'))

            content = [line.strip().split(' ') for line in lines]
            bbox_names = [x[0] for x in content]
            bboxes = [[float(info) for info in x[4:8]] for x in content]

            gt_bboxes = []
            gt_labels = []
            gt_bboxes_ignore = []
            gt_labels_ignore = []
```

```

# filter 'DontCare'
for bbox_name, bbox in zip(bbox_names, bboxes):
    if bbox_name in cat2label:
        gt_labels.append(cat2label[bbox_name])
        gt_bboxes.append(bbox)
    else:
        gt_labels_ignore.append(-1)
        gt_bboxes_ignore.append(bbox)

data_anno = dict(
    bboxes=np.array(gt_bboxes, dtype=np.float32).reshape(-1, 4),
    labels=np.array(gt_labels, dtype=np.long),
    bboxes_ignore=np.array(gt_bboxes_ignore,
                           dtype=np.float32).reshape(-1, 4),
    labels_ignore=np.array(gt_labels_ignore, dtype=np.long))

data_info.update(ann=data_anno)
data_infos.append(data_info)
#print(gt_bboxes)
#print(gt_labels)
return data_infos

```

### ***Modify mmdet/datasets/\_init\_.py***

The `_init_.py` should be modified by adding the dataset as follow:

```

from .builder import DATASETS, PIPELINES, build_dataloader,
build_dataset
from .cityscapes import CityscapesDataset
from .coco import CocoDataset
from .custom import CustomDataset
from .dataset_wrappers import (ClassBalancedDataset, ConcatDataset,
                              RepeatDataset)
from .deepfashion import DeepFashionDataset
from .lvis import LVISDataset, LVISV1Dataset, LVISV05Dataset
from .samplers import DistributedGroupSampler, DistributedSampler, GroupSampler
from .utils import (NumClassCheckHook, get_loading_pipeline,
                   replace_ImageToTensor)
from .voc import VOCDataset
from .wider_face import WIDERFaceDataset
from .xml_style import XMLDataset
from .kitti import KittiDataset #afnan
from .collated import CollatedDataset #afnan

__all__ = [ # afnan added dataset
    'KittiDataset', 'CollatedDataset', 'CustomDataset', 'XMLDataset', 'CocoDataset',
    'DeepFashionDataset',
    'VOCDataset', 'CityscapesDataset', 'LVISDataset', 'LVISV05Dataset',
    'LVISV1Dataset', 'GroupSampler', 'DistributedGroupSampler',
    'DistributedSampler', 'build_dataloader', 'ConcatDataset', 'RepeatDataset',
    'ClassBalancedDataset', 'WIDERFaceDataset', 'DATASETS', 'PIPELINES',
    'build_dataset', 'replace_ImageToTensor', 'get_loading_pipeline',
    'NumClassCheckHook'
]

```

### ***Add collated\_detection.py to configs/\_base\_/datasets***

Next is the creating the detection setting file which gives all the parameters such as mean, image scale and flip ratio. It also sits the location of the dataset images.

Following the content of the file:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Jun 30 13:49:09 2022

@author: afnan
"""

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Oct 12 17:13:34 2021

@author: afnan
"""

# dataset settings
dataset_type = 'CocoDataset'
CLASSES = ('pedestrian', 'bicycle-group', 'person-group-far-away', 'scooter-group', 'co-rider',
'scooter', \
    'motorbike', 'bicycle', 'rider', 'motorbike-group', 'rider+vehicle-group-far-away', None, \
    'buggy-group', 'wheelchair-group', 'tricycle-group', 'buggy', 'wheelchair', 'tricycle')
data_root = 'data/Collated/'
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True),
    dict(type='Resize', img_scale=(1920, 1024), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels'])]
```

```

]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(1920, 1024),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(type='Normalize', **img_norm_cfg),
            dict(type='Pad', size_divisor=32),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img']),
        ])
]
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type='RepeatDataset',
        times=8,
        dataset=dict(
            type=dataset_type,
            ann_file=data_root + 'Fusedday_train_all.json',
            img_prefix=data_root,
            pipeline=train_pipeline)),
    val=dict(
        type=dataset_type,
        ann_file=data_root + 'Fusedday_val_all.json',
        img_prefix=data_root,
        pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file=data_root + 'Fusedday_val_all.json',
        img_prefix=data_root,
        pipeline=test_pipeline))
evaluation = dict(interval=1, metric='bbox')

```

## Modify the class\_names.py file

Having the dataset detection file in place it is necessary to ensure that the classes are edited and added in the class name file:

```
# Copyright (c) OpenMMLab All rights reserved.

import mmcv

def kitti_classes(): #afnan
    return [
        'car', 'person', 'cyclist'
    ]
def europcity_classes(): #afnan
    return [
        'pedestrian', 'bicycle-group', 'person-group-far-away', 'scooter-group', 'co-rider', 'scooter', \
            'motorbike', 'bicycle', 'rider', 'motorbike-group', 'rider+vehicle-group-far-away', None, \
            'buggy-group', 'wheelchair-group', 'tricycle-group', 'buggy', 'wheelchair', 'tricycle'
    ]
def collated_classes(): #afnan
    return [
        'pedestrian', 'bicycle-group', 'person-group-far-away', 'scooter-group', 'co-rider', 'scooter', \
            'motorbike', 'bicycle', 'rider', 'motorbike-group', 'rider+vehicle-group-far-away', None, \
            'buggy-group', 'wheelchair-group', 'tricycle-group', 'buggy', 'wheelchair', 'tricycle'
    ]

dataset_aliases = {
    'voc': ['voc', 'pascal_voc', 'voc07', 'voc12'],
    'imagenet_det': ['det', 'imagenet_det', 'ilsvrc_det'],
    'imagenet_vid': ['vid', 'imagenet_vid', 'ilsvrc_vid'],
    'coco': ['coco', 'mscoco', 'ms_coco'],
    'kitti': ['kitti'], #afnan
    'fussed': ['fussed'], #afnan
    'collated': ['collated'], #afnan
    'europcity': ['eurocity'], #afnan
    'wider_face': ['WIDERFaceDataset', 'wider_face', 'WIDERFace'],
    'cityscapes': ['cityscapes'],
    'oid_challenge': ['oid_challenge', 'openimages_challenge'],
    'oid_v6': ['oid_v6', 'openimages_v6']
}

def get_classes(dataset):
    """Get class names of a dataset."""
    alias2name = {}
    for name, aliases in dataset_aliases.items():
        for alias in aliases:
            alias2name[alias] = name

    if mmcv.is_str(dataset):
        if dataset in alias2name:
            labels = eval(alias2name[dataset] + '_classes()')
        else:
            raise ValueError(f'Unrecognized dataset: {dataset}')
    else:
        raise TypeError(f'dataset must a str, but got {type(dataset)}')
```

```
return labels
```

### ***Modify the `mmdet\core\evaluation\__init__.py` file***

The last modification is related to the evaluation in which dataset must be introduced in order to recognise data when called for training and testing the modified file is located in core folder under evaluation folder. Modification should be as follows:

```
from .builder import DATASETS, PIPELINES, build_dataloader, build_dataset
from .cityscapes import CityscapesDataset
from .coco import CocoDataset
from .custom import CustomDataset
from .dataset_wrappers import (ClassBalancedDataset, ConcatDataset,
                               RepeatDataset)
from .deepfashion import DeepFashionDataset
from .lvis import LVISDataset, LVISV1Dataset, LVISV05Dataset
from .samplers import DistributedGroupSampler, DistributedSampler, GroupSampler
from .utils import (NumClassCheckHook, get_loading_pipeline,
                   replace_ImageToTensor)
from .voc import VOCDataset
from .wider_face import WIDERFaceDataset
from .xml_style import XMLDataset
from .kitti import KittiDataset #afnan
from .collated import CollatedDataset #afnan

__all__ = [ # afnan added dataset
    'KittiDataset', 'CollatedDataset', 'CustomDataset', 'XMLDataset', 'CocoDataset',
    'DeepFashionDataset',
    'VOCDataset', 'CityscapesDataset', 'LVISDataset', 'LVISV05Dataset',
    'LVISV1Dataset', 'GroupSampler', 'DistributedGroupSampler',
    'DistributedSampler', 'build_dataloader', 'ConcatDataset', 'RepeatDataset',
    'ClassBalancedDataset', 'WIDERFaceDataset', 'DATASETS', 'PIPELINES',
    'build_dataset', 'replace_ImageToTensor', 'get_loading_pipeline',
    'NumClassCheckHook'
```