ROBUST AI BASED PERCEPTION AND GUIDANCE FOR AUTONOMOUS VEHICLES

CHUYAO WANG

Supervisor: Prof Nabil Aouf

This dissertation is submitted for the degree of

Doctor of Philosophy

CITY UNIVERSITY OF LONDON

Department of Engineering, School of Science & Technology

November 2024

I, Chuyao Wang confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Sign:

Abstract

In the domain of autonomous vehicles, artificial intelligence (AI) now plays a crucial role in achieving fully autonomous systems that can navigate complex and dynamic environments without human intervention. To reach full automation, an autonomous driving system must possess not only the ability to make intelligent decisions but also exhibit high levels of robustness and real-time performance. The core of autonomous driving lies in processing and analysing enormous streams of data generated by a range of sensors, such as cameras, LiDAR, radar, and GPS. AI algorithms take on multiple critical tasks, including perceiving the surrounding environment, detecting and classifying objects, and understanding road conditions. Then the decision-making components determine the optimal course of actions, from lane changes to responding to unexpected obstacles. However, despite the progress made, AI-based autonomous driving systems still face numerous challenges, particularly in ensuring reliability in highly unpredictable environments, both in scene understanding and decision-making. Therefore, developing reliable and robust solutions for these components remains a critical area of research.

This thesis explores the novel deep-learning based methods for the perception and the decision-making modules in learning-based autonomous driving, aiming to improve efficiency and robustness of these modules. The proposed solutions tackle the current challenges and contribute to the overarching goal of achieving full autonomy in autonomous driving systems.

To analyse the elements in the environment, a learning-based approach is proposed for monocular semantic segmentation in urban driving scenarios. This method consists of two components: pyramid fusion spatial attention and fusion channel attention, which are designed to capture contextual dependencies while maintaining a lightweight architecture and achieving state-of-the-art performance. To further locate these elements in the 3D world, a 3D object detection method is proposed that uses only monocular camera input. To compensate for the lack of depth information in monocular images, additional adaptive depth supervision signals are introduced, which also achieve the goal of avoiding excessive computational burdens. Following this, a depth acquisition method is proposed to understand the 3D geometry of the entire scene. As part of this research, a synthetic depth completion dataset is collected by combining LiDAR and stereo camera data, addressing the shortcomings of existing datasets that lack dense ground truth.

To test the efficiency of the depth module, a guidance system is proposed that exploits the strengths of both Imitation Learning (IL) and Deep Reinforcement Learning (DRL). Results demonstrate that incorporating depth images improves the performance of the guidance network. Subsequently, the robustness of a single-agent driving system against adversarial attacks is investigated. This study presents a defence algorithm to mitigate state perturbations, ensuring the concrete robustness of the driving system in worst-case scenarios. Additionally, an explainable attack detector is introduced to accurately predict adversarial attacks and visualise the decision-making process, thereby enhancing the reliability of the proposed robust algorithm. The robustness of the complete approach is demonstrated through several synthetic test cases involving various strong perturbations and domain transfer. Lastly, the robustness is explored in the multi-agent systems. A connected, cooperative multi-agent system is introduced to enhance the efficiency of cooperative tasks in ideal environments. However, the challenges of adversarial attacks escalate significantly in MARL systems compared to single-agent systems, due to the increased complexity of dynamics and information sharing. To solve this, this method follows the idea of constrained objective function introduced in the single-agent case, and further adopt it to the multi-agent context with proposed safety criteria guarantee.

Keywords: Deep Learning, Deep Reinforcement Learning, Computer Vision, Optimisation, Adversarial Attacks, Explainability, Decision Making, 3D Object Detection, Semantic Segmentation, Depth Completion, Attention Module, Imitation Learning.

Table of Contents

Abstract ii

List of Tables ix

List of Figures x

Acknowledgements xiii

Abbreviations xiv

1 Introduction 1

1.1 Motivation 2

1.2 Autonomous Driving 3

1.2.1 Overview 4

1.2.2 AD Scene Understanding 5

1.2.3 Guidance Decision Making 9

1.3 Thesis Objectives 11

1.4 Thesis Structure and Contributions 13

1.5 Published and Under-reviewed Manuscripts 14

2 Background and Tools 16

2.1 Convolutional Neural Networks 17

2.1.1 Basic Concept 17

2.1.2 Advancements in CNN Architectures 19

2.1.3 CNN as Feature Extractor 20

2.2 Deep Reinforcement Learning 22

2.2.1 On-Policy DRL 22

2.2.2 Off-Policy DRL 24

2.3 Loss Functions . . . 26
2.3.1 Classification Loss Functions . . . 26
2.3.2 Regression Loss Functions . . . 27
2.3.3 Segmentation Loss Functions . . . 28
2.3.4 Robust and Advanced Loss Functions . . . 28
2.4 Sensing and Imaging . . . 29
2.4.1 Sensors in Autonomous Driving . . . 30
2.4.2 Transformation in Sensors . . . 31
2.4.3 Stereo Disparity . . . 34
2.4.4 Disparity Post-Filtering . . . 35
2.5 Simulation Tools . . . 36
2.5.1 CARLA (Car Learning to Act) . . . 36
2.5.2 TORCS (The Open Racing Car Simulator) . . . 37
2.5.3 Duckietown . . . 38
2.6 Datasets . . . 39
2.6.1 KITTI Dataset . . . 40
2.6.2 Cityscapes . . . 41
2.6.3 nuScenes . . . 42
2.6.4 Waymo Open Dataset . . . 43

3 Fusion-Attention Monocular Semantic Segmentation 46
3.1 Overview . . . 47
3.2 Related Works . . . 50
3.2.1 Semantic Segmentation . . . 50
3.2.2 Attention Model . . . 51
3.3 Methodology . . . 52
3.3.1 Preliminary . . . 53
3.3.2 Network Structure . . . 53
3.3.3 Pyramid Fusion Spatial Attention . . . 54
3.3.4 Fusion Channel Attention . . . 57
3.3.5 Attention Module Arrangement . . . 60
3.4 Experiments . . . 61
3.4.1 Dataset and Evaluation Standard . . . 61

3.4.2 Implementation Details . . . 62
3.4.3 Results . . . 63
3.5 Summary . . . 68

4 Monocular 3D Object Detection 69
4.1 Overview . . . 70
4.2 Related Works . . . 73
4.2.1 Monocular 3D Object Detection . . . 73
4.2.2 Feature Pyramid . . . 75
4.2.3 Contextual Dependency . . . 75
4.3 Methodology . . . 76
4.3.1 Framework Overview . . . 76
4.3.2 Feature Enhancement Pyramid Module . . . 77
4.3.3 Auxiliary Dense Depth Estimation (ADDE) . . . 80
4.3.4 Augmented centre Depth Estimation (ACDE) . . . 81
4.3.5 Multi-Head Detectors . . . 85
4.3.6 Adopted Training Loss Functions . . . 86
4.4 Experiments . . . 87
4.4.1 Dataset and Evaluation Standard . . . 87
4.4.2 Implementation Details . . . 88
4.4.3 Results . . . 88
4.5 Summary . . . 99

5 Guidance on Depth Completion 102
5.1 Overview . . . 103
5.2 Related Works . . . 106
5.2.1 Deep Reinforcement Learning with Imitation Learning . . . 106
5.2.2 Depth Completion . . . 108
5.3 Methodology . . . 110
5.3.1 Soft Actor-Critic . . . 110
5.3.2 SAC with Imitating Learning . . . 112
5.3.3 Reward Function . . . 114
5.3.4 Depth Completion . . . 115
5.4 Experiments . . . 121

5.4.1 Implementation Details . . . 121
5.4.2 Dataset Collection . . . 122
5.4.3 Results . . . 124
5.5 Summary . . . 127

6 Single-Agent Robustness Against Perturbations 128
6.1 Overview . . . 129
6.2 Related Works . . . 131
6.2.1 Deep reinforcement learning based autonomous driving . . . 131
6.2.2 Adversarial attack on DRL . . . 132
6.2.3 Explainability for Deep Learning . . . 133
6.3 Methodology . . . 134
6.3.1 Framework Overview . . . 134
6.3.2 Markov Decision Process with Perturbation . . . 135
6.3.3 Problem Formulation . . . 136
6.3.4 Optimal Adversary Generation . . . 136
6.3.5 Robust Proximal Policy Optimisation . . . 139
6.3.6 Reward Function Design . . . 145
6.3.7 Attack Detection Network and Explainability . . . 146
6.4 Experiments . . . 148
6.4.1 Simulator and Scene Settings . . . 148
6.4.2 Implementation Details . . . 149
6.4.3 Results . . . 150
6.5 Summary . . . 162

7 Multi-Agent Robustness Against Perturbation 164
7.1 Overview . . . 165
7.2 Related Work . . . 168
7.2.1 MARL for Autonomous Driving . . . 168
7.2.2 Adversarial Attacks on DRL . . . 169
7.2.3 Mitigation Against Adversarial Attacks . . . 170
7.3 Methodology . . . 171
7.3.1 Framework Overview . . . 171
7.3.2 Mean-Field Communicated Multi-Agent Structure . . . 172

7.3.3 Gradient-based Attacker . . . 174
7.3.4 Risk Assessment Formulation . . . 176
7.3.5 Constrained Robust Cooperative-MARL . . . 177
7.3.6 Network Structure . . . 185
7.4 Experiments . . . 186
7.4.1 Implementation Details . . . 186
7.4.2 Results . . . 188
7.5 Summary . . . 195

8 Conclusions and Future Work **197**
8.1 Conclusions . . . 198
8.2 Future Work . . . 202
8.2.1 Perturbation Denoising . . . 202
8.2.2 Sim-to-Real . . . 202
8.2.3 Complete AD Systems . . . 203

References **205**

List of Tables

3.1 Result comparison on the validation set of Cityscapes. 63
3.2 Detailed mIoU on 19 Categories of Cityscapes. 64
3.3 Individual Attention Module Efficiency. 66
3.4 Results on PASCAL VOC 2012 Validation Set. 67

4.1 Results on nuScenes dataset in order of NDS scores. 90
4.2 PAC3DNet mAP Results on each categories. 91
4.3 Inference time testing results. 91
4.4 Ablation studies of proposed components. 95

5.1 Task success rate in various environments and categories. 124
5.2 Performance comparison with state-of-the-art methods. 126

6.1 Hyper-parameters for the experiments. 149
6.2 Evaluation of different algorithms with different epsilon value. 153
6.3 Evaluation in the unseen heavy rain weather condition. 157

7.1 Hyper-parameters for the experiments. 188
7.2 Detailed performance comparison in various attack conditions. 190
7.3 Detailed performance metrics in normal observation condition. 191

List of Figures

1.1 Scene understanding modules in autonomous vehicles. 6
1.2 Deep-learning-based decision making modules for AD. 10

2.1 The example of convolutional operation. 18
2.2 The example of max pooling operation. 18
2.3 The example of ReLU activation. 19
2.4 LiDAR point cloud projection to the image plane. 32
2.5 Disparity in stereo camera system. 34
2.6 The sensor setup of KITTI dataset. 40
2.7 The sample data from KITTI. 40
2.8 The sample data from Cityscapes. 41
2.9 The sensor setup in the nuScenes dataset. 42
2.10 The sample annotations of 3D object detection in nuScenes. 43
2.11 The sensor setup in the Waymo dataset. 44
2.12 The sample LiDAR annotations in Waymo. 44

3.1 Overview of the Fusion Attention Network. 54
3.2 Design of Pyramid Spatial Attention Module. 55
3.3 Design of Channel Attention Module. 58
3.4 Two types of attention module arrangements. 60
3.5 Visual comparison of prediction and ground truth. 65

4.1 Overview of the framework. 76
4.2 Overview of the feature enhancement pyramid module. 77
4.3 Asymmetric Fusion Module. 79
4.4 The Fusion block 1 and 2. 79
4.5 The Lateral Feature Aggregation. 79

4.6 Four top vertices and the centre of a bounding box. 82
4.7 The yaw between the perceived object and world coordinates. 84
4.8 Training progress. 89
4.9 Training progress of depth estimators. 89
4.10 Visualization with and without thresh hold filters. 92
4.11 Visualization of detection results in various distance. 92
4.12 Visualization of detection results in various conditions. 93
4.13 Visualization of detection results in urban and rural areas. 94
4.14 Efficiency visualisation of depth-assisted module. 95
4.15 Attention map visualisation of FEPM module. 97
4.16 Attention map visualisation of different level features from FEPM. . . 98

5.1 Depth annotation comparison of KITTI and the collected dataset. . . 105
5.2 The framework overview of the proposed method. 113
5.3 The pro-processing pipeline for the collected synthetic dataset. 117
5.4 The proposed depth completion network. 118
5.5 Image reconstruction example. 119
5.6 The training environment CARLA Town 02. 122
5.7 Visualisation of the ground truth and the predicted depth maps. . . . 126

6.1 The framework overview. 135
6.2 Adversarial perturbation impacts on a trained agent. 137
6.3 An Adversarial Example. 138
6.4 Network structure of the robust DRL model. 143
6.5 Network structure of our adversarial attack detection model. 147
6.6 Environment for experiments in CARLA. 148
6.7 Training curves of the proposed model and its variant. 150
6.8 The attack success rates at different settings. 151
6.9 Detailed comparisons between the proposed method and baseline. . . 155
6.10 Visualisation of the road in different weather. 156
6.11 Performance comparison in average accumulative reward. 157
6.12 Training curve of the explainable attack detection model. 158
6.13 Saliency maps for steer and throttle actions of under-trained (a) and well-trained (b) policies under normal observations. 159

6.14 Steering saliency maps for the proposed model vs baseline under normal and adversarial observations. 160

6.15 Steering saliency maps under varying adversarial attack strengths ε . 161

7.1 The framework overview. 171

7.2 Illustration of the rhombus area indicating potential collision between two agents. 176

7.3 Network structure of the proposed robust cooperative communicated multi-agent DRL model. 185

7.4 Experiment scenarios with two different intersections. 187

7.5 Comparison of R-CCMARL, its variant, and MAPPO under attacks with $\varepsilon = 0.1$ and $iteration = 20$. 192

Acknowledgements

First, my most sincere and heartfelt gratitude goes to my supervisor, Professor Nabil Aouf, for his invaluable guidance, unwavering support, and insightful feedback throughout the course of my research. His expertise and encouragement have been instrumental in shaping both this thesis and my academic growth. It is my honour to be supervised by him.

I would like to thank my co-supervisor Dr. Abdelhafid Zenati for his continuous advice and discussions.

I would like to thank the team at the Robotics, Autonomy & Machine Intelligence Group (RAMI) at City, University of London, my colleagues Zakaria Chekatta, Ziwei Wang, Duarte Rondao, Jianing Song, Zuyuan Zhu, Leo Pauly, Maxwell Hogan, Abdulla Tammam, Thomas Hickling, Burak Inan, Amar Khan, Xi Guo, and Mohmoud Abdulsalam. Thank you for your camaraderie, support, and the insightful discussions that have enriched my research experience.

Aside from the academic teams, I would like to thank my parents, Sihan and Bin, for their support and encouragement throughout my decision to pursue this PhD. Their belief in me has been a constant source of motivation. I am grateful to Xueting for never ceasing to believe in me and for being there every step of the way.

Abbreviations

2D Two-dimensions

3D Three-dimensions

AAE Average Attribute Error

AD Autonomous Driving

AI Artificial Intelligence

AOE Average Orientation Error

AP True Positive

ASE Average Scale Error

ATE Average Translation Error

AVE Average Velocity Error

Adam Adaptive Moment Estimation

BN Batch Normalisation

CAD Computer-Aided Design

CNN Convolutional Neural Network

DL Deep Learning

DRL Deep Reinforcement Learning

FCN Fully-Connected Network

FOV Field of View

FPN Feature Pyramid Network

FPS Frames Per Second

GPS Global Positioning System

GPU Graphics Processing Unit

IL Imitation Learning

IMU Inertial Measurement Unit

IOU Intersection over Union

MAE Mean Absolute Error

MARL Multi-Agent Reinforcement Learning

NDS NuScenes Detection Score

RGB-D Red-Green-Blue-Depth

RGB Red-Green-Blue

RMSE Root Mean Squared Error

ReLU Rectified Linear Unit

SGBM Semi-Global Block Matching

SGD Stochastic Gradient Descent

WLS Weighted Least Squares

Chapter 1

Introduction

In this introduction chapter, the background context and initial foundations of this thesis are introduced, with an overview of how deep learning techniques are integrated to address the end-to-end autonomous driving problems. Particularly, this thesis mainly focuses on improving the scene understanding capabilities of the vehicles and investigating the robustness of deep-learning-based planning approaches in adversarial environments.

1.1 Motivation

In recent years, deep learning has made significant advancements across various domains, including computer vision, robotics, language processing, and in particular autonomous driving (AD) systems. Autonomous driving represents a complex and ambitious goal in modern technology, aiming to develop vehicles capable of navigating safely and efficiently without any human intervention. This technology has the potential to revolutionise transportation by improving road safety, reducing traffic congestion, and enhancing mobility. The core problem of AD lies in ensuring that these systems can perceive, interpret, and respond to a wide range of real-world driving scenarios, from complex urban environments to unpredictable conditions. Achieving this requires advanced perception and decision-making systems that are both accurate and robust. However, the complexity of real-world environments presents significant challenges for traditional rule-based approaches. To navigate in dynamic, uncertain, and often unpredictable conditions, autonomous vehicles require advanced decision-making capabilities that can continuously learn and adapt. This is where learning-based approaches, particularly deep learning, have shown great promise.

These methods allow systems to learn directly from data, making it possible to generalise across diverse environments and handle complex tasks such as lane detection, object detection, and decision-making under uncertainties. By leveraging large-scale datasets and continuous learning, learning-based approaches offer a promising route to achieve high-level autonomy in driving systems. However, the-state-of-the-art methods have yet providing precise perception and control while operating under stringent performance requirements to ensure safety, real-time processing, and robustness. This field of research still faces numerous challenges.

One of the major challenges lies in RGB-based perception, particularly due to the varying conditions under which images are captured, such as lighting changes, reflections, and various weather conditions. Autonomous driving systems must

interpret sensor data accurately in these conditions, often requiring advanced image processing and perception techniques to maintain performance.

Moreover, the complexity of autonomous driving increases when deployed in complex urban environments, where dynamic interactions with other vehicles, pedestrians, and unpredictable obstacles occur. Handling these scenarios requires a robust and adaptive system capable of real-time decision-making while ensuring safety and efficiency. Inaccurate or delayed decisions can lead to catastrophic consequences, making reliability a serious concern. On top of that, the robustness and generalisation ability of these models in the face of real-world challenges, such as adversarial attacks, perception errors, domain transfer, and dynamic multi-agent interactions, remain critical research areas.

Motivated by the potential and challenges in current autonomous driving research, this thesis aims to enhance the robustness and efficiency of autonomous driving systems by incorporating deep learning techniques. This work focuses on improving the perception module as well as the decision-making modules. Methods for semantic segmentation, 3D object detection, and depth completion are explored to enable autonomous vehicles to better understand their surroundings. For the guidance, the proposed approaches cover both single-agent and multi-agent systems, with a focus on robust deep reinforcement learning (DRL) under adversarial conditions and the application of explainable models to increase transparency and trust in decision-making.

1.2 Autonomous Driving

Automation is increasingly at the forefront of transportation research, with the potential to bring fully autonomous vehicles (AVs) to the roads in the coming years. This evolution represents a transformative shift in how we approach mobility, safety, and urban planning. The development of AD encompasses various technological advancements, including sophisticated sensors and machine learning algorithms that enable vehicles to navigate complex environments without human intervention. This section provides a holistic look at the multifaceted aspects of AD, including the

advancements in technology and investments that support its deployment. It also explores the regulatory frameworks and ethical considerations essential for ensuring the safe integration of AVs into existing transportation systems. Furthermore, the section examines AI-based techniques and components that underpin AD, highlighting their role in enhancing vehicle performance and decision-making capabilities.

1.2.1 Overview

Technology Development and Investment

The technology underpinning AD has advanced rapidly in recent years, driven by innovations in sensors, machine learning, and connectivity. Governments, particularly in the UK, have recognised the potential benefits of connected and autonomous vehicles (CAVs) in enhancing mobility, reducing congestion, and improving road safety. To facilitate this, significant investments have been made through initiatives like the UK's CAV Programme, which has funded various projects aimed at developing and testing autonomous technologies [1]. In the UK, the government has committed over £300 million to CAV research and development since 2015, supporting numerous projects that explore the integration of CAVs into existing transport systems. These investments focus on creating a robust infrastructure that enables safe and efficient autonomous operations, including trials for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. Additionally, partnerships between private companies and academic institutions are fostering innovation, ensuring that the UK remains at the forefront of AD technology [2].

Regulation and Ethics

As the technology for AD evolves, so too must the regulatory frameworks governing its implementation. Policymakers face the challenge of establishing regulations that ensure safety, accountability, and public trust in autonomous vehicles. Ethical considerations are paramount, especially concerning liability in the event of accidents involving autonomous vehicles. Questions arise about who is responsible—the manufacturer, the software developer, or the vehicle owner? [3] The UK government

has been proactive in addressing these issues through the establishment of guidelines and standards for testing and deploying CAVs. The Centre for Connected and Autonomous Vehicles (CCAV) has developed a comprehensive regulatory framework that emphasises safety and risk assessment while promoting innovation. Furthermore, public consultations and ethical reviews are critical to addressing societal concerns about privacy, data security, and the potential impact of autonomous vehicles on employment in driving-related jobs [4].

Safety Concerns for AI-Driven AD

Artificial intelligence plays a crucial role in the development of AD systems, enabling vehicles to perceive their environment, make decisions, and navigate safely. Advanced algorithms analyse vast amounts of data from sensors such as LiDAR, cameras, and Radar to understand complex driving scenarios in real-time. However, the reliance on AI raises safety concerns that must be carefully addressed. The potential for AI systems to malfunction or make erroneous decisions can have serious consequences. Concerns about bias in AI algorithms, the need for transparent decision-making processes, and the robustness of these systems in unpredictable conditions are critical areas of focus [5]. The development of rigorous testing protocols and safety standards is essential to ensure that AI-driven autonomous vehicles can operate safely alongside human drivers. Moreover, stakeholders in the AD ecosystem, including manufacturers, regulators, and researchers, must collaborate to create a framework for ongoing evaluation and improvement of AI systems. This will not only enhance safety but also build public confidence in the technology [6].

1.2.2 AD Scene Understanding

Perception is one of the most critical components of AD systems, as it enables vehicles to observe and understand their surroundings in real time. The perception module is responsible for gathering and processing sensory data to interpret the driving environment, which forms the basis for decision-making and planning as demonstrated in Fig 1.1. Without reliable perception, the vehicle would be unable to detect obstacles, recognise lanes, or anticipate potential hazards, all of which are

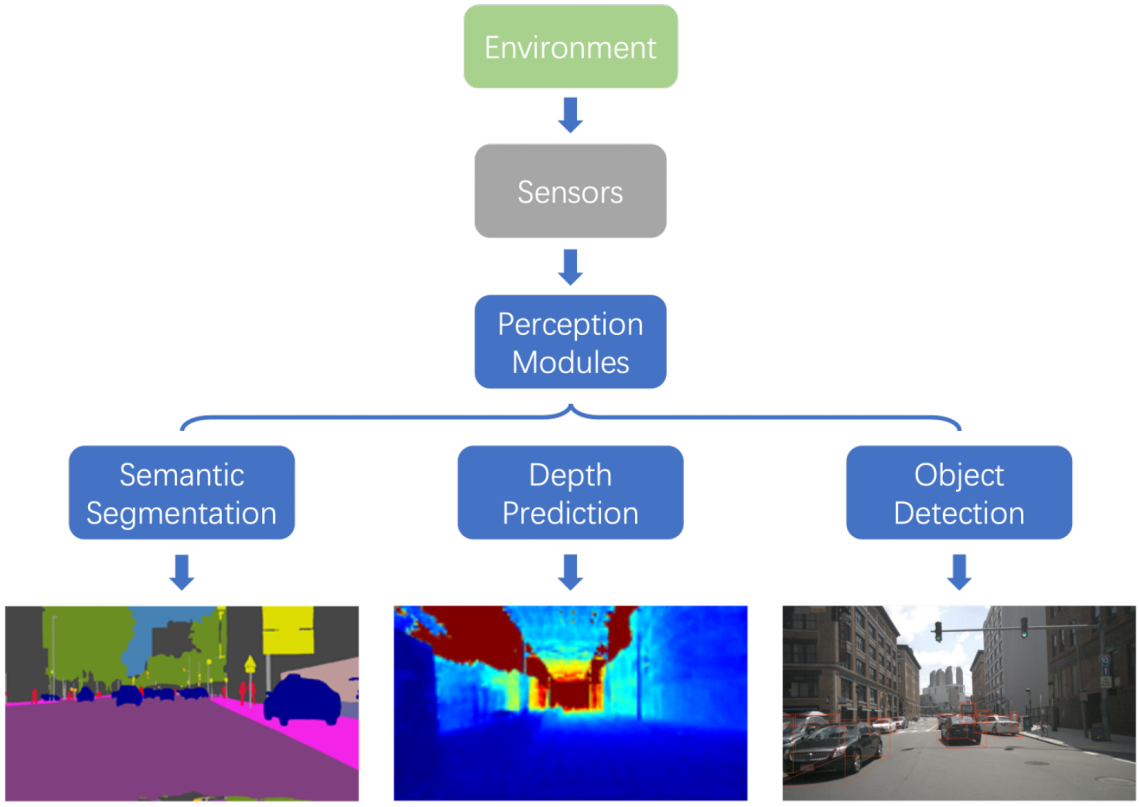

Figure 1.1 Scene understanding modules in autonomous vehicles. The sensors receive raw data from the environment, which is then processed by various perception modules to transform it into meaningful information for the vehicle's next decision-making stage. [Wang et al.]

essential for safe navigation. The core tasks of perception include depth prediction, semantic segmentation, and 3D object detection, each of which plays a vital role in ensuring the autonomous system's situational awareness.

Classic Perception Techniques

Before deep learning, AD perception was relied on explicitly defined algorithms, often using geometry-based approaches and feature engineering. For instance, stereo vision systems used multiple cameras to triangulate and compute depth information. This method relies on identifying matching points between two or more images, which allows for depth reconstruction in structured environments. However, stereo vision struggles in situations with low texture, occlusions, or in environments with poor lighting, such as at night [7]. Similarly, structure-from-motion (SfM) techniques estimate depth from the movement of the camera and its environment, but they require

precise motion estimation and are vulnerable to noise and changing conditions [8]. Other traditional methods, like edge detection (e.g., Canny, Sobel) and optical flow algorithms, focused on identifying motion between frames to infer object movements and road boundaries. These methods typically employed hand-tuned parameters for feature extraction and segmentation. In lane detection, the Hough Transform was widely used to identify lines in images, which worked effectively in well-marked lanes but struggled in cases of faded or non-standard markings [9]. Similarly, feature-based object detection uses techniques like SIFT and SURF to recognise objects through keypoint matching. These methods, while computationally efficient, were prone to failure in the presence of noise, lighting changes, or non-ideal weather conditions like rain or fog.

Although traditional approaches required less data and were more interpretable, they were often brittle and failed to generalise across different environments. They were largely rule-based, meaning they had limited adaptability to unseen or unpredictable situations. In comparison, deep learning methods learn features automatically from large datasets, providing more robust solutions for complex environments. They surpass traditional techniques in perception tasks like object detection, depth prediction, and semantic segmentation by being less dependent on hand-crafted features and better equipped to handle diverse scenarios.

Deep-learning-based Monocular-Based Perception

Of particular importance in this thesis is the focus on deep-learning-based monocular camera-based perception, which offers promising generalisation capability and unique advantages over more expensive and hardware-intensive sensor setups like LiDAR and Radar. Monocular-based perception involves using a single camera to extract depth, scene understanding, and object information from the environment. One of the most significant advantages of monocular systems is their cost-effectiveness, as they only require a single camera compared to multi-sensor setups like LiDAR or stereo camera systems. This significantly reduces both the hardware cost and the power consumption, making monocular setups more suitable for real-time AD applications, especially in cost-sensitive deployments.

Monocular cameras also offer rich boundary and shape information, which is crucial for tasks like lane detection and object segmentation. Moreover, monocular systems are lightweight, easy to integrate, and simpler to calibrate compared to LiDAR or multi-camera systems. Despite their inability to directly measure depth, advancements in deep learning, particularly convolutional neural networks (CNNs), have made it possible to infer depth from 2D images with high accuracy, bridging the gap between monocular systems and more hardware-heavy solutions [10, 11].

Recent research has proposed methods that integrate monocular cameras with deep learning models to perform accurate depth estimation and scene understanding. For instance, Godard et al. proposed an unsupervised learning framework that predicts depth from monocular images by enforcing geometric consistency between different viewpoints, offering a low-cost alternative to stereo systems [11]. Additionally, depth completion techniques, which combine monocular images with sparse depth maps from cheaper sensors like Radar, have further improved the depth prediction capabilities of monocular systems [12]. These developments have significantly enhanced the utility of monocular cameras in AD, enabling precise and efficient depth inference without relying on expensive LiDAR systems.

Semantic segmentation, the task of classifying every pixel in an image, is vital for autonomous vehicles to understand their surroundings. Monocular cameras, combined with deep learning techniques, are well-suited for this task as they provide high-resolution visual data that can be processed to identify objects such as cars, pedestrians, road signs, and lane markings. CNN-based architectures like U-Net and DeepLab have shown remarkable success in segmentation tasks [13, 14]. Monocular semantic segmentation models allow vehicles to better perceive their environment while keeping the system lightweight and efficient.

Monocular 3D object detection aims to detect and localise objects in three-dimensional space using a single camera. While challenging due to the lack of direct depth information, modern deep learning techniques have made it possible to achieve competitive results using monocular inputs. Techniques such as depth-aware feature extraction and geometric constraints are used to improve the accuracy of monocular

3D detection. These methods allow vehicles to estimate the size, position, and motion of nearby objects, even when using only a single camera [15].

1.2.3 Guidance Decision Making

After the perception stage, decision-making modules process the gathered information or raw sensor data to ensure safe and efficient navigation. The objective is to make real-time decisions regarding vehicle actions, such as steering, braking, and accelerating, while adapting to dynamic environments.

Classic Decision Making Techniques

Early approaches to decision-making in autonomous vehicles primarily relied on rule-based systems and behaviour cloning. Rule-based systems involved the use of predefined rules or logic to handle various driving scenarios. For example, systems like finite state machines (FSMs) were used to model different driving behaviours (e.g., lane following, lane changing, stopping at intersections) by transitioning between states based on sensor inputs [16]. These systems were interpretable and easy to implement but lacked flexibility and struggled in unstructured or unpredictable environments. As traffic scenarios became more complex, rule-based methods struggled to generalise beyond their initial design, resulting in brittle performance. Another approach was model-based planning, which often involved optimisation techniques such as Model Predictive Control (MPC). MPC solves a constrained optimisation problem to determine the optimal control actions over a future time horizon, taking into account the vehicle's dynamics and environmental constraints [17]. While effective for low-speed driving or well-structured environments like highways, model-based planning often struggled in dynamic, urban environments with uncertain behaviours from other road users. Furthermore, these techniques require accurate models of vehicle dynamics and are computationally expensive when scaled to complex driving scenarios.

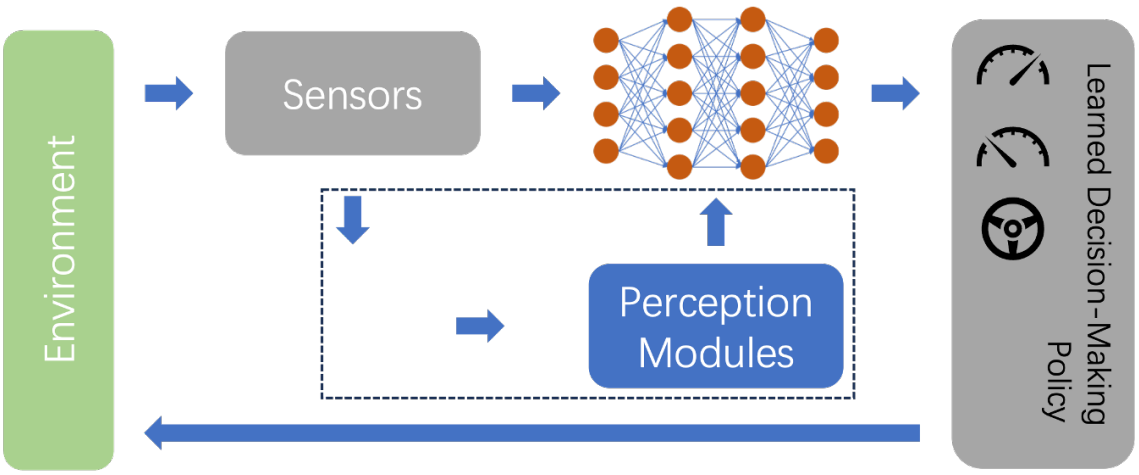

Figure 1.2 Deep-learning-based decision making modules for AD. End-to-end DL models take the feed directly from the sensor or from the interpreted information from the perception modules, and generate actions to interact with the environment. [Wang et al.]

Deep-Learning-based Decision Making Techniques

Over time, decision-making strategies have progressed from rule-based systems to learning-based approaches. Some early machine learning approaches, like behaviour cloning, attempted to learn decision-making policies by mimicking human drivers. These methods, which involved supervised learning of driving actions from human-labelled datasets, were limited by the quality and diversity of the training data. They also suffered from distributional drift, where errors compound over time as the model diverges from the states it was trained on [18]. With advancements in deep reinforcement learning, a more promising potential solution is found for decision making. Reinforcement learning (RL), in particular, offers a framework where an autonomous vehicle learns to make sequential decisions by interacting with the environment and receiving feedbacks in the form of rewards. One well-known method is Deep Q-Networks (DQN), where deep neural networks approximate the action-value function to learn optimal policies [19]. DQN has been extended to handle more complex driving tasks through multi-agent frameworks or continuous action spaces, which are necessary for tasks like lane-changing or negotiating intersections. A key advantage of RL is its ability to learn from trials and errors, allowing the system to encounter all kinds of situations as much as possible. Therefore, RL-

based systems often require extensive training and suffer from sample inefficiency. Techniques like Proximal Policy Optimisation (PPO) and Actor-Critic methods have been introduced to improve the stability and convergence of RL algorithms, making them more practical for real-world applications [20]. In contrast to traditional planning, end-to-end deep learning approaches allow autonomous vehicles to learn decision-making policies directly from raw sensor data like images, without requiring handcrafted features or explicit environment models. These are also referred as model-free methods. For instance, convolutional neural networks (CNNs) are employed to process visual inputs and output driving actions such as steering angles or speed adjustments [21]. These approaches bypass the need for modular perception, planning, and control stages by learning a direct mapping from input to output as shown in Fig. 1.2. However, even with the advancement of end-to-end methods that process raw sensor data for decision-making in AD, deep-learning-based perception modules still significantly enhance performance [22]. Another direction is the use of deep imitation learning (IL), considered as the improved version of behaviour cloning, where the model dynamically learns driving policies by observing expert demonstrations. Methods like Generative Adversarial Imitation Learning (GAIL) have shown success in generating human-like driving behaviours while handling more diverse driving scenarios, which bridges the gap between supervised learning and RL by leveraging expert data while also allowing for learning in interactive environments [23].

1.3 Thesis Objectives

Given the background context introduced in this chapter, the goal of the thesis is clear. In comparison to traditional methods in autonomous driving, deep learning offers a highly promising approach for achieving full automation due to its ability to handle complex and dynamic environments. This thesis aims to address the current limitations in perception and decision-making modules by leveraging advanced deep learning techniques. Specifically, the two key objectives of this work are to:

1. Design deep-learning-based methods for computer vision tasks to enhance the performance and keep light-weight computation of perception modules in autonomous driving systems. This enables better observation and understanding of the driving scenarios that autonomous vehicles encounter.

2. Develop robust and reliable guidance decision-making algorithms that can handle the worst-case conditions. These algorithms should prioritise safety and ensure optimal efficiency, enabling autonomous driving systems to navigate complex environments and adapt to environment changes or adversarial situations.

In detail, considering the computational and practical budget, monocular cameras are primarily used as the sole sensors in this thesis. Compared to LiDAR or Radar, monocular cameras offer a more cost-effective and lightweight solution, making them well-suited for real-time applications in autonomous driving. While LiDAR and Radar provide more direct depth information, monocular cameras offer advantages such as lower cost, rich boundary and shape information, and easier integration into existing systems. However, they also present challenges, such as the need to infer depth from 2D images and the sensitivity to weather conditions, where this thesis makes efforts to addresses through proposed methods. Additionally, LiDAR and binocular cameras are explored for depth completion tasks, providing complementary data to enhance the accuracy and robustness of perception modules in scenarios where monocular cameras may fall short.

For the objective of decision-making, by incorporating the proposed methodology, the goal is to create both single-agent and cooperative multi-agent systems that not only guarantee consistent performance under normal conditions but also demonstrate resilience in the face of worst-case adversarial scenarios. Assumptions are made that the adversarial attackers have the full access to the parameters of the driving systems, which creates the most serious misguidance to them.

1.4 Thesis Structure and Contributions

This thesis consists of eight chapters and the structure and contributions are presented as follows:

Chapter 2 provides a background review for current research for autonomous driving systems, including the knowledge of convolutional neural networks (CNN), deep reinforcement learning (DRL), loss functions used in deep learning, the sensors used and the data transformation between them, computer vision, the simulation tools for experiments, and the open datasets.

Chapter 3 proposes a method for deep-learning-based semantic segmentation using a monocular camera in urban driving scenarios, which is crucial for scene understanding in autonomous vehicles. Semantic segmentation provides more precise subject information than raw RGB images, thereby enhancing the performance of autonomous driving systems. This method introduces a novel dual-attention mechanism that efficiently captures contextual dependencies in high-level features, resulting in more accurate object predictions. Within the dual structure, the proposed pyramid pooling and fusion technique enhances performance while maintaining lightweight computation.

Chapter 4 solves the 3d object detection problem using only single camera input. The introduced feature enhancement pyramid module further enhances the feature representation, where the rich information within benefits the regression networks for individual tasks which help construct the final 3D bounding boxes. By introducing additional adaptive depth supervision signals and auxiliary depth estimator, the missing 3D information in RGB image is compensated, which also avoids bringing computational burdens.

Chapter 5 investigates a Deep Reinforcement Learning (DRL)-based guidance system for ground vehicles, utilising depth images as input. The system operates in two stages: imitation learning followed by reinforcement learning. The depth module, derived from a binocular camera, combines supervised and self-supervised learning

techniques. Experimental results show that using depth images greatly enhances the performance of the guidance network.

Chapter 6 proposes a defence algorithm to mitigate state perturbations, ensuring the robustness of the driving model in worst-case scenarios. The robustness of a single deep reinforcement learning (DRL) agent is tested against state adversarial attacks, which act as directional interference to mislead the agent and can represent adversarial attacks or real-world uncertainties. In addition, an explainable attack detector is introduced to accurately identify the presence of adversarial attacks and provide a visual explanation of the decision-making process, further enhancing the reliability of the proposed algorithm.

Chapter 7 builds on the constrained objective function from Chapter 6, adapting it to the multi-agent context with proposed safety criteria guarantees. The robustness of multi-agent deep reinforcement learning (MARL) is investigated, introducing a connected and cooperative system to improve task efficiency in synthetic urban environments. However, compared to single-agent systems, MARL faces greater challenges from adversarial attacks due to the added complexity of dynamics and the information sharing. The results demonstrate that the proposed method addresses these challenges.

Chapter 8 concludes all proposed frameworks and algorithms for autonomous driving systems, and discusses extension for the future work.

1.5 Published and Under-reviewed Manuscripts

Conferences

1. Wang C, Aouf N. Deep Reinforcement Learning based Planning for Urban Self-driving with Demonstration and Depth Completion[C]//2021 21st International Conference on Control, Automation and Systems (ICCAS). IEEE, 2021: 962-967.

2. Wang C, Aouf N. Fusion attention network for autonomous cars semantic segmentation[C]//2022 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2022: 1525-1530.

Journals

1. Wang C, Aouf N. Depth-Enhanced Deep Learning Approach For Monocular Camera Based 3D Object Detection[J]. Journal of Intelligent & Robotic Systems, 2024, 110(3): 101.

2. Wang C, Aouf N. Explainable Deep Adversarial Reinforcement Learning Approach for Robust Autonomous Driving[J]. IEEE Transactions on Intelligent Vehicles, 2024.

Under-reviewed Journals

Wang C, Aouf N. Robust Multi-Agent Reinforcement learning Against Adversarial Attacks for Cooperative Self-Driving Vehicles. Submitted to IET Radar, Sonar & Navigation and under review.

Chapter 2

Background and Tools

This chapter provides the background knowledge of convolutional neural networks and deep reinforcement learning, as well as the necessary tools in autonomous driving field.

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have been a cornerstone in the development of deep learning, particularly in fields such as computer vision, natural language processing, and more recently, autonomous driving. Introduced by Yann LeCun in the late 1980s, CNNs have evolved significantly, becoming one of the most effective architectures for visual data analysis due to their ability to automatically and adaptively learn spatial hierarchies of features through backpropagation.

2.1.1 Basic Concept

Convolutional Neural Networks are typically composed of multiple specialised layers designed for processing images and spatial data effectively. The main types of layers in a CNN are as follows:

1. **Convolutional Layer**: This layer applies a set of filters to the input image or feature map to capture spatial features. As shown in Fig. 2.1, filters slide over the input data based on the stride, padding, and kernel size, producing feature maps that highlight specific aspects of the input, like edges or textures. Each convolution operation helps to detect patterns within different regions of the image, allowing the network to recognise complex features across layers.

2. **Pooling Layer**: Pooling layers reduce the spatial dimensions of the feature maps, which decreases the number of parameters and computation, while preserving important features. Common types include max pooling, which takes the maximum value in a window, and average pooling, which calculates the average. The pooling operation is demonstrated in Fig. 2.2. Pooling helps make the model more invariant to small changes, like slight shifts or rotations in the input.

3. **Activation Layer**: After convolution, non-linear activation functions, such as ReLU (Rectified Linear Unit), shown in Fig. 2.3, are applied to introduce non-linearity into the model, allowing it to learn complex patterns. This

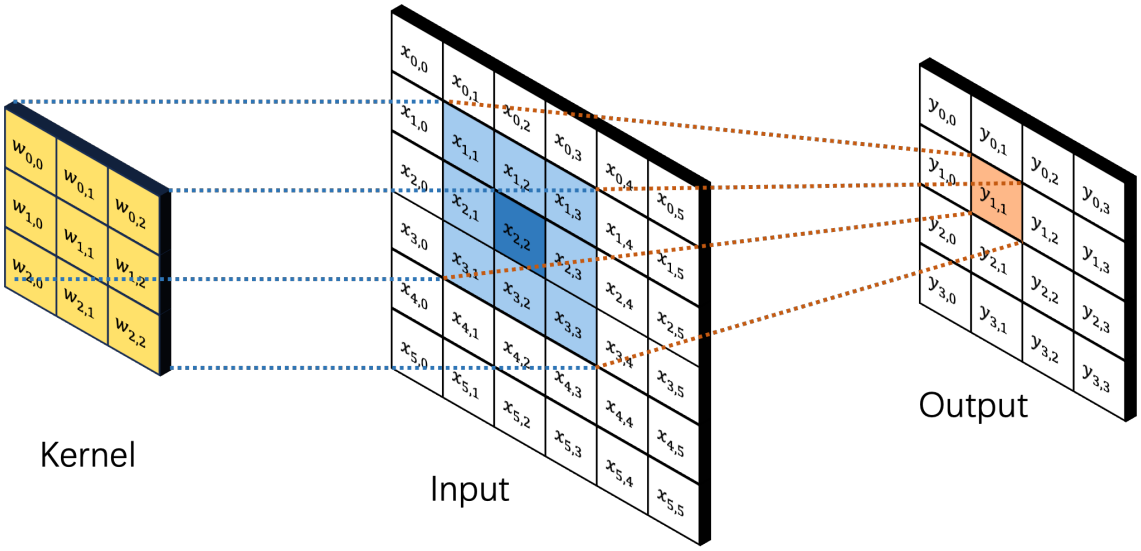

Figure 2.1 The example of convolutional operation. A 3×3 convolutional kernel with stride 1 and padding 0, slides on a 6×6 image/feature map, and results in a 4×4 feature map. [Wang et al.]

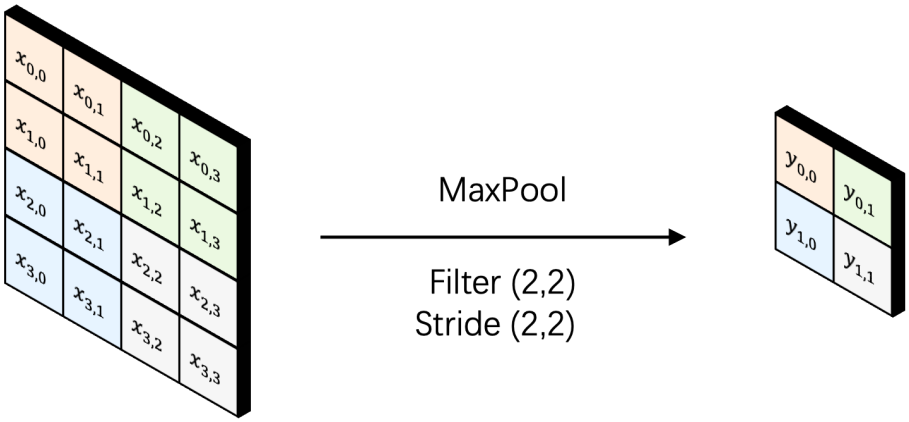

Figure 2.2 The example of max pooling operation. A 2×2 max pooling layer with stride 2 and padding 0, slides on a 4×4 image/feature map, and results in a 2×2 feature map. [Wang et al.]

activation layer enables CNNs to approximate any function, making them adaptable to various tasks.

4. **Fully Connected (Output) Layer**: In the final stages, fully connected layers consolidate the features detected by previous layers to make the final prediction. This layer typically connects every neuron from the previous layer to each neuron in the next, and it often concludes with a softmax or sigmoid activation, providing the probabilities for classification or regression tasks.

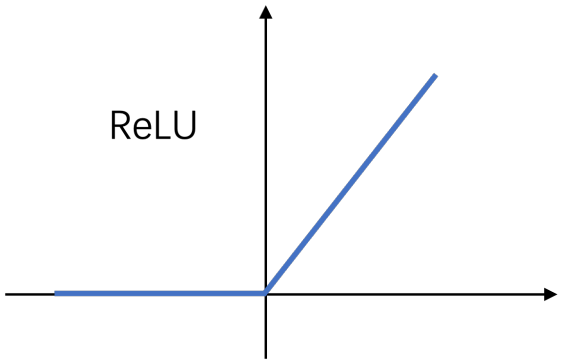

Figure 2.3 The example of ReLU activation. The output is only activated when the values in the input is over 0. [Wang et al.]

In a convolutional neural network, usually multiple combinations of a convolutional layer, a pooling layer, an activation layer are stacked sequentially, allowing the network to learn increasingly complex features from the input data. Each convolutional layer applies several filters (kernel) to the input, generating feature maps that highlight different aspects of the data. Following the convolutional layers, pooling layers are interspersed to downsample these feature maps, reducing their spatial dimensions while preserving the most salient information. This combination can be repeated multiple times. After a series of convolutional and pooling layers, the output from these layers is flattened and passed through one or more fully connected layers, leading to the final output layer that produces predictions. The connections between these layers facilitate the hierarchical extraction of features, enabling the CNN to learn from raw pixel values to high-level representations, ultimately enhancing performance in tasks like image classification and object detection.

2.1.2 Advancements in CNN Architectures

The early CNN, known as LeNet-5 [24], was developed for digit recognition tasks such as handwritten digit classification for postal codes. It laid the foundation for the modern architecture of CNNs, which consist of three main types of layers as mentioned before: convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters to input data to automatically extract feature maps, whereas traditional methods rely on human knowledge and experience to

determine the features. This enables the possibility of learning the feature extraction by the deep neural network itself for later researches.

One of the pivotal moments in the success of CNNs was the breakthrough achieved by AlexNet [25] in 2012. AlexNet, developed by Krizhevsky et al., won the ImageNet competition with a large margin, demonstrating the power of deep architectures when combined with GPU acceleration and large datasets. This success spurred significant research interests in CNNs. Since AlexNet, CNN architectures have evolved rapidly. VGGNet (2014) [26] introduced the idea of using small (3x3) filters in deeper networks, emphasising the depth of the network as a key factor in feature extraction. GoogLeNet (2015) [27] proposed an Inception module, which utilises multiple convolutional filters of different sizes in parallel to capture multi-scale features, reducing the computational cost while maintaining performance. ResNet (2016) [28] tackled the degradation problem in deep networks by introducing residual learning through skip connections, allowing CNNs to be trained with significantly more layers.

In recent years, architectures like DenseNet [29] and EfficientNet [30] have further refined CNN efficiency. DenseNet connects each layer to every other layer in a feed-forward manner, improving feature propagation and reducing the vanishing gradient problem. EfficientNet, on the other hand, balances network depth, width, and resolution using a compound scaling method, enabling a significant reduction in parameters while improving accuracy.

2.1.3 CNN as Feature Extractor

To this day, as both industry and academia strive to tackle increasingly complex tasks with large datasets, handcrafted CNNs often fall short of the required performance. Consequently, using a backbone as a feature extractor has emerged as a common approach, applicable not only in computer vision tasks but also in deep reinforcement learning (DRL) networks. The choice of backbone is critical and can be categorised into three main types based on performance and efficiency.

Large backbones, such as ResNet-101 and ResNet-50, are designed to prioritise performance. ResNet-50 has 50 layers and approximately 23 million parameters, resulting in a model size of around 98 MB, while ResNet-101 boasts 101 layers with approximately 44 million parameters, leading to a size of about 167 MB [31, 28]. These architectures are renowned for their depth and the incorporation of residual connections, which facilitate the training of deep networks by allowing gradients to flow more effectively during backpropagation. ResNet-101 is particularly popular for image classification tasks, achieving state-of-the-art results on benchmark datasets like ImageNet [32].

Middle-scaled backbones, exemplified by VoVNet-V2 [33], strike a balance between performance and lightweight design. Similar to ResNet, VoVNet-V2 can have various configurations, with the smallest version containing about 5 million parameters and a model size of around 20 MB, achieving a balance between accuracy and efficiency. VoVNet-V2 utilises a dynamic scaling approach, allowing the model to adjust based on input resolution, thus maintaining high accuracy while reducing computational overhead. Its improved bottleneck structure enables the extraction of multi-scale features, making it effective for applications such as semantic segmentation, where understanding context and fine details is crucial. The architecture has shown to outperform other models in various tasks while requiring fewer parameters, making it suitable for real-time applications.

Lightweight backbones, such as MobileNet [34], focus on efficient computation, making them ideal for deployment in mobile and edge devices where resource constraints are prevalent. MobileNet employs depthwise separable convolutions, which divide the convolution operation into two layers, significantly reducing the number of parameters (around 5 million for MobileNetV1) and leading to a model size of approximately 16 MB, while maintaining competitive performance. This efficiency enables real-time processing, which is essential for applications such as object detection and facial recognition on mobile platforms.

The selection of an appropriate backbone plays a pivotal role in optimising feature extraction across various applications. Large backbones offer superior performance

for complex tasks, middle backbones like VoVNet-v2 provide a balanced approach with efficient feature extraction, and lightweight backbones like MobileNet ensure that computational efficiency is prioritised without severely sacrificing accuracy. Understanding the trade-offs between accuracy, speed, and resource consumption is key in optimising performance in autonomous driving applications.

2.2 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) combines reinforcement learning (RL) principles with deep neural networks, allowing agents to learn complex, high-dimensional policies for sequential decision-making tasks. Central to DRL is the Markov Decision Process (MDP) [35], which provides the mathematical framework for decision-making. In an MDP, an agent interacts with an environment in discrete time steps, moving through states s_t by choosing actions a_t that lead to rewards r_t and transitions to new states s_{t+1}. Formally, an MDP is defined by the tuple (S, A, P, R, γ), where S and A are the sets of states and actions, $P(s_{t+1}|s_t, a_t)$ is the transition probability function, $R(s, a)$ is the reward function, and γ is the discount factor, which a bigger one prioritises future rewards, while a smaller one values recent rewards. An episode in DRL refers to the sequence of steps an agent takes in an environment, starting from the initial state and ending upon success or failure defined by specific criteria. The primary objective of DRL is to update the agent network parameters in a way that maximises cumulative rewards throughout each episode, which means the agent is making optimal decisions.

Based on how agents learn from interactions with the environment, DRL methods are usually categorised into on-policy and off-policy learning approaches

2.2.1 On-Policy DRL

In on-policy learning, the current policy of the agent is the same policy that the agent is improving during training. This means the agent collects data based on the current policy and updates itself to optimise this specific policy. The advantage of on-policy methods is the stability, as they directly evaluate and update the current

policy without needing to account for older experiences. However, the downside is that they are data-inefficient, as they require fresh trajectories with each update, making them costly in environments with limited access to the interactions [36] as the older trajectories are discarded. Key developments in on-policy algorithms are discussed as follows.

State-Action-Reward-State-Action (SARSA)

SARSA [37] is a straightforward on-policy algorithm where the agent learns the Q-value function based on the action taken by the current policy. This approach follows the "SARSA" trajectory: after observing a state (s_t), the agent chooses an action (a_t), receives a reward (r_t), and then observes the next state (s_{t+1}) and action (a_{t+1}). SARSA updates the Q-value using this experience, ensuring that the agent refines its action choices based on ongoing exploration. It is known for being relatively stable in noisy or uncertain environments, as its updates come directly from the current policy of the agent. However, a significant drawback of SARSA is its slower convergence and potential for suboptimal performance in more complex tasks. Since SARSA only evaluates actions directly linked to its current policy, it may miss out on learning more optimal actions outside of this immediate scope, which can limit its effectiveness in exploration-heavy scenarios or complex tasks requiring broader generalisation.

Trust Region Policy Optimization (TRPO)

TRPO [38] is notable for providing stability and performance consistency in complex environments. It improves training stability by enforcing a trust region constraint that limits the extent of policy changes with each update. This gradual adjustment helps avoid drastic, unstable updates, making TRPO ideal for tasks where reliable performance improvements are crucial. Another advantage of TRPO is its capacity to achieve monotonic policy improvement, which ensures each iteration results in better or at least non-worsening performance, an asset in environments requiring consistent learning. However, the downside is that TRPO can be computationally intensive due to its second-order optimisation, making it slower than simpler algorithms

and challenging to implement and tune effectively in large-scale applications. The theoretical foundation on trust regions has made it a foundation for robust on-policy training methods, influencing the development of PPO.

Proximal Policy Optimization (PPO)

PPO [20] is an advanced on-policy algorithm that is designed to balance the stability of TRPO with greater computational efficiency. PPO achieves this by using a simpler, first-order optimisation approach, adjusting the policy in small steps and clipping the probability ratio of the new to old policies to keep updates within a proximal range. This clipping prevents overly large policy updates, which helps maintain stability and prevents divergence, similar to the trust region of TRPO but without the need for complex second-order calculations. Additionally, it also allows multiple epochs of updates with mini-batch samples, enhancing sample efficiency. Therefore, PPO is widely appreciated for its versatility, as it performs well in various environments, making it one of the most popular algorithms in continuous and discrete action spaces. However, the algorithm does have some trade-offs. Though generally more efficient than TRPO, PPO still requires a significant amount of data, as it discards old trajectories after each update. Additionally, while the clipping mechanism is effective, it introduces a hyper-parameter (the clipping threshold) that needs tuning to balance exploration and stability, which can be sensitive in different tasks. Despite these challenges, its balance of simplicity, efficiency, and reliable performance has led it to become a standard algorithm in many reinforcement learning applications, including robotics and autonomous driving environments.

2.2.2 Off-Policy DRL

In contrast, off-policy DRL algorithms allow agents to learn from past experiences stored in a replay buffer, which contains actions and rewards that may differ from the current policy of the agent. Q-Learning [39] and Deep Q-Networks (DQN) [36] are classic off-policy methods, which use experience replay to sample from previous experiences and learn optimal action values $Q(s, a)$, independent of the policy followed at the time of collection. Off-policy algorithms are more data-efficient since they can

learn from older data and continue to improve without discarding past interactions. However, instability is introduced as involving high-variance approximations when training with old data [40]. Key developments in off-policy algorithms are discussed as follows.

Deep Q-Network (DQN)

DQN enhances Q-Learning by using a neural network to approximate the Q-function, allowing it to handle complex and high-dimensional state spaces. DQN introduced two main innovations: experience replay, where experiences are stored in a buffer and sampled randomly for training, and target networks, which stabilise Q-value updates by keeping a separate network for Q-target calculations. These enhancements led to remarkable successes in environments like Atari games, where DQN achieved near-human performance across multiple games. DQN has since been widely used for sequential decision-making tasks in gaming and simplified autonomous driving scenarios. However, DQN still has challenges, such as inefficiency in continuous action spaces and instability in environments with sparse rewards.

Deep Deterministic Policy Gradient (DDPG)

DDPG [41] is an important advancement in DRL for continuous action spaces, which traditional Q-learning cannot handle effectively. DDPG extends the Q-learning approach by combining an actor-critic architecture with deterministic policy gradients, making it particularly well-suited for tasks such as autonomous driving, where precise control in continuous action spaces is crucial. In DDPG, two networks operate in parallel: an actor network that directly outputs actions based on the current policy and a critic network that estimates Q-values for state-action pairs. The actor network learns a deterministic policy, generating specific actions for each observed state rather than probability distributions over actions, simplifying the gradient calculation process. A defining feature of DDPG is the use of target networks for both actor and critic networks, which are slowly updated versions of the main networks. These target networks improve stability by making conservative updates and reducing fluctuations in Q-value estimates.

Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 [40] is an enhanced algorithm that addresses the limitations of DDPG by introducing twin Q-networks to mitigate overestimation bias associated with value function approximation. By taking the minimum Q-value from these two networks when updating the policy, TD3 effectively avoids the instability in training. Additionally, TD3 introduces target policy smoothing to further enhance the reliability of its action selection process. This technique involves adding noise to the target policy during updates, making the Q-value estimates less sensitive to minor changes in the actions. This smoothing effect encourages exploration while ensuring that the policy remains stable and consistent, which is especially critical in continuous control tasks where precision is paramount. By combining the strengths of actor-critic methods with these advanced techniques, TD3 represents a significant step forward in the field of deep reinforcement learning, offering improved convergence properties and overall reliability in continuous action spaces.

2.3 Loss Functions

Loss functions play a critical role in deep learning, which provide feedback to the learning model during training by measuring how well the predictions of the model align with the expected outputs. In supervised learning, loss functions help minimise the difference between the predicted and true labels, guiding the backpropagation of errors through gradient descent or its variants. Common loss functions are typically grouped into categories such as classification, regression, and advanced tasks like object detection and generative models.

2.3.1 Classification Loss Functions

For classification tasks, where the goal is to assign an input to one of several categories, loss functions like Cross-Entropy Loss and Hinge Loss are most commonly used.

Cross-Entropy Loss (Log Loss): This is one of the most widely used loss functions in classification problems, particularly with softmax outputs. It measures

the distance between the predicted probability distribution and the true distribution (one-hot encoded labels). Cross-entropy penalises large deviations from the correct classification and encourages the model to output high probabilities for correct classes. Cross-entropy loss is given by:

$$L = -\sum_{i} y_i \log(\hat{y}_i) \tag{2.1}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability. It has been foundational in training deep neural networks, especially for image classification tasks.

Hinge Loss: Commonly used in support vector machines (SVMs), Hinge Loss aims to ensure that the margin between classes is maximised. Unlike cross-entropy, hinge loss is more suitable for binary classification and encourages a robust margin between classes:

$$L = \sum_{i} \max(0, 1 - y_i f(x_i)) \tag{2.2}$$

where y_i is the true class label and $f(x_i)$ is the predicted score.

2.3.2 Regression Loss Functions

For regression tasks, where the goal is to predict a continuous value, loss functions such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) are commonly employed.

Mean Squared Error (MSE): This is the most commonly used loss function for regression. It measures the average of the squared differences between the predicted and actual values:

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.3}$$

MSE heavily penalises larger errors due to squaring, which makes it sensitive to outliers. However, it remains popular due to its smooth gradients, which make optimisation easier.

Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual values:

$$L = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.4}$$

MAE is less sensitive to outliers compared to MSE, but its gradient is not smooth near zero, which can make optimisation less stable.

2.3.3 Segmentation Loss Functions

For tasks like semantic segmentation, where pixel-wise classification is required, loss functions such as Dice Loss and Jaccard Loss are frequently used, especially for imbalanced data scenarios.

Dice Loss: Originally used in medical image segmentation, Dice Loss is a measure of overlap between the predicted segmentation and the ground truth. It is particularly useful in cases of class imbalance:

$$L = 1 - \frac{2 \times |X \cap Y|}{|X| + |Y|} \tag{2.5}$$

where X, Y are the sets of predicted pixels and ground truth pixels, respectively. Dice Loss helps improve the overlap of segmented regions with ground truth regions, which is critical for precise segmentation.

Jaccard Loss (Intersection over Union): Similar to Dice Loss, Jaccard Loss focuses on the overlap between the predicted and actual segmented areas. However, it tends to penalise mismatches more harshly:

$$L = 1 - \frac{|X \cap Y|}{|X \cup Y|} \tag{2.6}$$

2.3.4 Robust and Advanced Loss Functions

Recent research has focused on developing loss functions that enhance the robustness and efficiency of deep learning models. For example:

Focal Loss: Focal Loss [42] is designed to address the imbalance between easy and hard examples in tasks like object detection. By down-weighting well-classified examples, it forces the model to focus on hard, misclassified cases:

$$L = -\alpha(1-\hat{y})^{\gamma}\log(\hat{y}) \tag{2.7}$$

Here, γ is a tunable focusing parameter that adjusts the weight on hard-to-classify examples.

Huber Loss: A hybrid between MSE and MAE, Huber Loss [43] is less sensitive to outliers than MSE and more stable than MAE. It is particularly effective for regression tasks involving noisy data:

$$L = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \leq \delta \\ \delta \cdot (|y - f(x)| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{2.8}$$

where δ is a threshold parameter that determines the transition point between the two cases.

Loss functions have continued to evolve alongside the growth of deep learning applications. Research is increasingly focusing on robust loss functions that handle noisy labels and adversarial examples, as well as loss functions tailored to specific tasks, such as contrastive loss for self-supervised learning and triplet loss for metric learning. Additionally, multi-objective loss is also gaining traction, as it enables models to optimise for multiple objectives simultaneously, thereby improving overall performance.

2.4 Sensing and Imaging

In the realm of autonomous driving systems, sensing technology is fundamental in perceiving the environment accurately and safely. Three prominent technologies are LiDAR, monocular cameras, and stereo cameras, each with distinct characteristics, advantages, and limitations.

2.4.1 Sensors in Autonomous Driving

LiDAR (Light Detection and Ranging) is a sophisticated sensor technology that utilises laser pulses to precisely measure distances between the sensor and various objects in the surrounding environment. By emitting laser beams and analysing the time it takes for the light to return after reflecting off surfaces, LiDAR generates highly accurate 3D point clouds that represent the spatial layout of the environment. Typical LiDAR systems have a range of up to 200 metres and can operate effectively in diverse lighting conditions. For instance, sensors such as the Velodyne HDL-64E offer a vertical field of view (FOV) of approximately 26.8 degrees and a full horizontal FOV of 360 degrees, allowing them to capture detailed information about their surroundings [44]. LiDAR systems generally operate at a frequency of around 900 nm, which contributes to their high level of accuracy (± 2 cm) in distance measurement. Despite these advantages, LiDAR systems can be expensive, often costing several thousand dollars, and may face significant challenges in adverse weather conditions, such as heavy rain or fog, which can scatter the laser beams and compromise measurement accuracy [45].

Monocular Cameras employ a single lens to capture 2D images of the environment, typically offering a field of view (FOV) ranging from 60 to 120 degrees. This wide FOV enables them to capture a broad area in a single frame, with resolution varying based on specific application requirements. Monocular cameras are cost-effective, lightweight, and relatively easy to integrate into vehicles, making them a practical choice for many autonomous driving applications. Additionally, they provide rich visual information, including colour, texture, and scene detail, which are valuable for tasks such as object classification and semantic segmentation [46]. However, inferring depth from monocular cameras presents significant challenges. Unlike stereo vision or LiDAR, which rely on multiple perspectives or active sensing, monocular cameras must estimate depth using advanced computer vision techniques. These methods often struggle with issues such as perspective distortion, where objects at varying distances appear differently scaled, and occlusions, where closer objects block the view of those farther away. These limitations can reduce the accuracy of depth

estimation, particularly in complex urban environments. Despite these weaknesses, monocular vision remains a compelling choice for tasks where lightweight design, affordability, and rich image data outweigh the constraints of depth perception.

Stereo Cameras consist of two or more cameras positioned at same or different angles, mimicking human binocular vision. By comparing the images from each camera, stereo vision systems can calculate depth information through triangulation, providing a more robust solution for depth perception compared to monocular cameras. These systems typically have a horizontal FOV similar to that of monocular cameras but can offer more accurate depth perception within a range of about 10 to 50 meters. This technology enhances object detection capabilities and facilitates the generation of 3D maps of the environment. However, stereo cameras can be more complex to calibrate and process, requiring significant computational resources to handle the data they produce [47].

2.4.2 Transformation in Sensors

In autonomous driving, it is common to use multiple types of sensors. Sensor fusion is important as it enables autonomous driving systems to utilise different types of data, strengthening the deep learning models for various tasks. Specifically, the integration of LiDAR and camera data is crucial for creating comprehensive environmental representations. This process often involves projecting LiDAR point clouds onto the camera plane, resulting in RGB-D images. The principle is demonstrated in Fig. 2.4.

Point Cloud Projection

Coordinate Transformation: The first step is to align the coordinate systems of the LiDAR and the camera. This is achieved through a calibration process that determines the extrinsic parameters between the two sensors. The transformation is typically represented using a transformation matrix T that converts LiDAR coordinates to camera coordinates:

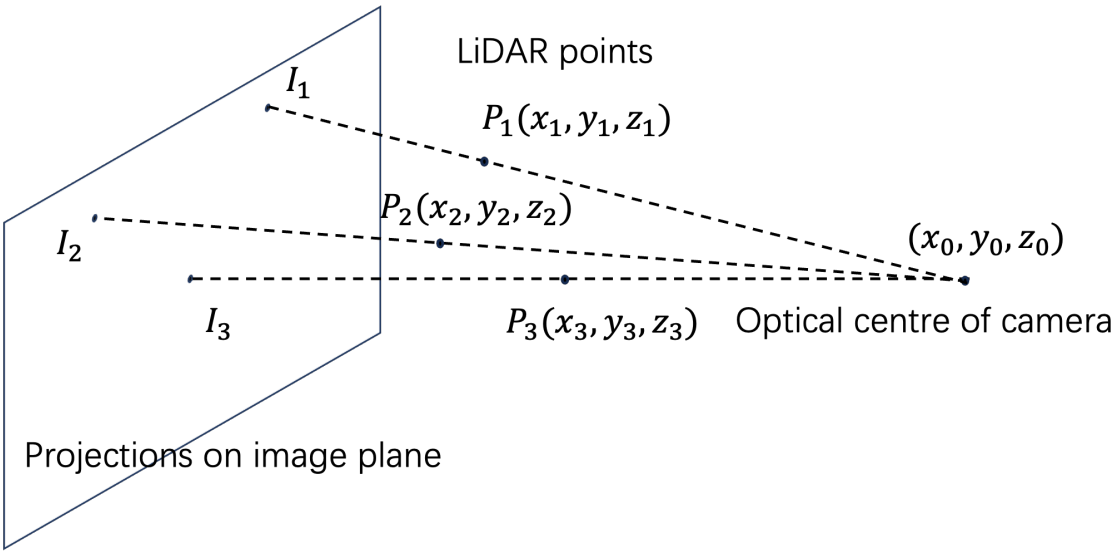

Figure 2.4 LiDAR point cloud projection to the image plane. [Wang et al.]

$$T = [R|t] = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_x \\ R_{2,1} & R_{2,2} & R_{2,3} & t_y \\ R_{3,1} & R_{3,2} & R_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

where R is the rotation matrix that defines the orientation of one frame relative to another. In the context of LiDAR and camera frames, R describes how to rotate points from the LiDAR frame to align with the camera orientation. t is the translation vector that defines the positional offset between the two coordinate frames. It shifts the origin of the LiDAR frame to match the position relative to the camera frame. The last row $[0 \quad 0 \quad 0 \quad 1]$ is called homogeneous row that ensures that the transformation can handle translations when working with 3D homogeneous coordinates.

Projection: Once the coordinates are aligned, the 3D points from the LiDAR can be projected onto the 2D camera image using the camera intrinsic parameters. The projection is done using the pinhole camera model, which can be expressed

mathematically as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K_{in} \cdot T \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.10}$$

where (u, v) are the pixel coordinates on the image, K_{in} is the intrinsic matrix of the camera, T is the LiDAR to Image transformation matrix, and (X, Y, Z) are the LiDAR point coordinates. K_{in} defines the transformation from 3D camera coordinates to 2D image coordinates. This matrix contains parameters that account for the camera internal characteristics, such as focal length and optical centre, which are typically fixed once the camera is manufactured. K_{in} is given by:

$$K_{in} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

f_x and f_y represent the focal lengths of the camera along the x and y axes, respectively. They scale the 3D points based on the distance to the camera sensor and are typically measured in pixels. c_x and c_y denote the optical centre (principal point) in the image plane, indicating where the optical axis intersects the sensor. This centre is essential for correct alignment, especially when the camera is not perfectly centred. $[0 \quad 0 \quad 1]$ is the homogeneous row. The RGB-D images are then obtained.

RGB-D information is valuable in applications such as autonomous driving. The RGB component captures detailed visual data on colour and texture, supporting tasks like object detection, classification, and semantic segmentation. The depth component, meanwhile, provides precise spatial information on the distance between the camera and objects, enhancing 3D scene understanding and spatial accuracy, which could also benefit the visual tasks.

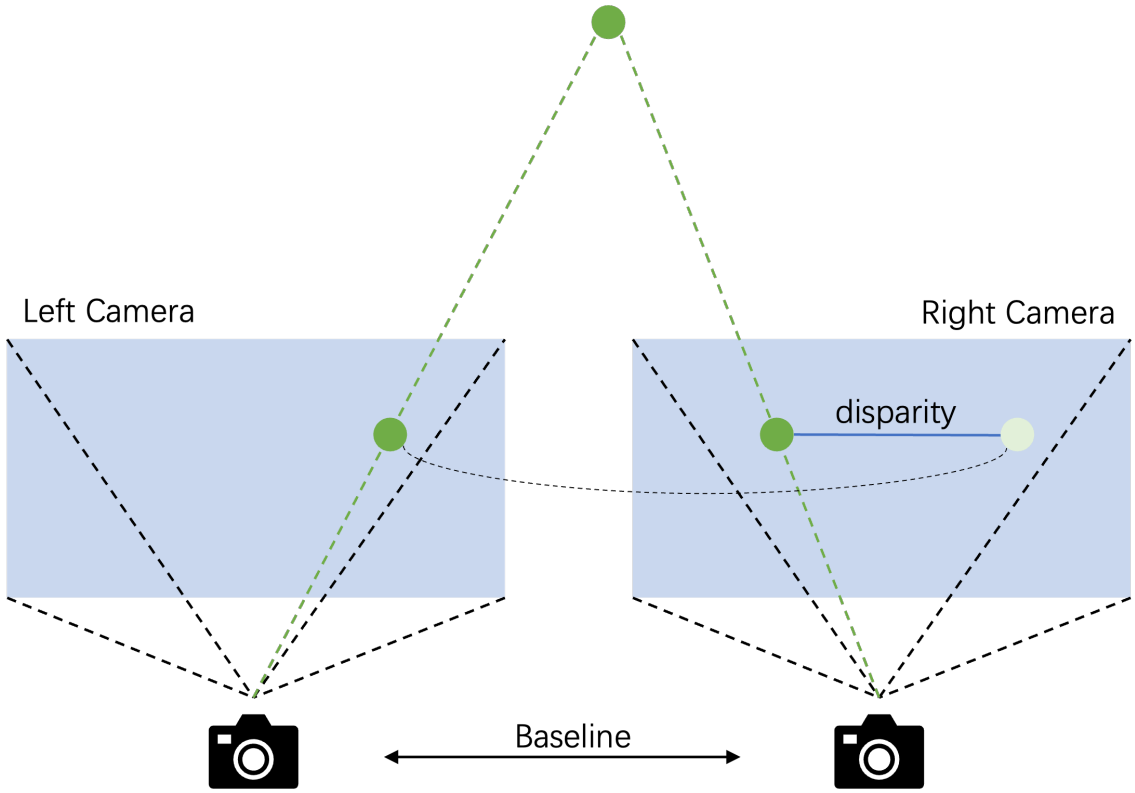

Figure 2.5 Disparity in stereo camera system. Due to the different perspective, the same object appears in different positions in the left and right cameras, the difference is disparity. [Wang et al.]

2.4.3 Stereo Disparity

Besides LiDAR, stereo cameras also offer alternative way to acquire 3D information. Stereo disparity refers to the difference in the horizontal position of a point in two images captured by stereo cameras, as shown in Fig. 2.5. This disparity enables depth perception by triangulating the location of objects based on the offset between the two images, simulating human binocular vision.

In stereo vision, the two cameras set apart by a baseline distance B, the disparity d can be defined as the difference between the x-coordinates of a point projection in the left and right images:

$$d = x_L - x_R \tag{2.12}$$

where x_L and x_R are the horizontal coordinates of the same object in the left and right image planes. The depth Z of the object can be computed using the following formula:

$$Z = \frac{f \cdot B}{d} \tag{2.13}$$

where f is the focal length of the camera. In this formula, higher disparity (closer points) results in a smaller depth value, indicating proximity to the cameras, whereas lower disparity (farther points) corresponds to greater depth. Advanced techniques, such as block matching (StereoBM) and semi-global block matching (SGBM) [48], further improve the calculation of depth information based on disparities, by matching small image blocks between the left and right images to find corresponding pixels.

2.4.4 Disparity Post-Filtering

Current stereo matching algorithms can calculate disparity maps in real time, enabling rapid depth estimation for applications like autonomous driving. However, these algorithms often struggle in challenging scenarios, such as scenes with uniform, textureless surfaces, areas of occlusion where objects block portions of the scene, or regions with strong reflections. These issues can lead to inconsistent or inaccurate disparity maps, which reduce the reliability of the depth information.

To address these issues, post-filtering techniques are applied to refine disparity maps. The Weighted Least Squares (WLS) filter is commonly used in disparity post-processing to enhance disparity maps while reducing noise. WLS operates by minimising the difference between the observed pixel values and the expected pixel values, where weights are applied to each pixel according to its spatial characteristics. This helps in preserving important features of the image, such as edges, while smoothing out noise. The weights in the WLS filter can be derived from various sources, such as the local gradient magnitude, which emphasizes edge-preserving characteristics. This means that regions with high gradient values (edges) receive higher weights, leading to less smoothing in those areas compared to smoother regions.

The WLS optimisation problem can be formulated as follows:

$$E = \sum_{i,j} w_{i,j}(I_{i,j} - g_{i,j})^2 \tag{2.14}$$

where E is the energy to minimise, $I_{i,j}$ is the input image pixel value, $w_{i,j}$ is the weight for each pixel. By minimising E, the output pixel value $g_{i,j}$ will generate a smoother disparity map, thus improving the final depth transformation.

2.5 Simulation Tools

Simulation tools play a crucial role in the development of autonomous driving systems, as conducting experiments in real-world environments presents numerous challenges, including safety concerns, high costs, and the variability of driving conditions. These simulators provide a controlled and flexible alternative, allowing researchers to develop and test algorithms and vehicle dynamics under a wide range of scenarios that would be difficult or impossible to replicate in reality.

2.5.1 CARLA (Car Learning to Act)

CARLA [49] has emerged as one of the most widely used open-source simulators for autonomous driving research. Developed by Intel Labs and the Computer Vision Center in Barcelona, it offers a highly realistic simulation environment specifically tailored for self-driving car applications. CARLA virtual environment replicates urban driving scenarios with high-fidelity, featuring multiple towns, complex road networks, intersections, roundabouts, and dynamic objects such as pedestrians, cyclists, and vehicles. The simulation also supports various weather conditions and lighting scenarios, which makes it ideal for training autonomous systems to handle diverse driving conditions.

For deep reinforcement learning (DRL) applications, CARLA provides an environment where agents can learn from trial and error by interacting with the environment. The agents receive rewards based on their performance in tasks like staying in the correct lane, avoiding obstacles, and obeying traffic rules. Through these interactions, they

gradually learn optimal driving policies. In the context of imitation learning (IL), CARLA can generate expert driving data through pre-programmed agents or human drivers. These data sets are then used to train IL models to replicate the behaviour of expert drivers.

One of CARLA's key strengths is the range of sensor data it provides, which is critical for perception tasks in autonomous driving. CARLA simulates cameras (RGB), depth sensors, LiDAR, radar, and even semantic segmentation masks, allowing researchers to develop algorithms for tasks such as semantic segmentation, 3D object detection, and depth completion. These inputs provide essential information for the decision-making and control modules of self-driving cars. The availability of high-quality ground truth labels in the CARLA environment allows for a detailed evaluation of model performance, making it an excellent tool for testing perception and control algorithms.

Evaluation metrics used in CARLA simulations typically include safety-related measures such as collision rates, lane deviation, and the number of infractions (e.g., running red lights or violating speed limits). Other performance metrics focus on task completion, such as the percentage of routes completed successfully or the time taken to reach a destination. Additionally, researchers can measure generalisation across different towns or weather conditions, which helps evaluate the robustness of trained models.

2.5.2 TORCS (The Open Racing Car Simulator)

TORCS [50], originally developed as a racing simulator, has been adapted for autonomous driving research, particularly in the field of reinforcement learning. While it was not initially intended for real-world driving simulation, TORCS offers a valuable platform for training and testing driving policies in simplified environments. The simulator provides a range of tracks and road types, along with the ability to modify vehicle dynamics, which makes it suitable for investigating the low-level control of autonomous vehicles.

In TORCS, reinforcement learning algorithms can be trained by rewarding the agent for completing laps, staying on the road, and avoiding collisions with other vehicles or track boundaries. This makes it a popular choice for research focused on learning to control speed, steering, and acceleration based on the vehicle sensor input. Since TORCS features different tracks with varying levels of difficulty, it also supports curriculum learning approaches, where agents progressively learn from simpler to more complex driving tasks.

While TORCS is limited in terms of sensory diversity compared to more advanced simulators like CARLA, it remains a valuable tool for experimenting with control strategies and reinforcement learning algorithms. Its simplicity allows for faster simulation times, making it ideal for prototyping and testing basic control policies before moving to more computationally intensive simulators. Researchers often use TORCS for experiments where the focus is on low-level vehicle control rather than high-level decision-making, such as learning to handle sharp turns, acceleration, and braking.

In terms of evaluation, TORCS mainly focuses on metrics such as lap time, the number of completed laps, and collision rates. Additionally, the ability of the agent to maintain stability during high-speed manoeuvres or sharp turns is often evaluated to assess the effectiveness of control policies.

2.5.3 Duckietown

Duckietown [51] is an innovative open-source platform designed to advance research in autonomous driving and robotics, particularly focusing on education and experimentation. It consists of a miniature city environment equipped with roads, intersections, traffic signs, and various dynamic elements such as other vehicles and pedestrians, all represented in a highly simplified and engaging manner. This setting serves as a testbed for various algorithms related to navigation, perception, and control, allowing researchers and students to explore autonomous driving concepts in a more accessible and manageable format.

The most significant innovation of Duckietown is its setup for real-world experiments, which includes Duckiebots, the small robotic vehicles equipped with cameras, and configurable maps that create an interactive miniature city for the Duckiebots to navigate. This setup replicates the features in the simulator, facilitating a sim-to-real application where algorithms can be trained in simulated environments and tested in real-world scenarios. This transition from simulation to reality is crucial for validating autonomous driving technologies.

In terms of evaluation, Duckietown provides various metrics for assessing the performance of autonomous agents, including navigation accuracy, task completion rates, and safety measures such as collision avoidance. The simplicity and engaging nature of the Duckietown environment make it an excellent tool for research purposes, bridging the gap between theoretical concepts and practical applications in the field of autonomous systems.

2.6 Datasets

Datasets are the core of deep-learning-based algorithms for both training and evaluation. In supervised learning, models rely on labelled datasets to learn mappings from input features to the desired output, either to classify or to regress, often requiring large amounts of diverse data to generalise well across real-world scenarios. High-quality datasets serve to standardise evaluation, making it possible to fairly compare model performance on common tasks. For example, autonomous driving requires diverse datasets capturing varied lighting, weather conditions, and road environments to ensure the perception robustness across different driving situations. In addition, using well-structured datasets helps prevent bias, supports performance benchmarking, and aids in diagnosing model errors. These datasets often include detailed and careful human annotations, such as segmentation masks and bounding boxes, to enhance model training in complex tasks like object detection and scene understanding, ultimately improving the reliability of decision-making in real-world applications.

Figure 2.6 The sensor setup of KITTI [52] dataset. It contains a stereo camera set facing the front, a LiDAR, and a GPS/IMU.

Figure 2.7 The sample data from KITTI [52] including labels of 3D object detection (left), semantic segmentation (centre), and depth prediction (right.)

2.6.1 KITTI Dataset

The KITTI dataset [52], introduced in 2012 by the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago, remains one of the foundational datasets for autonomous driving research. KITTI is designed to facilitate tasks such as 3D object detection, depth estimation, stereo matching, and odometry, offering a comprehensive set of benchmarks. It includes data collected from urban, suburban, and highway driving scenarios using stereo cameras, LiDAR, Global Positioning System (GPS), and Inertial Measurement Unit (IMU) sensors. The detailed sensor setup is demonstrated in Fig. 2.6. The dataset features over 200,000 images and point cloud data, with annotations. The sensor data with ground truth examples are shown in Fig. 2.7. A key strength of KITTI is its multimodal sensor setup, allowing researchers to experiment with sensor fusion and depth perception techniques in the

Figure 2.8 The sample data from Cityscapes [53]. This dataset contains finely annotated semantic labels (upper) as well as coarse labels as supplementary (lower).

earlier stage of deep learning. Despite its influence, KITTI has some limitations. It captures data primarily in clear weather and daylight conditions, meaning that it lacks the variability in lighting and weather that autonomous driving systems must navigate in real-world settings. Additionally, the dataset is relatively small by modern standards, with just over 15,000 3D bounding boxes. Nonetheless, KITTI's clear annotations and various well-defined tasks have set the foundation for numerous developments in the field of autonomous vehicle perception, particularly in 3D vision. It has served as a baseline for developing deep learning models aimed at enhancing object detection and depth perception using LiDAR and camera fusion.

2.6.2 Cityscapes

Released in 2016, the Cityscapes dataset [53] is a large-scale resource designed for urban scene understanding, with a particular emphasis on pixel-level semantic segmentation. The dataset contains 5,000 finely annotated images from 50 different cities across Europe, capturing a wide variety of urban environments, including roadways, pedestrians, vehicles, and other relevant objects. Cityscapes is unique in its high-resolution imagery and dense pixel-level annotations, making it particularly useful for training and testing segmentation models. The dataset also includes an additional 20,000 coarsely annotated images, which serve as supplementary training data for deep learning models for less detailed segmentation. The dense and coarse annotations are demonstrated in Fig. 2.8. Cityscapes has become one of the most widely used datasets for semantic segmentation in autonomous driving due to its comprehensive annotations and focus on urban environments. It supports a variety

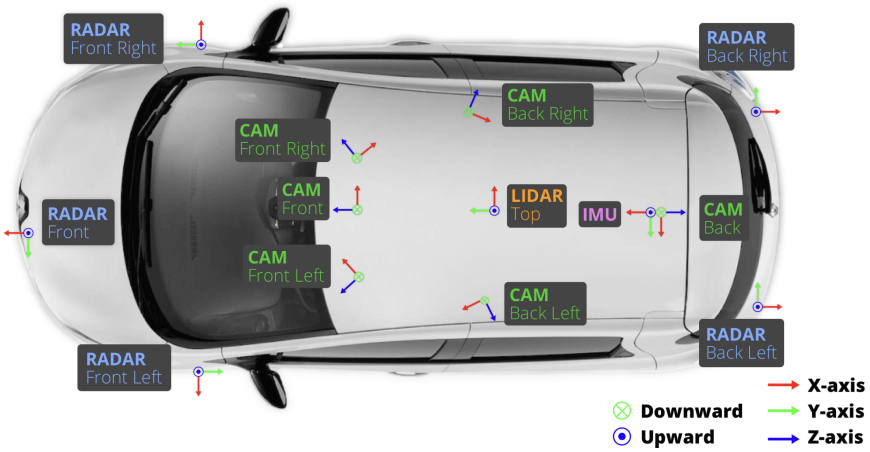

Figure 2.9 The sensor setup in the nuScenes dataset [54] provides extensive data from a range of sensors, including six cameras that cover 360° around the vehicle. These cameras are positioned to face the front, front right, front left, back right, back, and back left, giving a comprehensive view of the surrounding environment. Radar sensors are similarly oriented in multiple directions, addressing the occlusion challenges presented by setups with a single radar. The LiDAR sensor is mounted on the top of the vehicle, and an IMU is included to provide accurate positioning and motion data.

of tasks, including semantic and instance segmentation, object detection, and lane detection. Researchers have used this dataset to develop and refine models that can handle the complex, cluttered environments typical of city driving, where precise understanding of road conditions, lane markings, and pedestrian behaviour is crucial. However, like KITTI, Cityscapes lacks diversity in terms of weather conditions and sensor modalities—it only provides RGB images, without LiDAR or radar data, limiting its applicability to sensor fusion research.

2.6.3 nuScenes

nuScenes [54], released by Aptiv in 2019, marked a significant step forward in the development of multimodal datasets for autonomous driving. Captured in the cities of Boston and Singapore, nuScenes contains data from a rich sensor suite, including six cameras, five Radars, LiDAR, GPS, and IMU. The sensor setup is shown in Fig. 2.9. The dataset features 1,000 driving scenes, each about 20 seconds long, making it one of the most extensive datasets for dynamic scene understanding. nuScenes includes 1.4 million images, 390,000 LiDAR sweeps, and over 1.4 million annotated

Figure 2.10 The sample annotations of 3D object detection in nuScenes [54] from the six cameras at one frame.

3D bounding boxes across 23 different object classes. It also provides high-level map data, such as road lanes and intersections, which enhances its utility for tasks like path planning and autonomous navigation. One of the key innovations of nuScenes is its multimodal nature, allowing researchers to develop and test models that fuse data from cameras, LiDAR, and radar for 3D perception tasks. This has enabled significant progress in areas such as sensor fusion, 3D object detection, and motion forecasting. Unlike older datasets, nuScenes also includes data collected under diverse weather conditions, such as rain and overcast skies, which is crucial for developing models that generalise well to real-world driving scenarios. The 3D bounding box examples of nuScenes are shown in Fig. 2.10.

2.6.4 Waymo Open Dataset

The Waymo Open Dataset [55], released in 2019 by Waymo, is one of the largest and most comprehensive datasets for autonomous driving. Collected from Waymo fleet of self-driving cars, the dataset includes over 1,000 hours of driving data, 12 million 3D labels, and millions of 2D labels, covering a wide range of environments, weather conditions, and lighting scenarios. The dataset features high-quality data from multiple LiDARs (including both top-mounted and side-mounted LiDAR sensors), high-resolution cameras, GPS, and IMU. The detailed sensor setup is shown in Fig. 2.11. Waymo dataset provides dense annotations for vehicles, pedestrians, cyclists,

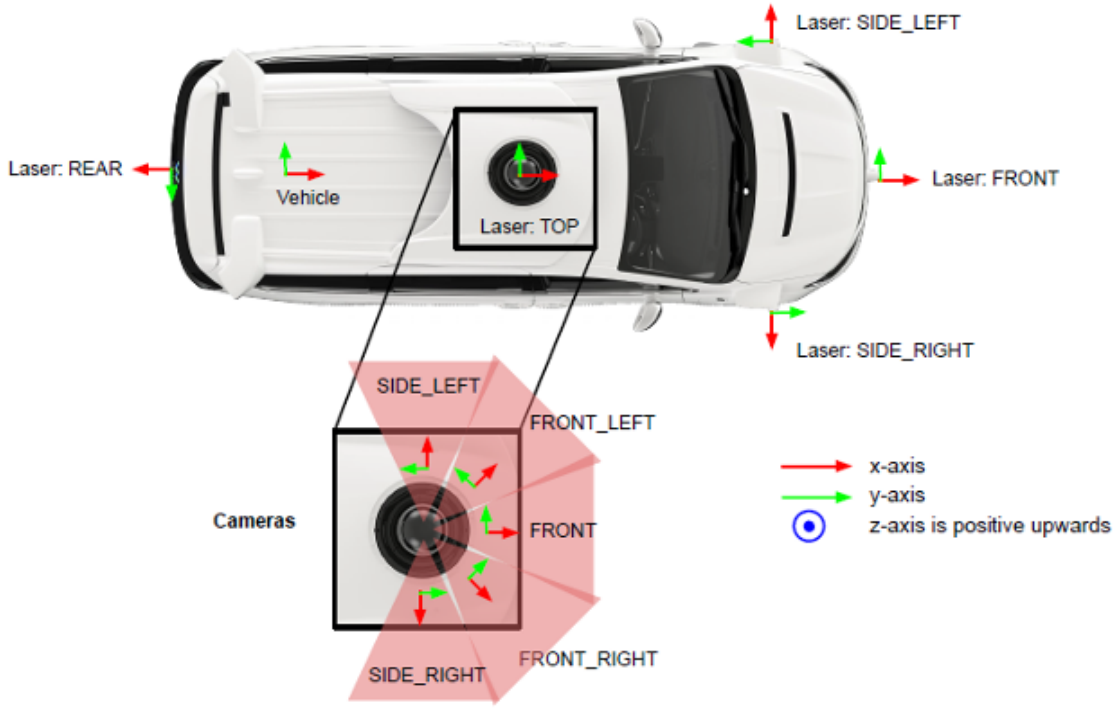

Figure 2.11 The sensor setup in the Waymo dataset [55] distinguishes itself by including a larger number of LiDAR sensors compared to other open datasets, making it especially valuable for research focused on LiDAR-based perception methods. This comprehensive sensor configuration enhances spatial perception accuracy, supporting detailed 3D scene understanding essential for autonomous driving applications.

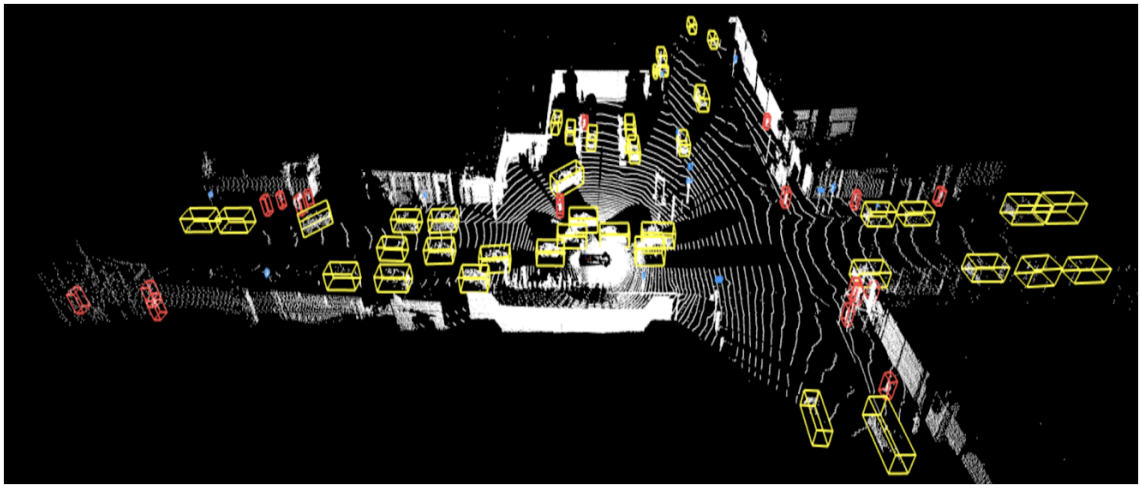

Figure 2.12 The sample annotations of 3D object detection in Waymo [55] from the LiDAR sensor set.

and other road users, making it particularly useful for 3D object detection, tracking, and semantic segmentation tasks. An example LiDAR based 3D bounding boxes

is shown in Fig. 2.12. One of the standout features of the Waymo Open Dataset is its sheer size and diversity, which far surpasses earlier datasets like KITTI and Cityscapes. It includes data captured in a variety of conditions—such as day and night, rain, fog, and different geographical locations—enabling researchers to develop more generalisable models. Furthermore, Waymo dataset supports a wide range of tasks, including object detection, lane detection, and sensor fusion, making it one of the most versatile datasets available for autonomous driving research.

Chapter 3

Fusion-Attention Monocular Semantic Segmentation

This chapter establishes a framework for deep-learning based monocular semantic segmentation in urban driving scenarios, which is crucial for autonomous vehicle scene understanding. It provides more precise subject information than raw RGB images, thereby enhancing the performance of autonomous driving systems.

Associated Publications This chapter is based on the following published work:

Wang C, Aouf N. Fusion attention network for autonomous cars semantic segmentation[C]//2022 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2022: 1525-1530.

Recently, self-attention methods have demonstrated significant improvements in learning based image semantic segmentation. Attention mechanism facilitates scene parsing by capturing abundant relationships between every pixel in an image. However, it is computationally demanding. Moreover, existing methods typically focus either on channel attention, which neglects pixel positional factors, or on spatial attention, which disregards the inter-dependencies between channels. To address these issues, in this chapter, the Fusion Attention Network is designed to capture rich contextual dependencies. This model comprises two chains: a pyramid fusion spatial attention module and a fusion channel attention module. The pyramid sampling is employed in the spatial attention module to reduce the computational complexity of generating spatial attention maps. The channel attention module follows a similar structure to the spatial attention module. Finally, the outputs from the spatial and channel attention modules are concatenated to form an enhanced attention map, resulting in improved semantic segmentation performance. Furthermore, the arrangement of attention module is discussed on aggregating the contextual dependencies extracted from both attention chains. Extensive experiments are conducted on popular datasets under various settings, alongside an ablation study, to demonstrate the efficiency of our approach. The proposed method outperforms state-of-the-art methods on the Cityscapes dataset [53] and also exhibits strong generalisation capabilities on PASCAL VOC 2012 [56].

3.1 Overview

Semantic segmentation is a fundamental task in computer vision with a wide range of applications in robotics, medical imaging and more importantly in autonomous driving. It aims to detect various objects, such as vehicles, roads, traffic signals, and pedestrians, and segment them with pixel-level precision. Recent studies on semantic segmentation have relied on convolutional encoder-decoder architectures, which have shown certain success. The encoder generates high-level, low-resolution features, where the decoder then up-samples to produce pixel-wise segmentation labels for each class. Fruitful segmentation approaches [57, 58] employ fully convolutional

network (FCN) [59] to accomplish this task. However, with the complex scene layout and enormous categories, and the spatial information lost during pooling operations, these methods are limited by the poor local contextual representation which leads to the model performance far from desire.

To address the limitations of simple Fully Convolutional Networks (FCN), several studies suggest that incorporating long-range and global dependencies can enhance performance. Multiple approaches [13, 60] have been proposed to fuse multi-scale contextual information by aggregating feature maps from various dilated convolutions and pooling operations. However, encoders that rely solely on convolutions struggle to capture these long-range dependencies. One possible reason is that the receptive field of a single convolutional layer is insufficient to cover the correlated regions effectively even with dilation. To capture richer contextual information, some works [61] attempt to enlarge the kernel size using a decomposed structure or by introducing an effective encoding layer on top of the network. Nevertheless, these methods often result in inefficiency due to the increased number of parameters and the subsequent computational demands.

Various efforts have been made to enhance feature representation after the standard encoder. To generate dense, pixel-wise contextual information, PSANet [62] learns to aggregate contextual information for each feature through a predicted attention map. Non-local Networks [63] leverages a self-attention mechanism [64], allowing a single feature at any position to perceive features from all other positions. CCNet [65] introduces a criss-cross attention module that gathers contextual information from all features along a cross-shaped path. Specifically, it captures cross-shaped dependencies at once and obtain the full-image contextual information by employing a recurrent operation. However, the performance of this method is dependent on the number of recurrences. Although the aforementioned methods have demonstrated improvements, they do not explore channel-wise attention.

Contextual dependencies refer to the influence that every pixel exerts on every other pixel in a high-level feature map. A feature map contains $H \times W \times C$ elements, where C represents the number of channels, and H and W denote the height and

width, respectively. To capture the full scope of global contextual information, it needs to calculate the correlation of every element in $H \times W \times C$ on every other element, leading to a computational complexity of $(H \times W \times C) \times (H \times W \times C)$. This level of computation is extremely demanding. To achieve better performance while maintaining efficiency, the channel and spatial information are harvested separately, introducing a Fusion Attention Network for the semantic segmentation task.

In this chapter, the contextual information from both the channel map and the spatial map is analysed, and combined into an enhanced global attention map. Specifically, a dilated ResNet [66] is employed and followed by two parallel attention modules: one for spatial attention and the other for channel attention. To address the high computational complexity, the spatial module is integrated with the proposed pyramid samplers, which efficiently reduce the computational load. In each sampler, high-level features are divided into several layers by channel and processed individually, before being recombined to establish concrete relationships with lower computational cost. For the channel module, high-level channel maps are similarly divided into layers, but along the spatial dimension, allowing the model to predict and combine the co-dependent effects across all channel maps. Finally, the outputs of these two attention modules are fused via concatenation, further enhancing the feature representations. The model is trained and evaluated on Cityscapes and PASCAL VOC 2012 datasets. In summary, the key contributions in this chapter are as follows:

1. The fusion attention modules that integrate both channel and spatial attention to capture global contextual dependencies.

2. The improved results by enhancing the attention maps without increasing the number of parameters, thanks to the pyramid and fusion structure.

3. The state-of-the-art performance on popular benchmarks, including the Cityscapes and PASCAL VOC datasets.

This chapter is organised as follows: Section 3.2 gives an overview of the current approaches of learning based semantic segmentation, as well as recent approaches

for applying attention mechanism. Section 3.3 describes the design methodology. Section 3.4 shows several test experiments with varying benchmark datasets and provides a proof-of-concept of the performance of the proposed method. Finally, Section 3.5 summaries the chapter.

3.2 Related Works

3.2.1 Semantic Segmentation

Semantic segmentation plays a crucial role in providing detailed environmental information by assigning a class label to each pixel in an image. This pixel-level classification significantly enhances the understanding of environmental context, particularly in tasks like autonomous driving, where precise scene comprehension is vital. John et al. [67] proposed one of the early approaches using a convolutional neural network (CNN)-based encoder-decoder architecture for semantic segmentation. Their network processes input images to perform pixel-wise classification and labeling, achieving a prediction accuracy of 88% for vehicles and 96% for roads, showcasing the effectiveness of CNNs in segmentation tasks. Building on this foundation, fully convolutional networks (FCNs) and FCN-based models advanced image semantic segmentation by directly operating on the spatial dimensions of input images. Several models have been developed that incorporate encoder-decoder architectures, such as Unet [14], SPGNet [68], RefineNet [58], and DFN [69]. These models effectively combine low-level and high-level features to produce more accurate segmentation predictions. The encoder captures detailed local information, while the decoder refines this information by integrating contextual knowledge, improving the overall accuracy of the predictions. In parallel, other methods focused on enhancing contextual aggregation to further improve segmentation performance. For instance, Deeplabv2 [13] and Deeplabv3 [70] introduced atrous spatial pyramid pooling, which uses parallel dilated convolutions to embed multi-scale contextual information. PSPNet [60] also contributed to this area by proposing pyramid pooling to capture information from different scales, effectively enhancing the network's ability to understand objects

at various levels of detail. With the success of the attention mechanism in natural language processing tasks, it was soon applied to image-based tasks, including semantic segmentation. Attention mechanisms allow models to focus on specific parts of the image, making them more effective at capturing both spatial and contextual dependencies. This shift has opened up new opportunities for improving the accuracy and efficiency of semantic segmentation models, marking a new phase in the development of this field.

3.2.2 Attention Model

Attention modules are widely recognised for their ability to model both long-range and short-range dependencies, enhancing the most important regions of a feature map. These modules help neural networks focus on the most relevant parts of the input, boosting the performance of tasks like semantic segmentation by refining the features being learned. A notable example is Squeeze-and-Excitation Networks (SENet) [71], which applies spatial-wise average pooling to capture global information across the image. SENet employs two fully connected layers to model channel-wise relationships, allowing the network to re-calibrate the importance of each channel through an attention mechanism, ultimately enhancing the representational power of the network by giving more weight to significant channels. Building on these principles, Li et al. [72] proposed a more complex attention structure called split, fuse, and select. This structure starts with a split operation, a multi-branch process where different excitation cores are used to extract diverse features from the input. Each core focuses on capturing different aspects of the feature map, providing multiple perspectives for the network to process. The fuse operation then combines the information from these multiple branches, merging them into a rich contextual representation. Finally, the select operation combines feature maps of various kernel sizes based on learned selection weights, allowing the network to dynamically focus on the most relevant features. This multi-stage process helps the model better capture complex dependencies in the input data. In parallel, Vaswani et al. [73] introduced a multi-head attention block, which became influential in many fields, including image processing. This block enables the network to compute multiple

self-attention representations through different heads, allowing it to capture a variety of dependencies and features simultaneously. By processing the information from different heads, the model can learn complementary attention maps, further improving its understanding of the input. Other methods have also sought to exploit attention mechanisms to improve contextual modelling in semantic segmentation. OCNet [74] and DANet [75], for instance, utilised the Non-local module, which computes contextual information by considering all positions in the feature map. This enables the model to capture dependencies across the entire image, which is particularly useful for segmenting large objects or objects spread over multiple regions of the image. However, the Non-local module is computationally expensive and involves a large number of parameters. To address this, CCNet [65] introduced a criss-cross attention mechanism, which significantly reduces the computational burden and complexity of the Non-local module. CCNet achieves this by computing relationships only between pixels in horizontal and vertical directions, rather than all pairs of pixels, making the attention process more efficient while still capturing sufficient contextual information. Similarly, EPSANet [76] focuses on enhancing attention by harvesting attention maps from features at different scales. By considering multi-scale information, EPSANet can better handle objects of varying sizes and improve the overall performance of the model in semantic segmentation tasks. These advancements illustrate the power of attention mechanisms in refining feature representations, allowing models to better understand complex spatial and contextual relationships within images. Through different innovations—such as channel-wise re-calibration, multi-branch operations, and computationally efficient attention modules—these methods have contributed significantly to improving the accuracy and efficiency of semantic segmentation models.

3.3 Methodology

This section presents the general framework of the proposed network. Next, the fusion attention module is detailed, including both the channel-wise fusion attention and pyramid spatial-wise fusion attention modules, which are designed to capture

full-image contextual information in the channel and spatial dimensions, respectively. Finally, various methods for aggregating these modules are compared to achieve the best performance.

3.3.1 Preliminary

Self-attention is a mechanism that allows models to dynamically focus on different parts of the input when processing each element, enabling them to capture long-range dependencies. In this mechanism, three key concepts are employed: $Query(Q)$, $Key(K)$, and $Value(V)$. These are derived through learned transformations of the input sequence, and they play distinct roles in determining how attention is distributed across the sequence.

1. $Query(Q)$: The query vector represents the current token's request for information. When processing a particular token, the query is used to determine which other tokens in the sequence are most relevant to it.

2. $Key(K)$: The key vector corresponds to each token and is used to match with the query. It helps evaluate how much attention should be given to different tokens by measuring the similarity between the query and each key.

3. $Value(V)$: The value vector holds the actual content or information of the token. Once the relevance between tokens is determined (using Q and K), the value vectors are weighted accordingly, and the final output for each token is produced by combining these weighted values.

The self-attention mechanism computes attention scores by performing a dot product between the $Query$ and Key vectors of all tokens, and then these scores are used to weight the $Value$ vectors. This allows the model to consider relationships across the entire input sequence when processing each token.

3.3.2 Network Structure

The network architecture is illustrated in Fig. 3.1. As shown, two types of attention

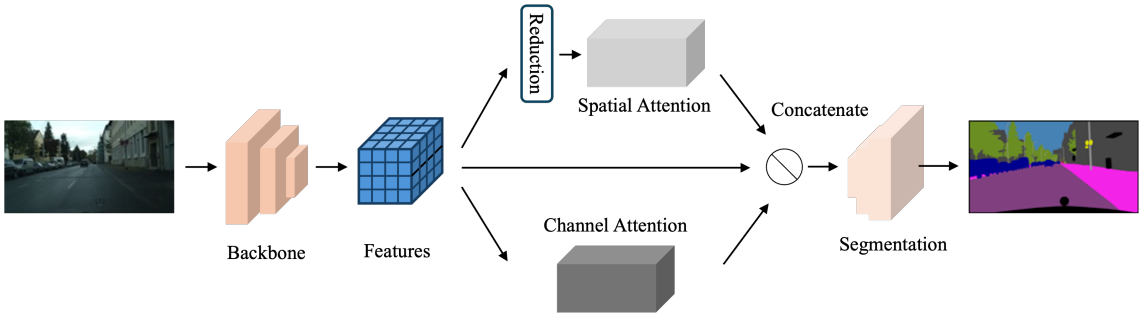

Figure 3.1 Overview of the Fusion Attention Network. [Wang et al.]

modules are designed to capture global context over local features generated by a dilated residual network, thus enhancing feature representations for pixel-level prediction. A pretrained residual network with a dilated strategy [13] is employed as the backbone. By removing the last two down-sampling operations and incorporating dilated convolutions in the subsequent ResNet blocks, the output feature map size is enlarged back to 1/8 of the input image, allowing us to extract more information without increasing the number of parameters. After obtaining the feature map, a convolutional layer is applied for dimension reduction to prepare inputs for both spatial and channel attention modules. In the self-attention mechanism, the first step involves generating an attention matrix that models the relationships between any two pixels within the feature maps. The spatial and channel attention is treated separately, utilising a fusion technique and pyramid sampler for spatial attention, while employing channel-wise fusion attention to capture long-range contextual information in the channel dimension. Subsequently, the matrix multiplication is performed between the attention matrix and the query vectors. Finally, an element-wise sum operation is executed on the multiplied result and the original features to obtain the final representations that reflect long-range contexts. With outputs from both the spatial and channel modules, they are aggregated to enhance feature representations for pixel-level prediction.

3.3.3 Pyramid Fusion Spatial Attention

To model dense pixel relationships over local feature representations while ensuring high performance and lightweight computation, the pyramid fusion spatial attention

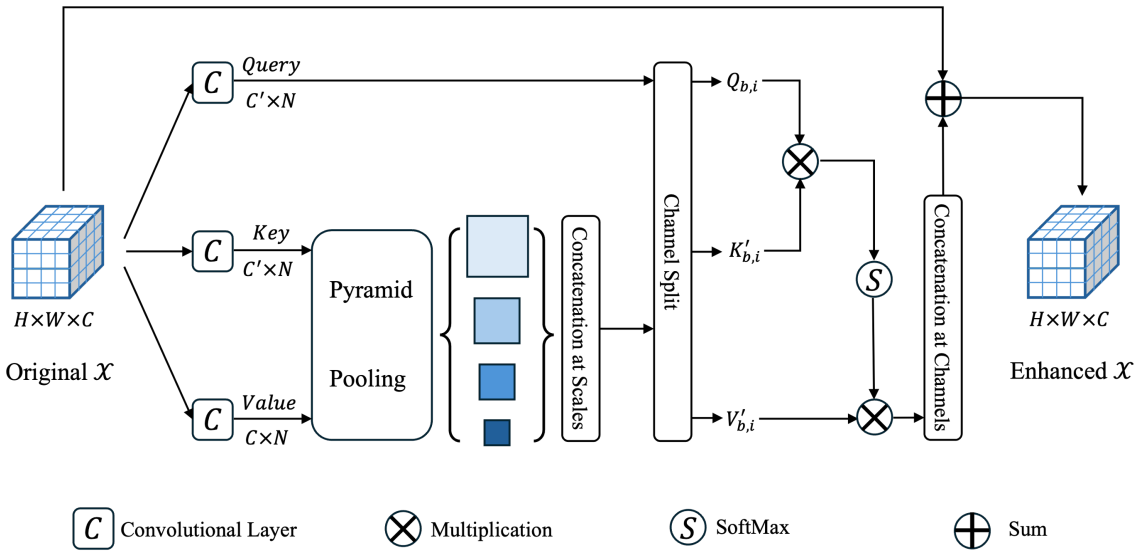

Figure 3.2 Design of Pyramid Spatial Attention Module. [Wang et al.]

module is introduced in this section. This module effectively captures rich contextual information, enhancing the pixel-level representational capability.

As illustrated in Fig. 3.2, given a local feature map $\mathcal{X} \in \mathbb{R}^{H\times W\times C}$, the module first applies two convolutional layers with filter size 1×1 on $\mathcal{X}$ to generate two new feature vectors $Query(Q)$ and $Key(K)$, where $Q, K \in \mathbb{R}^{C^{'}\times W\times H}$ and $C^{'}$ is the reduced channel number for the new feature vectors for less computation. This refers to the reduction block in the Fig. 3.1. After that Q and K are reshaped at their spatial dimension to $\mathbb{R}^{C^{'}\times N}$ for latter calculation, where $N = W \times H$. Simultaneously, another convolutional layer, also with a 1×1 filter is applied to $\mathcal{X}$ to generate $Value(V)$, where $V \in \mathbb{R}^{C\times W\times H}$, and then reshape to $V \in \mathbb{R}^{C\times N}$ without channel reduction. At this point, Q and K are typically subjected to matrix multiplication followed by SoftMax to generate a large attention map, which requires significant computing and memory resources. To mitigate this, pyramid sampling is introduced on both feature vectors K and V. For instance, if the size of the feature map $\mathcal{X}$ is 96×96, the attention map size will be $N \times N = (W \times H) \times (W \times H) = 84.9M$. By applying the designed pyramid pooling layers with $kernal\_size = 8, 6, 3, 2$, the new features sizes become 12×12, 16×16, 32×32 and 48×48. As the attention map is calculated at each kernel size, the overall complexity is $\mathcal{O} = (12 \times 12)^2 + (16 \times 16)^2 + (32 \times 32)^2 + (48 \times 48)^2 = 6.6M$,

resulting 92.2% of computational reduction when for attention map calculation. Next, the fusion structure is presented. First, the Pyramid Pooling on K and V generates K_1, K_2, K_3, K_4 and V_1, V_2, V_3, V_4 at various lower scales. These are then concatenated, resulting in $K^{'} \in \mathbb{R}^{C^{'} \times N^{'}}$ and $V^{'} \in \mathbb{R}^{C \times N^{'}}$. Next, Q, $K^{'}$, and $V^{'}$ are divided into m blocks along the channel dimension, where each block contains $\frac{C}{m}$ elements for both $Q_{b,i}$ and $K^{'}_{b,i}$, and $\frac{C^{'}}{m}$ elements for $V^{'}_{b,i}$. This division allows each block to learn distinct information within the attention map by focusing on different channel-specific features. For each block, matrix multiplication is performed between the transposed $Q_{b,i}$ and $K^{'}_{b,i}$:

$$S_i = Q^T_{b,i} \in \mathbb{R}^{N \times \frac{C^{'}}{m}} \times K^{'}_{b,i} \in \mathbb{R}^{\frac{C^{'}}{m} \times N^{'}}, i = 1, \ldots, m \tag{3.1}$$

Here, S_i represents the similarity matrix, which measures the degree of correlation between each position in $Q_{b,i}$ and $K^{'}_{b,i}$. It is important to note that the size of S_i is $\mathbb{R}^{N \times N'}$, rather than $\mathbb{R}^{N \times N}$, where N' is the spatial dimension of $K^{'}$. A SoftMax layer is then applied to S_i to compute the spatial attention map $A_i \in \mathbb{R}^{N \times N'}$. Subsequently, matrix multiplication is performed between the transposed A_i and $V^{'}_{b,i}$:

$$V^{'}_{b,i} \in \mathbb{R}^{\frac{C}{m} \times N'} \times A^T_i \in \mathbb{R}^{N' \times N} = \mathbb{R}^{\frac{C}{m} \times N} \tag{3.2}$$

Equation 3.2 generates the results for each of the blocks we previously divided. After obtaining the results from all channel blocks, we concatenate the m blocks of size $\mathbb{R}^{\frac{C}{m} \times N}$ along the channel dimension to form $M \in \mathbb{R}^{C \times N}$. Then, M is reshaped to $\mathbb{R}^{H \times W \times C}$, aligning with the size of the original feature map $\mathcal{X}$. The contextual information is gathered by performing an element-wise sum between the feature map $\mathcal{X}$ and the summed results from the Pyramid Fusion Spatial Attention module.

$$Y_j = \gamma M_j + X_j \tag{3.3}$$

Where j denotes each position within the original feature map $\mathcal{X}$, and γ is a learnable scale parameter, which plays a critical role in adjusting the contribution of the contextual information during the learning process. Initially set to 0, γ gradually

increases its value as the model learns, allowing it to assign more weight to the contextual features over time. This gradual learning process ensures that the model starts by relying on the original feature map and slowly incorporates more contextual information as it learns to refine its predictions. By adding the original feature map $\mathcal{X}$ to the aggregated contextual information, the model is able to maintain its original feature representation while enriching it with the additional context. This allows the model to capture both local features and broader contextual information, providing a more holistic view of the image. The selective aggregation of these contexts is guided by the spatial attention map, which helps the model focus on the most relevant regions of the image based on the learned spatial relationships. The combination of the original features and the contextual information leads to the generation of more robust feature representations. By allowing these similarity features to interact and complement each other, the model achieves mutual gains. This means that the contextual features help improve the quality of the original feature map, and vice versa, creating a synergistic effect. As a result, the representation capacity of the network is significantly enhanced, enabling it to better capture complex patterns and relationships within the input data.

3.3.4 Fusion Channel Attention

In the previous section, while the proposed method successfully enhances feature maps, it overlooks the dependencies between individual channels, where high-level semantic information could be more effectively explored. By leveraging the inter-dependencies between channel maps, the model can emphasise interdependent feature maps, thereby improving the representation of specific semantic features. However, existing channel attention methods often fail to consider the positional relationships between channel maps. To address this, the concept of a fusion structure is extended to the channel dimension. In this approach, the spatial dimension is partitioned into blocks, enabling the model to apply distinct attention to different positions within each block, allowing for more focused attention across various spatial locations.

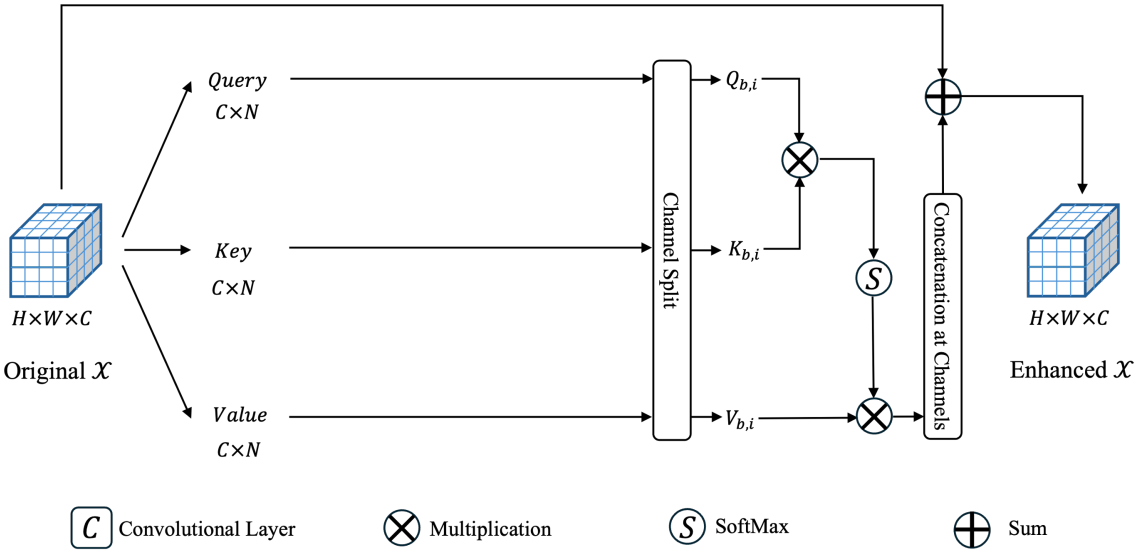

Figure 3.3 Design of Channel Attention Module. [Wang et al.]

The channel attention structure is illustrated in Fig. 3.3. In the spatial attention module, several convolutional layers are employed to reduce the dimensions of the feature map $\mathcal{X}$ for improved efficiency. However, in the channel attention chain, dimension reduction does not lead to a significant reduction in parameters because convolutional layers cannot effectively reduce the spatial dimensions. Instead, it results in the loss of important information within the channel maps, which are the primary focus of investigation. Therefore, the feature map $\mathcal{X}$ is directly inherited from the original features and used as Q, K, and V, which are then reshaped into $\mathbb{R}^{C\times N}$. The fusion strategy is subsequently applied by first dividing each feature map into m blocks. In contrast to the spatial attention module, each block here has a shape of $\mathbb{R}^{C\times\frac{N}{m}}$. Every block contains distinct spatial information, leading to improved attention maps. By employing a similar process of reshaping, transposing, and performing matrix multiplication, the similarity matrix $S_i \in \mathbb{R}^{C\times C}$ is obtained:

$$S_i = Q_{b,i} \in \mathbb{R}^{C\times\frac{N}{m}} \times K_{b,i}^T \in \mathbb{R}^{\frac{N}{m}\times C}, i = 1, \ldots, m \tag{3.4}$$

Where b, i indicates the index of blocks. S_i demonstrates the degree of correlation between each channel in $Q_{b,i}$ and $K_{b,i}$. A SoftMax layer is used to obtain the attention map A_i. Following this, a matrix multiplication is performed between the transpose

of V and A_i:

$$V_{b,i}^{'\ T} \in \mathbb{R}^{\frac{N}{m} \times C} \times A_i \in \mathbb{R}^{C \times C} = \mathbb{R}^{\frac{N}{m} \times C} \tag{3.5}$$

Through spatial-wise concatenation and reshaping, the results from each block are fused into a feature of size $\mathbb{R}^{C \times H \times W}$, matching the dimensions of the original feature map $\mathcal{X}$. Finally, the contextual channel information is gathered by performing an element-wise sum between the feature map $\mathcal{X}$ and the concatenated result:

$$Y_j = \gamma R_j + X_j \tag{3.6}$$

Where j represents each channel map in the feature map $\mathcal{X}$, and γ is a learnable scale parameter that initially starts from a value of 0. This scale parameter is essential for the model's adaptability, allowing it to adjust the contribution of contextual information as training progresses. As the model learns, γ gradually increases, facilitating a more significant integration of contextual features into the final representation. Similar to the spatial attention mechanism, where attention is focused on specific regions of the input feature map, channel attention enhances the model's performance by focusing on the most relevant channels. In spatial attention, the emphasis is placed on specific spatial locations, allowing the model to weigh the importance of various parts of the image. This is particularly valuable in semantic segmentation, as it enables the model to prioritise areas with critical features, such as object edges and important textures, that contribute to accurate segmentation. In contrast, channel attention focuses on the inter-dependencies between different channels within the feature map. By adjusting the importance of each channel, channel attention allows the model to highlight features that are particularly relevant to the segmentation task while suppressing those that may add noise or ambiguity. For instance, in distinguishing between overlapping objects in a scene, channel attention can help the model better capture the unique characteristics of each object by enhancing the features that define them, regardless of their spatial location.

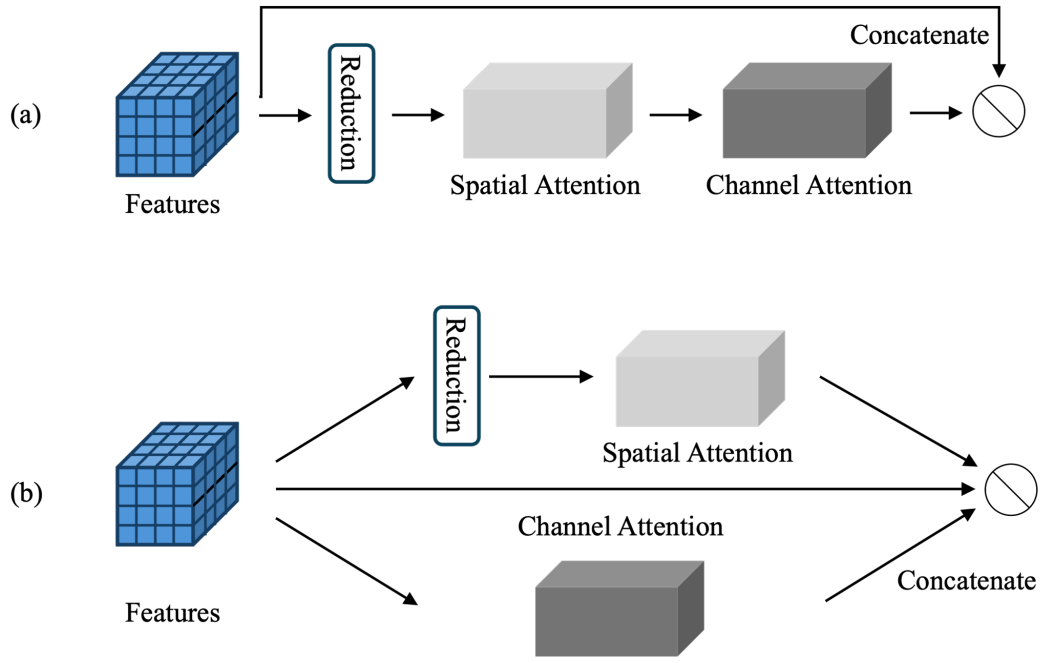

Figure 3.4 Two types of attention module arrangements. (a) Sequential connection. (b) Parallel connection. [Wang et al.]

3.3.5 Attention Module Arrangement

Given an input image, the two attention modules are designed to compute complementary forms of attention, with one focusing on the spatial dimension and the other concentrating on the channel dimension. These two distinct forms of attention allow the model to capture both where the relevant features are located within the image (spatial attention) and what specific channels or feature maps are most important (channel attention). Intuitively, these modules can be arranged in different configurations, such as in parallel or sequential order as shown in Fig. 3.4. The parallel configuration processes the spatial and channel information simultaneously, while the sequential arrangement processes one form of attention after the other.

Through our experiments, we observed that the parallel arrangement consistently yields better performance compared to the sequential configuration. This suggests that when the two attention modules are aligned in parallel, the dependencies between spatial and channel information are preserved more effectively, allowing the model to make full use of both forms of attention without disrupting their mutual relationship. On the other hand, the sequential arrangement introduces additional complexity by

forcing the model to first process one form of attention, followed by the other. This may lead to a breakdown in the synergy between spatial and channel dependencies, making it harder for the model to capture these inter-dependencies in an efficient manner. We hypothesise that the parallel structure ensures a more balanced fusion of spatial and channel information, whereas the sequential connection introduces an unnecessary layer of complexity that does not enhance the overall performance.

Despite the two ways of arrangements, both attention mechanisms are complementary and can be used in tandem to improve the model's overall performance in semantic segmentation. While spatial attention provides a mechanism for the model to concentrate on significant areas of the input, channel attention allows for a deeper exploration of the relationships among feature channels. This dual approach enhances the model's capacity to understand complex scenes, leading to more precise and accurate delineation of object boundaries. By leveraging both spatial and channel attention, the model can achieve a robust feature representation that captures essential contextual information while maintaining focus on the most critical features within the input data. This integrated approach ultimately enhances the model's performance in semantic segmentation tasks, enabling it to deliver more accurate and detailed segmentation results.

3.4 Experiments

The proposed method is trained and evaluated on two semantic segmentation datasets, Cityscapes[7] and PASCAL VOC 2012[9]. The experimental results demonstrate that our method achieves state-of-the-art performance on Cityscapes.

3.4.1 Dataset and Evaluation Standard

Dataset

Cityscapes The dataset has 5,000 images captured from 50 different cities. Each image has 2048×1024 pixels, which have high quality pixel-level labels of 19 semantic classes. There are 2,975 images in training set, 500 images in validation set and 1,525

images in test set. For evaluation, we use mIoU (mean of class-wise inter-section over union) as the metrics.

PASCAL VOC 2012 This dataset contains 1464 of training images, 1449 images for validation with different scales. It has 21 classes of objects which including a background and 20 foreground object classes. With data augmentation, the training images are expanded to 10582.

Evaluation

Mean Intersection over Union (mIoU) The mean of class-wise Intersection over Union (mIoU) is widely used as the evaluation metric for semantic segmentation tasks. This metric provides an intuitive and reliable assessment of the model's performance by measuring how well it segments different objects or regions in an image. The performance is measured across all classes, and the final evaluation score is the mean mIoU, which is the average IoU computed over all classes:

$$mIOU = \frac{1}{C}\sum_{i=1}^{C}\frac{P_i \cap G_i}{P_i \cup G_i} \tag{3.7}$$

Where C is the total number of classes and P_i and G_i are the predictions and ground truths of class i.

3.4.2 Implementation Details

This approach is implemented with PyTorch. Stochastic gradient descent (SGD) [77] is used as the optimizer. For the Cityscapes dataset, the initial learning rate is set to 0.01, and following [78, 65], the polynomial learning rate policy is adopted, where the initial learning rate is multiplied by $1 - \left(\frac{iteration_{current}}{iteration_{max}}\right)^{power}$, with power set to 0.9. The momentum and weight decay are set to 0.9 and 0.0001, respectively. The batch size is set to 8, and train for 242 epochs in total. For the PASCAL VOC 2012 dataset, the learning rate is set to 0.001 and follows the same decay policy as Cityscapes. The batch size is 16 for VOC, and training is conducted for 200 epochs. For both datasets, a joint objective is employed in the loss function. Specifically, after the fusion attention module, three feature maps are generated: one from the spatial

Methods	Backbone	mIOU %
Non-Local [63]	ResNet101	75.1
GCN [61]	ResNet101	78.1
PSPNet [60]	ResNet101	78.5
DeepLabV3 [70]	ResNet101	78.5
ASPP [13]	ResNet101	78.9
DeepLabV3+ [79]	Xception-71	79.6
CCNet [65]	ResNet101	81.3
Non-Local	ResNet50	73.3
GCN	ResNet50	76.2
PSPNet	ResNet50	76.4
ASPP	ResNet50	77.1
CCNet	ResNet50	78.1
Proposed Method	ResNet101	**81.9**

Table 3.1 Result comparison on the validation set of Cityscapes. [Wang et al.]

attention, one from the channel attention, and the fused feature map. Each feature map is passed through the bottleneck for semantic segmentation prediction. During training, the objective is to minimise the difference between the ground truth and the predictions from the spatial-enhanced features, the channel-enhanced features, and the fused final features. After training, only the fused final prediction is used for evaluation. Cross-entropy loss function is used for the three supervised signals.

3.4.3 Results

Results on Cityscapes

The proposed method is first evaluated on the Cityscapes validation set. The results of the evaluation are presented in Table 3.1. For the feature extraction backbone, ResNet101 is employed, known for its ability to capture rich hierarchical features. Additionally, state-of-the-art models with the same ResNet101 backbone, as well as ResNet50, are selected for comparison to ensure a fair evaluation of the method's effectiveness.

Table 3.2 provides the detailed mIoU values for each class in the dataset. As observed, the segmentation accuracy for categories such as road, building, sidewalk, vegetation, sky, car, and bus exceeds 85%. This high performance is expected, as these objects

Category	**Road**	**Sidewalk**	**Building**	**Wall**	**Fence**
mIOU %	98.6	88.6	94.1	74.1	77.9
Category	**Pole**	**Traffic Light**	**Traffic Sign**	**Vegetation**	**Terrain**
mIOU %	70.2	75.6	83.7	93.4	71.9
Category	**Sky**	**Person**	**Rider**	**Car**	**Truck**
mIOU %	95.1	84.9	68.3	95.5	70.0
Category	**Bus**	**Train**	**Motorcycle**	**Bicycle**	
mIOU %	86.0	79.3	67.3	81.6	

Table 3.2 Detailed mIoU on 19 Categories of Cityscapes. [Wang et al.]

typically exhibit a consistent appearance and occupy large portions of the image. For instance, roads and buildings have clearly defined edges, making them easier to detect and segment. Similarly, cars and buses, which are relatively large and well-defined objects, are consistently well-segmented by the network. For mid-sized objects like traffic signs, persons, trains, and bicycles, the mIoU values fall within a respectable range of 75% to 85%. These classes are often more variable in shape and size compared to larger objects but still maintain a degree of regularity that allows the network to generalise effectively. For example, pedestrians and bicycles, though dynamic and sometimes difficult to predict, are often well-localised, contributing to decent segmentation results. Trains, though sometimes challenging due to their varying shapes and speeds, are still segmented reasonably well within this range. However, there are certain classes where the network struggles to achieve high accuracy. Categories such as wall, fence, pole, train, motorcycle, and terrain show comparatively lower performance. These classes pose unique challenges due to several factors:

1. Fence and pole suffer from poor segmentation due to their thin and inconsistent structures. Since these objects often appear small or are partially occluded in complex urban scenes, it becomes difficult for the network to distinguish them accurately.

2. Motorcycle and truck exhibit weaker recognition primarily due to the limited number of samples for these classes. Since these objects appear far less frequently than cars or buses, the model has fewer opportunities to learn their unique features, which contributes to lower segmentation accuracy. Further-

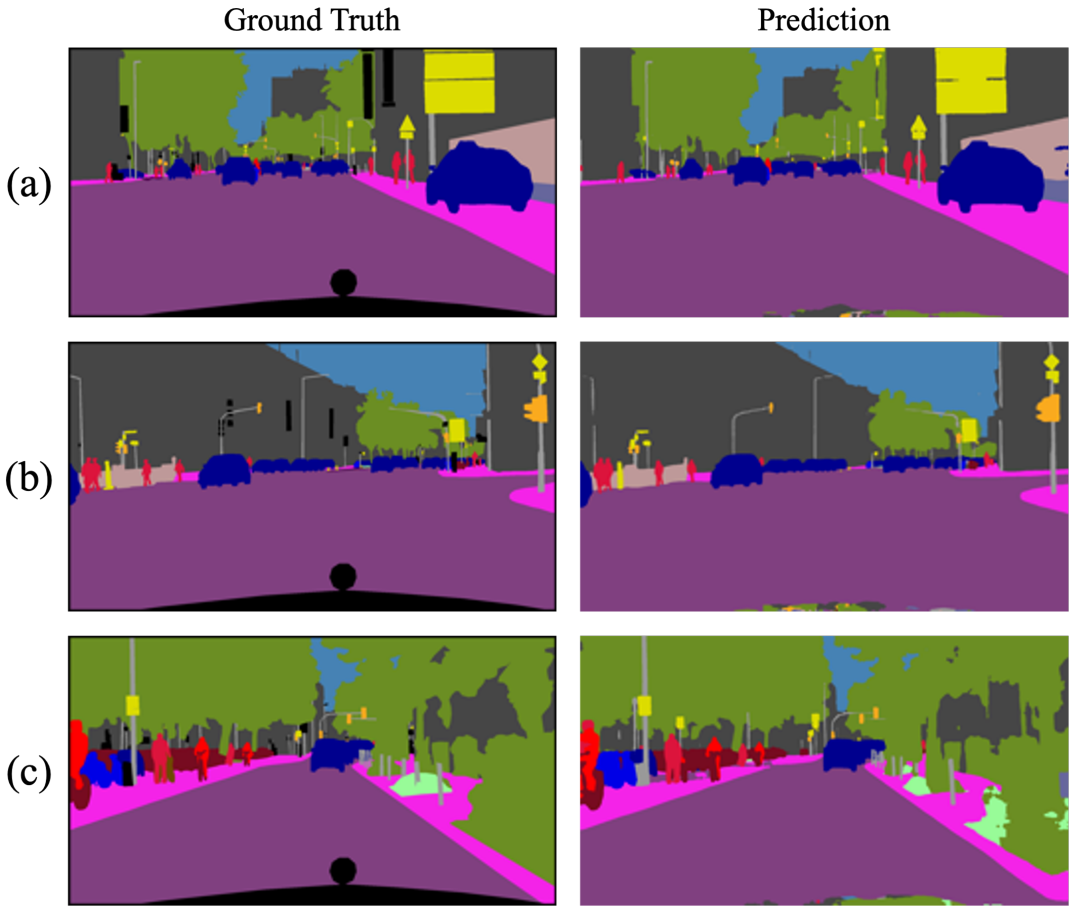

Figure 3.5 Visual comparison of prediction and ground truth. [Wang et al.]

more, large objects like trucks often appear only partially in the input images when close to the camera, making it challenging for the model to predict their full structure. This incomplete visibility compromises accurate segmentation, as the model lacks the necessary context to identify the whole object.

3. Train and terrain also present difficulties. Trains can be challenging due to their varying appearances and motion in the scene, while terrain, which may encompass different types of surfaces (e.g., grass, dirt), often blends with surrounding objects, making it harder to segment distinctly.

To show the overall performance, the visual comparison between the predicted semantic segmentation and the ground truths is demonstrated in Fig. 3.5. As can be seen, though there are some flaws in predicting the labels in vegetation and overlapping motorcycles, the proposed method perfectly segments the three scenes with the vital objects for autonomous vehicles such as roads, sidewalks, vehicles and

Backbone	Attention Module	mIOU%
ResNet101	Pyramid Fusion Spatial	76.2
ResNet101	Fusion Channel	81.4
ResNet101	Spatial and Channel Fusion	**81.9**

Table 3.3 Individual Attention Module Efficiency. [Wang et al.]

pedestrians. Note that as the ground truth in the bottom of the image (the hood of the ego vehicle), the predicted semantic segmentation contains blur information.

Ablation Studies

To demonstrate the efficiency of the proposed attention modules and their components, experiments are conducted on the model with various configurations using the Cityscapes dataset, which more effectively reveals the segmentation capability in driving scenarios. The performance of each attention module is evaluated on the Cityscapes validation set using a ResNet101 backbone.

Pyramid pooling layer sizes of 16, 24, 32, 48 are applied to divide the high-level feature map into four different scales. This multi-scale approach enables the network to capture information at varying resolutions, making it more efficient and computational-friendly in segmenting objects of different sizes, especially in complex driving scenarios. The results of these experiments are presented in Table 3.3. When using only the spatial attention module, the network achieves a mean Intersection over Union (mIoU) of 76.2%. This demonstrates the module's ability to enhance segmentation by focusing on spatial structures. However, the channel attention module delivers stronger performance, achieving an mIoU of 81.4%, a 5.2% improvement, which indicates that capturing inter-channel relationships adds more value in terms of feature representation. When the spatial and channel attention modules are combined, the network achieves its best result with an mIoU of 81.9%. This combined approach yields a 5.7% improvement over the spatial module alone, showcasing the complementary nature of the two modules. By integrating both spatial and channel-wise attention, the network benefits from enhanced spatial context and enriched semantic information, leading to more accurate segmentation results.

Methods	mIOU%
HyperSeg [80]	80.6
DeepLab-CRF [81]	77.6
DeepLabv3 [70]	76.5
Proposed Method	**81.6**

Table 3.4 Results on PASCAL VOC 2012 Validation Set. [Wang et al.]

The visualised results are provided in Fig. 3.5, illustrating the effectiveness of the combined attention mechanism.

Results on PASCAL VOC 2012

In this section, further experiments are conducted on the PASCAL VOC 2012 dataset to demonstrate the generalisation capability of the proposed method. The results, presented in Table 3.4, show a strong performance compared to other state-of-the-art methods. The same backbone, ResNet101, is employed and trained on the trainaug set of VOC 2012. The proposed method achieves a mean Intersection over Union (mIoU) of 81.6%, surpassing well-known models like HyperSeg, DeepLab-CRF, and DeepLabv3. Specifically, the mIoU score of 81.6% shows a 1.0% improvement over HyperSeg (80.6%), which is known for its efficiency in semantic segmentation tasks. Compared to DeepLab-CRF, which scores 77.6%, the proposed method outperforms it by a margin of 4.0%. Additionally, DeepLabv3, another prominent segmentation model, achieves a lower score of 76.5%, with the proposed method demonstrating a significant 5.1% improvement.

This performance boost highlights the robustness and generalisation capability of the proposed attention mechanism across different datasets. By effectively capturing both spatial and channel-wise dependencies, the method consistently enhances feature representations, allowing for more accurate segmentation across a wider range of object classes in PASCAL VOC 2012. The competitive results further demonstrate the method's capability to generalise beyond the Cityscapes dataset, making it a strong candidate for various semantic segmentation tasks.

3.5 Summary

In this chapter, a novel Fusion Attention Network is proposed for semantic segmentation in autonomous driving scenarios, leveraging a self-attention mechanism. The primary goal of this network is to enhance the understanding of global contextual dependencies, which is critical for accurately segmenting complex scenes in real-time driving environments. To achieve this, the network incorporates two key modules: a pyramid fusion spatial attention module and a fusion channel-wise attention module. The pyramid fusion spatial attention module effectively reduces the number of parameters required for generating attention maps by dividing the feature maps into multiple scales, allowing the model to capture both fine-grained and large-scale features more efficiently. Simultaneously, the fusion channel-wise attention module ensures that relationships between different feature channels are fully exploited, enriching the network's ability to discern different semantic elements within the image. By employing both spatial and channel attention mechanisms, the proposed network can take full advantage of the information contained in each element of the image, ensuring that no relevant details are overlooked, whether they pertain to small objects or large structures. This dual approach to attention processing enhances the feature extraction capabilities of the network, making it well-suited for handling the diverse and dynamic nature of driving environments.

Extensive experiments are conducted on two widely recognised benchmark datasets, Cityscapes and PASCAL VOC 2012, to validate the performance of the Fusion Attention Network. The results demonstrate that the proposed model consistently outperforms existing methods, achieving superior segmentation accuracy across a wide range of object classes. This outstanding performance indicates the potential of the Fusion Attention Network to be deployed as an efficient and reliable on-board perception module in future autonomous driving systems. Its ability to segment scenes accurately and in real-time makes it a promising solution for enhancing the safety and effectiveness of autonomous vehicles, especially in complex urban environments where precise object detection and scene understanding are paramount.

Chapter 4

Monocular 3D Object Detection

This chapter solves the 3d object detection problem using only single camera input by introducing additional adaptive depth supervision signals, which also avoids bringing computational burdens.

Associated Publications This chapter is based on the following published work:

Wang C, Aouf N. Depth-Enhanced Deep Learning Approach For Monocular Camera Based 3D Object Detection[J]. Journal of Intelligent & Robotic Systems, 2024, 110(3): 101.

Automatic 3D object detection using monocular cameras presents significant challenges in the context of autonomous driving. Precise labeling of 3D object scales requires accurate spatial information, which is difficult to obtain from a single image due to the inherent lack of depth information in monocular images, compared to LiDAR data. In this chapter, a novel approach is proposed to address this issue by enhancing deep neural networks with depth information for monocular 3D object detection. The proposed method comprises three key components: 1)Feature Enhancement Pyramid Module: The conventional Feature Pyramid Networks (FPN) is further explored by introducing a feature enhancement pyramid network. This module fuses feature maps from the original pyramid and captures contextual correlations across multiple scales. To increase the connectivity between low-level and high-level features, additional pathways are incorporated. 2)Auxiliary Dense Depth Estimator: An auxiliary dense depth estimator is introduced that generates dense depth maps to enhance the spatial perception capabilities of the deep network model without adding computational burden. 3)Augmented centre Depth Regression: To aid centre depth estimation, additional bounding box vertex depth regression based on geometry is employed. The experimental results demonstrate the superiority of the proposed technique over existing competitive methods reported in the literature. The approach (PAC3DNet) showcases remarkable performance improvements in monocular 3D object detection, making it a promising solution for autonomous driving applications.

4.1 Overview

Autonomous driving is an evolving research topic, with object detection being a key technology alongside planning and guidance systems [82–85]. Existing works in 2D object detection such as [86, 42, 87–89] have made significant progress in recent years. However, 3D attributes as location, size, orientation are required for more precise and safety-guaranteed applications like autonomous driving. Therefore research on deep learning based 3D object detection has gained popularity. Classically, existing 3D object detection approaches are based on LiDAR sensor data or RGB images.

State-of-the-art methods [90–92] rely on the accurate depth information provided by LiDAR point clouds. While achieving descent performance, their implementations are expensive and computational demanding. In order to propose attractive solutions which are characterized by low hardware-costing, low-computational and flexible deployment implementation, monocular 3D object detection methods [15, 93] are explored with impressive progress in prediction accuracy relying on consistency between 2D detection and 3D detection priors. However, the performance is still far from satisfaction due to the natural drawback of image data compared to LiDAR data although the latter lacks of spatial information. [94] uses an independent depth estimator to reconstruct 3D point cloud as an enhanced input representation. The data from 2D detection deep neural network and depth generator are fused and then sent to the 3D detector, which makes the framework miscellaneous. Nevertheless, regressing depth from monocular images is a challenging computer vision problem. Errors in depth estimation heavily affect the detection precision of methods that depend on accurate depth, therefore it becomes the major reason for the performance gaps between pseudo-LiDAR and LiDAR-based detectors.

Besides focusing on the detector framework, recent research shows interest in feature extraction for better performance. Among these works, FPN [95] is an effective framework that is adopted by many solutions as their feature extractor for object detection. In Convolutional Neural Networks (CNNs), the network depths correspond to different levels of semantic features. The small network has high resolution and learns more detailed features, while the deep network has low resolution and learns more semantic features. FPN proposes a feature fusion method using different resolutions. The feature maps of high resolution, and the up-sampled low-resolution features are element-wise added, so that the features of different levels are enhanced. Since this method only performs cross-layer connection and element-wise summation on the basis of the network, the increase of calculation is minor, while with an excellent performance improvement. Furthermore, PANet [96] finds the long path from low-level structure to topmost features, increases the difficulty to access accurate localization information. [97] further explores the inner connection among the feature pyramids and proposes to gather these information and fuse them into one feature.

In this chapter, a 3D object detector is proposed utilising enhanced depth information to locate the object positions. VoVNet−v2 [33] is adopted as the backbone connected to a feature pyramid structure. The estimation of an object's 3D location is classically decoupled to the 2D centre with an offset to the projected 3D center, and its depth [98]. Rather than estimating single depth for each object, an additional branch is used to regress vertex depths assisting the centre depth formatting. Considering the uncertainties of vertices depth estimation and direct regression, the final estimation is formulated as a confidence-weighted average estimation problem. The proposed combination allows the model to flexibly choose more suitable estimators for robust and accurate predictions. Although the vertex depth estimator provides improvement to object locating, it does not change the ill-posed nature of point depth prediction, which is lacking contextual information from surrounding pixels in the regression mechanism. To this end, the auxiliary dense depth estimator is introduced to update the parameters in the feature extractor (the VoVNet backbone and the feature enhancement pyramid module) which effectively assists point depth prediction. During inference time, ADDE is not in use and removed to avoid increasing computational burden. To further boost the detection accuracy from the source feature, an efficient feature enhancement pyramid module is designed to capture the intact global contextual information from all feature levels.

The proposed model is trained and evaluated on the popular dataset nuScenes [54]. As this dataset only provides centre depth ground truth and to generate the additional ground truth data it needs for the validation of the method, the existing label attributes and the geometry constraints among them are investigated. The dense ground truth depths are created by LiDAR point cloud projections.

This chapter makes several key contributions, which can be summarized as follows:

1. A novel auxiliary dense depth estimator is introduced to enhance the model's perception of depth information. This auxiliary module effectively improves depth estimation capabilities without adding excessive computational burden, making the overall model lightweight and efficient.

2. To achieve more accurate depth estimation, an augmented centre depth module is designed. This module dynamically combines the outputs from the fundamental centre depth predictor and the vertex depth estimator, resulting in more robust and precise depth predictions.

3. The proposed feature enhancement pyramid module significantly enhances the contextual representation of the model. The module effectively fuses feature maps from the original pyramid, capturing contextual correlations across multiple scales. Additionally, it facilitates seamless integration into other detectors, leading to improved performance for various object detection tasks.

Overall, extensive evaluations on the widely-used benchmark dataset, nuScenes, demonstrate the effectiveness of the proposed algorithm when compared to state-of-the-art methods.

4.2 Related Works

4.2.1 Monocular 3D Object Detection

In recent years, many researchers develop 3D object detection based on camera feeds for the convenience of low-cost deployment compared to LiDAR based methods. Most of the previous approaches adopt additional networks in their architectures or auxiliary labelling data, such as keypoints, CAD models, instance segmentation or even the use of stereo cameras feed. Monocular 3D detection is more challenging due to the natural limitation of acquiring reliable 3D information based on a single image. To tackle this problem, RTM3D [93] predicts the keypoints of the 3D bounding box and additional properties while realizing real-time performance. [99] uses geometrical heuristics based on the assumption that the objects are always on the ground plane. Prior 3D shapes of vehicles are also leveraged to reconstruct the bounding box for autonomous driving. One of the pioneers, Deep MANTA [100], reconstructs 3D object information utilizing 2D keypoints and template similarities from 3D CAD models. [101] accomplishes 3D reconstruction of vehicles on uneven roads based on monocular camera. The core of this framework is to estimate the 3D shape

and 6DOF pose from the monocular image. 3D-RCNN [102] based on R-CNN can predict the shape, attitude and size attributes of the vehicles, and render the scene at the same time. The obtained mask performs "render-and-compare" loss calculation with the ground truth depth map. MonoGRNet [103] regresses the 3D centre points, the rough instance depths and the approximate 3D positions. This work highlights the difference between the 2D bbox centre and the projected 3D bbox centre to the 2D image. The projected 3D centre point can be considered as an additional keypoint. Inspired by “2D proposal generation" methods, Mono3D [104] filters low-confidence bounding box proposals based on predefined priors (e.g. shape, height, location) to reduce the searching space. [105], on the other hand, uses an additional network to estimate the confidence map to filter meaningless proposals. However, these frameworks inevitably face a huge computational burden despite the reduced proposals they adopt. To this end, single-stage methods propose to directly predict classes and regress other components of the 3D boxes from each feature position, in a similar way of semantic segmentation. Groomed-NMS [106] proposes a detector that generates both 2D anchors and 3D anchors for the given images. The anchor generation is highly related to its class label. CenterNet [107] in particular, utilizes key-point estimation to find the centre point and several regression heads are used to estimate the other attributes of the object, including depth, size, and orientation. Monopair [108] gets inspiration from CenterNet, and improves the final detection results via the spatial relationship between pairs of cars. Compared to CenterNet, the 3D bbox is directly predicted, and the constraint points between virtual pairs of matching cars are also predicted. In order to complete the spatial information, [109] proposes to use input from stereo camera and fuse the feature for proposal generation. The stereo image feeds allow the network to better learns the depth hints. [110] uses a multi-modal framework to fuse the depth feature from a single depth estimator and the RGB feature to capture the spatial cues. [111] utilizes the discrete depth and orientation representation to predict the 3D bounding boxes. An additional segmentation heatmap sub-network is applied for centre point regression, reducing the detection offset significantly. [112] further extends the idea with a fusion strategy by embedding dynamic weights and affinity to combine depth

features and RGB features in multiple network layers. Clearly, these methods could improve the accuracy of the detection, but they are computationally demanding with extra networks and labelled data.

4.2.2 Feature Pyramid

Exploration of using features from different deep neural network layers for computer vision tasks has been made through the years. LRR [113] fuses feature maps to get more details for semantic segmentation. Fully Connected Network (FCN) [59], U-Net [14] aggregate information from lower layers through simple skip-connections. TDM [114] constructs a top-down path with lateral connections and takes the highest resolution fused feature map for object detection. SSD [115], DSSD [116], MS-CNN [117] choose to infer from several feature levels. FPN [95] combines their advantages and becomes a widely used feature extractor for many object detectors. Optimizations have also been made based on the FPN framework. PANet [96] creates a bottom-up path augmentation based on FPN. It aims to shorten the information path and uses the precise positioning information stored in the low-level feature to improve the feature pyramid architecture. ThunderNet [118] up-samples and broadcasts low-level features and fuses them into one detection head. AugFPN [119] proposes consistency supervision to narrow the semantic gaps between features at different levels. For features of various sizes, it introduces adaptive spatial fusion. Same as ThunderNet, the detection head only contains a final feature fusion.

4.2.3 Contextual Dependency

Various studies have illustrated the impact of the contextual information on deep learning based computer vision problems including semantic segmentation [72, 73] as well as object detection [97]. Squeeze-and-Excitation Networks [71] uses spatial-wise average pooling, and two fully-connect layers to model the channel-wise relationships by attention mechanism, reinforcing the representational capability of the model. The self-attention method Non-local module [63] is followed by OCNet [74] and DANet [75], to calculate the contextual information. EPSANet [76] harvests attention map

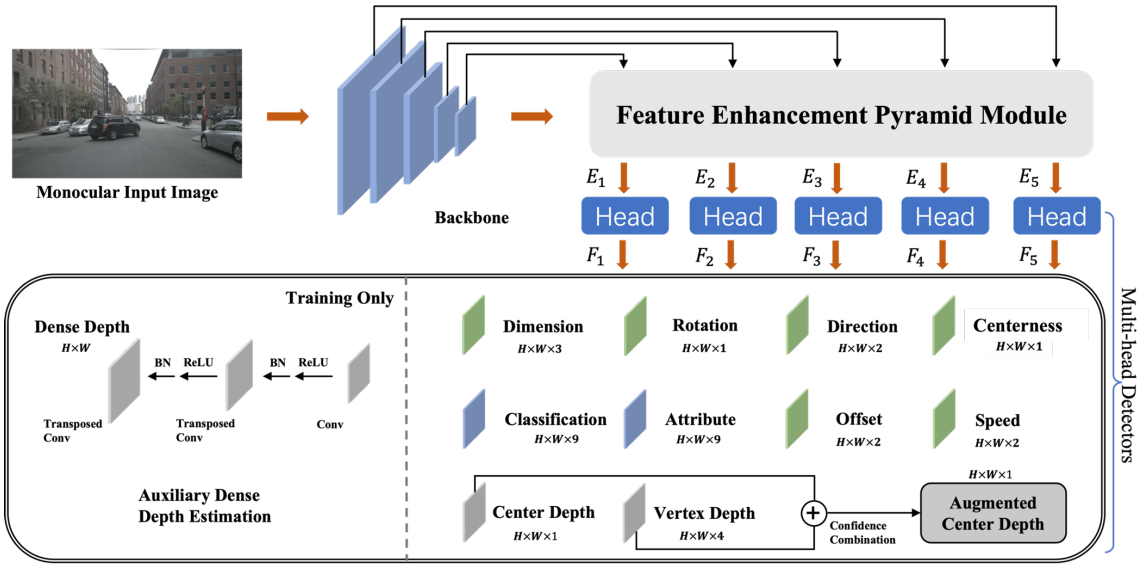

Figure 4.1 Overview of the framework. The features are first extracted from input by the backbone and processed by the feature enhancement pyramid module, which can be seen as a redesigned FPN. This FEPM utilize the asymmetric fusion module to strengthen the representation of feature pyramids. The multi-head detector shares parameters from the backbone and FEPM to regress the bounding boxes. [Wang et al.]

from different sized features pyramid. FAN [120] uses fusion attention which contains channel-wise and spatial-wise aggregation. A pyramid pooling technique is also proposed for computation reduction while the performance is guaranteed. CCNet [65] markedly reduces the parameters and the complexity of the non-local module through the computation of the partial dependencies. By repeating the attention module, it achieves a promising performance.

4.3 Methodology

4.3.1 Framework Overview

Fig. 4.1 illustrates the proposed model, which mainly consists of four sections: the backbone, the Feature Enhancement Pyramid Module (FEPM), the auxiliary dense depth estimation, and the 3D detection heads. VoVNet−v2 [33] is exploited as the backbone network, and take the features from the last four layers as the original feature pyramid. The FEPM module is designed to generate contextual enhanced

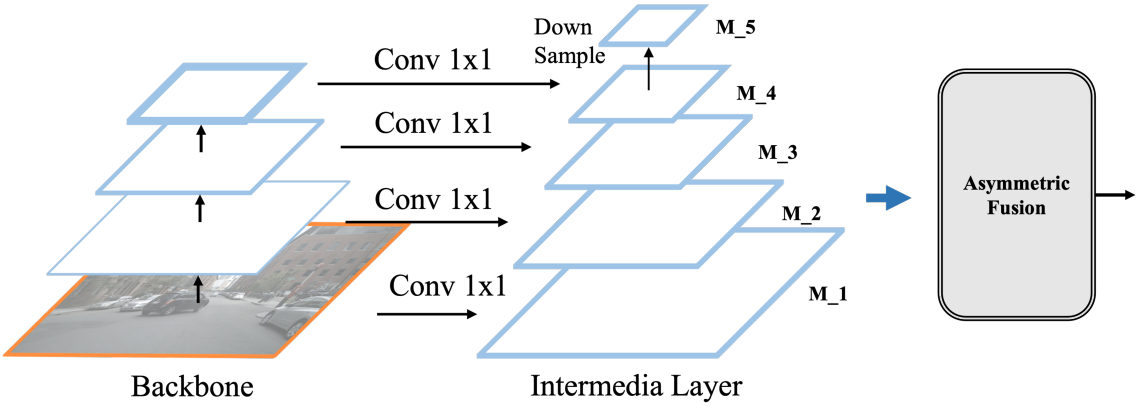

Figure 4.2 Overview of the feature enhancement pyramid module. [Wang et al.]

features from different scales, while several convolution and combination operations are applied to reverse the fusion feature back into pyramids. Then, with the feature pyramid, the auxiliary dense depth estimator (ADDE) will be trained to regress the dense depth maps, updating all the parameters in backbone and FEPM. Finally, the depth estimator is removed and replaced with 3D object detectors to train the multi-task branches including augmented centre depth estimation (ACDE) with vertex depth estimation inside.

4.3.2 Feature Enhancement Pyramid Module

FPN is widely used in 3D object detection tasks. It uses feature maps of different resolutions, generated by intermediate layers, to build a feature pyramid. In order to make up for both high-level and low-level features, FPN integrates the multi-scale context information, fusing features at different levels by skip-connections and element-wise summation. Although FPN achieves great improvement, there is still space to make it more effective. To that end, by implementing the asymmetric fusion module, the FEPM framework further enhances the network with the identifying capability of large instances in higher level features and the contextual dependencies of all levels. First of all, unlike FPN, the feature hierarchy is rebuilt by only applying lateral connections to feature maps from the last 4 layers of the backbone network, denoted as $M_i \in H_i \times W_i \times C_i, i = 1, 2, 3, 4$ as shown in Fig. 4.2. Each lateral connection consists of a 1×1 convolution layer, and reduces the channel number of all feature levels to C, resulting in $M_i \in H_i \times W_i \times C, i = 1, 2, 3, 4$. Note that this

operation does not include skip-connections between backbone features and pyramid features. To make the most of the feature pyramid, M_4 is further down-sampled to get M_5. Here, the asymmetric fusion block is introduced. The proposed asymmetric fusion module as part of the FPN, gathers information from the feature pyramid and generates an enhanced feature map. Then enhanced feature map is recovered back to the pyramid through lateral feature aggregation for detection heads. As shown in Fig. 4.3 and Fig. 4.4, all the feature maps from $M_i \in H_i \times W_i \times C$ are reshaped into $V_i \in N_i \times C$, where $N_i = H_i \times W_i, i = 1, 2, ..., 5$. In self-attention mechanism, Key, Query and Value vectors are used to model the correlation of the features. V_4, V_5 are concatenated, resulting in size $(N_4 + N_5) \times C$ and take the concatenated features as Value and Key vectors. Similarly, V_3 is taken as query vector. Then a matrix multiplication is operated between the Query and transposed Key to obtain:

$$S = Query \times Key^T \tag{4.1}$$

with $Query = V_3, \in (N_3 \times C)$ and $Key^T = (cat(V_4, V_5))^T, \in (C \times (N_4 + N_5))$ and S is the similarity matrix, demonstrating the degrees of correlation between each position in query and key. Note that the size of $S \in N_3 \times (N_4 + N_5)$. After that, a SoftMax layer is applied on S to calculate the attention map $A \in N_3 \times (N_4 + N_5)$. Then, a matrix multiplication is performed between the matrix A and the value vector generated above to obtain:

$$M^{'} = A \times Value \tag{4.2}$$

where Value equals to Key and $M^{'}$ has the size of $N_3 \times C$, same as M_3. Finally, the output Y_1 of fusion block 1 containing rich contextual information is obtained by an element-wise sum operation between feature map M_3 and the calculated $M^{'}$:

$$Y_{1(H,W)} = \gamma M^{'}_{H,W} + M_{3(H,W)} \tag{4.3}$$

Where H, W specify each position in $Y_1, M^{'}, M_3$. M_3 is the original feature map before reshaping and γ is a learning scale parameter. It is initialized as 0 and

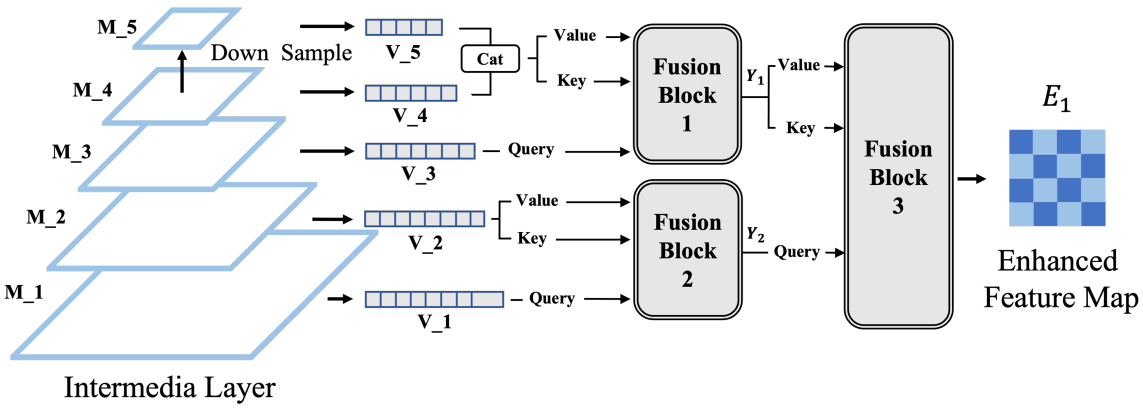

Figure 4.3 Asymmetric Fusion Module. [Wang et al.]

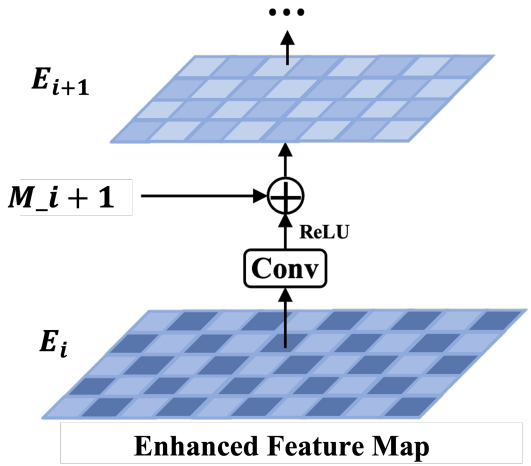

Figure 4.4 The Fusion block 1 and 2. [Wang et al.]

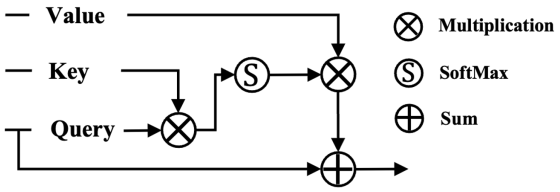

Figure 4.5 The Lateral Feature Aggregation, $i = 1, 2, 3, 4$. [Wang et al.]

gradually learns to increase with more weight [121]. By adding the original feature to contextual information, the representation capacity of the network is enhanced. Using Equation (4.1) and Equation (4.2), the same operations implemented on V_1 and V_2 to obtain the enhanced feature Y_2 with the difference that $Query = V_1$, $Key = V_2$ and $Value = V_2$. Y_2 is the output of fusion block 2 using (3). Y_1 and Y_2 are then used to calculate fusion block 3 output $E_1 \in H_1 \times W_1 \times C$, which is the final enhanced feature map. Note that he order of value, key and query assignments to intermediate layers are fixed to get the E_1 the same size as M_1. In the last step of the FEPM module, lateral feature aggregation, as shown in Fig. 4.5, is performed. The aggregation first takes the enhanced feature map E_1 and a coarser map M_2 through lateral connection and generates the new feature map E_2. Specifically, E_1 first goes through a 3×3 convolutional layers with stride 2 to reduce the spatial size. Then each element of the original feature map M_2 and the down-sampled enhanced map are added through lateral connection. The fused feature map E_2 is then manipulated by another 3×3 convolutional layer to generate E_3 for the following sub-blocks. This is an iterative process and terminates after obtaining the enhanced feature map E_5. All convolutional layers are followed by a ReLU [122]. Through lateral feature aggregations, the enhanced feature maps are obtained as a new enhanced feature map pyramid E_1, E_2, E_3, E_4, E_5 that is ready to use by the detection heads.

Compared to only using skip-connections to combine the information from different scales in traditional Feature Pyramid Network, the method gathers the information by applying the asymmetric attention mechanism. The correlation of two high-level feature maps and another correlation for three low-level feature maps are calculated, and then a global feature map is obtained with enhanced the contextual information by execute the same operation on the two outcomes from before.

4.3.3 Auxiliary Dense Depth Estimation (ADDE)

One of the draw-back of monocular 3D object detection is the inaccurate depth estimation. As the dataset to be worked on, NuSenes, does not consider monocular depth prediction task, the LiDAR point cloud is projected to six camera view-

angles to generate depth ground truth. In the proposed Auxiliary Dense Depth Estimating (ADDE) module, per-pixel depth predictions are implemented on all levels of feature maps obtained in Feature Enhancement Pyramid Module. Fig. 4.1 details the architecture of the proposed ADDE. Each feature level is connected to a convolutional layer, followed by two transposed convolutional layers. Extra ReLU and batch normalization layers are used for fast converging. Then, the ADDE network is updated by minimizing the inverse smooth L_1 norm loss function below:

$$\ell(d,p) = \sum_{n=1}^{w*h} l_n \tag{4.4}$$

with

$$l_n = \begin{cases} 0.5(d_n - p_n)^2/beta & \text{if } |d_n - p_n| < beta \\ |d_n - p_n| - 0.5 * beta & \text{otherwise} \end{cases} \tag{4.5}$$

where d is the ground truth depth and p represents the inverse predicted depth values in each position of (width, height). Instead of directly predicting the depth, the log of depth is regressed, which is $p = e^{d_{predict}}$. As it can be seen in Fig. 4.1, the ADDE and the augmented centre depth estimator (Center Depth and Vertex Depth) share the same backbone and feature pyramid network. During training, the depth predicting capability of the auxiliary dense depth estimator allows the model to better regress the augmented centre depth, while also benefits other targets learning in the framework solution proposed with effective transfer in a multi-task learning scheme. Note that the augmented centre depth prediction does not directly rely on the output of the ADDE. During inference time this prediction is realized as a regression branch on its own, as shown in Fig. 4.1. Doing this will significantly reduce computation.

4.3.4 Augmented centre Depth Estimation (ACDE)

Single centre depth estimation for object detection is unstable and inaccurate. It is even harder for the centre depth regression branch to converge in a multi-task

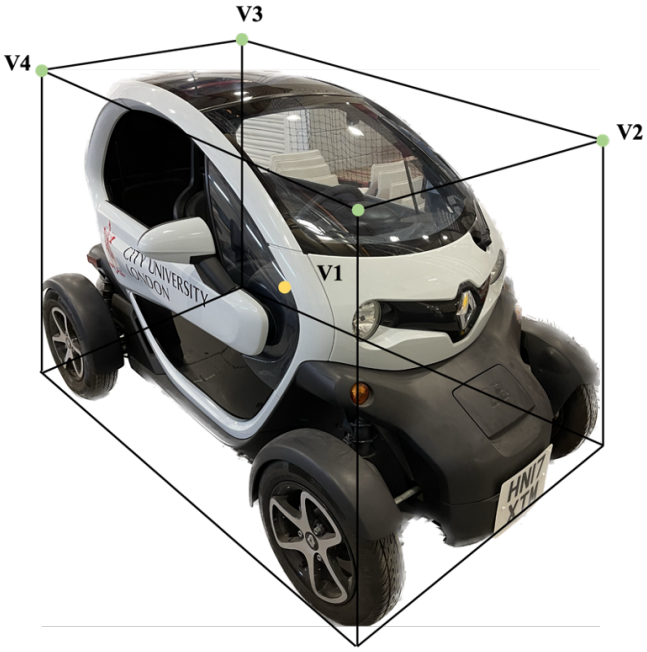

Figure 4.6 Four top vertices (green dots) and the centre (yellow dot) of a bounding box. [Wang et al.]

regression structure, as the modality of depth prediction is far distinct from other branch tasks such as classification and dimension estimation. Based on this, a better position reasoning approach is explored for the depth regression branch and further take advantage of the Auxiliary Dense Depth Estimator (ADDE). The proposed ACDE module in the section includes two regression branches, centre depth and vertex depth. Extra depth is regressed from vertices and utilise the geometry of the 3D detection bounding box to finalize the augmented centre depth prediction. Based on geometry constraints showed in Fig. 4.6, the centre depth equals to the average value of vertex depths:

$$d_v = (d_{v1} + d_{v2} + d_{v3} + d_{v4})/4 \tag{4.6}$$

Assuming that the ground is flat, the four column frames of a bounding box are always perpendicular to the ground plane, which means the four vertices on the top have the same depth as their corresponding vertices on the bottom. In order to lighten the computation of this vertex depth regression branch, only the top four corner vertex depths are regressed. This branch also benefits from the network

parameters trained by the ADDE. To make sure the parameters in this branch and ADDE are updated in the same way, keeping the consistency between different depth estimators, the same loss function as $\ell(d, p)$ is applied with an additionally introduced confidence factor δ, resulting in L_{depth} as follows:

$$L_{depth} = \frac{\sum_{i=1}^{4} \ell(d,p)_i}{\delta} + log(\delta) \tag{4.7}$$

with

$$\ell(d,p) = \sum_{n=1}^{w*h} 0.5(d_n - p_n)^2/beta \tag{4.8}$$

if $|d_n - p_n| < beta$. otherwise:

$$\ell(d,p) = \sum_{n=1}^{w*h} |d_n - p_n| - 0.5 * beta \tag{4.9}$$

where $\ell(d,p)_i$ indicates the loss of four vertex depth. When calculating the centre depth, there is only l_n with δ in Equation (4.7). And $p_n = e^{d_{predict_n}}$, which is the error between the ground truth and the log of predicted depth. δ models the uncertainty of centre and vertex depth regression tasks. To minimise this loss, the network needs to have high uncertainty value which demonstrates its confidence of the prediction. The term $log(\delta)$ can avoid trivial solutions and encourage the model to be optimistic about accurate predictions. Same for the vertex depth estimation, confidence prediction is added to the centre depth branch as following: the average vertex depth value d_v and the centre depth value are combined according to their confidence ratio as shown in Equation (4.10). The confidence combination can assign more weights to the outputs of the more confident estimator; therefore being robust to potentially inaccurate predictions.

$$d = \frac{\delta_c * d_c + \delta_v * d_v}{\delta_c + \delta_v} \tag{4.10}$$

Similar to the dense depth learning, vertex depth ground truth is also unavailable from the dataset. This information is acquired based on the centre depth and other

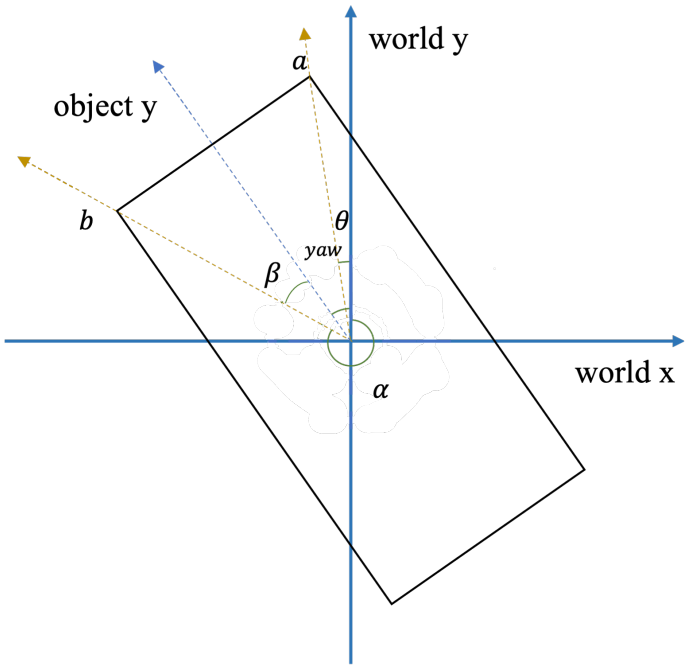

Figure 4.7 The angle (yaw) between the perceived object and world coordinates, which is deified by the ego camera. [Wang et al.]

existing annotations. In Fig. 4.7, yaw is the angle between the world coordinate and the ego coordinate. θ and α are the angles of the diagonal and world coordinate. The centre of the bounding box (noted as bb in the following) is at the origin, and the position of vertex a can be located on the four quadrants. The location situations of vertex a are further simplified into two types: 1. the first and third quadrants; 2. the second and fourth quadrants. The diagonal and β can be calculated by:

$$diag = \sqrt{(length_{bb})^2 + (width_{bb})^2} \tag{4.11}$$

$$\beta = \arctan(width_{bb}/length_{bb}) \tag{4.12}$$

Then the depth of the four bounding box vertices is calculated as following:

If $\beta < \text{yaw} < \pi * 0.5$ or $-\pi - \beta < \text{yaw} < -0.5 * \pi$

$$\begin{cases} d_{v1} = d_{center} + \cos(\beta - yaw) * diag \\ d_{v2} = d_{center} - \cos(\beta - yaw) * diag \\ d_{v3} = d_{center} + \cos(\beta + yaw) * diag \\ d_{v4} = d_{center} - \cos(\beta + yaw) * diag \end{cases} \tag{4.13}$$

otherwise,

$$\begin{cases} d_{v1} = d_{center} + \sin(\beta - yaw) * diag \\ d_{v2} = d_{center} - \sin(\beta - yaw) * diag \\ d_{v3} = d_{center} + \cos(\beta + yaw) * diag \\ d_{v4} = d_{center} - \cos(\beta + yaw) * diag \end{cases} \tag{4.14}$$

where d_{center} denotes the bounding box centre depth ground truth.

4.3.5 Multi-Head Detectors

After the feature enhancement pyramid module, as shown in Fig. 4.1, five detection heads (blue heads) are respectively connected to the five pyramid feature maps $(E_1, E_2, E_3, E_4, E_5)$, which were aggregated from the FEPM. Each head consists of two sets of four convolutional layers, with kernel size $3 * 3$, stride 1 and padding 1. The first set of four convolutional layers in one head is for classification tasks (blue squares) and the other set is for the rest (green and grey squares) as regression tasks. Finally, the features processed and output by the detection heads $(F_1, F_2, F_3, F_4, F_5)$ will be passed to a convolutional layer with different output dimensions for multiple purposes. The detection heads and the task layers together are called multi-head detectors. The output of each tasks is in the form of a heatmap, and the width and height are the same as the corresponding input level feature maps. The offset regression branch predicts the offset between the 2D centre and the projected 3D centre on the image plane. With the regressed centre depth and offsets, the 3D centre point $[X, Y, Z]$ can be retrieved based on the camera intrinsics. The dimension

regression branch predicts the size of the object. The rotation regression branch predicts the yaw angle, and the outputs from the direction regression branch help to solve the controversial angle situations (when this object in opposite directions). The speed regression branch and the attribute regression branch predict velocity and additional attributes of the detected objects in order to obtain the overall score required for the nuScenes dataset. Following [87], the centerness regression branch is adopted as a filter to retrieve and display only the high-quality 3D detection bounding box predictions.

While training the auxiliary dense depth estimator neither regression branch nor classification branch is attached to the detection heads, as the intermediate step of the whole training. Instead, as shown in Fig. 4.1, two transposed convolutional layers are introduced to predict dense depth information. It has been found more efficient to have extra ReLU and batch normalization layers between convolutional layers and transposed convolutional layers for the dense depth training.

4.3.6 Adopted Training Loss Functions

To train the full 3D detection model proposed in Fig. 4.1, three different loss functions are used. First, cross-entropy loss is used for direction, attribute and centerness predictions:

$$\mathcal{L}_{ce} = \mathcal{L}_{direc} + \mathcal{L}_{attri} + \mathcal{L}_{centr} \tag{4.15}$$

For the rest of the regression branches, smooth $L1$ norm loss is applied as specified in Equation (4.4) and Equation (4.5) but without inverse the prediction values. The loss of the offset, dimension, rotation and speed branches, together are denoted as $\mathcal{L}_{sml1}$. Note that the augmented centre depth branch uses a modified $L1$ norm loss with confidence assignment as shown in Equation (4.7), Equation (4.8), and Equation (4.9). The loss of centre depth and vertex depth branches is denoted as $\mathcal{L}_{depth} = \ell(d, p) + L_{depth}$, where the two terms are defined in Equation (4.4) and Equation (4.7) respectively. A simple focal loss is used for the classification branches,

denoted as $\mathcal{L}_{focal}$. To sum up, the total loss is defined as:

$$\mathcal{L}_{total} = \mathcal{L}_{ce} + \mathcal{L}_{sml1} + \mathcal{L}_{focal} + \mathcal{L}_{depth} \tag{4.16}$$

4.4 Experiments

4.4.1 Dataset and Evaluation Standard

Dataset

nuScenes3D The nuScenes 3D detection benchmark dataset [54] is adopted in the experiments and it consists of 1000 multi-modal videos with 6 cameras on the top of a car, covering the full 360-degree field of view from the ego car. the dataset is split into 700 videos for training, 150 for validation, and 150 for testing. The 3D detection models are evaluated by regressing 3D bounding boxes of 10 object classes, from multiple types of vehicles to pedestrians, over an amount of frames from videos. NuScenes is becoming one of the definitive benchmarks for 3D object detection because of its variety and quantity of scenarios and labels.

Evaluation

nuScenes Detection Score The detection performance is evaluated by the official metrics adopted in nuScenes and are distance-based mAP (Average Precision metric) and comprehensively defined NDS (NuScenes Detection Score), which is a more intuitive overall score to assess the 3D detection model performance on nuScenes dataset. The mAP defines the match between the ground truth and the predicted bounding boxes that have the smallest 2d center-distance under a certain threshold, where NDS is computed by the weighted sum of the mean average precision(mAP), average translation error(mATE), average scale error(mASE), average orientation error(mAOE), average velocity error(mAVE) and average attribute error(mAAE). To calculate NDS, the true positive error needs to be transformed to true positive

scores(TP), and normalize the weighted score sum as:

$$NDS = \frac{1}{10}[5 * mAP + \sum max(1 - TPerror, 0)] \tag{4.17}$$

The model is tested on the validation dataset, and report NDS and mAP, along with all the five true-positive metrics that are critical to 3D detection.

4.4.2 Implementation Details

VoVNet−v2 [33] is adopted as the backbone network, with input size of 1600×900. Each feature level map from the FEPM connects to a detection head, attached to four $H \times W \times 256$ convolutional layers, a BatchNorm layer [123], and a ReLU layer, plus another $H \times W \times num$ convolutional layer for different classification and regression tasks, where num is the output size. The whole model is trained using stochastic gradient descent (SGD) [124] optimizer with an initial learning rate of $2e^{-3}$ and weight decay as $1e^{-4}$, warm-up iterations at 500 and warm-up ratio of 0.33. The multi-task learning weight is set to 1. The model is trained for 12 epochs with a batch size of 16 on a single Nvidia A100 GPU, and finetune the loss weights of depth-related regression branches for another 12 epochs. The random horizontal flip is adopted as the only data augmentation used here.

4.4.3 Results

In the result section, the quantitative and qualitative results are presented. A detailed ablation study is given as it is shown to prove the efficiency of the different modules introduced in the work.

Quantitative and Qualitative Analysis

The training progress is shown in Fig. 4.8 and Fig. 4.9. As expected it is harder for depth regression to converge than other tasks which proves again that the problem tackled in this research is tough since a multi-task training is conducted. The plunge in loss responds to the drop of learning rate during training. Then the performance of the proposed method on nuSecnes validation set is showed. Comparisons are

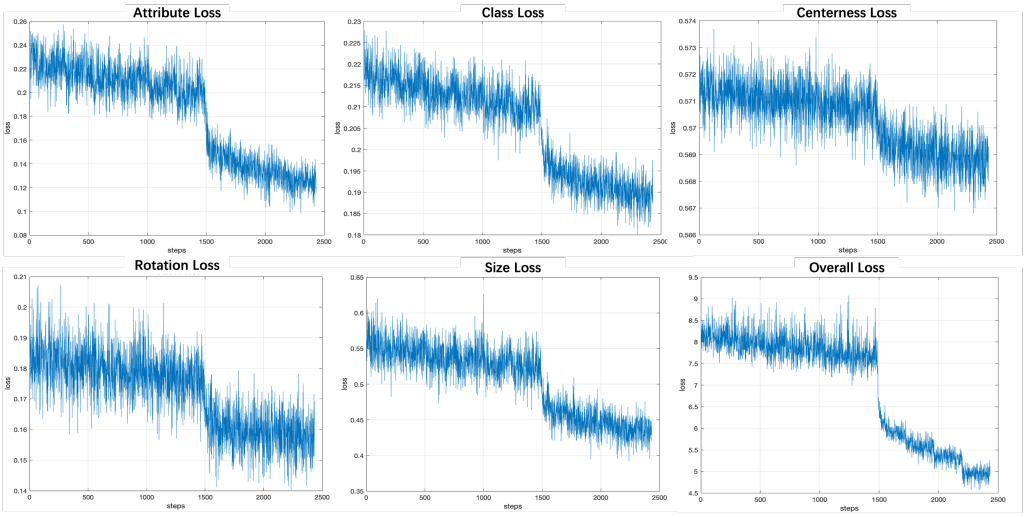

Figure 4.8 Training progress. As a multi-task learning approach, the losses of multiple tasks are displayed. Note that one step means training on a batch size. [Wang et al.]

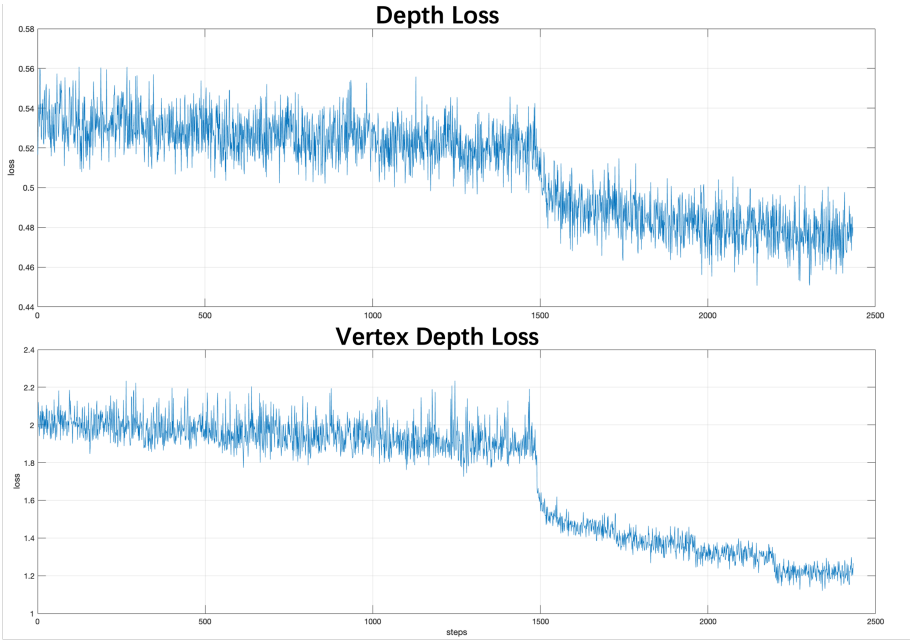

Figure 4.9 Training progress of centre depth estimator and vertex depth estimator. [Wang et al.]

Methods	Modality	mAP	mATE	mAVE	mAAE	mASE	mAOE	NDS
CenterNet[98]	camera	0.306	0.716	1.426	0.658	0.264	0.609	0.328
MonoDIS[15]	camera	0.304	0.738	1.553	0.134	0.263	0.546	0.384
FCOS3D[125]	camera	0.343	0.725	1.292	0.153	0.263	0.422	0.415
PGD[126]	camera	0.369	0.683	1.268	0.185	0.260	0.439	0.428
AIML-ADL	camera	0.352	0.696	1.592	0.122	0.696	0.392	0.429
DD3Dv2[127]	camera	0.431	**0.570**	-	-	0.250	**0.380**	**0.480**
PointPillars[90]	LiDAR	0.305	0.520	0.316	0.368	0.290	0.500	0.450
CVFNet[128]	LiDAR	0.548	0.291	0.349	0.139	0.248	0.389	0.633
PAC3DNet	camera	**0.434**	0.581	**1.246**	**0.053**	**0.238**	0.614	0.461

Table 4.1 Results on nuScenes dataset in order of NDS scores. The proposed method Deep Assisted 3D Object Detection (PAC3DNet) outperforms existing camera based methods. The comparisons with LiDAR methods are also demonstrated, which has nature advantages in position related tasks. Although they have higher overall NDS scores, PAC3DNet still has competitive performance in mAAE, mASE and mAP. [Wang et al.]

displayed of the results with other state-of-the-art monocular 3D detectors, as shown in Table 4.1. It can be observed that the approach achieves better performance than other camera-based methods in terms of the mAP which is the most important locating metric in the benchmark. Furthermore, it is worth noting that the approach outperforms other non-extra-depth-assisted methods by large margins. For instance, compared to FCOS3D, CenterNet, MonoDIS, PAC3DNet exceeded their mAP by 9.1%, 12.8% and 13.0%, respectively. Thanks to the depth reasoning modules ADDE and ACDE, it is worth note that the method achieves much better mAVE than the three methods mentioned above. Despite the absence of continuous multi-frame data as input, the depth information strongly helps the prediction of the speed. Compared to LiDAR based methods, PAC3DNet surpasses PointPillars at mAP by 12.9%. However, as expected, LiDAR based methods naturally got better NDS than most camera based approaches and better in mAVE category, due to the accurate point cloud information. To recover the missing distance information in monocular images, some depth-assisted methods like ours achieve better results than regular methods. Compared to PGD, AIML-ADL, DD3Dv2, PAC3DNet still has 6.5%, 8.2% and 0.3% of improvement in mAP, and 22%, 35% in mAVE compared to PGD and AMIL-ADL. The detailed scores on mAP in each classes can be found in Table 4.2. the model works good for traffic cones, barrier and most importantly, cars. For categories of big objects such as trucks and buses, the precision still needs improvement. The

car	truck	bus	trailer	construction vehicle
0.607	0.364	0.481	0.239	0.156
pedestrian	**motorcycle**	**bicycle**	**barrier**	**traffic_cone**
0.510	0.397	0.381	0.604	0.602

Table 4.2 PAC3DNet mAP Results on each categories. [Wang et al.]

Methods	NDS	Inference Time (s/100task)
FCOS3D[125]	0.415	**4.23**
PGD[126]	0.428	5.19
PAC3DNet	**0.461**	4.55

Table 4.3 Inference time testing results, calculated by second per 100 frames of detection. [Wang et al.]

performance drop for this case and on this challenging dataset is due to the frequent occlusion by smaller objects and those objects being out of images. Further work is required and expected in the future.

The inference time of other two code-available monocular-based methods are tested and showed in Table 4.3, FCOS3D and PGD (a depth-assisted method) on the same hardware—a single RTX 4090 graphics card to compare the computational efficiency of the method. As shown in the Table 4.3, the method achieved the highest nuScenes detection score (NDS). Regarding computational efficiency, the inference time is 14% faster than PGD, supporting the claim that the depth module avoids additional computational burden. Additionally, the asymmetric attention mechanism in the Feature Enhanced Pyramid Module (FEPM) is more efficient than the traditional attention mechanism. When compared to FCOS3D, although the method is 0.32 seconds slower, PAC3DNet surpasses their NDS by 4.6%.

A more comprehensive analysis of the visualized results is attained using the nuScenes dataset. The showcases are categorised into comparisons based on differences in range and environmental conditions. To ensure the presentation of only the most precise bounding boxes, a threshold is implemented for box display, thereby disregarding lower-scoring boxes solely for display purposes. Fig. 4.10 illustrates an example of the filtering process applied to generated bounding boxes. As shown in the distance category of Fig. 4.11, the 3D detection deep network model demonstrates remarkable precision in detecting most objects across all ranges. For instance, in the parking lot

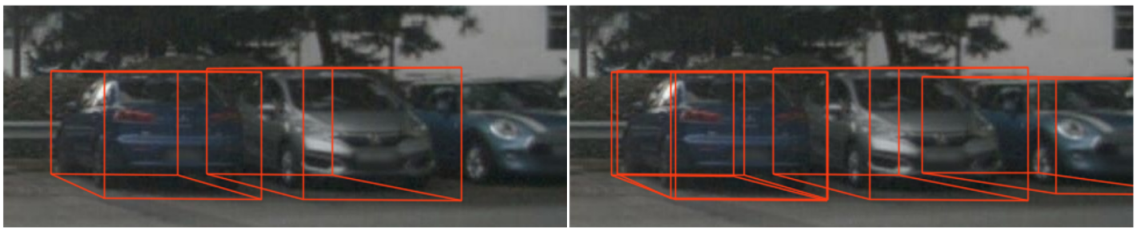

Figure 4.10 Visualization with (left) and without (right) thresh hold filters. [Wang et al.]

Figure 4.11 Visualization of detection results, categorised by distances of the objects. [Wang et al.]

scenario illustrated in Fig. 4.11 (d), the algorithm accurately generates bounding boxes even amidst heavy occlusions between cars. Similarly, the image featuring large trucks and two cars in Fig. 4.11 (c) highlights the robustness of the 3D detection approach. However, objects situated at considerable distances, as showcased in examples like Fig. 4.11 (d) and Fig. 4.11 (e), pose challenges for precise tracking by the model. Additionally, incomplete objects, visible less than half, due to proximity

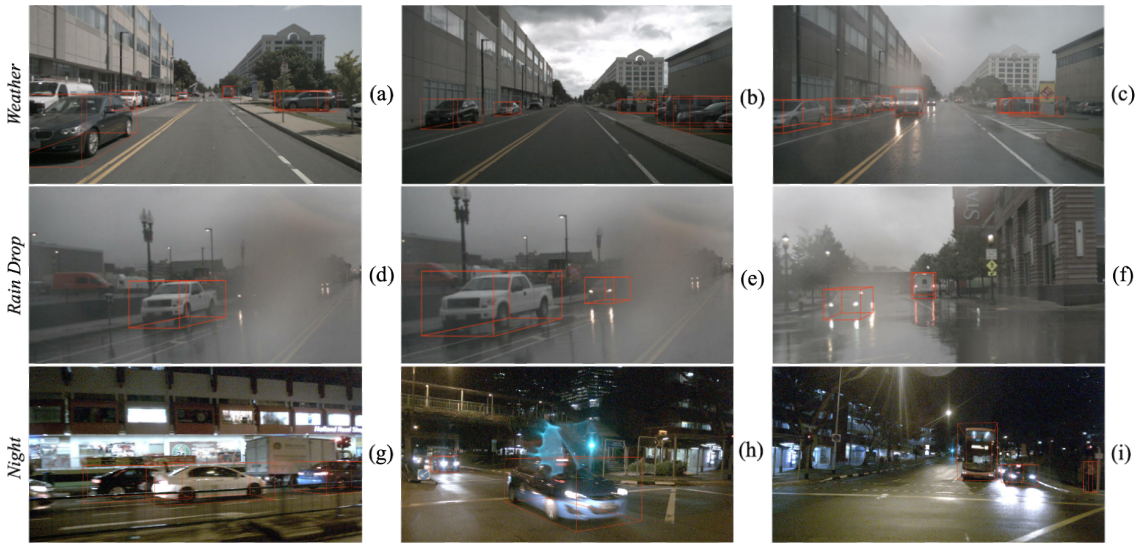

Figure 4.12 Visualization of detection results, categorised by different environment conditions. [Wang et al.]

to the camera also present difficulty for accurate detection. To increase the detecting difficulties, the comparisons under various environmental conditions is considered to better show the robustness of the model. As shown in Fig. 4.12, the performance tests are conducted in scenes characterized by sunny (Fig. 4.12 (a)), cloudy (Fig. 4.12 (b)), and rainy (Fig. 4.12 (c)) weather conditions. Remarkably, the model exhibits flawless handling across all moderate weather condition changes, showcasing its adaptability and reliability. Then the model's performance is compared under heavy rain conditions with raindrops on the camera, which significantly impact the visual quality. In Fig. 4.12 (e) and Fig. 4.12 (f), despite heavy occlusions affecting the red cars, the model adeptly detects them and generates bounding boxes surpassing the threshold. However, under extreme conditions illustrated in Fig. 4.12 (d), where the red car is nearly invisible due to raindrop blockage, the detection falls below the threshold. In night conditions, the model exhibits robustness against poor illumination and reflections. In Fig. 4.12 (g) and Fig. 4.12 (i), despite challenging conditions such as low light and blurriness, the model accurately bounds the front objects, although the bounding box for the overlapped truck is filtered out. In Fig. 4.12 (h), the method successfully bounds the car even in the presence of reflections from the traffic light, simulating a camera failure scenario.

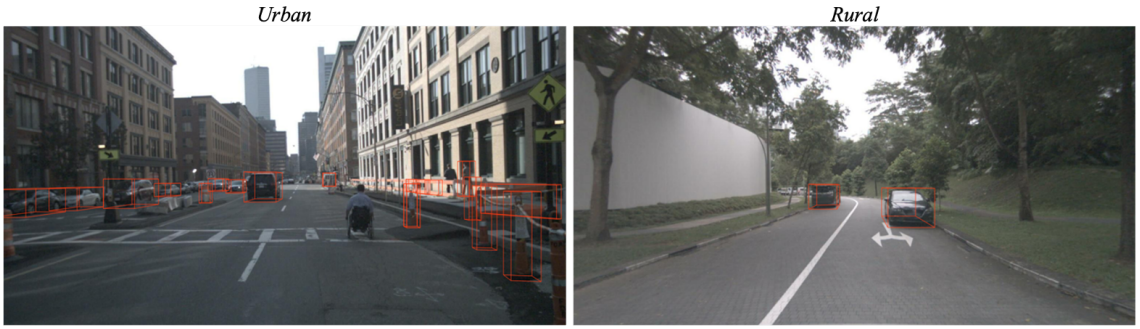

Figure 4.13 Visualization of detection results in urban and rural areas. [Wang et al.]

Moreover, it is discovered that the dataset assumes the road to be flat by default. In Fig. 4.12 (i), the slanted bus results from the gradient of the road. While the generated bounding box effectively locates the object, it's worth noting that the default ground is perpendicular to the x-axis of the camera coordinate (world coordinate), causing the bounding boxes to align parallel to the ground. Addressing this challenge in future research involves exploring methods to detect the roll angle information of these objects and adjust their bounding boxes accordingly. Fig. 4.13 illustrates the comparison of environment layouts in different areas. In urban environments, characterized by high complexity and a higher density of objects, the task of detection becomes more challenging due to the increased clutter. However, even in such demanding scenarios, the model maintains robust performance. Conversely, in rural environments with more vegetation, where the scene complexity differs, the model continues to demonstrate consistent and reliable performance.

In conclusion, the method exhibits robust performance across a wide spectrum of detection ranges, ranging from close proximity to distant objects. Furthermore, it proves its adaptability and reliability across diverse environmental conditions, including variations in lighting, weather, and scene complexity.

Ablation Studies

An in-depth comparison is conducted among the three proposed components, with Table 4.4 showcasing their performance across various evaluation metrics. Initially, the results of the Feature Enhancement Pyramid Module (FEPM) is presented. It is evident that the FEPM module yields a significant improvement, contributing

Parts	mAP	mATE	mAVE	mAAE	NDS
Baseline	0.343	0.725	1.292	0.153	0.415
FEPM	0.374	0.714	1.275	0.104	0.427
ADDE	0.412	0.653	1.249	0.072	0.446
ACDE	**0.434**	**0.581**	**1.246**	**0.053**	**0.461**

Table 4.4 Ablation studies of proposed components on nuScenes dataset. [Wang et al.]

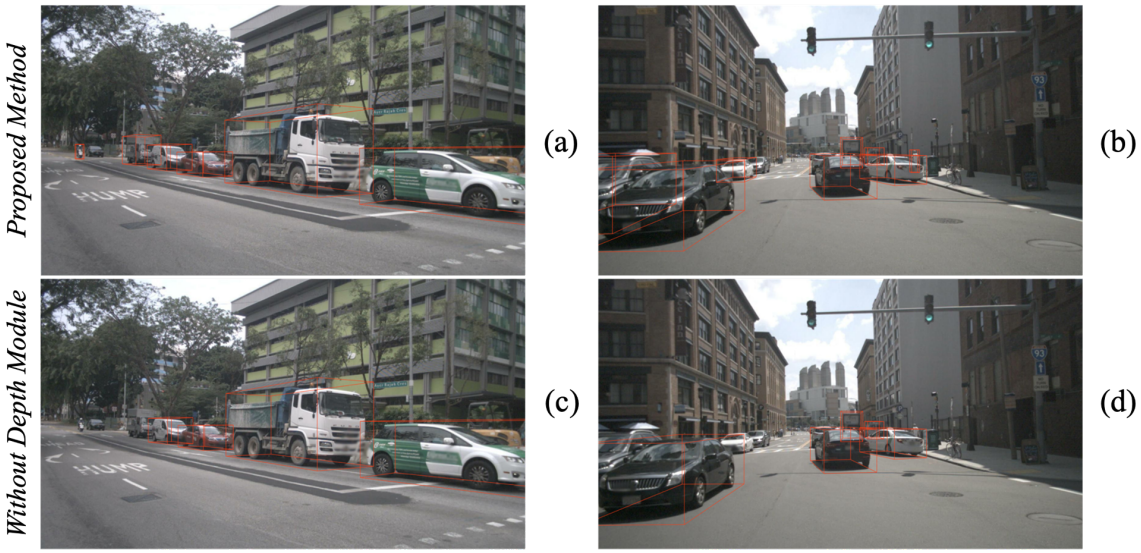

Figure 4.14 Visualization of detection results regarding the efficiency of the depth-assisted module. [Wang et al.]

to a 1.2% enhancement in the overall NDS compared to the baseline (FCOS3D). A noticeable boost shows in the mAAE (attribute errors). Thanks to the asymmetric fusion block, the fused feature benefits of the strong contextual information from different feature levels. With the Auxiliary Dense Depth Estimator (ADDE), the performance is further raised to 0.412 in mAP. The ADDE module offers the recognition of spatial information, and this capability remains in the shared parameters leading to a better NDS at 0.446. The Augmented centre Depth Estimation (ACDE) also improves the results. The regression of it is actually done the same way as centre depth, however, the confidence voting mechanism provides more robustness when one of the regression branches is not sure about its prediction. Including ACDE in the 3D object detection module achieves state-of-the-art performance at 0.434 mAP and 0.461 NDS.

Visualizations is included comparing the results from the depth-assisted model (combining ACDE and ADDE) with those from the depth-less model. the decision to combine ACDE and ADDE into one model stems from their significant contribution to performance gains and their role as depth-assisted components. Therefore, the two models tested are the one with FEPM and the full method. As detailed in last section, during inference, a bounding box score threshold is employed to mitigate the occurrence of multiple boxes on a single object. The absence of boxes on an object indicates that the generated bounding boxes fall below this threshold. Conversely, a higher number of bounding boxes signifies more precise predictions. When comparing the full method to solely using FEPM, it becomes apparent the proposed method generates more bounding boxes within the same scenes in Fig. 4.14. However, the missing boxes primarily occur in regions distant from the camera and in overlapping areas, which are known challenging aspects of the 3D object detection task. This observation underscores the effectiveness of the proposed depth-assisted modules in addressing these challenges and enhancing the overall performance of the detection system.

Assessment on FEPM

This section includes a comprehensive impact analysis of the Feature Enhancement Pyramid Module (FEPM). While original feature maps and enhanced feature maps from intermediate layers can be directly visualized, the distinctions are not apparent due to the complexity of high-level features in deep neural networks. To gain a better understanding of the proposed method's impact, Grad-CAM is used to generate heatmaps for the original feature pyramid and the proposed enhanced feature pyramid, respectively, which are both then projected onto the input images.

Four sets of comparisons using different input images are presented in Fig. 4.15, with colours representing the importance of pixels in predicting 3D bounding boxes. Warmer colours indicate higher importance, signifying areas where the network focuses its attention. Upon examination of the four examples, it becomes evident that with the original feature maps, the highlights are randomly dispersed, suggesting that the network makes less efficient predictions based on imperfect information. In

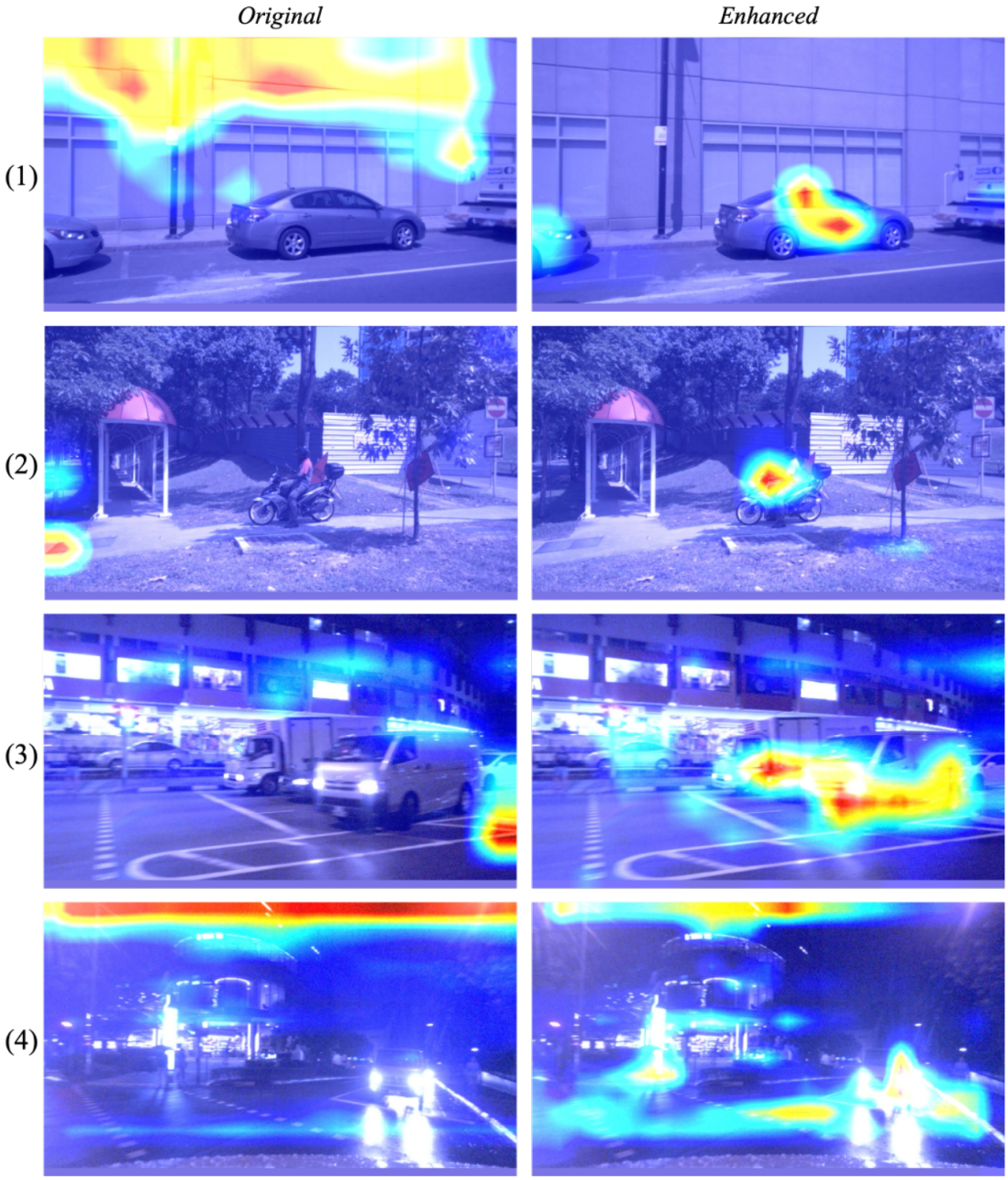

Figure 4.15 Visualization of attention maps on the impact of FEPM module. The Grad-CAM [129] is presented on the input images to show the important regions when making a prediction. Warmer colours mean most influential areas and cooler colours mean less influence. [Wang et al.]

contrast, the enhanced feature maps enable the detection network to concentrate more accurately on the main objects (cars and pedestrians), resulting in better regression of bounding boxes. Particularly in well-illuminated conditions (examples 1 and 2), the feature map enhancement allows the network to precisely focus on

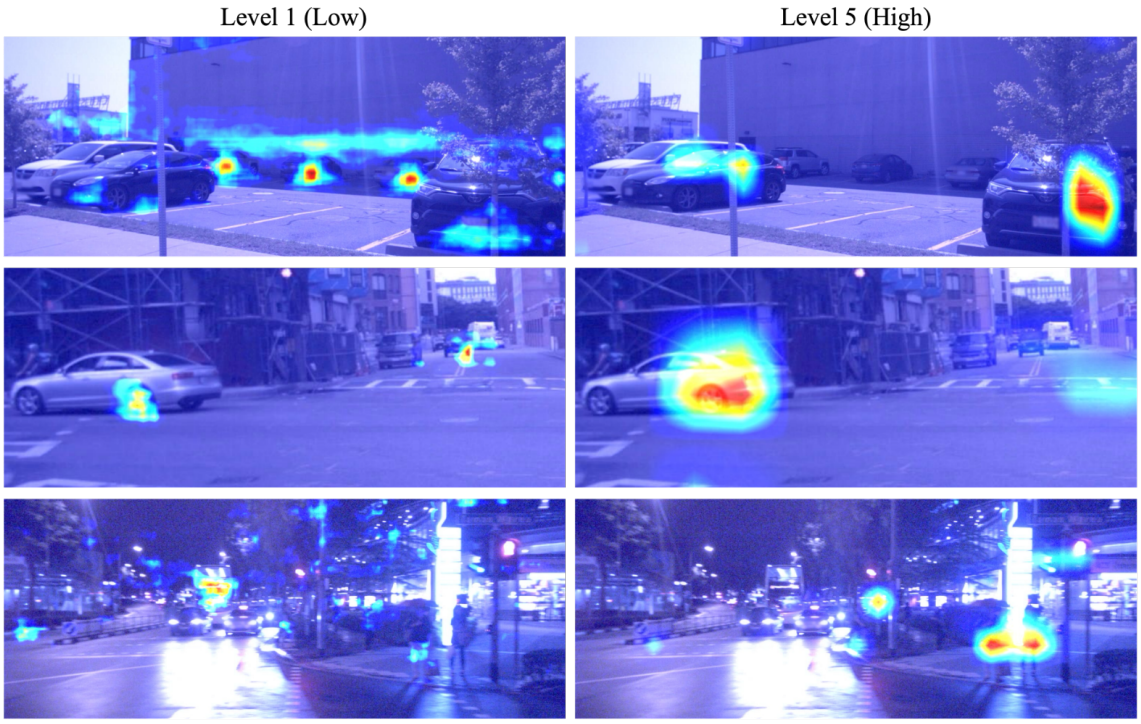

Figure 4.16 Attention visualization of enhanced lower level feature maps and higher level feature maps from FEPM. [Wang et al.]

objects while disregarding irrelevant information. Even under undesirable conditions such as poor illumination and light reflection (examples 3 and 4), the attention map still covers most of the relevant areas. When combined with the results presented in Table 4.4, the feature enhancement demonstrates its efficacy in facilitating the precise prediction of 3D bounding boxes, centre points, attribute classification, and, notably, boosting the accuracy of object speed estimation. This enhancement leads to better object localization in the detection process.

Furthermore, to assess the efficiencies of different levels within the enhanced feature pyramid, three examples of Grad-CAM images from level 1 (representing low-level features) and level 5 (representing high-level features) are presented in Fig. 4.16. In the context of CNN applications, it is widely understood that lower level features encompass more primitive information, such as basic shapes, edges, colours, or textures, while higher level features capture more abstract and complex information. In the specific object detection task, it is observed that the rules reflect to detecting of objects in various sizes and distances. At level 1, smaller and more distant objects are better detected, but there is a partial loss of focus on larger objects, such as

the bus located in the middle of the street and the pedestrians closer to the camera. Conversely, at level 5, the network prioritizes nearer objects in its predictions while paying relatively less attention to distant ones. By leveraging the benefits of different feature levels, the multi-head detector achieves versatile object detection performance across all objects, regardless of their sizes and distances. This indicates the efficacy of the proposed feature enhancement pyramid module in improving object detection capabilities.

4.5 Summary

In summary, this chapter introduces a novel approach for monocular 3D object detection that leverages depth-enhanced deep learning techniques, incorporating spatial information to improve detection accuracy. By incorporating an auxiliary task for dense depth estimation, the network's feature extractor gains a strong awareness of depth, without significantly increasing computational complexity or burden. This auxiliary training allows the model to better interpret the spatial relationships within an image, enhancing its overall understanding of object positioning in 3D space. To take full advantage of the improved depth perception, a simple yet effective vertex depth regression technique is devised. This technique uses a fusion approach that combines both centre depth and vertex depth estimations through a confidence voting mechanism, which greatly enhances the robustness and precision of the overall depth estimation. This process ensures that the model not only predicts object location but also does so with greater accuracy, even in challenging scenarios where depth cues may be ambiguous or incomplete. Moreover, to further strengthen the feature representation from the input data, the Feature Enhancement Pyramid Module (FEPM) is introduced. This module captures contextual dependencies across different feature levels, effectively integrating information from multiple scales. By doing so, it preserves high-resolution details and rich semantic features throughout the network, ensuring that finer, critical details are retained in the final predictions. This combination of depth awareness, robust depth estimation, and enhanced feature

representation makes the proposed method highly effective for 3D object detection in monocular images.

Looking ahead, several challenges and opportunities for future research are acknowledged, which could further refine and enhance the proposed approach. One such challenge lies in accurately computing the roll and pitch angles of objects. Current methods primarily focus on estimating object positions and their orientation on flat surfaces; however, real-world driving scenarios often involve more complex road conditions, such as inclines, ramps, and uneven terrains. Accurately determining the roll (tilt) and pitch (slope) angles of objects in these conditions would greatly improve 3D object detection, especially for applications in autonomous driving where understanding the precise orientation of vehicles, pedestrians, or other obstacles is crucial for safety and decision-making. Tackling this problem requires more advanced algorithms that can integrate these angular estimates into the existing framework without overloading the computational complexity. Another key direction for future research is optimising weight assignment for the various sub-tasks involved in the overall model. As the approach integrates multiple tasks—such as dense depth estimation, vertex depth regression, and object classification—finding the optimal balance between these tasks is essential for maximising performance. Assigning appropriate weights to each task can be intricate, as overemphasising one task may detract from another. However, achieving this balance holds the potential for significant performance gains. Developing a more adaptive and dynamic weight-assignment strategy could help streamline this process, allowing the network to automatically adjust its focus based on the task at hand and the complexity of the environment.

Moreover, while the proposed depth-enhanced monocular 3D object detection approach, along with the incorporation of spatial information and the Feature Enhancement Pyramid Module (FEPM), has demonstrated promising results, there is still room for innovation. Refinements in feature extraction, integration of multi-scale contextual information, and improvements in training efficiency could yield even more robust models. These advancements would not only apply to autonomous driving but also extend to other domains such as robotics, augmented reality, and

smart surveillance systems. Ultimately, the hope is that this work will inspire further advancements in 3D object detection research, pushing the boundaries of what is possible in monocular vision systems.

Chapter 5

Guidance on Depth Completion

This chapter explores Deep Reinforcement Learning (DRL)-based guidance for ground vehicles, using depth images as input. The guidance system consists of two phases: an imitation learning phase and a reinforcement learning phase. The depth module, based on a binocular camera, integrates both supervised and self-supervised learning approaches. Experiments demonstrate that incorporating depth images significantly improves the performance of the guidance network.

Associated Publications This chapter is based on the following published work:

Wang C, Aouf N. Deep Reinforcement Learning based Planning for Urban Self-driving with Demonstration and Depth Completion[C]//2021 21st International Conference on Control, Automation and Systems (ICCAS). IEEE, 2021: 962-967.

In the previous two chapters, semantic segmentation and 3D object detection techniques were discussed as key perception modules, aimed at improving scene understanding for autonomous driving vehicles (ADVs). Depth images are also critical for ADVs to comprehend the 3D layout of their environment, which is essential for autonomous planning. Current techniques for decision-making in driving policies tend to be modular and hand-designed, making them both expensive and inefficient. With advancements in machine learning, learning-based approaches have become a dominant research direction. However, performance in urban driving scenarios remains limited due to the brittle convergence of deep reinforcement learning (DRL) algorithms and degraded observation data. To address these limitations, this chapter proposes a learning-based method that combines DRL with imitation learning (IL), alongside a novel depth completion model designed to enhance perception. The framework is built upon the Soft Actor-Critic (SAC) algorithm [130] and introduces a novel updating strategy to tackle the difficulties that current DRL-IL methods face when transitioning from the IL phase to the DRL phase. Instead of relying on RGB input, a reconstruction-constrained deep fusion depth completion network is proposed to predict a comprehensive and accurate depth map of the environment, using pre-processed synthetic datasets. In experiments, the autonomous driving agent using the proposed algorithm successfully transitions from Imitation Learning (IL) to Deep Reinforcement Learning (DRL) during training, outperforming state-of-the-art methods in challenging urban environments. Furthermore, the driving model maintains competitive performance when using predicted depth images as input, compared to ground-truth data.

5.1 Overview

Self-driving is an expansive research field that combines control systems, computer vision, and artificial intelligence [131]. In recent years, emerging autonomous driving technologies have progressed from academic research to practical applications [132, 133]. Some companies have already tested autonomous vehicles in public environments. However, creating a highly intelligent planning and decision-making model

for self-driving in complex urban scenarios remains a significant challenge. Previous approaches to autonomous driving have largely relied on inefficient hand-designed, model-based methods, which can easily fail in dynamic, real-world situations [134]. With the major successes of deep reinforcement learning (DRL) in various applications, such as AlphaGo and retro game playing, researchers have turned to machine learning to address these challenges. An ideal end-to-end, model-free system is expected to solve both perception and motion planning tasks. However, current raw sensor data, like that from RGB cameras and LiDAR, does not provide the level of accuracy required for high-quality perception. Additionally, the action and state space in driving environments is vast, making it difficult for a policy network to find an optimal driving policy. This has led many researchers to focus on solving sub-tasks within the broader navigation problem, or to develop systems that are limited to completing specific tasks, such as lane keeping [21], where a complete solution still remains challenge.

A depth image is a crucial input for autonomous driving, as depth information is essential for 3D vision tasks, helping to interpret the 3D geometry of a scene. While existing sensors such as LiDAR, radar, and others to acquire depth information, the generated depth maps are too sparse to satisfy the driving systems in complex outdoor environments. This shortcoming hinders the accuracy needed for tasks in autonomous driving. As a result, depth completion remains an important and valuable area in today's robotics-related industries. Recently, many approaches have used monocular cameras to predict depth maps, leveraging the rich texture information. Eigen et al. [10, 135] approached depth estimation as a regression problem using Convolutional Neural Networks (CNNs), applying a local fine-scale network to refine the output of a global coarse-scale network for depth estimation. Wang et al. [136] explored the relationship between depth and semantic information, finding that objects with similar semantic labels tend to have similar depth values. However, these methods are limited by changes in lighting conditions. Other techniques [137] use sparse depth maps alongside guided RGB images. Depth completion, especially when based on deep learning, requires a large amount of offline training data, which remains a challenge due to the sparse measurements and pixel-level annotations available from

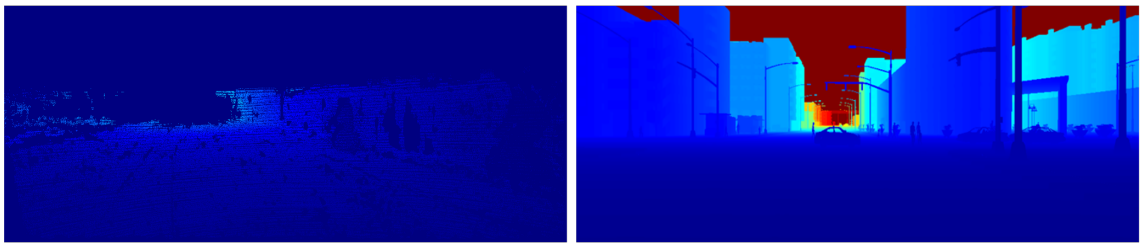

Figure 5.1 A set of depth annotations comparison of KITTI dataset (left) and the dataset collected in this chapter (right). Comparing to KITTI, the collected data contains dense depth information of the whole scene. Colours are used to describe the object distance to the camera. Blue indicates close objects and red indicates far objects. [Wang et al.]

modern LiDAR sensors. For example, the widely-used KITTI dataset [52] provides only 5.9% density point clouds captured by the Velodyne-HDL64, as shown in Fig. 5.1.

In this work, an end-to-end, model-free, learning-based framework is proposed to learn an optimal driving policy in urban environments. The approach is built on a maximum entropy reinforcement learning algorithm, leveraging imitation learning to stable the eventual reinforcement learning. Furthermore, a reconstruction-constrained, deep-fusion depth completion model is introduced, utilising pre-processed synthetic dataset for reliable 3D-aware perception. Since existing open datasets lack dense depth ground truth, a novel dataset is collected using the Carla simulator [49], which offers dense and highly accurate ground truth data, as shown in Fig. 5.1. This new dataset addresses the limitations of existing datasets, and features urban scenarios close to the complexity of real-world environments, making it suitable for further research in this field. The depth completion model is built upon ResNet-34 [28] as the baseline, utilising stereo images and sparse depth data as inputs. A deep fusion technique is employed to integrate different modalities for precise depth prediction. The contributions of this chapter are summarised as follows:

1. A guidance model combining Imitation Learning (IL) and Deep Reinforcement Learning (DRL) for autonomous driving. This approach accelerates the training process and delivers superior results, outperforming state-of-the-art methods.

2. A novel update strategy that enables a smooth transition from the IL phase to the DRL phase. This strategy not only trains the policy network during the IL phase but also updates the value function and Q-function. Experiments demonstrate that this method facilitates seamless transitioning from demonstration to reinforcement learning.

3. A novel dataset containing dense depth information has been collected. The raw data comprises stereo images and LiDAR signals, and an assembly method is employed to fuse all the raw data into dense depth maps.

4. A depth completion model is developed, accurately estimating depth maps using the pre-processed dataset. The model integrates effectively with the driving agent, enhancing overall performance.

5.2 Related Works

5.2.1 Deep Reinforcement Learning with Imitation Learning

Currently, two main learning-based approaches are widely used in autonomous driving: imitation learning and reinforcement learning. Imitation learning enables a driving system to learn an optimal policy by mimicking human expert driving behaviour through supervised training. The underlying idea is straightforward—by analysing expert demonstrations, the system can approximate human decision-making processes, allowing it to handle driving tasks. However, imitation learning comes with several limitations. The most significant is the need for vast amounts of high-quality expert data, which is both time-consuming and costly to collect. Even with a large dataset, imitation learning may still fail to generalise to all possible real-world scenarios that an autonomous vehicle may encounter. Unseen hazards, such as unexpected obstacles or rare events not included in the training data, could be catastrophic for a vehicle relying solely on imitation learning. Furthermore, in complex planning tasks, errors made early in the decision-making process can propagate and compound, a phenomenon known as compounding errors [138]. These errors can reduce the

effectiveness of the model, especially when it is required to handle long sequences of actions, such as navigating through busy urban environments.

On the other hand, deep reinforcement learning (DRL) has demonstrated promising results in various fields, such as robotics and gaming, where agents can learn through trial and error by interacting with their environment. In DRL, the agent continuously improves its policy based on feedback from the environment, learning which actions lead to better outcomes. However, applying DRL to autonomous driving has proven challenging due to the complexity of real-world driving environments and the requirement for safety-critical decision-making. DRL models often require extensive exploration, which is not always feasible or safe in autonomous driving. Additionally, DRL convergence properties tend to be brittle, leading to slow or unstable learning, especially in scenarios where the agent faces degraded or noisy observations. These issues make DRL alone insufficient for handling the intricacies of autonomous driving. To address these limitations, there has been growing interest in hybrid approaches that combine reinforcement learning with imitation learning to capitalise on the strengths of both methods. For example, Hester et al. [139] proposed Deep Q-learning from Demonstration (DQfD), a technique that accelerates the learning process by using a small set of expert demonstrations. This method enhances the efficiency of the standard Deep Q-Network (DQN) approach by pre-training the model on demonstration data, which allows it to learn useful driving strategies before interacting with the environment. Similarly, Vecerik et al. [140] explored a hybrid approach that combines expert demonstrations with online transitions sampled in a replay buffer. This method, using the Deep Deterministic Policy Gradient (DDPG) algorithm, enables the agent to learn from both human demonstrations and its own experiences, thereby improving learning stability and performance. Liang et al. [141] further refined this approach by proposing controllable imitative reinforcement learning, which integrates a command gating function to reduce meaningless exploration, allowing the agent to focus on relevant tasks while still benefiting from reinforcement learning exploration capabilities.

In summary, combining reinforcement learning with imitation learning offers a more efficient and robust approach to training autonomous driving systems. By leveraging human expertise while still allowing the model to explore and learn from its own experiences, these hybrid approaches can overcome the limitations of each method individually. This work builds on this idea, adopting a maximum entropy reinforcement learning framework to further improve the stability and performance of autonomous driving models, ensuring they can handle the complex urban driving scenarios.

5.2.2 Depth Completion

Sparse LiDAR data, while valuable for depth estimation, lacks the density required for detailed scene understanding, necessitating the development of methods that can fill in the gaps, or "complete" the depth information. Uhrig et al. [142] introduced a sparsity variant CNN to tackle this issue, utilising a binary mask operation and input normalisation. This approach ensures the network maintains consistent performance despite varying levels of depth sparsity, making it a foundational technique in depth completion. Many subsequent methods improved depth completion by using RGB images as additional guidance, leveraging the rich, dense visual information they provide. Schneider et al. [143], for example, use guided RGB images alongside pixel-wise structured edge detection and semantic scene annotation to sharpen object boundaries in the depth maps. While this method enhances the depth completion process, it highlights a key challenge: RGB images and sparse depth data are inherently different. RGB images are dense and structured, providing continuous pixel information, while point cloud data from LiDAR sensors is sparse and unordered. These differences make feature fusion difficult to achieve effectively at the input or feature level.

Typically, there are three main fusion strategies employed in the depth completion field: early fusion, deep fusion, and late fusion. Early fusion combines the data before it is input into the network, but this can be problematic because it does not fully address the distinct characteristics of each data modality. Deep fusion, on the

other hand, processes each data stream independently, extracting features separately with specialised feature extractors before merging them at a multi-scale semantic level. This allows the model to handle the unique properties of each modality more effectively. Both early and deep fusion are considered feature-level fusion techniques, focusing on integrating features from multiple inputs within the network. Mohan et al. [144] introduced a deep fusion method in which feature maps are extracted from multiple inputs during the encoding phase and concatenated at various stages of the network. This multi-stage fusion allows the model to incrementally integrate information from the different modalities, making it more flexible in processing both RGB and sparse depth data. Late fusion, in contrast, is a decision-level approach where predictions from each modality are generated independently and then combined at the final decision stage. This method often yields better results compared to early fusion, as it allows each modality to contribute without being forced into alignment at earlier stages of processing. Studies like [145] have shown that late fusion frequently outperforms early fusion, particularly in tasks that require precise depth estimation. In this chapter, a hybrid fusion approach that combines elements of both early and deep fusion techniques is implemented. During the pre-processing of the collected dataset, early fusion is employed by performing pixel-level combinations, effectively creating dense input data that merges RGB and sparse depth information at an initial stage. This allows exploiting the rich visual information in the RGB images while still making use of the depth data from LiDAR. Deep fusion is then applied during the feature extraction process, where each data stream is processed independently to capture the unique properties of both RGB and sparse depth inputs. By fusing the feature maps at multiple stages within the framework, the strengths of both data modalities are effectively integrated, leading to improved depth prediction accuracy. Comparing to state-of-the-art methods, the proposed model demonstrates competitive performance.

5.3 Methodology

In this section, a guidance network based on Soft Actor-Critic (SAC) with Demonstrations is proposed for urban driving scenarios, leveraging both imitation learning and deep reinforcement learning. The framework is considered as a continuous decision-making problem which follows the Markov Decision Process (MDP) [35].

The basic MDP consists of a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. Specifically, the driving agent interacts with the environment by receiving a successive state $s_i \in \mathcal{S}$ and taking a corresponding action $a_i \in \mathcal{A}$ based on the current policy. This leads to the next state $s_{i+1} \in \mathcal{S}$ with the transition probability $p_i = \mathcal{P}(s_{i+1}|s_i, a_i)$ and a reward $r_i \in \mathcal{R}$. The goal of deep reinforcement learning is to find an optimal policy that maximises the expected long-term return, represented by the Q-function.

5.3.1 Soft Actor-Critic

The proposed method builds upon an off-policy model-free DRL algorithm, Off-policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor (SAC). Model-free DRL algorithms face two main challenges: high sampling complexity and fragile convergence, which heavily depends on hyper-parameter tuning, limiting the broader application of DRL to real-world scenarios. SAC addresses these issues by introducing three key elements: an actor-critic structure with separate policy and value networks, a soft off-policy update based on experience, and the introduction of the maximum entropy to ensure stability and encourage exploration. The standard maximum expected reward is given by:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \sum_{t=0}^{T} [r(s_t, a_t)] \tag{5.1}$$

For SAC, the objective is to maximise both the expected reward and the entropy of the policy for better exploration, defined as:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \sum_{t=0}^{T} [r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))] \tag{5.2}$$

Where the temperature parameter α is a coefficient that controls the weight of the entropy term in the objective function in the SAC algorithm. The role of this parameter is to balance the trade-off between exploration and exploitation. s_t and a_t represent the state and action at time t respectively. To achieve the best performance possible, the entropy should vary in different training stages. For example, when the optimal action is clear, the entropy should be smaller; whereas during the exploration stage, the entropy should be larger. SAC proposes an automated entropy adjustment for the temperature parameter through training by solving:

$$\alpha_t^* = arg\min_{a_t \sim \pi_t^*} \left[-\alpha_t log \pi_t^*(a_t|s_t;\alpha_t) - \alpha_t \overline{\mathcal{H}} \right] \tag{5.3}$$

Where $\overline{\mathcal{H}}$ is the minimum expected entropy.

SAC follows a actor-critic architecture, considers a policy network $\pi(a|s)$ as the actor network, two soft Q-networks $Q_\phi^1(s,a)$ and $Q_\phi^2(s,a)$, a value network $V_\psi(s)$ and a target value network $V_{\bar{\psi}}$. The value function is used to stabilise the training process, which is updated by:

$$J_V(\psi) = \mathbb{E}_{s_t \sim D} \left[\frac{1}{2} \left(V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi} \left[min \left(Q_\phi^1(s_t,a_t), Q_\phi^2(s_t,a_t) \right) - log \pi(a_t|s_t) \right] \right)^2 \right] \tag{5.4}$$

Where D is the replay buffer which contains a certain amount of experiences. This objective function ensures the estimated long-term rewards in specific state under the policy is close to estimated long-term rewards in specific state and taking the most likely action of the action distribution under the policy, with an entropy term. For latter, the minimum estimation of the two Q-networks is chosen. The agent is encouraged to explore actions that are less likely, thus promoting diversity in the actions chosen.

For the soft Q-function, it can be updated by:

$$J_{Q,i}(\phi) = \mathbb{E}_{(s_t,a_t) \sim D} \left[\frac{1}{2} \left(Q_{\phi,i}(s_t,a_t) - \hat{Q}(s_t,a_t) \right)^2 \right] \tag{5.5}$$

for $i = 1, 2$ with the target Q-value, which consists of the immediate reward and γ discounted future rewards:

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} V_{\bar{\psi}}(s_{t+1}) \tag{5.6}$$

$V_{\bar{\psi}}$ is the target value network. It is updated slower than the value network V_ψ using the Polyak Averaging [146**?**] providing a stable target that reduces the variance in the Q-value estimates:

$$\bar{\psi} \longleftarrow (1 - \tau)\bar{\psi} + \tau\psi \tag{5.7}$$

Finally, the policy network can be updated by approximating the policy gradient by:

$$J(\pi) = \mathbb{E}_{(s_t, a_t) \sim D} \left[min \left(Q_\phi^1(s_t, a_t), Q_\phi^2(s_t, a_t) \right) - \alpha log(\pi(a_t|s_t)) \right] \tag{5.8}$$

5.3.2 SAC with Imitating Learning

The proposed method combines imitation learning and deep reinforcement learning to accelerate the training process and achieve better performance. The framework overview is shown in Fig. 5.2. The SAC-IL training process consists of two parts: the imitation pre-training and the deep reinforcement learning. In DRL methods, the subsequent policy network, value network and Q-network are updated in one gradient step, however, IL methods only focus on updating the policy network, where the driving model makes decisions based on the observations. This poses a dilemma in transferring the IL model to DRL that the policy generates good action while the value network and the Q-network generate highly inaccurate predictions, which undermines the policy network in the latter updating.

Therefore, to ensure smooth transferring, it is necessary to update all the subsequent networks in SAC. Instead of storing state-action pairs, transitions $(s_t, a_t, r_t, s_{t+1}, d)$ is collected in CARLA and stored in the replay buffer as the expert data. The loss

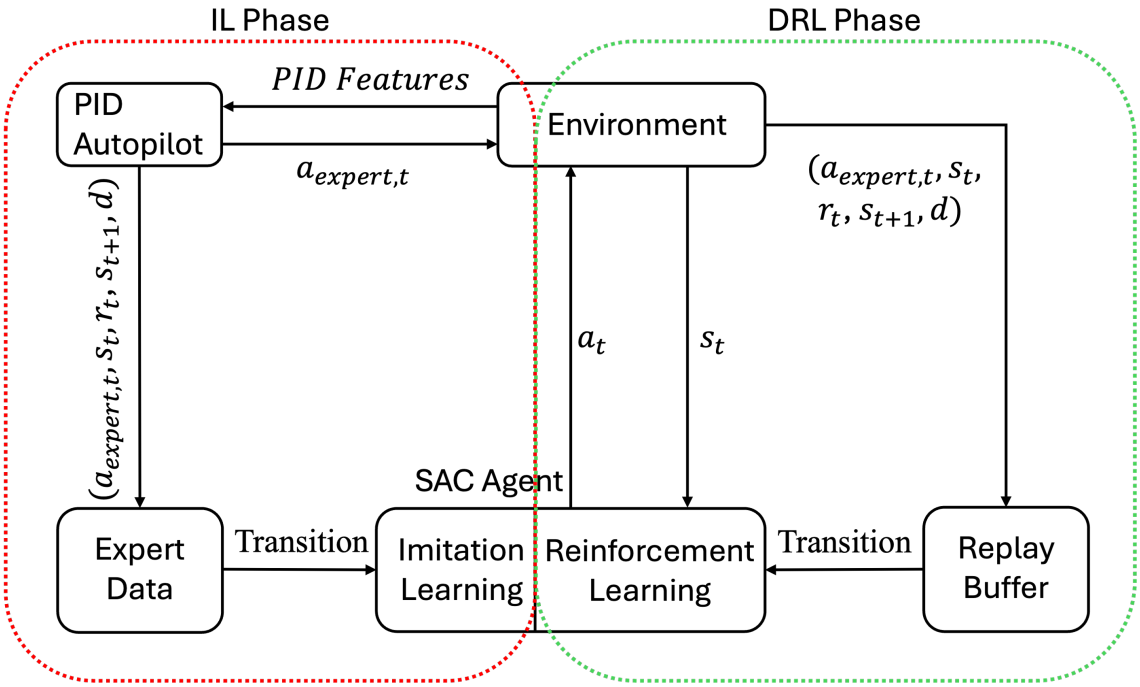

Figure 5.2 The framework overview of the proposed method. The model first finishes the imitation learning on the left side then transfer to deep reinforcement learning on the right side. [Wang et al.]

function for optimising the policy network is defined as:

$$\mathcal{L}_{IL}(\pi) = \sum_{t=0}^{T} \Big(\pi(\cdot|s_t) - a_{expert,t})^2 - \alpha log(\pi(a_t|s_t))\Big) \tag{5.9}$$

where $a_{expert,t}$ is the expert action from the demonstration replay buffer at time t.

To train the value function and Q-function simultaneously with the policy network, we add the weighted L_{IL} to the gradients of Equations (5.4) and Equation (5.5) respectively. After the imitation learning phase, the pre-trained model is then transferred to pure reinforcement learning. With the help of the pre-trained network, the second stage of training performs significantly better at the beginning compared to random exploration, thereby accelerating the entire training process. In contrast, deep reinforcement learning (DRL) training explores a wider range of scenarios and driving strategies, enhancing the agent driving capabilities in ways that the imitation learning (IL) model cannot achieve. This dual approach not only improves the efficiency of training but also allows the agent to adapt to more complex and

dynamic driving environments, ultimately leading to superior performance in real-world applications.

5.3.3 Reward Function

The driving task in this chapter is for the agent vehicle starts from the starting point and navigate autonomously to reach the target point while avoid collisions in urban driving scenarios. To ensure the agent accomplishes the objective while maintaining safety and efficiency, the hand-designed reward function is defined as the combinition of multiple terms:

$$r = r_{speed} + r_{collision} + r_{steer} + r_{success} + r_{distance} + C \tag{5.10}$$

r_{speed} encourages the agent to travel at a desired speed and penalises overspeeding in terms of safety. Specifically, the speed reward is defined as:

$$r_{speed} = \begin{cases} s_{current} & s_{current} < s_{desired} \\ s_{desired} - s_{current} & otherwise \end{cases} \tag{5.11}$$

$r_{steering}$ limits the frequency of steering actions, improves the passenger comfort, and also prevents the vehicle from driving in circles by keeping the same steer in the early stages:

$$r_{steer} = -a_{steer}^2 \tag{5.12}$$

Due to the complex road topology, the agent does not always find the target. Relying solely on the network could lead to poor performance in a new environment. In other word, the agent needs a direction. A relative distance reward is defined to assist in decision-making during crossroad situations.

$$r_{distance} = 0.001 \times (d_{t-1} - d_t) \tag{5.13}$$

where d is the distance between current position and target position.

$r_{collision}$ punishes any detected collision with a penalty of -50, while C represents a constant time penalty for efficiency.

5.3.4 Depth Completion

Depth images offer rich spatial and geometric information, however, they are not directly available in real world. In this chapter, a depth completion network is proposed to provide dense depth observations for the driving agent, using a RGB image and a pre-processed incomplete depth map as input.

Dataset Collection and Pre-processing

The KITTI Dataset is a well-known benchmark in the autonomous driving domain, particularly popular for depth completion tasks in training and evaluation. However, the ground truth depth labels it provides are sparse due to the limitations of the real LiDAR, which limiting the performance of networks. By employing the Carla simulation software, accurate ground truth depth maps can be generated and collected for depth completion training. Using the simulator, real traffic scenarios is replicated, incorporating traffic rules, moving cars, non-motor vehicles, and pedestrians. The sensors are configured to closely resemble the KITTI Dataset, ensuring that the proposed algorithm trained on the synthetic dataset can be transferred to real-world applications. The dataset comprises three types of sensors: a stereo camera, LiDAR for raw input signals and a depth camera for ground truth annotations.

The pre-processing pipeline consists of four steps for one of the input data.

Disparity Acquisition First, the stereo RGB images are converted into disparity maps. To achieve this, Semi-Global Block Matching (SGBM) [48] is employed to compute and aggregate the matching cost between the stereo image pairs. The cost refers to how well two pixels, one from each image, correspond to the same 3D point, and it is computed based on pixel similarity. SGBM evaluates this cost and aggregates it over multiple directions, improving the disparity estimation in areas with weak textures or complex structures. Since the initial disparity map often contains noise and imperfections, particularly in regions with less textures or at

object boundaries, a weighted least squares (WLS) filter is applied to fill in the holes and smooth the edges by propagating reliable disparity values from neighbouring pixels, guided by pixel intensity similarities thus improving the overall quality of the disparity map. This refinement ensures subsequent processing stages for better depth annotations.

Depth Acquisition Next, the enhanced disparity maps are further converted into real depth maps based on the following equation:

$$depth = \frac{f \times baseline}{disparity} \tag{5.14}$$

where f is the focal length and $baseline$ is the distance between the two cameras in the stereo camera setup. The disparity value is inversely proportional to the depth. To ensure the accurate depth of the far background, such as the sky, a depth value of 255 is assigned to any disparity values less than or equal to 3.

LiDAR Projection After that, LiDAR point clouds are projected to the 2D depth image plane using the camera intrinsic K_{in} and the LiDAR-to-Camera transformation matrix T:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K_{in} \cdot T \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{5.15}$$

Where u, v are the 2D pixel coordinate of projected LiDAR points on the image plane and x, y, z are the LiDAR points 3D coordinates.

Combination Finally, a combination strategy is presented for the converted depth maps and LiDAR projections at the pixel level. The detailed steps are shown as followed:

1. Direct Combination: On the final depth image plane, if LiDAR points are projected, the LiDAR values are preferred as the depth values. Otherwise, the converted depth maps are used.

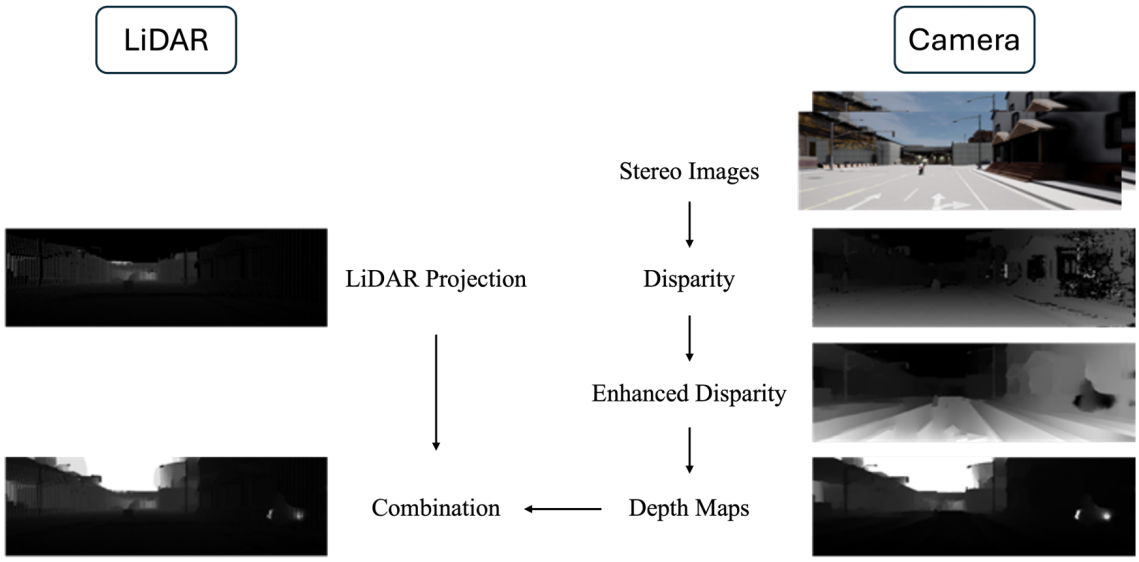

Figure 5.3 The pro-processing pipeline for the collected synthetic dataset. Raw stereo camera images and LiDAR point cloud are processed and combined into a final incomplete depth image. [Wang et al.]

2. Depth Refinement: It is understood that the depth values in the converted depth maps can be inaccurate, however, the relative depth relationship among pixels can be trusted. Firstly, if a projected LiDAR point coincides with a depth map pixel, the depth value of the corresponding depth map pixel will be replaced by the projected LiDAR point value. Secondly, since the relative depth is trustworthy, every pixel in the converted depth map that shares the same value as the coincided point will be replaced with that LiDAR value. This approach enhances the reliability of the depth map pixels in areas where there are no LiDAR points projected. The dataset preparation process is illustrated in Fig. 5.3.

Through the four steps of pre-processing, an enhanced incomplete depth map is created, taking most advantages of the raw data available in the dataset. This improved incomplete depth map will later serve as input for the depth completion network, resulting in accurate dense depth maps for the DRL-IL driving agent.

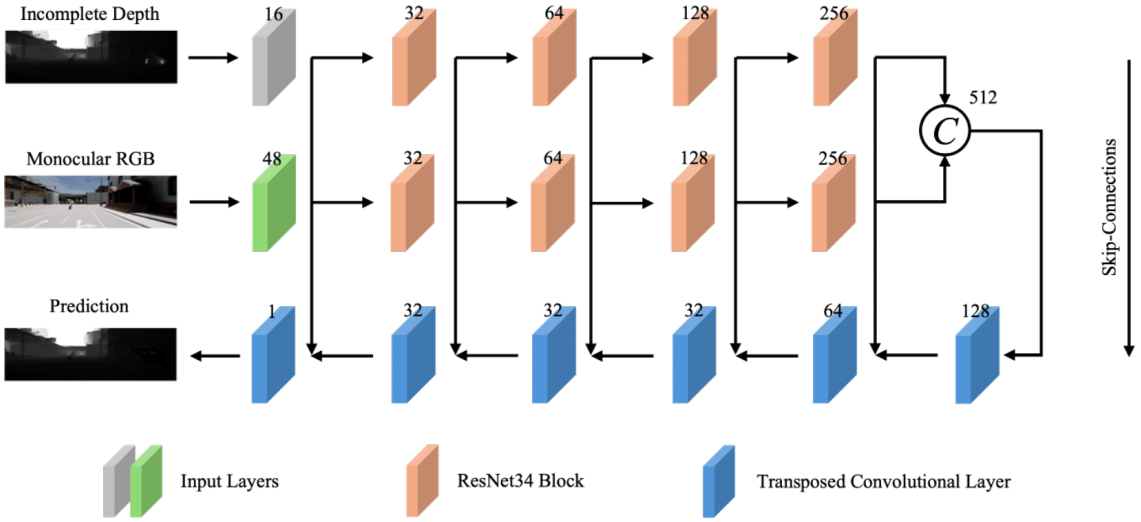

Figure 5.4 The proposed depth completion network. The numbers on the layers indicate the output channel. The network takes an incomplete depth map and a monocular RGB image, and late fusion is applied to fuse the different modalities. Skip-connections are used to enhance the feature representations. [Wang et al.]

Depth Completion Network Architecture

The proposed model consists of an encoder-decoder structure shown in Fig. 5.4. The encoder consists of a parallel series of layers with increasing filters for down-sampling the input images. Each input layer starts with an initial convolutional layer, followed by four blocks of ResNet-34. The RGB layer outputs a tensor of 48 channels as the RGB has three channels in contrast to one channel in incomplete depth. One branch is dedicated to the pre-processed incomplete depth maps, and the other processes RGB images. Then a deep fusion structure [147] is employed to concatenate the features after the ResNet-34 blocks for the later decoder.

The decoder has a reverse architecture to the encoder, up-sampling the image back to its original size with one channel for dense depth predictions. Outputs from each encoding layer are passed to their corresponding decoding layers through skip-connections, concatenating with the output from the previous decoding layer into one tensor as input for the next transposed convolutional layer. The skip-connections [148] effectively mitigate gradient vanishing and exploding problems in deep networks. All convolutional layers have a kernel size of 3, a stride of 2, and padding of 1 to ensure 1/2 down-sampling and 2× up-sampling. The model accepts inputs of any

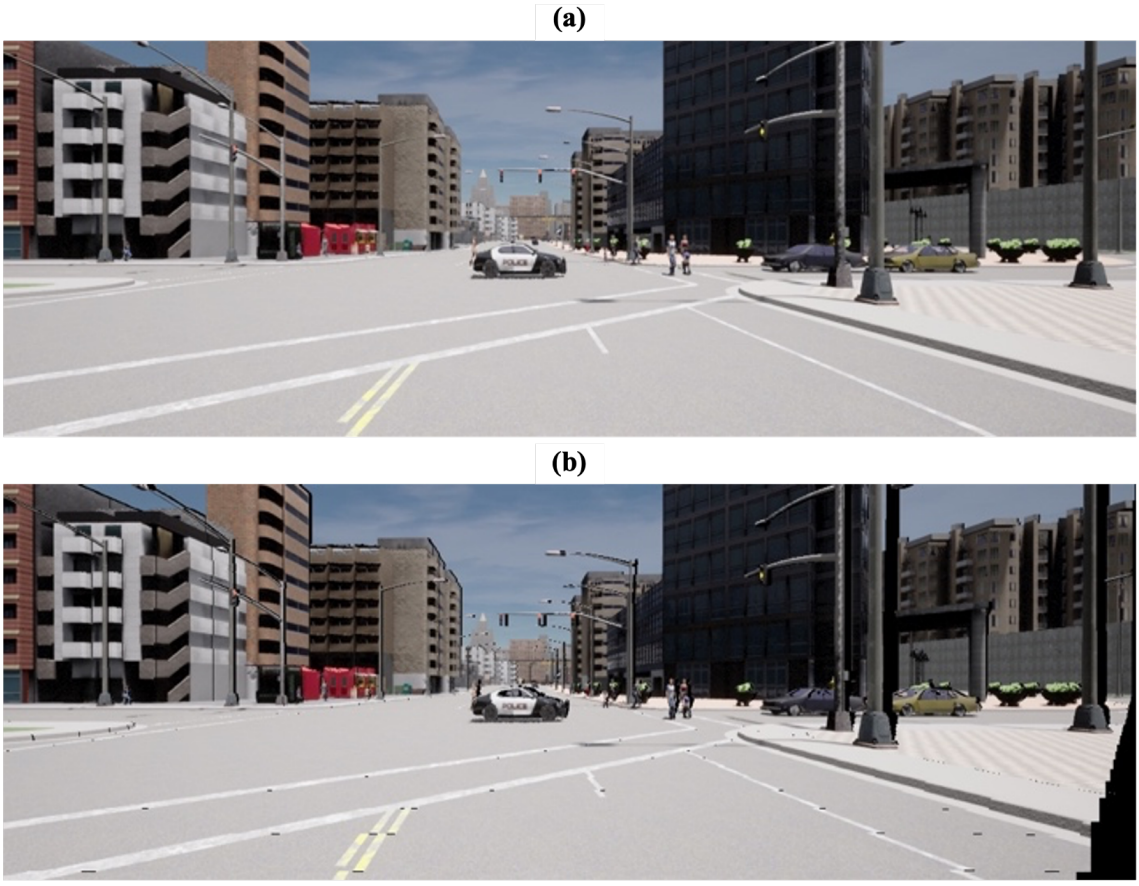

Figure 5.5 Image reconstruction using stereo images involves utilizing a pair of stereo images to generate a new image based on depth information. Each pixel in one image (a) can be mapped to a corresponding pixel in the other image, allowing for the reconstruction of an alternate view (b). [Wang et al.]

size and generates the predicted depth maps with the same resolution of inputs.

Image Reconstruction Restraints

Most existing work relies on training with the supervision of dense depth ground truth. However, with stereo images available, incorporate self-supervised signals in the traditional supervised learning has the potential of improving the network performance. This leverages the inherent relationships between stereo images to generate additional training signals, thus reducing the dependence on dense ground truth. To do this, it is essential to understand the 3D geometry of the environment, which is obtained from the depth maps from the model predictions. For every pixel located at (x, y_1) in the left image, there is a corresponding pixel (x, y_2) in the right

image:

$$L(x, y_1) = R(x, y_2) \tag{5.16}$$

Given a left image L and its associated depth map D predicted by the network, the left image L can be reconstructed or warped into the right image R following the equation:

$$y_2 = y_1 - \frac{f \times baseline}{1000 \times D(x, y_1)} \tag{5.17}$$

Where f denotes the camera focal length and *baseline* is the distance between the two cameras in the stereo camera setup. $D(x, y)$ indicates the depth value of the pixel. Then the warped image is obtained $L_{warped}(x, y_2)$ as demonstrated in Fig. 5.5. Even with ground truth depth, the warped image may contain vacancies, as some pixels from one image may not correspond to any pixels in the other image due to differences in view angles. The accuracy of depth predictions directly influences the quality of the reconstructed image; the better the model predicts the depth, the more accurately it can reconstruct the image. By comparing the warped left image with the actual right image, additional self-supervision signals can be integrated to enhance the depth predictions indirectly.

Loss Function

The loss function consists of three terms. First term penalises the difference between the model predictions and the dense depth annotations, for achieving a sufficiently accurate pixel-wise mapping.

$$\mathcal{L}_{depth} = \frac{1}{N} \sum_{i=1}^{N} (D_{annotaion,i} - D_{pred,i})^2 \tag{5.18}$$

Second, to encourage smoother predictions, the mean error of the second-order derivatives of the predicted depth along both the x and y axes is penalised. This smoothness loss, $\mathcal{L}_{smooth}$, aims to minimise abrupt changes in depth values by

quantifying the curvature of the depth map. It is formulated as follows:

$$\mathcal{L}_{smooth} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{d^2\nabla D_{pred,i}}{d^2x} + \frac{d^2\nabla D_{pred,i}}{d^2y}\right| \tag{5.19}$$

By penalising high-frequency variations, the model is encouraged to produce depth maps with smoother transitions, enhancing overall prediction quality. Third, the image reconstruction constraints are utilised to minimise the difference between the actual right image and the warped left image therefore improving the network performance indirectly:

$$\mathcal{L}_{recon} = \frac{1}{N}\sum_{i=1}^{N}(R_i - L_{warped,i}) \tag{5.20}$$

As mentioned before, due to the different viewpoints, there will be holes in the warped image. Therefore, the loss is calculated only where the pixel values are greater than 0 in the warped images. To summarise, the entire loss function for the model is:

$$\mathcal{L} = \mathcal{L}_{depth} + a \times \mathcal{L}_{smooth} + b \times \mathcal{L}_{recon} \tag{5.21}$$

Where a and b are the hand-designed hyper-parameters for the second and third terms.

5.4 Experiments

5.4.1 Implementation Details

In implementation, the Town 02 map in the CARLA simulator is used as the training environment, as shown in Fig. 5.6. To set up the environment as realistically as possible, 50 vehicles and 30 pedestrians are randomly spawned and controlled by the autopilot controller. Stereo cameras and a LiDAR are attached to the agent vehicle for both data collections and training.

Figure 5.6 A shot of the training environment CARLA Town 02. Comparing to other maps in CARLA, Town 02 has narrow driving roads and frequent intersections, which is difficult for the agent to navigate around. [Wang et al.]

5.4.2 Dataset Collection

Expert Driving Data

The expert data is collected by driving around in the environment using the CARLA PID controller, which works as a fully observed algorithm. It uses all the information collected by the Motion Planner Stage to generate throttle, brake, and steering commands. The process consists of three stages:

1. Localization – This obtains the position and velocity of all vehicles and creates future waypoints based on the previous trajectory.
2. Hazard Detection – Collision and traffic light detection occurs in this stage.
3. Motion Planner – This estimates actions based on information from the previous stages.

During experiments, the expert agent occasionally crashes due to "car accidents" caused by surrounding vehicles which also proves the limit of PID controller but the collision data could also be helpful for the later reinforcement learning as introducing the hazard scenarios. The ego agent is spawned randomly in the environment and 50 episodes of demonstration data are collected with a success rate of 85%.

In the reinforcement learning phase, the environmental settings remain the same with those used during the demonstrations. The pre-training SAC with Imitation Learning model (SIL) uses a depth image of size 256×128 generated by the proposed depth completion model Reconstruction Constrained Deep Fusion Network (RDFNet) as visual input. This depth image is concatenated with additional matrix inputs including the current speed and the relative distance between the target and current position as the final input for the policy network, the value network and the Q-network in SIL. The target point is fixed, and in each episode, the ego car is spawned randomly in the map. The policy network of SIL outputs an action consisting of throttle, brake, and steering. Additionally, an early-stop mechanism is implemented to terminate an episode when the agent moves into unreasonable space, such as grass and areas deep in the buildings, effectively preventing meaningless exploration during training.

Depth Completion Data

For the depth completion task, a total of 8328 frames of data are collected, including stereo images, LiDAR pointcloud, and the ground truth depth, with a resolution of 256×128 by driving around in Town 02 and Town 03. Among them, 2854 frames are used as the validation set, resulting 5474 frames for training. The detailed sensor configurations are listed below:

1. LiDAR – (a) Laser Range: A maximum range of 120 metres; (b) Vertical Field of View: from -24° to +2°; (c) Horizontal Field of View: 360°; (d) Scan Frequency: 10 Hz.

2. Stereo Camera - (a) Baseline: 0.54 metres; (b) Resolution: 256×128; (c) Field of View: 90°

In the experiment of depth completion task, the model RDFNet is trained with collected dataset on a 12 GB Nvidia RTX 3060 graphic card for over 30 epochs. Parameters a and b in Equation (5.20) are treated as hyper-parameters. Additionally, two variants of RDFNet are tested to show the efficiency of the proposed method.

Scenario	Input (Depth)	Drive Straight	Turn Making	Navigation	Full Navigation
Town 02	Ground Truth	1.0	0.96	0.74	0.68
(Trianing Environment)	Predicted	1.0	0.94	0.70	0.68
Town 03	Ground Truth	1.0	1.0	0.94	0.88
	Predicted	1.0	0.96	0.90	0.86
Town 04	Ground Truth	1.0	0.94	0.92	0.88
	Predicted	1.0	0.96	0.84	0.80
Town 05	Ground Truth	1.0	1.0	0.76	0.68
	Predicted	1.0	0.96	0.72	0.66
CIRL Town 02 [141]		1.0	1.0	0.53	0.41

Table 5.1 Task success rate in various environments and categories. Both using ground truth depth and predicted depth as inputs are considered. [Wang et al.]

First one is using early fusion instead of deep fusion, the other variant is trained without reconstruction loss. Root mean squared error (RMSE) is employed as the standard metric for evaluation benchmark.

5.4.3 Results

Results of Driving Agent

The driving model SIL is tested in three different unseen maps (Town 03, Town 04, Town 05) and the training environment (Town 02) to evaluate its performance generalisation capability. To provide a meaningful comparison with other works, three additional tasks are included: straight driving, making turns, and navigation without surrounding cars (Navigation), in addition to the original task of navigating with surrounding cars (Full Navigation) to a target point for evaluation. A total of 50 test runs are performed in each scenario. The success percentages for these four tasks are shown in Table 5.1, with higher percentages indicating better performance. The results are compared with the state-of-the-art method, Controllable Imitative Reinforcement Learning for Vision-based Self-Driving (CIRL), as the tasks are similar and conducted in the same environment.

As illustrated in Table 5.1, the driving model SIL demonstrates good performance levels across different urban environments. When provided with ground truth depth input, SIL effectively managed sub-tasks such as straight driving and making turns, achieving results that are comparable to those of CIRL. In the more challenging navigation task that require a series of decision making, SIL continues to yield

competitive results. The model scores over 70% in these scenarios, with some instances reaching as high as 94%. Such performance highlights the effectiveness of the learning algorithm in dynamic environments that demand continuous decision making. In the full navigation task, the most challenging and realistic task including surrounding vehicles and pedestrians, SIL still performs admirably. Results range between 66% and 88%, reflecting the model capability to handle complex interactions in a busy urban setting. In contrast, the performance of CIRL shows a sharp decline when transitioning from making turns to executing navigation tasks, with scores dropping from 71% to 53% and further down to 41%. This significant drop illustrates the challenges faced by CIRL in maintaining effective driving strategies under more complex conditions.

However, a closer look of the results reveals noticeable performance difference between Town 03 and Town 04 compared to Town 02 and Town 05. This discrepancy can be attributed to the broader roads found in Town 03 and Town 04, which provide a more forgiving environment for minor errors and leaving room allowing the ego vehicle to correct any deviations from the intended driving trajectories, thus enhancing overall success rates. At the same time, these results also prove the robust generalisation capability of the SIL model, indicating its ability to adapt to unseen scenarios while maintaining effective driving strategies.

While the performance of the model using the predicted depth is slightly lower than using the ground truth inputs, it still outshines the results produced by CIRL in each tested category. This comparison enable the possibilities of building on-board perception module with the DRL driving model for the future researches.

Results of Depth Completion

Evaluations are also conducted for the depth completion model RDFNet independently. First before training, the incomplete depth maps pre-processed from the collected data are tested, with the RMSE already impressive at 38.34. To compare with state-of-the-art techniques, several methods are selected and evaluated on the collected synthetic datasets. The depth range is capped at 70 metres, any objects

Method	CAP	RMSE	MAE
Sparse-to-dense (gt) [149]	70	17.45	5.02
Revisiting Sparsity [150]	70	19.76	5.98
Sparsity Invariant CNNs [142]	70	34.37	9.91
Early Fusion	70	5.67	2.27
Without Warping	70	8.99	2.82
RDFNet	70	**2.66**	**0.77**

Table 5.2 Performance comparison with state-of-the-art methods on RMSE and MAE. The depth range is capped at 70 metres, which are the practical range of real-world applications. [Wang et al.]

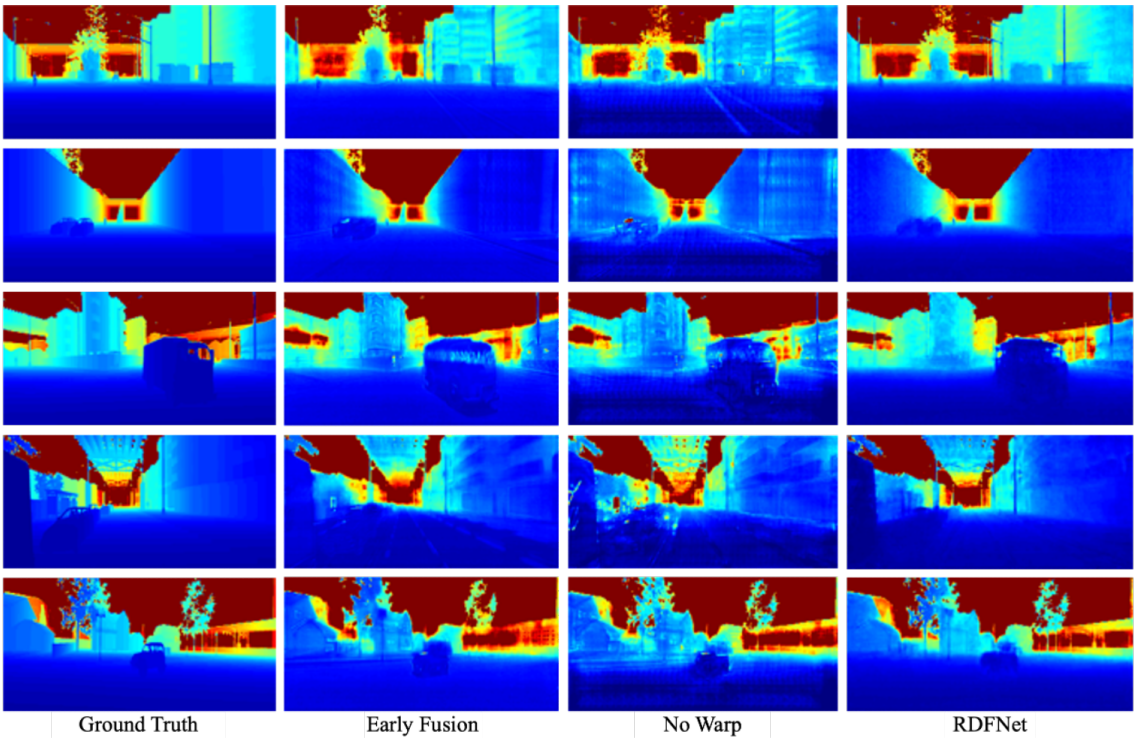

Figure 5.7 Visualisation of the ground truth and the predicted depth maps. [Wang et al.]

further that 70 metres are set at depth value 70 for a more realistic application, which is common in this field. As shown in Table 5.2, comparing to other methods, RDFNet demonstrates better performance with an RMSE of 2.66 and a Mean Absolute Error (MAE) of 0.77.

In the ablation study, the Early Fusion variant achieves an RMSE of 5.67, indicating that the deep fusion technique contributes to a reduction of approximately 3.01 in RMSE. The other variant, Without Warping, only reaches an RMSE of 8.99,

highlighting the effectiveness of the reconstruction constraint loss proposed in this chapter, which brings a 6.33 improvement in RMSE.

In addition to the quantitative results, Fig. 5.7 presents five example scenes that offer visual comparisons between the ground truth depth maps and the predicted depth maps generated by RDFNet and two other variants. The depth maps are colourised to enhance visual differentiation, making it easier to assess the accuracy of the predictions.

5.5 Summary

In summary, the proposed method for guidance agent, SAC with Imitation Learning, demonstrates a significant improvement in performance over previous approach in the realm of urban autonomous driving. By integrating deep reinforcement learning with imitation learning, this method effectively leverages the strengths of both algorithms. The innovative updating strategy employed allows the policy network to make more efficient transfer from using expert data during the imitation learning phase, accelerating and enhancing the overall learning process. This leads to a more robust and adaptive agent capable of navigating complex driving scenarios. Furthermore, the proposed depth completion model effectively addresses critical perception challenges encountered in autonomous driving. By transforming available raw data into more dense depth maps, the depth model receives better inputs.

Chapter 6

Single-Agent Robustness Against Perturbations

In this chapter, the robustness of a single deep reinforcement learning (DRL) agent is investigated against state adversarial attacks. The adversarial attacks work as a directional interference to mislead and determine the performance of the trained DRL agent, which can be considered as cyber-attacks or real-world uncertainties. This study proposes a defence algorithm to mitigate the state perturbations, ensuring the concrete robustness of the driving model in worst-case situations. Furthermore, an explainable attack detector is introduced to accurately predict the presence of adversarial attacks and provide an explainable visualisation of the decision-making process, further enhancing the reliability of the proposed robust algorithm.

Associated Publications This chapter is based on the following published work:

Wang C, Aouf N. Explainable Deep Adversarial Reinforcement Learning Approach for Robust Autonomous Driving[J]. IEEE Transactions on Intelligent Vehicles, 2024.

Deep Reinforcement Learning (DRL) has made promising progress in autonomous driving planning and guidance within dynamic urban scenarios. As the potentials for real-world applications increase, so does the demand for a safe and robust driving system. For example, an end-to-end deep reinforcement learning agent makes guidance decisions based on the observation states extracted from perception sensor inputs, which can be interfered by unpredictable adversarial attacks. The latter causes adversarial observation states, which easily leads autonomous driving agent to incorrect decisions and ultimately to unintended accidents. In this chapter, both attack and defence approaches are proposed for robust learning-based self-driving agent. The optimal observation perturbation is realised using an efficient augmented gradient-based method. An attack detection deep network with saliency map based explainability is then proposed to flag up to the users the existing danger of the attacks on the sensor perception. Furthermore, to ensure safe driving under these perceptional perturbations, a deep adversarial reinforcement learning based approach is proposed for robust autonomous driving in roundabout passing scenes. PPO (Policy Proximal Optimisation) [20] is adopted as the baseline guidance algorithm and a theoretical supported constraint, multi-object objective function optimisation is developed to efficiently mitigate the effect on the deep guidance autonomous driving policy from strong adversarial attacks. Extensive experiments and evaluation are conducted on the proposed robust model under various adversarial attack configurations in traffic scenarios. The method shows significant improvements coping with optimal adversary in dynamic environments.

6.1 Overview

In recent years, autonomous driving has attracted major research interests because of its capacity to bring the prospect of greater convenience, mobility efficiency and safety to the automotive industry. Researchers have made significant progress in different applications in autonomous driving field, such as scene understanding (perception), localisation, mapping and path planning. In particular, state-of-the-art

deep reinforcement learning based algorithms are able to handle the challenging decision making (guidance) tasks in dynamic environments [134, 151–153].

Despite of existing successes, these simulation based methods work well under ideal experimental conditions, in which the sensors are all time properly functioning. The observations delivered by these sensors may inherently contain uncertainties due to various factors such as unavoidable sensor errors or natural equipment inaccuracies, and adversarial attacks. Among them, adversarial attacks are the strongest directional interference. A policy of a DRL agent that is not robust to such perturbations can encounter catastrophic failures in more realistic environments. In light of these risks, autonomous vehicles are required to ensure that their decision making system can handle the adversarial perturbations from perceptual sensing module. Having stated the importance of such problem, only few researches investigated deeply the aforementioned challenge. Some works [154–156] adopt adversarial training from supervised learning scheme to improve the robustness of such deep based guidance schemes. Specifically, the agent is occasionally attacked and the adversarial trajectories are generated during the data collection. Then some existing DRL algorithms are trained with the replay buffer containing adversarial transitions generated before. However, for most environments, naive adversarial training leads to unstable training and deteriorates DRL agent performance [157, 158] or does not significantly improve their robustness under strong attacks.

To address this problem, this chapter introduces an explainable deep adversarial reinforcement learning approach for autonomous vehicles to improve the performance and robustness of driving policies against the observation adversarial attacks. The key contributions can be summarized as follows:

1. A Markov Decision Process (MDP) with perturbations is established to model the decision making behaviors of an autonomous vehicle under policy constraints and observation perturbations for roundabout scenarios.

2. A white-box optimal attack method based on FGSM is proposed to approximate the worst observation perturbations due to sensor attacks. In the testings,

this attack technique can efficiently lead the non-robust deep reinforcement learning agent to failures.

3. A robust proximal policy optimisation algorithm is proposed to obtain a robust policy for roundabout making under bounded strong observation perturbations.

4. Furthermore, an explainable adversarial attack detection network is presented to provide an insightful visualisation on how the DRL agent is making the decision based on the sensor inputs and how is the adversarial attack influencing on that, as well as additional information to alarm the users of the system whether the guidance agent is under attack or not.

Extensive experiments are carried out in the realistic unreal-engine powered simulator CARLA [49] with dynamic surrounding traffic and various attack strengths to evaluate the performance of the proposed methods. The proposed optimal adversaries show strong effects on the baseline DRL agent. This results further prove the improvement of performance and robustness of the adversarial defence method. Synthetic data is collected to train and evaluate the detection network, which shows highly reliable detection accuracy.

6.2 Related Works

6.2.1 Deep reinforcement learning based autonomous driving

Following the major breakthroughs of deep reinforcement learning (DRL) in recent years [159, 36], researchers have started to use DRL to address the decision-making problems in autonomous driving [160, 151, 161]. DRL based methods can greatly decrease the heavy reliance on the large amount of data because they do not specifically need large labeled driving data for training comparing to supervised learning scheme adopted methods such as behavior cloning and imitation learning. Alternatively, they learn and enhance their scene understanding and decision-making capability via interactions with the environment. Besides, human pre-collected data could be biased to cover the hazard and failure situations where DRL based

methods can be used in the collision or close-collision scenarios, where self-driving vehicles learn to deal with dangerous scenarios in the simulation. Dong Li et al [162] solves the lateral control for autonomous driving by introducing an individual perception module for the track features prediction and the DRL control module for the control action prediction. They claim that their framework outperforms the conventional linear quadratic regulator (LQR) controller and model predictive control (MPC) controller on different tracks. [163] introduces a velocity control model for autonomous driving to focus on delivering safe, efficient, and comfortable experience. By analysing human driving data and combining driving features related to safety, efficiency, and comfort, this method develops a reward function encouraging the driving agent to maintain stable while avoiding obstacles. To further exploit the human prior knowledge, [164] combines deep reinforcement learning and human expert demonstrates based imitation learning with uncertainty estimation. Then the DRL agent learns by regularising the KL divergence between the DRL agent's policy and the imitative expert policy.

Unlike supervised learning methods, DRL approaches can mitigate the high cost of data collection in hazardous scenarios by training models within virtual simulation environments, allowing for cost-effective failure occurring learning. However, as they running in a perfect conditioning simulation environment, many DRL based decision-making methods do not take observation uncertainty due to sensor failure or attacks into consideration, leading to unstable and less robust driving strategies that could make the vehicle unsafe under more realistic environment settings.

6.2.2 Adversarial attack on DRL

Though deep learning models recently achieve significant improvement, research shows that these well-trained models are still very vulnerable to adversarial attacks. Adversarial attacks on camera sensor often lead to visually similar images to the normal images from a human perspective, yet they can deceive deep learning models into generating inaccurate predictions. Generally there are two types of adversarial

attack methods, white-box attacks and black-box attacks, depending on if the attacker has full access to the models' parameters or not.

In [165], a Bayes optimisation based approach was proposed to generate the painting of black lines on the road to counterfeit lane lines and make the vehicle deviate from the original orientation. Experiments were conducted in CARLA simulator, and results showed that end-to-end driving models were attacked and deviated to the orientation chosen by attackers. He et al. [166] combined Bayesian optimisation and Jensen-Shannon (JS) divergence to measure average variation distance of the policies attacked by the observation perturbations for optimal black-box attacks. Behzadan and Munir [167] studied black-box attacks on DQNs with discrete actions via transferability of adversarial examples. Pattanaik et al. [168] further enhanced adversarial attacks to DRL with multi-step gradient descent and better engineered loss function. They required a critic or Q function to perform attacks. Typically, the critic network learned during agent training. For white-box approaches, Huang et al. [169] evaluated the robustness of deep reinforcement learning policies through an FGSM based attack on Atari games with discrete actions. Kos and Song [170] proposed to use the value function to guide adversarial perturbation search. Lin et al. [171] considered a more complicated case where the adversary is allowed to attack only a subset of time steps, and used a generative model to generate attack plans luring the agent to a designated target state. In this chapter, a FGSM based method is introduced to generate optimal adversary examples by maximising the pre-defined collision risk. Results show that with a small strength parameter ε and minimal visual difference, our method can efficiently misguide the well-trained agent.

6.2.3 Explainability for Deep Learning

Despite the success in deep learning, its explainability is limited due to its predominantly black-box nature. Researchers have been investigating ways to provide explanations for the decisions made by deep neural networks. A major trend of works focus on attributing importance or relevance to different input features. These methods aim to identify the parts of the input that contribute most to the model's

output. Examples include SHapley Additive exPlanations (SHAP) [172], Local Interpretable Model-agnostic Explanations (LIME) [173], and Gradient-based Class Activation Mapping (Grad-CAM) [129]. Among them, gradient-based methods excel in providing better visual explainability, which are easier for human to understand. T. Nathan et al. [174] use a layer-ordered visualisation of information to visualise individual scale/layer contributions, and combine them into a single saliency map. By exploiting saliency map order equivalence, their method exceed traditional Grad-CAM and Grad-CAM++ without increasing the computational costs. Ekrem et al. [175] propose an attention-based saliency map prediction model to interpret the relationship between saliency and driving decisions. Similar to gradient based saliency methods, attention based methods also provide comprehensive visualisation. During training, the network focuses on specific parts of the inputs by nature, and the attention mechanism will gradually increase the weights of important parts through training. Kim and Bansal [176] propose an attentional bottleneck architecture and combine with visual attention to predict trajectories for autonomous driving cars and improve model explainability. To understand the intrinsic mechanism of the trajectory predictions, [177] utilise a transformer model with multiple prediction heads to retrieve the influencing factors of the predictions. Lei et al. [178] further explain DRL using both visual and texture explanation. In our work, the proposed saliency map based attack detection model not only provides detects adversarial attacks but also explainability to interpret the decisions made by our robust driving agent.

6.3 Methodology

6.3.1 Framework Overview

In this section, an explainable deep adversarial reinforcement learning approach for autonomous driving is proposed to enhance the robustness of RL agent against the observation perturbations. Our framework consists of three main components, including:

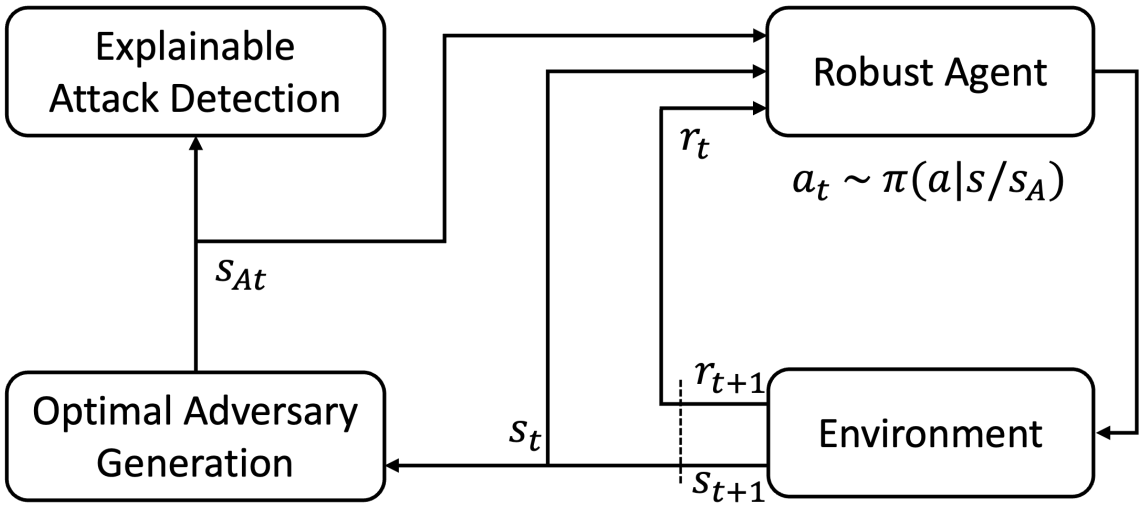

Figure 6.1 The framework overview. [Wang et al.]

1. An optimal adversarial attack generation to model the strongest observation perturbations on the perception of the driving agent.

2. An adversarial deep reinforcement learning scheme against the introduced optimal attacks to guarantee a robust guidance performance.

3. An efficient adversary detection model based explainability to show whether the guidance DRL network is under adversarial attacks.

Furthermore, with the explainability mechanism, the detection network provides more insightful information on the behavior of the driving agent under observation perturbations. The system overview is shown in Fig. 6.1.

6.3.2 Markov Decision Process with Perturbation

A deep reinforcement learning task can be considered as a continuous decision-making problem, which follows the Markov Decision Process (MDP) [35]. Basic MDP is defined as $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. The agent interacts with the environment of successive state $s \in \mathcal{S}$ with actions $a \in \mathcal{A}$, getting the step reward $r \in \mathcal{R}$ and the transition probability $p \in \mathcal{P}$ to the next state $s^{'} \in \mathcal{S}$. γ indicates the discount factor. The goal of deep reinforcement learning is to find an optimal policy that has the maximum accumulative discounted returns. In this section, to ensure the robust policy against

perturbation, the Markov Decision Process with observation adversary (OA-MDP) is defined.

In contrast to MDP, OA-MDP consists of a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{S}_{\mathcal{A}}, \mathcal{A}_{\mathcal{A}})$, where $s_A \in \mathcal{S}_{\mathcal{A}}$ is the set of states from S under optimal adversarial attack, and $a_A \in \mathcal{A}_{\mathcal{A}}$ is the action the agent responded to s_A. The attacked state s_A is a shifted state, which models the worst case perturbation due to attacks on the sensor leading to a_A sub-optimal than a. A well-trained guidance policy network may be able to cope with a weak and quick perturbation, turning back to the actual actions and the desired trajectory after the state observations get back to normal. However under strong and continuous adversarial attacks, this guidance policy network can easily fail.

6.3.3 Problem Formulation

To get a DRL agent robust to adversarial attacks, the goal is to find the optimal policy π^* that maximises the accumulative long-term rewards under series of worst-case perturbed states $\mathcal{S}_{\mathcal{A}}$. This can be formulated as:

$$J_{\pi^*} = \underset{\pi}{max} \underset{\mathcal{S}_{\mathcal{A}}}{min} J_{\pi}(\pi, \mathcal{S}_{\mathcal{A}}) \tag{6.1}$$

It means that the overal problem is decoupled as two sub-problems. First is to solve the inner minimum, finding the state $\mathcal{S}_{\mathcal{A}}$ under the worst-case adversarial attacks. The details of optimal adversary generation is introduced in the later section. Second is the outer maximisation of the long-term rewards under these adversaries, and here the robust proximal policy optimisation is presented.

6.3.4 Optimal Adversary Generation

In this section, the inner minimisation part of the problem is solved modelling the optimal adversarial attacks on observations based on a white-box technique. This chapter requires an adversarial attack generation method which prioritizes ease of implementation and computational efficiency. The Fast Gradient Sign Method

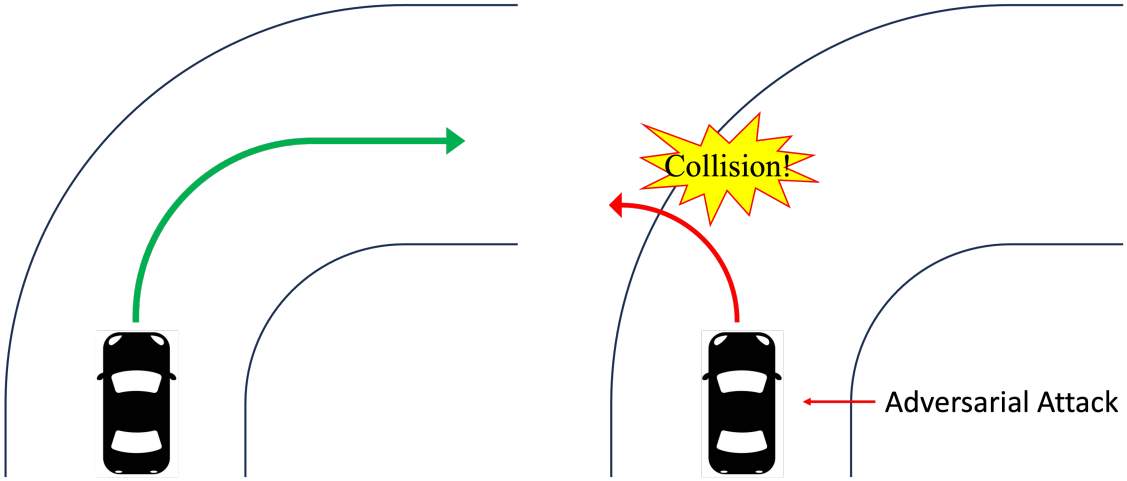

Figure 6.2 An example of how our adversarial perturbation works on a trained agent. [Wang et al.]

(FGSM), introduced by Goodfellow et al [179], fulfils these criteria. Additionally, FGSM offers flexibility, allowing for the application of techniques like Basic Iterative Method (BIM) [180] to introduce iterations in the calculations and enhance the adversarial attack instead of increasing the perturbation parameter ε.

FGSM works by using the gradients of the neural network to create an adversarial example. For deep reinforcement learning, the method uses the gradient $\bigtriangledown_s L(\pi, s, a)$ of the loss with respect to the input observation to create a new observation that maximises this loss. The π refers to the policy network parameters, while the a is the output action. The new masked observation is obtained by:

$$s_{adv} = s + \varepsilon * sign(\bigtriangledown_s L(\pi, s, a)) \tag{6.2}$$

This new observation s_{adv} is called the adversarial state. The ε parameter controls the strength of the attack, and it varies from 0 to 1. The closer the value to 1, the more visually distinguishable the s_{adv} is. On the contrary, if the ε is small, it will be harder to detect the difference with normal state for human eyes. As discussed in the beginning of the section, to solve the inner $\underset{\mathcal{S}_\mathcal{A}}{min} J_\pi(\pi, \mathcal{S}_\mathcal{A})$, the generated perturbations should lead the policy to minimum rewards. In our Robust PPO training, minimum rewards refer to minimum advantage value, which is unobtainable during decision-making phase. In such situation, an alternative loss function is established instead

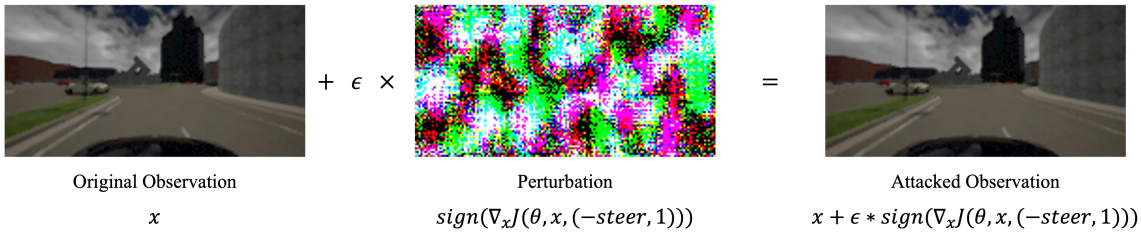

Figure 6.3 An Adversarial Example. The perturbation mask is enhanced for better visualisation. The alterations in the attacked observation are hard to recognise with the human eyes in reality. [Wang et al.]

of minimising the advantage value. Within the realm of autonomous driving, certain throttle and steering actions in certain scenarios may pose potential danger. Based on this idea, the gradient-based perturbations are introduced into the control system with the aim of maximising the throttle and reversing the steering, thereby deliberately maximising the risk of collisions. As shown in Fig. 6.2, in the turn making situation, optimally, the agent should slow down and make a right turn. After introducing our augmented FGSM, the agent will be encouraged to accelerate and make a sharp left turn to make the collision.

The problem of using FGSM is the effective sensitivity to ε. In practice, a larger ε is needed to make the perturbation effective on the deep guidance model decision. However during training, this will make perturbations predictable and tolerated by the model [155]. The BIM is employed to intentionally escalate the attack effeteness associated with autonomous driving. The BIM iterates the process of a single FGSM, which means a previous generated adversarial state will be the input of the next adversary generation. By iterating FGSM, the influence of the attacks can be increased without affecting the sensory image as much as by increasing the ε by a comparable amount. Finally, with the risk encouragement and BIM, the optimal adversary is obtained as:

$$\begin{aligned} s_{adv} &= s_n + \varepsilon * sign(\bigtriangledown_{s_n}(L_n)) \\ s_n &= s_{n-1} + \varepsilon * sign(\bigtriangledown_{s_{n-1}}(L_{n-1})) \end{aligned} \tag{6.3}$$

where

$$L_n = (-steer, 1) - (\pi \mid s_n) \tag{6.4}$$

s_{adv} indicates the optimal adversary, and n is the iteration number. These attacsk will lead the model to the worst actions in most of the cases. The test details of success rate of the attack will be carried out in results section. The process of one iteration of generating our adversarial sensory image sample is shown in Fig. 6.3.

6.3.5 Robust Proximal Policy Optimisation

To solve the outer maximisation with generated optimal adversaries, and learn the robust optimal driving policy, a robust proximal policy optimisation algorithm is introduced in this section. PPO is adopted as the baseline DRL. It is a model-free, on-policy, stochastic method. It shares the similar idea of TRPO (Trust Region Policy optimisation) algorithm [38], which restricts the policy update every step based on the probability ratios or the divergences between the new policy and the old policy. The most popular PPO variant employs a clipped surrogate objective to prevent the new policy differing too much from the old policy:

$$L^{clip}(\pi) = \hat{E}_t \left[min \left(\left(\frac{\pi(a|s)}{\pi^{'}(a|s)}, clip \left(\frac{\pi(a|s)}{\pi^{'}(a|s)}, 1-\varepsilon, 1+\varepsilon \right) \right) \hat{A}_t \right) \right] \tag{6.5}$$

Where $\hat{E}_t$ denotes the expectation over t. π is the new policy parameters of the deep guidance agent, and $\pi^{'}$ is the old policy, resulting $\frac{\pi(a|s)}{\pi^{'}(a|s)}$ the ratios of the probability of the new and old policy choosing action a under state s. ε stands for the constant clip term that limits the policy update by constraining the ratio within a specified range, preventing excessively large updates. $\hat{A}_t$ is the estimated advantage value, obtained by calculating the difference between the observed reward for taking the action a at the state s and the expected value predicted by the critic network at the state s:

$$\hat{A}_t = Q_t(s, a) - V_t(s) \tag{6.6}$$

As a stochastic method, the policy network in PPO outputs a probability distribution instead of an actual action. During training, actions are randomly sampled from this distribution to increase exploration. In the experiment, a multivariate Beta distribution is employed as the policy output to ensure a bounded action space.

A trained policy demonstrates robustness against adversarial attacks when the behaviours at non-attacked and attacked states show minimal divergences. This closeness in behaviour indicates that the policy remains effective and resilient in the presence of adversarial perturbations. Based on the theorem in [181], given a policy π and its value function $V_\pi(s)$, under the optimal perturbation s_A, for all $s \in \mathcal{S}$ and the corresponding $s_A \in \mathcal{S}_\mathcal{A}$, it holds that:

$$\max_{s\in\mathcal{S}, s_A\in\mathcal{S}_\mathcal{A}} \{V_\pi(s) - V_\pi(s_A)\} \leq \alpha \max_{s\in\mathcal{S}, s_A\in\mathcal{S}_\mathcal{A}} D_{TV}(\pi(s), \pi(s_A)) \tag{6.7}$$

Where the left indicates the long-term expected reward difference, which reflects the performance gaps between normal states and adversarial attacked states. $D_{TV}(\pi(s), \pi(s_A))$ is the Total Variation Distance between $\pi(s)$ and $\pi(s_A)$. The latter is the largest possible difference between the probabilities that the two probability distributions can assign to the same action. According to Pinsker's inequality [182], D_{TV} can be linked to another distance Kullback–Leibler (KL) divergence [183]:

$$D_{TV}(\pi(s), \pi(s_A)) \leq \sqrt{\frac{1}{2} D_{KL}(\pi(s), \pi(s_A))} \tag{6.8}$$

In practice, computing KL divergence is more efficient compared to the summing operation in D_{TV} and easier to realise in continuous space. The differentiable aspect of KL divergence also benefit the gradient based optimisation in the DRL algorithms. Therefore, Equation (6.7) is further amended with the KL divergence distance to:

$$\max_{s\in\mathcal{S}, s_A\in\mathcal{S}_\mathcal{A}} V(s, s_A) \leq \max_{s\in\mathcal{S}, s_A\in\mathcal{S}_\mathcal{A}} \sqrt{\frac{1}{2} D_{KL}(\pi(s), \pi(s_A))} \tag{6.9}$$

That is, if the KL divergences of the action probabilities are minimised under non-attacked states and attacked states, the total variation distance D_{TV} will also be

minimised, therefore the performance gaps could be minimised too. The objective function of the proposed robust proximal policy optimisation algorithm is then developed. Combining the divergence term and the accumulative expected reward, the optimisation problem is formulated as followed:

$$\begin{aligned} &\max_{\pi} \{J_\pi + KL_r\} \\ &subject \quad to \qquad \mu - L_2(f(s_t, \pi), s_t) \geq 0 \end{aligned} \tag{6.10}$$

Where

$$J_\pi = \hat{E}_t \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{6.11}$$

and KL_r is the reverse term of the KL divergence of the action probability under attacked and non-attacked states. Note that this pseudo term is only used here to match the maximisation problem and will be transferred back to standard format later. f is the optimal adversary generation function with policy π at current s. The constraint condition for the maximisation optimisation indicates that the L_2 norm of the state difference under normal and perturbed states is bounded by μ. t and γ denote the time step and discount factor respectively. Based on the Lagrange multiplier technique, the generalised Lagrange function of the original optimisation is obtained:

$$\mathcal{L}(\pi, \alpha) = \sum_{t=0}^{T} \Big(\gamma^t(s_t, a_t) + KL_r \ + \alpha(\mu - L_2(f(s_t, \pi), s_t))) \tag{6.12}$$

Where $\alpha \geq 0$ is the Lagrange multiplier, a new variable introduced to replace the constraint equation. Now consider a new function θ_P with respect to α:

$$\theta_P(\alpha) = \min_{\alpha \geq 0} \mathcal{L}(\pi, \alpha) \tag{6.13}$$

If policy π satisfies the constraint condition in Equation (6.10), it has:

$$\theta_P(\alpha) = \sum_{t=0}^{T} \Big(\gamma^t r(s_t, a_t) + KL_r \Big) \tag{6.14}$$

otherwise $\theta_P(\alpha) = -\infty$. The proof is shown as follows:

Under a fixed policy π, minimising the Lagrange function corresponds to minimising the Lagrange-multiplier-weighted constraint $\alpha(\mu - L_2(f(s_t, \pi), s_t))$. Given that $\alpha \geq 0$, if $(\mu - L_2(f(s_t, \pi), s_t)) < 0$, which violates the constraint, the following holds:

$$\begin{aligned} \min_{\alpha \geq 0} \sum_{t=0}^{T} \alpha(\mu - L_2(f(s_t, \pi), s_t)) \rightarrow -\infty, \\ \Rightarrow \min_{\alpha \geq 0}, \mathcal{L}(\pi, \alpha) \rightarrow -\infty, \\ \Rightarrow \theta_P(\alpha) = -\infty. \end{aligned} \tag{6.15}$$

Conversely, if $(\mu - L_2(f(s_t, \pi), s_t)) \geq 0$, which satisfies the constraint:

$$\begin{aligned} \min_{\alpha \geq 0} \sum_{t=0}^{T} \alpha(\mu - L_2(f(s_t, \pi), s_t)) \rightarrow 0, \\ \Rightarrow \min_{\alpha \geq 0}, \mathcal{L}(\pi, \alpha) = \sum_{t=0}^{T} \left(\gamma^t r(s_t, a_t) + KL_r \right), \\ \Rightarrow \theta_P(\alpha) = \sum_{t=0}^{T} \left(\gamma^t r(s_t, a_t) + KL_r \right). \end{aligned} \tag{6.16}$$

Thus, it is proved that when the constraint is not satisfied, the minimisation leads to $-\infty$, while satisfying the constraint yields the primal optimisation problem.

The proof above denotes that maximising the newly defined function with respect to policy $\max_{\pi} \theta_P(\alpha)$ is equivalent to the primal problem in Equation (6.10), as they both have the same solutions. Based on Equation (6.10) and Equation (6.14), the primal optimisation problem is reformulated as:

$$\max_{\pi} \theta_P(x) = \max_{\pi} \min_{\alpha \geq 0} \mathcal{L}(\pi, \alpha) \tag{6.17}$$

This optimisation is detailed by the two following steps:

$$\begin{aligned} &(1) \quad \min_{\alpha \geq 0} \sum_{t=0}^{T} \left(\gamma^t (s_t, a_t) + KL_r + \alpha(\mu - L_2(f(s, \pi), s)) \right) \\ &(2) \quad \max_{\pi} \sum_{t=0}^{T} \left(\gamma^t (s_t, a_t) + KL_r + \alpha(\mu - L_2(f(s, \pi), s)) \right) \end{aligned} \tag{6.18}$$

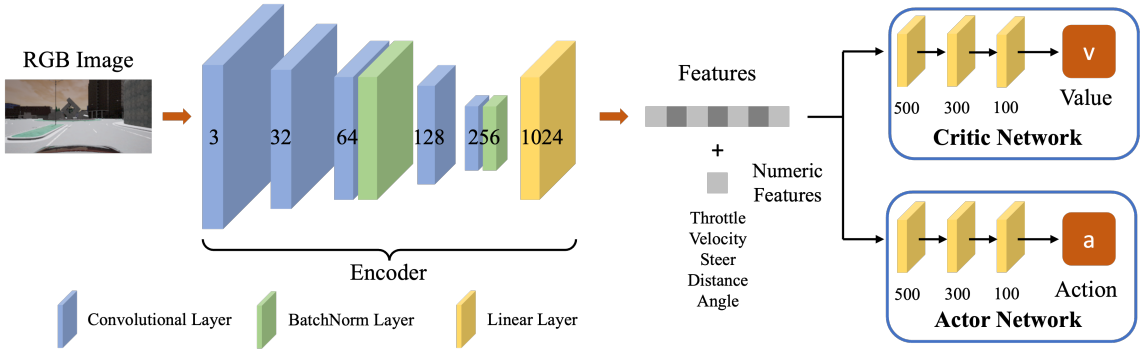

Figure 6.4 Network structure of the robust DRL model. It consists of a feature extract encoder, and the actor-critic network. The actor-critic network is fed with the concatenation of the numeric features and the extracted features. [Wang et al.]

First, the policy π is frozen and the Lagrange multiplier α is updated based on step (1). Then similarly, the updated Lagrange multiplier α is frozen to update the policy π according to step (2). Specifically, step (1) can be represented as:

$$\min_{\alpha \geq 0} \sum_{t=0}^{T} (\alpha(\mu - L_2(f(s_t, \pi), s_t))) \tag{6.19}$$

And step (2) can be represented as minimising the negative terms:

$$\begin{aligned} &\max_{\pi} \sum_{t=0}^{T} \Big(\gamma^t(s_t, a_t) + KL_r + \alpha(\mu - L_2(f(s_t, \pi), s_t))\Big) \\ =& \min_{\pi} \sum_{t=0}^{T} \Big(-\gamma^t(s_t, a_t) + D_{KL,t} - \alpha(\mu - L_2(f(s_t, \pi), s_t))\Big) \\ =& \min_{\pi} \sum_{t=0}^{T} \Big(\hat{A}_t + D_{KL,t} + \alpha L_2(f(s_t, \pi), s_t)\Big) \end{aligned} \tag{6.20}$$

Here as μ is independent from π, the term $\alpha\mu$ is removed. The KL divergence D_{KL} is transferred back from KL_r. Finally, the maxmin problem becomes a minmin problem. In practice, $L^{clip}(\pi)$ from equation 5 is used to replace $\hat{A}_t$ in Equation (6.20) for better performance. Gradient descent is applied to find both the optimal Lagrange multiplier α and the optimal policy π repeatedly through Equation (6.19) and Equation (6.20) in each batch update. The objective function is a jointly optimisation problem, where the parameters of the three terms $L^{clip}(\pi)$, $D_{KL,t}$ and $L_2(f(s, \pi), s)$ balance the performance under normal observation and the adversarial

Algorithm 1 Robust Proximal Policy Optimisation

Initialise the actor network parameter θ_0, the state-value critic network parameter ϕ_0, the replay buffer $\mathcal{D}$;
Initialise the Lagrange multiplier α;
for $k = 0, 1, 2, ...$ **do**
  for $t = 0, 1, 2, ...$ **do**
    Sample action $a_t \sim \pi(\theta_k)(a_t|s_t)$ based on the policy $\pi(\theta_k)$ and state s_t.
    Generate optimal adversary s_{At} through equation 10.
    Obtain transitions s_{t+1}, r_t by executing a_t.
    Store the transitions in $\mathcal{D}_k \leftarrow \{s_t, a_t, r_t, s_{t+1}, s_{At}\}$.
  end for
  Calculate the reward-to-go $\hat{R}_t$.
  Calculate estimated advantage $\hat{A}_t$ through equation 6.
  Obtain action distributions d, d_A under all s and s_A from $\mathcal{D}_k$
  Calculate KL divergence D_{KL} based on d, d_A.
  Calculate L_2 norm of the difference of attacked and normal states.
  Update the Lagrange multiplier α by minimising the step 1 objective function:
  $\alpha_{k+1} = \underset{\alpha \geq 0}{argmin} \quad \alpha(\mu - L_2(f(s, \pi), s))$
  Update the policy θ by minimising the step 1 objective function:
  $\theta_{k+1} = \underset{\theta}{argmin} \quad L^{clip}(\theta) + D_{KL} + \alpha L_2(f(s, \theta), s)$
  Update the state-value critic network parameter ϕ_0 by regression on mean-squared error:
  $\phi_{k+1} = \underset{\phi}{argmin} \quad \frac{1}{T} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)$
end for

defence capability. In the experiments, it is discovered the policy will learn to satisfy the constraint in Equation (6.10) and resulting the term $\alpha L_2(f(s, \pi), s)$ in step (2) to decrease during the gradient descent process, and α in step (1) will gradually approach zero. Consequently, step (2) will find the minimum after α becomes zero. For the parameter of $D_K L$, experiments are conducted with several potential values and the best performing one is chosen.

The proposed robust DRL actor-critic network structure is illustrated in Fig. 6.4. The model takes the monocular RGB image as input, and concatenates the high-level features extracted by the encoder with five additional numeric features. The concatenated features are then fed into actor and critic networks. With our optimal adversary generation and robust proximal policy optimisation, the problem formulated in Equation (6.1) is solved. The robust method is detailed in Algorithm 1.

6.3.6 Reward Function Design

The reward function design is one of the most crucial parts for DRL problems. To make sure the agent accomplishes the roundabout making task while ensure the safety, efficiency, the hand-designed reward function is defined as followed, which are mainly two signals:

$$r_{total} = r_{done} + r_{compound} \tag{6.21}$$

For the r_{done}, five distinct terminal conditions are defined, and three of which share the same reward value. These three conditions are characterized as 'Early Stop Signals' to prevent situations considered as meaningless explorations, thereby enhancing training efficiency:

1. Distance Threshold: If the distance between the center of the ego vehicle and the road center is beyond a certain criterion, it reveals that the car deviates too far from the desired trajectory.

2. Under-speed Threshold: If the velocity of ego vehicle is below 1 km/h for over certain time steps. This usually means that the agent chooses being still to avoid receiving the negative rewards, which is commonly seen in DRL training.

3. Over-speed Threshold: If the speed is over the pre-defined maximum speed. For safety concerns, the maximum speed is limited at 25 km/h.

All three of these Early Stop Signals are assigned with a reward value of -100, aiming to discourage actions associated with these conditions. Another terminal signal, collision, consists of three different types: Collision with pedestrians; Collision with other vehicles; and Collision with other objects. They are assigned respectively with -1000, -200, -100, to express the severity associated with each collision type. Finally, if the ego vehicle approaches the target position, the agent will receive a success reward of 100.

To calculate $r_{compound}$, two variables are introduced:

$$centering = 1 - \frac{distance\_from\_center}{max\_distance\_from\_center} \tag{6.22}$$

$$angling = 1 - \mid \frac{angle}{max\_angle} \mid \tag{6.23}$$

The $distance\_from\_center$ refers to distance between the center of the ego vehicle and the road center, and the $angle$ denotes the angle between the car and the vector of two neighboring way-points. Both of the variables are in range of [0, 1]. $r_{compound} = V * centering * angling$, and V changes depending on the vehicle speed. When the speed is between 12 km/h and 22 km/h, $V = 1$. If the speed is below 12, $V = \frac{velocity}{12}$ and if the speed exceeds 22 km/h, $V = 1 - \frac{velocity-22}{3}$. This setting aims to encourage the DRL agent to maintain a safe speed within the range of 12 km/h to 22 km/h. In summary, the designed reward function balances the encouragement for goal reaching, obstacle avoidance and safety. It is worth noting that the adversarial attack defence algorithm is independent from the reward function design. Both the baseline and the proposed approach are trained using the same reward function, meaning that the efficiency of the robust method is hardly affected by the design of the reward function.

6.3.7 Attack Detection Network and Explainability

In this section, an efficient explainability based adversarial attack detection network is introduced for attack awareness and better understanding the fundamental of the decision making process under adversarial attacks. Saliency maps are adopted to provide insights into the DRL model's decision-making process by highlighting the most influential regions in the input space. Saliency maps help understand which parts of the input contribute most to the agent's actions. With careful analysis, it is noted that the saliency maps generated from the same policy show certain discrepancies under normal states and adversarial attacked states. This spurred the decision to propose a deep network-based detector (CNN-based detector) exploiting

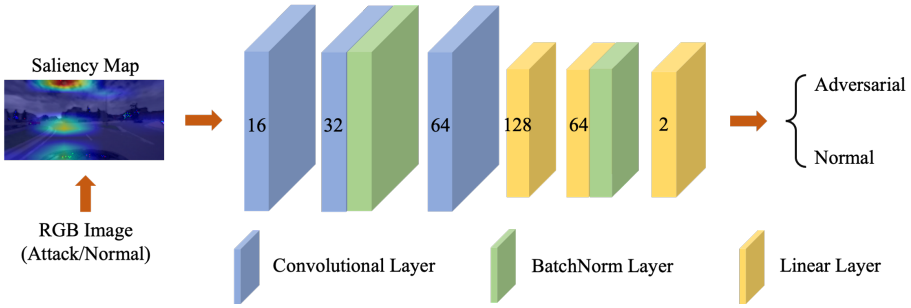

Figure 6.5 Network structure of our adversarial attack detection model. [Wang et al.]

explainability based saliency maps by monitoring the saliency maps generated by the gradients of the robust proximal policy optimisation agent. The detector will learn and find patterns in the saliency map images to detect adversarial attacks.

By calculating the gradients of the outputs with respect to the feature maps from the last convolutional layer of the robust DRL model, the saliency maps are obtained. These maps are represented as heatmaps, and they will be up-sampled to overlay on the sensor RGB image inputs to form the final visualisation. A novel adversarial attack detection data is collected in Carla using the trained agent to drive around. The details of the attack detection dataset will be explained in the experiment section.

An optimised convolutional neural network architecture is designed to balance the detection accuracy and the computation burdens. It consists of three convolutional layers with channel numbers of 16, 32, and 64, respectively. All convolutional layers have kernel size of 3, stride of 1, and padding of 1. Three linear layers are then connected to the last convolutional layer with output neurons of 128, 64, and 2 for the final prediction of adversarial or normal. Batch normalisation layers are employed after the second convolutional layer and the second linear layer to stabilise the training process. ReLU activation functions are used, and the network is initialised using Kaiming uniform initialisation. The details of the network is shown in Fig. 6.5. Cross-entropy loss function is used to fit the network outputs and the supervised signals.

Figure 6.6 Environment for experiments in CARLA. The training scenario comprises the roundabout and its connecting extension road. The green cross means starting point and red cross is destination. The yellow line is demonstrating trajectory. [Wang et al.]

6.4 Experiments

6.4.1 Simulator and Scene Settings

The training and evaluation of the proposed framework are implemented in CARLA simulator. CARLA is an open-source platform for development, training, and validation of autonomous driving systems. It has a rich library of vehicle models and realistic urban road modelling, hence being near ideal to urban driving simulation. The experiment takes place in Town 3, as shown in Fig. 6.6. The ego agent is randomly spawned around the starting point, and the task is to navigate through 3/4 of a roundabout while avoiding static and dynamic obstacles, such as road lamps, surrounding vehicles and pedestrians, where the latter move autonomously using PID controllers. The task is challenging due to multiple exists in the trajectory to avoid entering, and the unpredictable movement of pedestrians and vehicles which are randomly spawned in the map.

Hyper-parameter	Value
discount factor γ	0.99
RL network learning rate	0.0001
Lagrange multiplier learning rate	0.01
initial Lagrange multiplier α	0.01
memory size	5000
clipping ratio	0.2
constraint threshold μ	0.001

Table 6.1 Hyper-parameters for the experiments. [Wang et al.]

6.4.2 Implementation Details

The DRL agent takes 160*80 RGB images as inputs from a monocular camera and 5 numeric features: throttle, velocity, steer, the distance between the vehicle and the road centres, and the angle between the vehicle forward vector and the tangent to the road as shown in the network structure in Fig. 6.4. Specifically, the encoder extracts the high-level features from the RGB image, and they are concatenated with the five numeric features. The policy network and value network share the parameters in feature extractor, where the high-level information is shared to benefit both tasks during the training. For the constraint threshold μ, a suitable value is selected manually from candidate values through experiments. This threshold serves to restrict the gradient changes of the policy network outputs in response to adversarial attacks. It is sensitive to the scenario as it significantly affects the stability and performance of the training process. The main hyper-parameters used are shown in Table 6.1. To increase the uncertainties in the process of generating optimal adversarial attacks, a normal-distributed random strength controlling parameter ε is used, with $mean = 0.01$ and $std = 0.012$, and the value of ε is limited in range [0.01, 0.05]. Additionally, two methods are implemented to handle the bounding of adversarial attacks during training time. First, following [181], the attacks are manually constrained by employing a clipping technique on the adversarial states. This technique restricted the magnitude of changes resulting from the generated optimal adversaries. While this approach enhances training stability, it may potentially limit the overall performance of the driving policy. Secondly, the

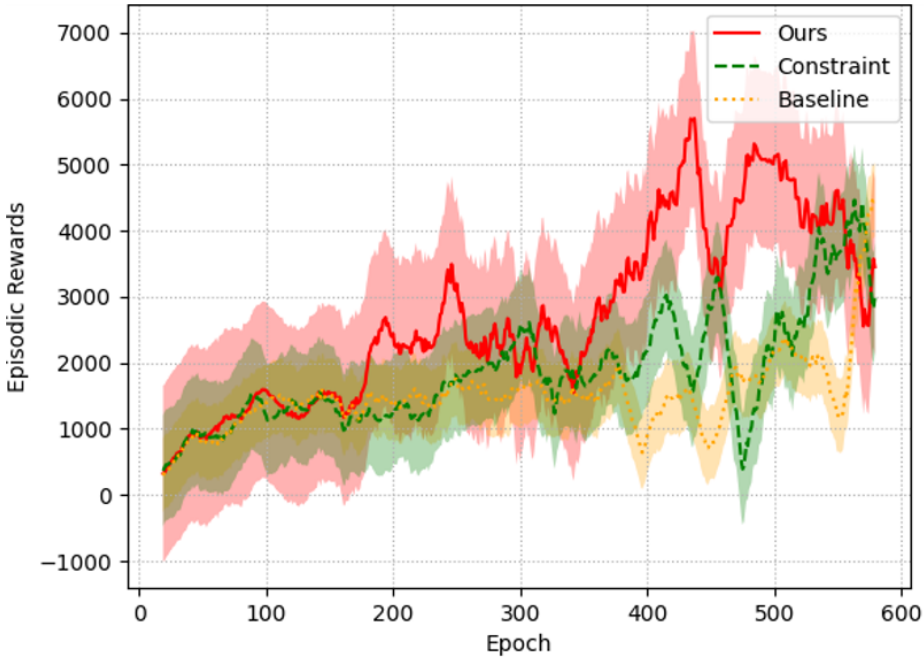

Figure 6.7 Training curves of the proposed model and its variant (constraint optimisation only), and the baseline model. [Wang et al.]

manual bounding process is disabled, enabling the algorithm to autonomously learn and naturally satisfy the constraint of the observation changes due to adversarial attacks within μ, as described in Equation (6.10). During the testing phase, manual bounding is not applied to either model, as it is unfeasible to confine the adversarial attacks or observation uncertainty within a specific range in real world. In the robust agent evaluation section, the performance comparison of the proposed method is conducted using these two implementations. The model runs at 250 fps in CARLA on a single RTX 4090 graphic card.

6.4.3 Results

In this section, experiments are carried out for the efficiency assessment of the optimal adversarial attacks, the robust guidance policy evaluation and the adversarial attack detection accuracy as well as explainability. To realise these tests, first the baseline (PPO), the proposed robust model with different attack bounding policies, and its variants are all trained. The training is conducted in the roundabout scenario with randomly spawned surrounding traffic. All algorithms are trained under the same applicable hyper-parameter settings and environment conditions. Fig. 6.7 shows the

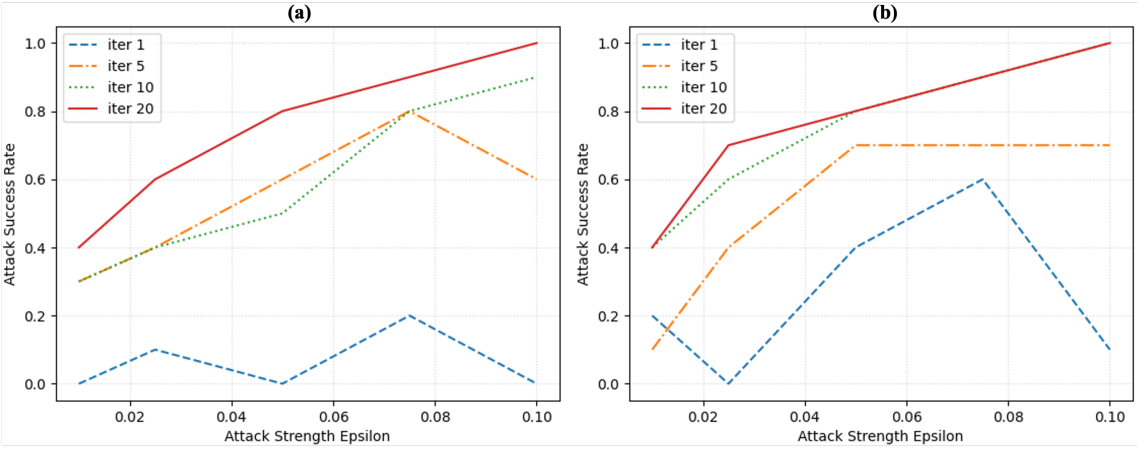

Figure 6.8 The attack success rates at different settings of 100 (a) and 200 (b) continuous frames of perturbation. [Wang et al.]

training curves of the proposed method (no manual bounding), proposed variant (constraint optimisation only) and the baseline. The proposed model demonstrates faster convergence compared to the other two. As the driving trajectories are generated by the actions based on the normal observations in training, the episodic rewards represent the performance under normal observations, but not necessarily reflect the robustness of the models. Detailed robustness tests will be conducted in the later section.

Adversarial Attack Evaluation

Before testing the robust deep reinforcement learning agent, the impact of the proposed optimal generated adversaries is evaluated at various experiment settings. The attack strength parameter ε is set from 0.01, to 0.025, 0.05, 0.075 and 0.1. At each ε, different numbers of iterations are applied at 1, 5, 10, and 20. As the model runs at a very high 250 fps, it needs more frames to be attacked to make a deviation in the DRL decision making process. Therefore two sets of attack frames are conducted at 100 and 200 continuous frames. It is observed that in the testing time, certain levels of adversarial attacks cause the agent to be seriously off the desired path, for example to the lawn and the sidewalk, but eventually the agent avoids collision or avoids out-of-lane threshold. To increase the difficulty of testing and distinguish the attack abilities, a successful attack is defined as leading the

self-driving agent directly or finally into collision or exceeding out-of-lane threshold. The adversarial attack evaluation in this section is tested on the non-robust trained normal PPO, with 10 episodes in each configurations. The results are shown in Fig. 6.8. The success rate is positively related to the number of iterations and the value of ε. It also can be seen that, higher iterations or higher strength parameters normally lead to higher success rates. The results in Fig. 6.8 (a) with 100 attack frames are first analysed. When the iteration is 1, which means that the attack method is just the FGSM, the impact on the trained deep guidance agent is unstable and relatively minor despite how strong the ε is. This also implies that the impact gained from increasing ε is not as substantial as increasing the iteration numbers when the iteration is small. The success rates are below 20%, and even invalid at 0.01, 0.05 and 0.1 of ε. The adversarial attacks have similar impact when the numbers of iterations are set at 5 and 10 (although relatively higher). Their success rates vary from 30% to 90% depending on the strength parameter settings. Lastly, the generated attacks hold the highest and stable impact performance when the number of iteration is 20. At the smallest ε, it still holds 40% of success attacks. Furthermore, with the increase of ε, it reaches 100% attack success rate. This result also indicates that a well-trained non-robust PPO has no chance in surviving under such strong adversarial perturbations.

The influence of the attacks is further elevated by increasing the continuous attack frames to 200. Fig. 6.8 (b) shows the evaluation results with the same configurations as 100 frame attacks. As expected, even with increased number of frames, the attack with 1 iteration as function of ε remains the lowest success rate and unstable. Iteration 5 does not show significant improvement comparing to 100 continuous frames attack, and reaches a limit of 70% success rate after ε is over 0.05. Iteration 20 and 10 share most part of the performance curve except at ε of 0.025. Same as before, both of them can hit a 100% of success rate. In a word, the proposed optimal adversary generation shows great performance on disturbing the decision making of the self-driving DRL agent, and with the proper settings the attack success rate can easily remain in 90-100% level. Note that due to the high fps the model running at, the 100 and 200 consecutive frame attacks only take 0.4 and 0.8 seconds of simulation

Epsilon	Method	Average Reward	Collision Rate	Success Rate
0.05	Baseline	1133.15±109.31	0.070±0.012	0.080±0.049
	Constraint	2187.47±254.83	0.099±0.018	0.670±0.078
	Ours Manual	2441.15±204.86	0.040±0.014	0.770±0.096
	Ours	**2656.04±341.03**	**0.038±0.013**	**0.809±0.113**
0.1	Baseline	803.08±142.49	0.230±0.012	0.000
	Constraint	1684.47±256.93	0.078±0.015	0.450±0.103
	Ours Manual	1928.81±142.49	0.044±0.020	0.530±0.096
	Ours	**2255.70±281.91**	**0.042±0.004**	**0.679±0.086**

Table 6.2 Evaluation of different algorithms with different epsilon value. The constraint refers to the model with only the constraint optimisation. Ours and Ours manual share the same algorithm but Ours is implemented without manual clipping for the adversarial attacks. [Wang et al.]

time, respectively.

Robust Agent Evaluation

In the previous section, the optimal adversary generation shows a great impact on the decision-making process of the baseline deep guidance agent. The performance gaps between 200 and 100 consecutive frame attacks become negligible at higher strength parameters ε and iteration numbers. To evaluate the performance of the proposed robust guidance algorithm under observation perturbations, two difficult attack configurations are chosen, $\varepsilon = 0.1$ and $\varepsilon = 0.05$ which have over 80% success rate in the previous evaluation. Both ε values are set with iterations of 20 and 100 consecutive frames adversarial attacks. For each category all models are evaluated for 100 episodes. Note that a stochastic ε value is used in the training, while the maximum value is only 0.05. In other words, most of the adversarial attacks in the evaluation are much stronger than the ones in the training process.

Four methods are evaluated, the baseline PPO, the proposed method with only constraint optimisation, the proposed method with manual attack bounding, and the proposed method without bounding. The categories are defined as followed:

1. Average Reward: The average accumulative reward per episode.

2. Average Success Rate: Current success times to reach the goal position over current episodes

3. Average Collision Rate: Current collision encounters over current episodes.

Note that despite the concerns about the bias in evaluation due to incorporating extra, non-attacked numeric data as inputs to enhance stability and performance, the results still show substantial impacts on the baseline model, resulting in high failure rates. The proposed method effectively handles strong adversarial attacks under the same conditions, indicating that the additional inputs do not compromise the evaluation outcomes. In Table 6.2, Ours (the proposed method without bounding) leads all the categories. Fig. 6.9 illustrates the performance of these models under the two settings in a more comprehensive way. The thin line is the mean value of each category and the standard deviation is visualised around the mean value. Overall as it can be seen, Ours outperforms the baseline model with large margin in terms of accumulative rewards, task success rate, and collision encounters. Specifically, with $\varepsilon = 0.05$ in Fig. 6.9 (a), the baseline deep guidance model is at 1133.15 per episode. In contrast, the proposed method achieves around 2656.04 accumulative reward an episode, which represents 134.4% gain from the baseline. Comparing to the Constraint (with only constraint optimisation), Ours achieves 21.4% improvement. And as expected, the manual attack bounding (Ours Manual) loses 8.8% of performance, but still averages 2441.15 in terms of reward. In the category of task success rate in Fig. 6.9 (b), Ours averages 80.9%, leading the baseline (at 8.0%) by 72.9%. With this attack strength, the baseline model can barely finish the task, due to the sub-optimal or worst decision making impacted by the adversarial attacks. The robust guidance model and its variants on the other hand keep resistant to the strong perturbations. All models demonstrate low tendency towards collisions in Fig. 6.9 (c), but Ours holds the advantage at the lowest 3.8% of collision rate. From this result it can inferred that surpassing the out-of-lane threshold contributes to the majority of failure cases in the attack evaluation section. The reason that the collision rate of baseline does not match the attack success rate in the previous section is that the out-of-lane situations are excluded here.

When the attack strength escalates to $\varepsilon = 0.1$, as expected, the baseline model shows worse robustness to the perturbations. In the reward category in Fig. 6.9 (d), the

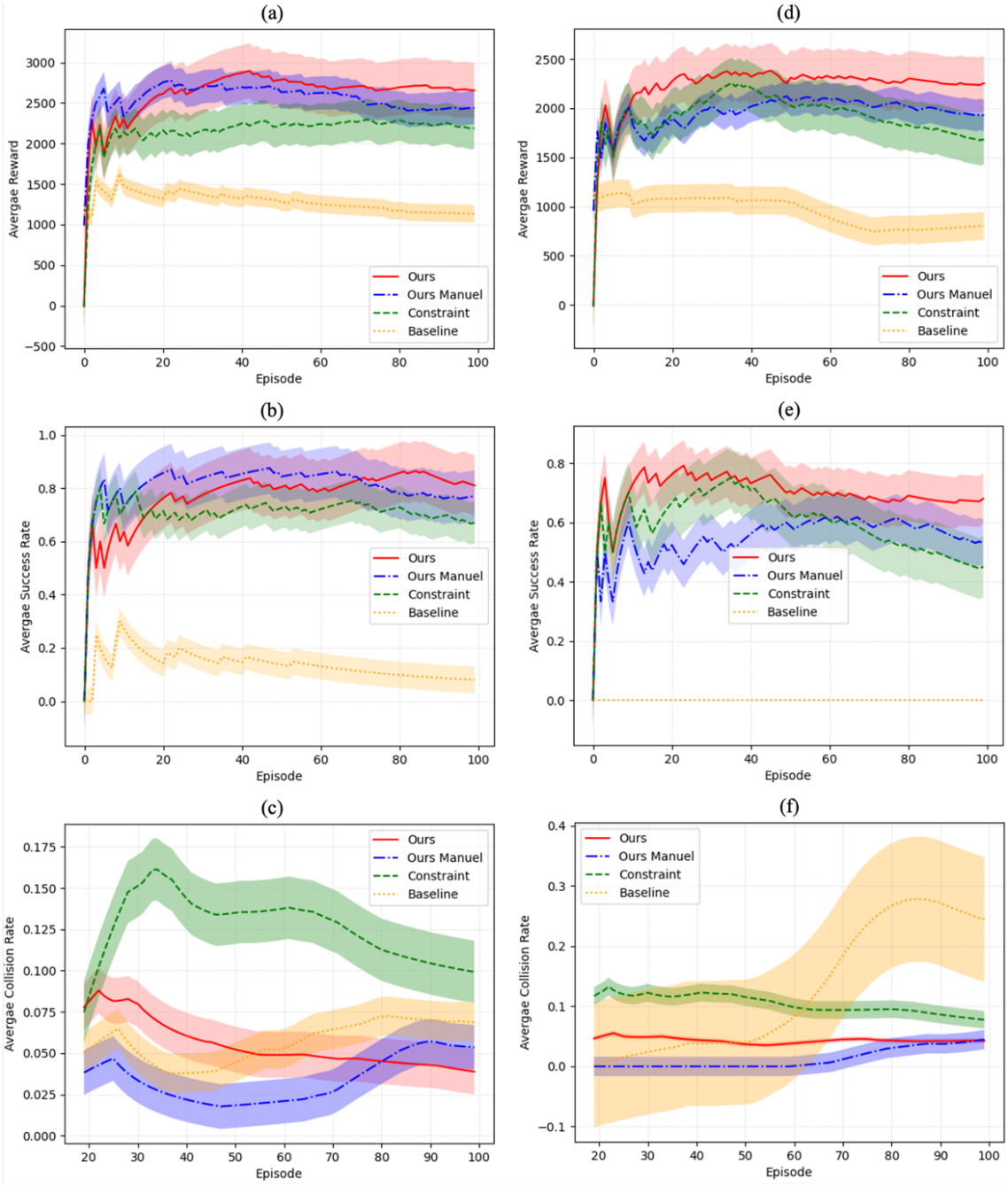

Figure 6.9 Detailed comparisons between the proposed method and baseline. The agent is attacked with 100 consecutive frames with the same *iteration* = 20 and surrounding traffic. In (a)(b)(c) $\varepsilon = 0.05$ and in (d)(e)(f) $\varepsilon = 0.1$. The curve line is the mean value while transparent area indicates standard deviation. [Wang et al.]

baseline decreases to 803.08, whereas Constraint reaches 1684.47, marking a 109.7% improvement already. Ours Manual achieves 1928.81, 140.2% better than the baseline. In comparison to manual bounding, Ours achieves an average cumulative reward of 2255.7, marking a 180.8% increase over the baseline. This demonstrates a 33%

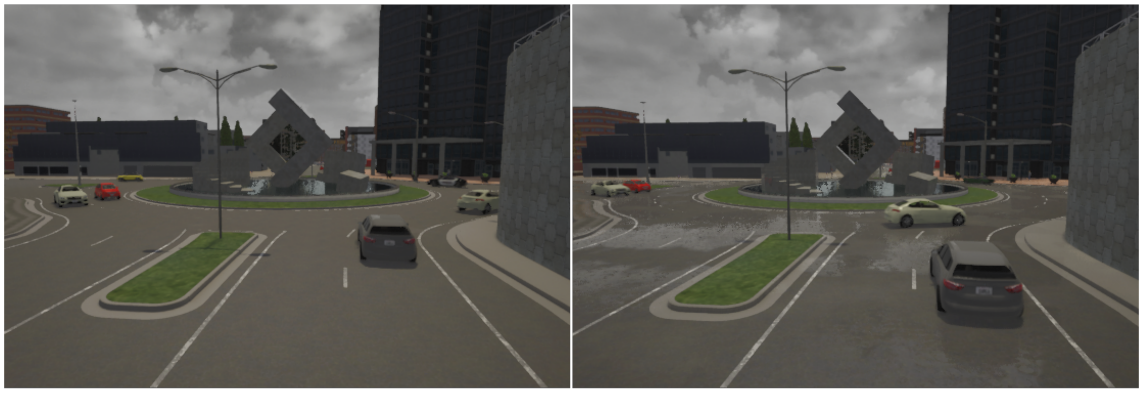

Figure 6.10 Visualisation of the road in the training cloudy noon (left) and the unseen heavy rain (right) weather condition. [Wang et al.]

improvement over the Constraint method and a 17% enhancement solely attributed to the autonomously bounding learned by the proposed algorithm. In terms of task success rate, the baseline model exhibits a complete failure, being unable to complete a single episode. Although the performance of Constraint, Ours Manual and Ours decline compared to the setting with $\varepsilon = 0.05$, they still successfully reach the goal point without collisions at 45%, 53% and 67.9% of the time respectively. Under the stronger attacks, the notable performance difference emerges in the collision category. Ours experiences a collision rate of 4.2%, close to the previous setting. The baseline encounters collisions 5.47 times higher than our model. Moreover, Ours demonstrates superior performance across all categories without the need for manual perturbation bounding. This highlights the algorithm's capacity to learn and restrict observation shifts caused by gradient-based adversarial attacks during training, thereby enhancing robustness against stronger, unbounded perturbations. The divergence based optimisation term further reduces the action probability generated under normal and attacked observations.

These tests demonstrate the capability of the proposed robust proximal policy optimisation guidance model to remain resilient against strong adversarial perturbations while maintaining proficient performance in the driving tasks. Moreover, to assess the generalisation capability of the proposed method in handling domain shifting, another challenging scenario that requires system robustness, both the proposed model and the baseline model are evaluated under the heavy rain weather condition,

Methods	Average Reward	Collision Rate	Success Rate
Baseline Normal	780.14±111.25	0.034±0.014	0.01±0.02
Ours Normal	2031.25±283.47	0.012±0.007	0.56±0.08
Baseline Attacked	764.71±103.88	0.123±0.025	0
Ours Attacked	1776.56±202.49	0.017±0.007	0.42±0.07

Table 6.3 Evaluation in the unseen heavy rain weather condition. The proposed method and baseline model are tested under normal observation and attacked observation with epsilon=0.05. [Wang et al.]

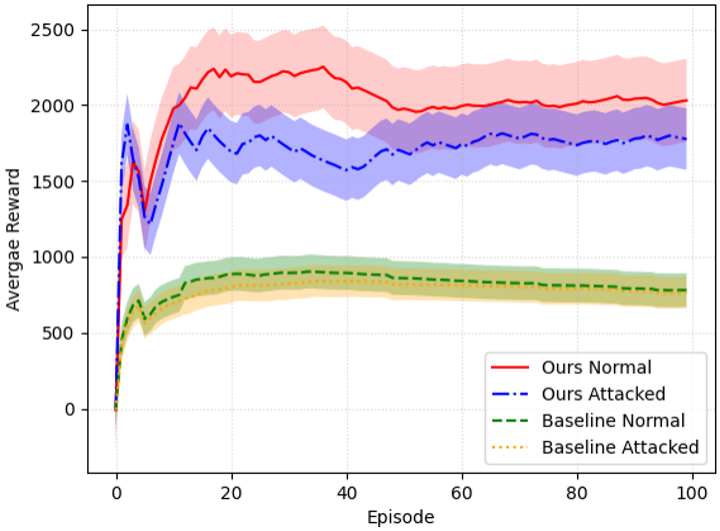

Figure 6.11 Performance comparison in average accumulative reward category of the proposed method and the baseline in heavy rain weather condition. [Wang et al.]

an unseen scenario for both models. Specifically, the precipitation parameter (rain) is set as 50/100, precipitation-deposits (puddle) is set as 50/100, and the wetness (water on the camera) is set as 20/100 in CARLA. The visual difference can be seen in Fig. 6.10. Two settings are applied as normal observations and attacked observations with $\varepsilon = 0.05$. The results are shown in Table 6.3. In the heavy rain weather condition, without any adversarial attacks encountered, the baseline model fails to adapt its driving policy to the new scenario. It averages a reward of 780.14 per episode and only completes the task once. In contrast, when faced with the unseen heavy rain condition, the proposed model exhibits great generalisation, achieving an average reward of 2031.25 per episode with a 56% success rate. When the adversarial perturbations are introduced, the performance of the baseline continues to fall to a collision rate of 0.123 and zero task success. The proposed model stays resilient to the perturbed unseen scenario, scoring 1776.56 episodic rewards and 42% of success

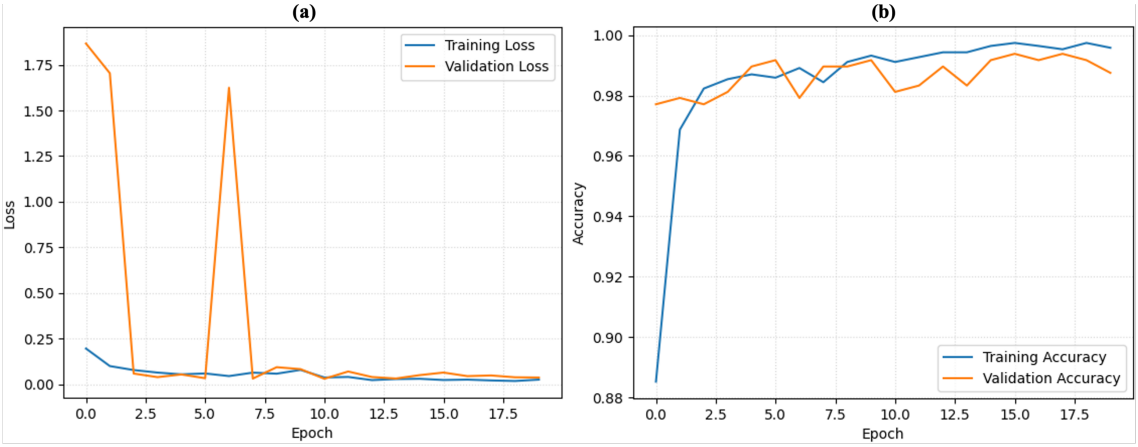

Figure 6.12 Training curve of the explainable attack detection model. (a) shows the loss and (b) demonstrates the accuracy of the training and validation during the training phase. [Wang et al.]

rate. In Fig. 6.11, the curves of the baseline in normal and attacked scenarios appear similar. This similarity suggests that the baseline model is already highly unstable due to weather changes. Consequently, adversarial attacks do not significantly escalate this situation.

In summary, the proposed algorithm not only handles optimal adversarial attacks within the training weather but also demonstrates robust generalisation in unforeseen scenarios, even in the presence of observation perturbations.

Attack Detection Evaluation and Explainability

In total 2200 frames of synthetic data are collected from CARLA by navigating the trained proposed model. Out of the collected frames, 1100 represent normal, non-adversarial sensor observations, while the remaining 1100 are adversarial examples generated by stochastic controlled adversaries with $\varepsilon \in [0.5, 1]$. This range aligns with the parameters used in the evaluation of the robust agent. The saliency maps are then generated by computing the gradients of the outputs from the final convolutional layer within the feature extractor of the self-driving agent, with respect to the agent action outputs. The data is divided into 1700 samples for training and 500 samples for evaluation purposes. Fig. 6.12 illustrates the training progress of the adversarial detection network. After 20 epochs of training, the detector reaches an accuracy of 99.58% on the training set and 99.38% on the validation set.

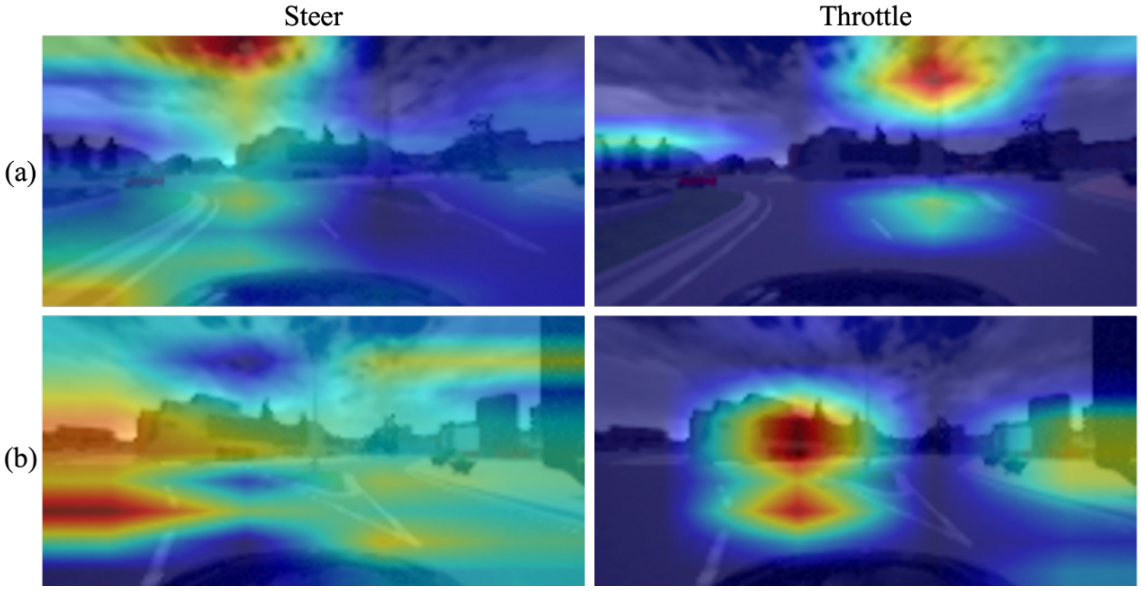

Figure 6.13 Visualisation of the explainable saliency maps for steer and throttle actions of an under-trained policy (a) and a well-trained policy (b) at the normal observation respectively. [Wang et al.]

Furthermore, to explore the explainability behind the behaviours of the self-driving model, the saliency maps used in the training of the adversarial detector are visualised. Overall in Fig. 6.13, it is clear that the steer and throttle actions depend on distinct information derived from the sensory input observation. In the normal driving condition, the background buildings, functioning as an obstacle, notably influences the throttle actions of the well-trained policy in Fig. 6.13 (b). The saliency map of a under-trained agent produces highlight in different pixels of the heatmap in Fig. 6.13 (a). As focusing on the wrong objects, the agent generates non-optimal actions, leading to collisions during training. For the steer action, the open space and terrain have significant influence on this specific action generation in Fig. 6.13 (b). On the other hand, steering in Fig. 6.13 (a) is based on the sky, which is unreasonable.

During the robust model evaluation, it is noticed that the driving policy is more vulnerable to adversarial perturbations in certain positions along the trajectories. The generated saliency maps confirm these findings in Fig. 6.14. Two spots are pinpointed in the driving scenario that exhibit different potential vulnerabilities under identical perturbation strengths on both proposed model and the baseline model. When the agent is at the potentially vulnerable position 1 and the camera

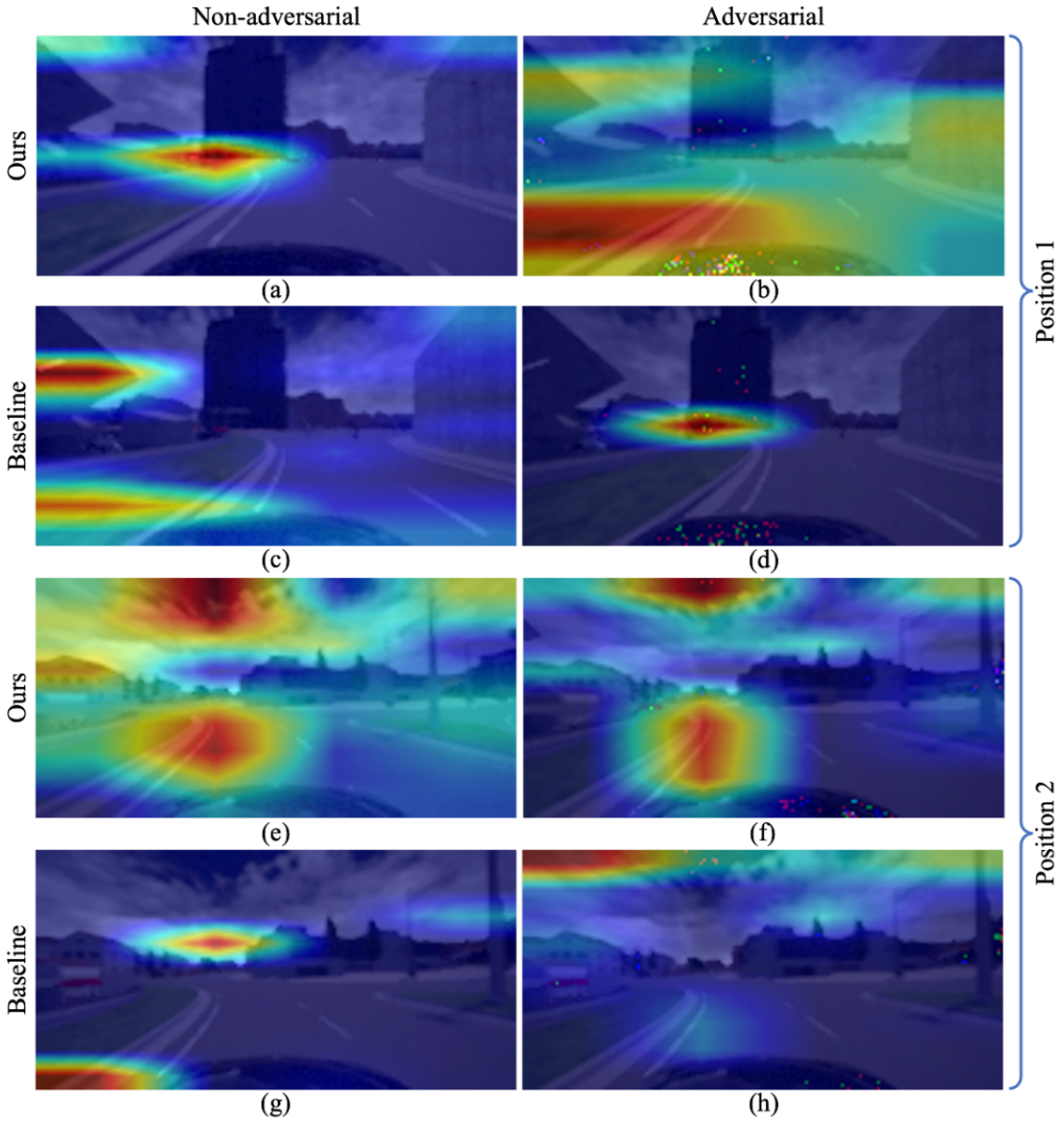

Figure 6.14 Steering saliency map comparisons between the proposed model and the baseline at two positions along the trajectories. Saliency maps under normal observations and same-strength adversarial attacked observations are displayed. [Wang et al.]

observations are shown in (a) and (c), the adversaries significantly alter the action generation logic shown in (b) and (d). Consequently, both agents output actions deviating in large margin from the optimal and thus potentially leading to collisions. On the other hand, when the robust agent is at position 2, the steering saliency maps (e) and (f) show very similar heatmaps, indicating that the actions generated by the proposed model are close enough whether the adversarial attacks are present

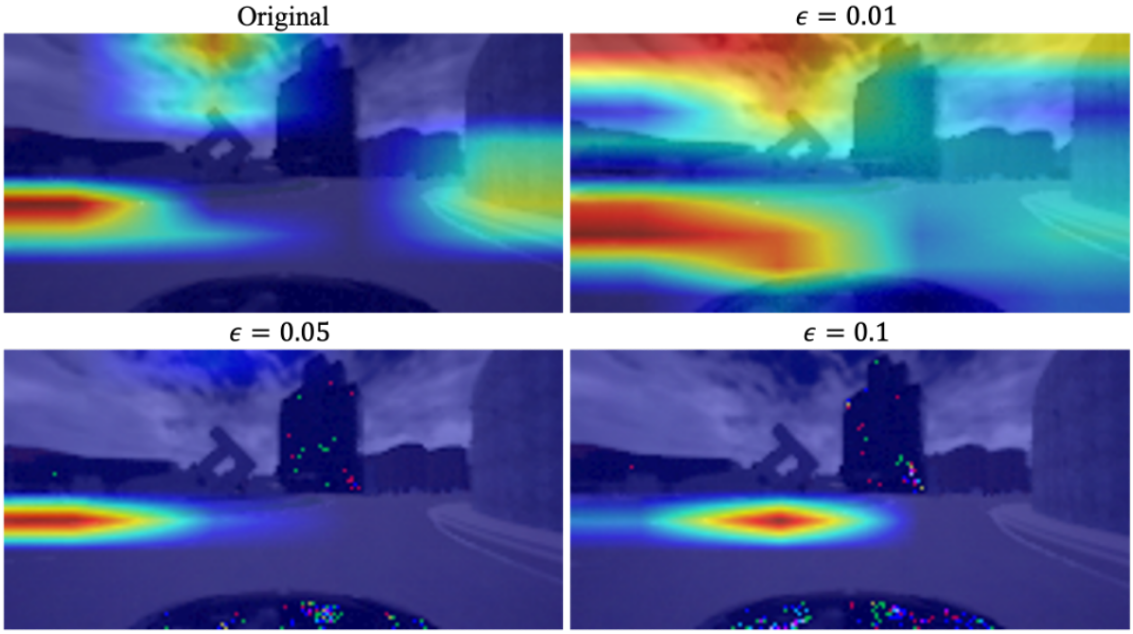

Figure 6.15 Steering saliency map comparisons with the same observation and various adversarial attack strength ε. [Wang et al.]

or not. Note that this is also the objective of the divergence-based optimisation term. In contrast, the decision-making process in the baseline model is significantly affected by the same perturbation strength at the same position, as shown in (g) and (h). This suggests that the position 2, considered invulnerable to the robust model remains potentially vulnerable to the baseline model due to the absence of the proposed robust optimisation training.

Fig. 6.15 demonstrates the changes to the decision-making process when the robust agent encounters adversarial attacks with different strengths. As expected, attack with $\varepsilon = 0.01$ produces very similar saliency map to the normal observation, implying that the agent stays resilient to this attack. For stronger attack with $\varepsilon = 0.1$, more changes in the saliency map can be seen. However, there is still similarity in the highlights, for example the left highlight keeps the same position. With the increase of ε to 0.1, the highlight part on the left starts to shift, signifying larger changes in the action's magnitude. Besides the changes in the heatmap, it is observed that isolated spots tend to appear predominantly on darker objects, such as the building and the hood of the ego vehicle in Fig. 6.15. These spots play the key role in distinguishing whether the observation is under adversarial attacks for the attack

detection network. The focus of driving agent on the input observation appears to be distracted, causing a spreading of the highlight of the heatmap onto other objects in the scene and finally luring the agent to the sub-optimal action generations.

6.5 Summary

Although in Chapter 5, a DRL driving agent is presented with reliable performance and proven generalisation capability, its robustness is limited to ideal and controlled environments, lacking a formalised approach to handle adversarial challenges. To address this limitation, in this chapter, a novel approach is introduced for deep reinforcement learning based robust autonomous driving against observation perturbations, along with an explainable adversarial detection model. Autonomous driving systems based on DRL are constantly challenged by safety-critical issues, which are not only from inevitable sensor failures and hardware limitations but also from the dynamic nature of real-world environments. These challenges are similar to adversarial attacks, which can degrade or compromise the ability of DRL agents to make precise and safe decisions in high-risk scenarios.

To approximate the worst-case perturbations that an autonomous driving agent may encounter, the proposed approach employs a white-box method. This method, which assumes full access to the internal parameters of the DRL agent, is developed to generate optimal adversaries. These adversaries act as training samples to strengthen the agent's robustness during reinforcement learning. The method is built upon FGSM, a widely used adversarial attack generation technique, and is enhanced with the novel collision risk maximum formulation, which ensures that the generated adversaries lead the DRL agents to collisions. Unlike traditional FGSM implementations that simply increase the attack strength by enlarging the ε parameter, the method uses iterative FGSM to apply repeated, making attacks harder to detect, thereby reducing the visibility of adversarial examples and simulating more realistic disturbances. Once the perturbed observations are generated, a Markov decision process (MDP) with perturbation is introduced to model decision-making in a complex urban environment, specifically focusing on roundabout navigation under constrained and uncertain

observational conditions. The agent must navigate this environment while managing the perturbations, aiming to learn an optimal policy that can adapt to real-world uncertainties. Then a Robust Proximal Policy Optimisation algorithm is presented to maximise the designed objective function with the constrained adversarial attacks to realise concrete robustness. Furthermore, a divergence-based optimisation term is introduced to mimic the performance gap between non-adversarial and adversarial states. Furthermore, the adversarial detection network is proposed. This network leverages explainable saliency maps, which provide visual insights into which parts of the input observations are influencing the decision-making process of the DRL agents. By highlighting these regions, the detection network offers transparency into the agent behaviour, making it easier to understand how adversarial attacks affect decision-making and enabling the reliability of the robust agents in safety-critical situations.

Extensive experiments have been conducted to thoroughly validate each component of this approach. Results reveal the notably high attack success rates achieved when the proposed attack method is applied to a well-trained baseline model. Additionally, the proposed robust agent exhibits resilience and robustness under strong perturbations across varied evaluation conditions. The detection model also demonstrates high accuracy in identifying attacks, providing insightful visualisations that help clarify the decision-making process under adversarial conditions. With single-agent robustness ensured, the next chapter, robustness in a more complex multi-agent system will be investigated.

Chapter 7

Multi-Agent Robustness Against Perturbation

In this chapter, the robustness of multi-agent deep reinforcement learning is investigated. A connected, cooperative multi-agent system is introduced to enhance the efficiency of cooperative tasks in ideal environments. However, the challenges of adversarial attacks escalate significantly in MARL systems compared to single-agent systems, due to the increased complexity of dynamics and information sharing. To solve this, this chapter follows the idea of constrained objective function introduced in chapter 6, and further adopt it to the multi-agent context with proposed safety criteria guarantee.

Associated Publications This chapter is based on the following submitted work:

Wang C, Aouf N. Robust Multi-Agent Reinforcement learning Against Adversarial Attacks for Cooperative Self-Driving Vehicles. Submitted to IET Radar, Sonar & Navigation.

Multi-Agent Deep Reinforcement Learning (MARL) for self-driving vehicles aims to address the complex challenges of coordinating multiple autonomous agents in shared road environments. MARL creates a more stable system and enhances vehicle performance in typical traffic scenarios compared to single-agent DRL systems. However, despite its sophisticated cooperative training, MARL remains vulnerable to unforeseen adversarial attacks. Perturbed observation states can lead one or more vehicles to make critical errors in decision-making, triggering chain reactions that often result in severe collisions and accidents. To ensure the safety and reliability of multi-agent autonomous driving systems, this chapter proposes a cooperative multi-agent proximal policy optimisation algorithm for self-driving vehicles, equipped with robust capabilities to handle strong and unpredictable adversarial attacks. Unlike most existing works, the proposed MARL framework employs a universal policy for each agent, realising a more practical, non-task-oriented policy network for real-world applications. In this way, it enables integrating shared observations with Mean-Field theory to model interactions within the MARL system. To further enhance robustness, a risk formulation and a risk estimation network are developed to minimise long-term risks while maximising the long-term rewards. This risk estimator is then used to construct a constrained optimisation objective function with a regulariser to maximise long-term rewards in worst-case scenarios. Experiments conducted in the CARLA simulator in intersection scenarios demonstrate that the proposed method remains robust against adversarial state perturbations while maintaining high performance, both with and without attacks.

7.1 Overview

In recent years, deep reinforcement learning has demonstrated promising decision-making capabilities for autonomous driving vehicles in various environments [134, 151, 152]. As it brings prospect of greater convenience, mobility efficiency, and safety to the automotive industry, an increasing number of self-driving vehicles will be deployed on roads in the near future. Research has shown that in complex cooperative tasks, Multi-Agent Reinforcement Learning (MARL) offers improved coordination in

collaboration, more efficient learning and overall performance compared to single-agent reinforcement learning [184, 185]. This is particularly relevant for applications such as autonomous driving, where the tasks require multiple agents to work together cooperatively and maintain awareness of the overall system rather than acting independently [186]. Although MARL has been extensively studied, challenges still persist in designing multi-agent systems. Existing works [187, 188] primarily train the decision-making process based on local observations. Even though these agents are trained within a multi-agent framework, they often lack a comprehensive awareness of individual contributions and the overall situation. Sunehag et al. [185] attempt to use global rewards to inform local agents about overall performance. Additionally, QMIX [189] employs a mixing network to enhance the accuracy of global Q-value estimation by incorporating global state awareness. However, this approach still neglects the individual contributions of each agent to the overall performance. The interactions each agent learns can be subtle, which means that centralised-training-decentralised-execution approaches may resemble multiple single agents working together, rather than fully achieving the potential of true multi-agent systems. Furthermore, in many multi-agent systems, each agent is designed to complete a specific task using its own policy network, meaning that an agent trained for one task cannot easily transfer to another. While this task-specific approach is necessary for some robotic applications, it does not suit autonomous driving. In this context, vehicles should operate uniformly on the road without requiring specialised training for each specific task.

Besides the fact that MARL methods have areas that could benefit from further development, they also face challenges as vulnerability to adversarial attacks [190]. These attackers are designed to affect the decision-making process of the MARL system. Given the high safety requirements of autonomous vehicles, it is important to develop a robust DRL agent or MARL system to prevent catastrophic failures due to perturbations in realistic environments. This robustness is crucial not only for defending against adversarial attacks but also for handling unavoidable sensor errors and natural equipment inaccuracies, which can impact the agents in a similar manner.

Recognizing the importance of model robustness, researches have investigated the single-agent DRL. Some works [154, 155] adopt adversarial training from supervised learning scheme to improve the robustness of such deep based guidance schemes. Specifically, the agent is occasionally attacked and the adversarial trajectories are generated during the data collection. However, simply training the agent with adversarial samples only brings limited improvement. Compared to these robust single-agent algorithms, enabling robustness for multi-agent scenarios faces more challenges, as not only the driving agents are influence by the adversarial attacks, but their behaviour after the attacks could affect other agents in the same tasks. Moreover, despite the improvements that communication brings to cooperative multi-agent autonomous driving under normal conditions, the shared observations can be perturbed, further compromising the agents that rely on this information. To address these problems, this chapter introduces a novel algorithm for robust cooperative multi-agent reinforcement learning based self-driving to solve the intersection passing scenarios against the observation adversarial attacks. The key contributions can be summarised as follows:

1. A cooperative MARL framework with communication is established. The communication and the universal policy network allow us to take account the interactions among agents based on Mean-Field theory in the training, thereby enhancing the MARL performance. Additionally, the universal policy indicates that each agent is not limited to a specific task. It ensures that one policy can manage all tasks after training.

2. A risk assessment formulation is defined to model both the system and individual risk levels at current state. Similar to Long-term rewards and the value network, the risks is minimised with a risk network at the same time. Moreover, the gap between system risks and individual risks is formulated as a credit assigning method, allowing the policy to be updated by accounting for each agent's contribution to the MARL system.

3. A robust proximal policy optimisation is proposed to obtain a robust policy for roundabout making under bounded optimal observation perturbations. A regulariser is used to solve the attacked information sharing.

Experiments are carried out to evaluate the performance of the proposed algorithm for intersection-passing tasks in the realistic unreal-engine powered simulator CARLA [49]. The results show the improvement of performance and robustness of the proposed adversarial defence method for MARL system.

7.2 Related Work

7.2.1 MARL for Autonomous Driving

Multi-agent deep reinforcement learning (MARL) aims to maximise team rewards, and several effective approaches have been developed. MADDPG [184] extends DDPG to multi-agent settings, where each agent has its own actor and critic networks, using global information to learn coordinated policies. MAAC (Multi-Agent Actor-Critic) [191] improves coordination by using attention mechanisms to prioritise relevant information from other agents, making it more efficient in complex environments. G2ANet [192] further enhances communication by incorporating graph attention networks to capture agent interactions dynamically. On the value-based side, QMIX uses a mixing network to combine individual Q-values into a global Q-value, ensuring a more accurate global Q-value estimation. QPD [193] decomposes the global Q-function into individual components for better scalability, while QPLEX [194] uses duplex duelling networks to model the interplay between agents. Mean-filed actor-critic method (MFAC) [195] applies the mean-field theory to MARL and thus successfully improves the scalability of MARL with a large number of agents.

Following the major breakthroughs of MARL in recent years, recent advancements in multi-agent deep reinforcement learning (MARL) have expanded the scope of autonomous driving systems by addressing complex interactions among multiple vehicles and agents. Unlike single-agent DRL, which focuses on optimising the performance of an individual vehicle, MARL considers the collaborative and competitive

dynamics in a shared driving environment [19, 196]. This approach is particularly useful for scenarios involving multiple autonomous vehicles that must navigate and negotiate their movements in real-time. MARL techniques can significantly enhance the coordination and cooperation among vehicles, improving overall traffic flow, safety, and efficiency. These methods allow vehicles to learn not only from their own experiences but also from the interactions with other agents, leading to more robust decision-making and adaptive behaviours. For instance, [162] introduces a cooperative control framework where autonomous vehicles use MARL to synchronise their movements and optimise lane merging and intersection crossing, resulting in smoother traffic management and reduced congestion. Furthermore, MARL can address the challenge of non-stationary environments where the behaviour of other agents is dynamic and uncertain. By leveraging multi-agent techniques, vehicles can develop strategies to anticipate and respond to the actions of neighbouring vehicles more effectively [163]. [164] presents a method that integrates MARL with communication protocols, allowing vehicles to share information about their intentions and local environment, thereby improving coordination and reducing the likelihood of accidents.

However, MARL systems face notable challenges, particularly regarding their vulnerability to adversarial attacks. The collaborative nature of MARL can make it more susceptible to disruptions caused by malicious agents or unexpected behaviours from other vehicles. Unlike single-agent systems, where robustness can be achieved through isolated adjustments, maintaining robustness in a multi-agent setting is more complex due to the interdependence among agents.

7.2.2 Adversarial Attacks on DRL

Though deep learning models recently achieve significant improvement, research shows that these well-trained models are still very vulnerable to adversarial attacks. Adversarial attacks on camera sensor often lead to visually similar images to the normal images from a human perspective, yet they can deceive deep learning models into generating inaccurate predictions. Generally there are two types of adversarial

attack methods, white-box attacks and black-box attacks, depending on if the attacker has full access to the models' parameters or not.

In [165], a Bayes optimisation based approach was proposed to generate the painting of black lines on the road to counterfeit lane lines and make the vehicle deviate from the original orientation. Experiments were conducted in CARLA simulator, and results showed that end-to-end driving models were attacked and deviated to the orientation chosen by attackers. He et al. [166] combined Bayesian optimisation and Jensen-Shannon (JS) divergence to measure average variation distance of the policies attacked by the observation perturbations for optimal black-box attacks. Behzadan and Munir [167] studied black-box attacks on DQNs with discrete actions via transferability of adversarial examples. Pattanaik et al. [168] further enhanced adversarial attacks to DRL with multi-step gradient descent and better engineered loss function. They required a critic or Q function to perform attacks. Typically, the critic network learned during agent training. For white-box approaches, Huang et al. [169] evaluated the robustness of deep reinforcement learning policies through an FGSM based attack on Atari games with discrete actions. Kos and Song [170] proposed to use the value function to guide adversarial perturbation search. Lin et al. [171] considered a more complicated case where the adversary is allowed to attack only a subset of time steps, and used a generative model to generate attack plans luring the agent to a designated target state. In this chapter, a FGSM based method is introduced to generate optimal adversary examples by maximizing the pre-defined collision risk. Results show that with a small strength parameter ε and minimal visual difference, the proposed method can efficiently misguide the well-trained agent.

7.2.3 Mitigation Against Adversarial Attacks

Defence methods against adversarial attacks have been explored recently. Zhang et al. [197] proposed a novel Markov decision process (SA-MDP) that considers state-adversarial perturbations, and provides a theoretical foundation for robust single-agent reinforcement learning. They developed the principle of policy regularisation that can possibly be applied to many DRL algorithms. Based on SA-MDP, Zhang

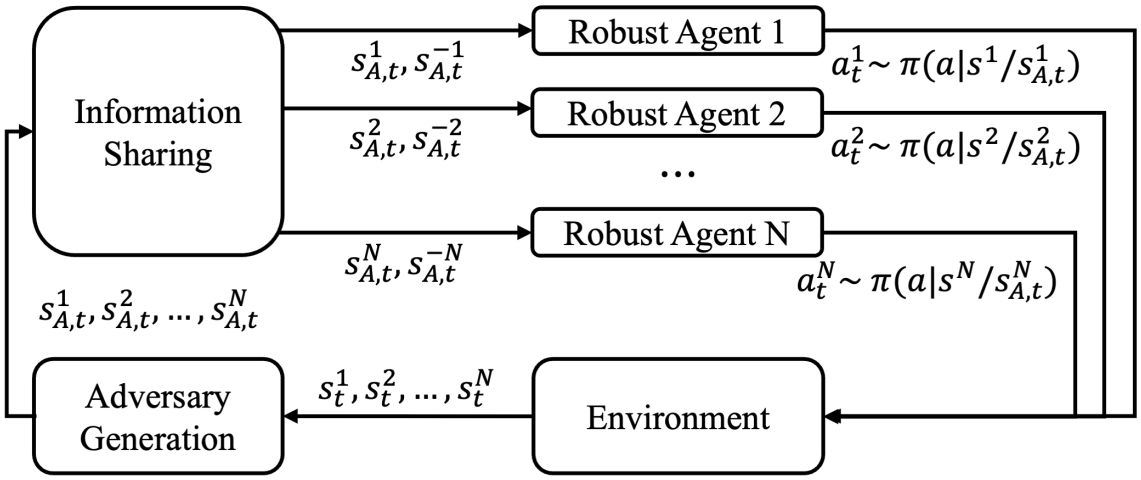

Figure 7.1 The framework overview. [Wang et al.]

et al. [198] proposed an alternate training framework with learned adversaries and developed a robust Markov game to address environmental uncertainty by introducing uncertainty into the reward function.. Oikarinen et al. [199] proposed the robust ADversarIAl loss (RADIALRL) method, which can improve the robustness of DRL under the ℓ_p norm boundary against attacks with lower computational complexity. Kumar et al. [200] proposed certified robustness by adding smoothing noise to the state. However, these methods are designed for single-agent RL systems and overlook the specific challenges of MARL, making them difficult to apply effectively. MARL systems are often more vulnerable to adversarial attacks, even when only a single agent is targeted [201]. To counter state-based attacks, Zhou et al. [181] proposed robust policies by minimising the cross-entropy loss between the actions of agents in non-perturbed and perturbed states. Compared to previous methods, our work focuses on mitigating perturbations in both local agent states and global shared states, addressing a more challenging problem than prior approaches.

7.3 Methodology

7.3.1 Framework Overview

The system overview is shown in Fig. 7.1. The cooperative MARL system includes N agents with an information sharing module and an adversarial attacker. During infer-

ence, the environment generates a tuple of local camera observations $(s_t^1, s_t^2, \cdots, s_t^N)$. The adversarial attacks are then applied on the clean states, resulting in a new perturbed tuple $(s_{A,t}^1, s_{A,t}^2, \cdots, s_{A,t}^N)$. Through the information sharing scheme for the connected multi-agent system, the observations will be processed as the perturbed local observation and the mean of all neighbor observations for each individual agent. All agents share a non-task-oriented, universal policy network with the exact same parameters. Based on the local observation and the shared observation (which could be perturbed or not), each agent outputs the actions and execute the actions in the environment. As previously discussed, developing a robust multi-agent system poses significant challenges, particularly due to the uncertainty regarding the number of agents that may be compromised by adversarial attacks. Additionally, while information sharing here enhances decision-making in non-perturbed contexts by incorporating diverse perspectives from neighboring agents, it can have a backwards impact when the shared observations themselves are exposed to attacks, leading to more erroneous decision-making.

In this section, the existing problems are addressed and a robust cooperative deep adversarial reinforcement learning approach is proposed for autonomous driving agents against strong observation perturbations.

7.3.2 Mean-Field Communicated Multi-Agent Structure

A multi-agent deep reinforcement learning task can be considered as a continuous decision-making problem, which follows the Stochastic Games (SG) [202]. SG is defined as a tuple $(\mathcal{S}, \mathcal{A}^1, \cdots, \mathcal{A}^N, \mathcal{R}^1, \cdots, \mathcal{R}^N, P, \gamma)$. $\mathcal{N}$ is the number of the agents, $\mathcal{A}^j$ is the action space of agent j, $\mathcal{R}^j$ is the step reward of agent j. The agents interact with the environment of successive joint state $\mathcal{S} : s^1 \times \cdots \times s^n$ with a joint action, getting the step rewards $(r^1 \in \mathcal{R}^1, \ldots, r^N \in \mathcal{R}^N)$ and the transition probability $p \in \mathcal{P}$ to the next joint state under the current joint state and the joint action. γ indicates the discount factor. Considering the state perturbations, the worst case perturbed joint state and the perturbed action responded to it $(\mathcal{S}_A, \mathcal{A}_A^j)$ are introduced to the original SG tuple. The goal of the robust multi-agent deep reinforcement learning is

to find a series of optimal policies that return the maximum accumulative discounted team returns under the worst adversarial attacked states:

$$Q^i_{\pi^*} = \max_{\pi} \min_{\mathcal{S}_\mathcal{A}} \left[\mathbb{E} \left(\sum_t^T \gamma^t r^i_t \left(s^i_t / s^i_{A,t}, a^i_t / a^i_{A,t} \right) \right) \right] \tag{7.1}$$

The attacked state s_A is a shifted state, which models the worst case perturbation due to attacks on the sensor leading to a_A sub-optimal than a^*. A well-trained guidance policy network may be able to cope with a weak and quick perturbation, turning back to the actual actions and the desired trajectory after the state observations get back to normal. However under strong and continuous adversarial attacks in the context of multi-agent, the guidance policy networks can easily fail.

One problem for multi-agent reinforcement learning is the difficulty to model the interactions among the agents. Mean-field actor-critic reinforcement learning (MFAC) uses the mean-field theory to transform the interaction of multiple agents into the interaction between two agents, which makes large-scale multi-agent reinforcement learning become possible. In MFAC, the long-term expected Q value for agent j at state s with the joint action a, $Q^i(s, a)$ is decomposed to the sum of Qs when interacting with each agents:

$$Q^i(s, a) = \frac{1}{N-1} \sum_{k \subseteq N-1} Q^i \left(s, a^i, a^k \right) \tag{7.2}$$

where the a^k counts as the local interaction between agent j with the neighbor agent k. In this case, a universal policy network is used for every agent in the MARL, to enable a universal driving agent that works in every task in the intersection scenario. Therefore, the decomposition becomes:

$$Q^i(s, a) = \frac{1}{N-1} \sum_{k \subseteq N-1} Q^i \left(s^i, a^i, s^k \right) \tag{7.3}$$

As a universal policy is used, the actions generated by different agents only depend on the observation. The action of neighbor can be replaced by the state of the

neighbor. It is proved by MFRL:

$$Q^i(s,a) \approx Q^i\left(s^i, a^i, \bar{s}^k\right) \tag{7.4}$$

where s^k is the mean state or fusion state of all neighbor agents. The local agent j makes decision based on the local observation and the information shared by other agents, which is processed through two FC layers and an attention module.

7.3.3 Gradient-based Attacker

In this section, the inner minimisation part of Equation (7.1) is solved by modelling the optimal adversarial attacks on observations based on a white-box technique. An adversarial attack generation method is required that prioritises ease of implementation and computational efficiency. The Fast Gradient Sign Method (FGSM), introduced by Goodfellow et al [179], fulfills these criteria. Additionally, FGSM offers flexibility, allowing for the application of techniques like Basic Iterative Method (BIM) [180] to introduce iterations and enhance the adversarial attack instead of increasing the perturbation parameter ε.

FGSM works by using the gradients of the neural network to create an adversarial example. For deep reinforcement learning, the method uses the gradient $\bigtriangledown_s L(\pi, s, a)$ of the loss with respect to the input observation to create a new observation that maximises this loss. The π refers to the policy network parameters, while the a is the output action. The new masked observation is obtained by:

$$s_A = s + \varepsilon \times sign\left(\bigtriangledown_s L(\pi, s, a)\right) \tag{7.5}$$

This new observation s_A is called the adversarial state. The ε parameter controls the strength of the attack, and it varies from 0 to 1. The closer the value to 1, the stronger effect on the targeted policy network, but this also makes s_A more visually distinguishable. On the contrary, if the ε is small, its impact is weaker, and it becomes harder for human eyes to detect differences from the normal state. As discussed in the beginning of the section, to solve the inner minimisation, the

generated perturbations should lead the policy to worst case situations. In the proposed cooperative MARL system, an unintended-action-leading loss function is introduced instead of directly minimising the long-term rewards. Within the realm of autonomous driving, certain throttle and steering actions in certain scenarios may pose potential danger. Based on this idea, the gradient-based perturbations are introduced into the control system by maximising the throttle and reversing the steering, thereby deliberately maximising the risk of collisions. After introducing the augmented FGSM, the agent will be encouraged to accelerate and make wrong steering to make the collision.

The problem of using FGSM is the effective sensitivity to ε. In practice, a larger ε is needed to make the perturbation effective on the deep guidance model decision. However during training, this will make perturbations predictable and tolerated by the model [155]. The BIM is employed to intentionally escalate the attack effeteness associated with autonomous driving. The BIM iterates the process of a single FGSM, which means a previous generated adversarial state will be the input of the next adversary generation. By iterating FGSM, the influence of the attacks can be increased without affecting the sensory image as much as by increasing the ε by a comparable amount. Finally, with the risk encouragement and BIM, the optimal adversary is obtained as:

$$\begin{aligned} s_A^i &= s_n^i + \varepsilon \times sign\left(\bigtriangledown_{s_n^i}\left(L_n^i\right)\right) \\ s_n^i &= s_{n-1}^i + \varepsilon \times sign\left(\bigtriangledown_{s_{n-1}^i}\left(L_{n-1}^i\right)\right) \\ &\dots \end{aligned} \tag{7.6}$$

where

$$L_n^i = \left(-steer^i, 1\right) - \left(\pi \mid s_n^i\right) \tag{7.7}$$

s_A^i indicates the optimal adversary, and n is the iteration number. Note that the attacker applies perturbations only on the targeted agent's observation, which is more realistic for the attacker to aim on the camera of that agent. The shared

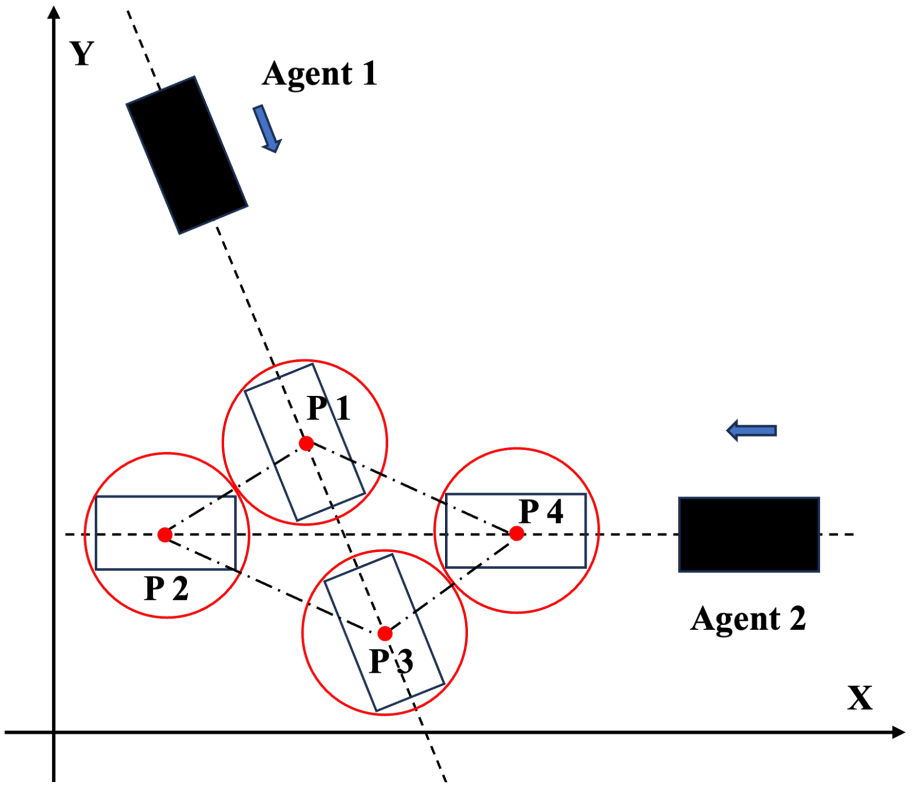

Figure 7.2 The illustration of the area (rhombus) that could lead to collision between two agents. [Wang et al.]

information could be perturbed through the cameras of neighbour agents, interfering with the decision-making on the receiving end. These attacks will lead the model to the worst actions in most of the cases.

7.3.4 Risk Assessment Formulation

To enhance the robustness of the MARL algorithm, a safety criteria is also introduced. The risks in the multi-agent framework are quantified as the collision probability of any two agents in the whole system. Autonomous driving reinforcement agents need to not only maximise the rewards and finish the tasks, but also be aware and reduce the risks during the navigation, which is particularly crucial for multi-agent self.driving. To simplify the risk analysis and disregarding the varying dimensions and shapes of individual vehicles, each vehicle is represented as a circle with a diameter equal to its diagonal. As shown in Fig. 7.2, considering two agents running towards each other with current speed v_1, v_2 and rotation to the world coordinate θ_1, θ_2, at

world locations $(x_1, y_1), (x_2, y_2)$. Assume the two agents maintain the current speed and rotation, they will meet at the intersection of their trajectories. If sliding the two circles representing the two vehicles along their respective trajectories (the dotted lines), the first point of collision is identified when agent 1 is at $P1$ and agent 2 is at $P2$, and so is the last point of contact when agent 1 is at $P3$ and agent 2 is at $P4$. This implies that between the moments when agent 2 passes $P4$ and leaves $P2$, while agent 1 remains between $P1$ and $P3$, there is a highly chance of collision between the two agents. Given the orientations, diagonals, and locations of the two agents, the coordinates of the four points $P1, P2, P3$ and $P4$ can be determined. Based on the agents' current speeds, the times associated with these positions are as follows: t_0 represents current time, t_1 is the time when agent 1 reaches P_1, t_3 is the time when agent 1 leaves P_3, t_4 is the time when agent 2 reaches P_4, t_2 is the time when agent 2 leaves P_2. The risk probability is then defined as:

$$P_{risk} = \frac{(t_3 - t_1) \cap (t_2 - t_4)}{(t_3 - t_1) \cup (t_2 - t_4)} \tag{7.8}$$

$P_{risk} \subseteq [0, 1]$ indicates the ratio of the overlap in the time intervals when both agents are within the collision zone to the total time interval between the moment the first agent entering the zone and the last agent leaving the zone. To calculate the overall risk in a MARL system, where P_{risk} describes the risk between two agents:

$$P_{tot} = 1 - \prod_{i=1}^{N-1} (1 - P_{risk,i}) \tag{7.9}$$

where N is the number of agents and $P_{tot} \subseteq [0, 1]$.

After the careful design of the risk evaluation, it is then embedded to the robust MARL.

7.3.5 Constrained Robust Cooperative-MARL

In this section, a robust multi-agent proximal policy optimisation is introduced with risk assessment integrated to solve the outer maximisation in Equation (7.1). Proximal Policy optimisation (PPO) is extended to multi-agent context, where each

agent learns a policy while coordinating with other agents, leveraging a centralised training with decentralised execution structure:

$$\mathcal{L}^i_{clip}(\pi) = \hat{E}_t \left[\sum_{i=1}^{N} min \left(\left(\frac{\pi\left(a^i|s^i\right)}{\pi^{'}\left(a^i|s^i\right)}, clip\left(\frac{\pi\left(a^i|s^i\right)}{\pi^{'}\left(a^i|s^i\right)}, 1-\varepsilon, 1+\varepsilon \right) \right) \hat{A}^i_t \right) \right] \tag{7.10}$$

Where π is the new policy parameter of the deep guidance agents, and $\pi^{'}$ stands for the old policy. N is the number of Agent. In this case, π is the universal policy parameters. $\hat{E}_t$ denotes the expectation over $t.$, while ε stands for the constant clip term that limits the policy update by constraining the ratio within a specified range, preventing excessively large updates. $\hat{A}_t$ is the estimated advantage value, obtained by the difference between the observed reward for taking an action a at a state s and the expected value predicted by the critic network at a state s:

$$\hat{A}^i_t = Q^i_t\left(s^i, a^i, s^{-k}\right) - V^i_t\left(s^i, s^{-k}\right) \tag{7.11}$$

Long-Term Risk Minimisation

The objective of equation 10 aims to maximise long-term expected returns. Similar to the long-term accumulative expected returns, based on the risk assessment, a risk objective can also be designed to minimise the long-term expected risks for the agent safety in the MARL. At every step, the risks of the MARL system is evaluated, and at the end of the episode, the overall risk is computed as the discounted cumulative risks. In order to fit in the maximisation form as the returns, $(1 - risk) \in [0, 1]$ is used as the safety term, and the safety-to-go at t is defined as:

$$S^i_t = \sum_{j=0}^{T-t} (\gamma\lambda)^j \left(\left(1 - P_{tot}\left(s^i_t, s^{-k}_t\right)\right) + \gamma S_\theta\left(s^i_{t+1}, s^{-k}_{t+1}\right) \right) \tag{7.12}$$

Where λ is the Generalised Advantage Estimation (GAE) [203] parameter used to control the bias-variance trade-off. Maximising the S_{target} is equivalent to minimising the risks. Similar to the critic network, the risk network S_θ will be iteratively updated to achieve accurate long-term expected safety prediction by Mean Square

Error (MSE):

$$\mathcal{L}^i_{risk} = \frac{1}{T}\sum_{t=0}^{T}\left(S^i_\theta\left(s^i_t, s^{-k}_t\right) - S^i\left(s^i_t, s^{-k}_t\right)\right)^2 \tag{7.13}$$

For policy optimisation, in addition to the advantage value obtained from equation 11, a risk advantage function plus single agent contribution is considered:

$$\hat{A}^i_{fi,t} = \hat{A}^i_t + \left(S^i - S^i_\theta\right) + Credict^i \tag{7.14}$$

Where $Credict^i$ is the weighted risk advantage value, which reflects the importance of agent i within the MARL system or the contribution of agent i to the system's overall performance. Credit assigning is crucial in cooperative MARL training, as it ensures individual actions are rewarded or penalised appropriately for the overall performance. Instead of only assigning the team reward uniformly to every agent, a risk-value-based method is introduced to enable non-linearity in credit assigning. The credit is defined as the difference between the total risk of the MARL system when agent i is included and excluded. :

$$Credict^i = \prod_{j=1,j\neq i}^{N-1}(1 - P_{risk,j}) - \prod_{j=1}^{N-1}(1 - P_{risk,j}) \tag{7.15}$$

By replacing $\hat{A}^i_t$ with $\hat{A}^i_{final,t}$ in Equation (7.10), the long-term expected risk is minimised efficiently in the C-MARL context.

Adversarial Regulariser

The proposed stochastic universal policy network outputs probability distributions of the predicted actions. A multivariate Beta distribution is employed as the policy output to ensure a bounded action space. The actions are randomly sampled from this distribution to increase exploration in training phase, and the mean value is chosen for inference phase. A Robust cooperative MARL system should behave similarly or close enough under perturbed observations and normal observations, which means the predicted action distributions with minimal divergence. The theorem in [82] is extended to the cooperative MARL scheme with universal policy: Given a policy π

and its value function $V_\pi(s^i, s^{-k})$ and considering the worst perturbation situation where all agents are attacked with optimal perturbation s^i_A, for all $s^i, s^{-k} \in \mathcal{S}$ and the corresponding $s^i_A, s^{-k}_A \in \mathcal{S}_\mathcal{A}$, it holds:

$$max\left(V_\pi\left(s^i, s^{-k}\right) - V_\pi\left(s^i_A, s^{-k}_A\right)\right) \leq \alpha max\left(D_{TV}\left(\pi\left(s^i, s^{-k}\right), \pi\left(s^i_A, s^{-k}_A\right)\right)\right) \quad (7.16)$$

where $D_{TV}(\pi(s^i, s^{-k}), \pi(s^i_A, s^{-k}_A))$ is the total variation distance between the predicted action distributions when all the agents are attacked and attack-free. This is the largest possible difference between the probabilities that the two probability distributions can assign to the same action. According to Pinsker's inequality [182], D_{TV} can be linked to another distance Kullback–Leibler (KL) divergence [183]:

$$D_{TV}\left(\pi\left(s^i, s^{-k}\right), \pi\left(s^i_A, s^{-k}_A\right)\right) \leq \sqrt{\frac{1}{2} D_{KL}\left(\pi\left(s^i, s^{-k}\right), \pi\left(s^i_A, s^{-k}_A\right)\right)} \quad (7.17)$$

which is more computationally efficient compared to the integral operation in D_{TV} as discussed in the previous chapter. Therefore, Equation (7.16) with the KL divergence to:

$$max\left(V_\pi\left(s^i, s^{-k}\right) - V_\pi\left(s^i_A, s^{-k}_A\right)\right) \leq max\sqrt{\frac{1}{2} D_{KL}\left(\pi\left(s^i, s^{-k}\right), \pi\left(s^i_A, s^{-k}_A\right)\right)} \quad (7.18)$$

Which means the minimisation on KL divergence of the action probabilities under non-perturbed and attacked states guarantees the minimisation on the total variation distance D_{TV}, therefore the performance gap could be minimised too. The regulariser will be added to the final objective function to update the policy network:

$$\mathcal{L}^i_{reg,t} = \sqrt{\frac{1}{2} D_{KL}\left(\pi\left(s^i_t, s^{-k}_t\right), \pi\left(s^i_{A,t}, s^{-k}_{A,t}\right)\right)} \quad (7.19)$$

Constrained Objective Function

Consider a well-trained cooperative MARL system. For every state $\mathcal{S}$ in all possible trajectories, there exists a subset actions $\mathcal{A}^i_{robust} \in \mathcal{A}$ under bounded worst-case state perturbations that leads to $\mathcal{S}^i_{safe} \in \mathcal{S}$ avoiding high-risk states for agent i. Therefore,

it is possible to guarantee a robust policy by constraining the policy to output safe trajectories and the worst case perturbations. Intuitively, there are two constraints in the objective function to design: one for a bounded adversarial perturbation and one for safe states.

Control Barrier Function (CBF) is widely deployed in autonomous driving applications for ensuring safety by providing mathematical guarantees that the vehicle will avoid unsafe situations, such as collisions, while respecting constraints like speed limits and obstacle avoidance. CBF enables real-time adaptability and seamless integration with existing control systems, ensuring safe and reliable navigation in dynamic environments. The CBF of the cooperative MARL is defined as the risk value network $h(s^i, s^{-k}) = S_\theta(s^i, s^{-k})$. The calculation of the proposed CBF is based on the non-perturbed states only, as the actual states of the agents are needed for the actual risk estimation. The safety set of states is then defined as:

$$\mathcal{C} = \left\{ s^i \in \mathcal{S} : h\left(s^i, s^{-k}\right) \geq \varepsilon \right\} \tag{7.20}$$

ϵ is the safety criteria, and the system dynamics is defined as:

$$\dot{s}^i = f(s^i) \tag{7.21}$$

As a model-free method, the agent learns a policy directly through the environment interactions without knowing or modelling the environment's dynamics. This presents a significant challenge when it comes to predicting how an action will transition the system from one state to the next. Based on [204], the continuous states of a well-trained model are Lipschitz-continuous. Therefore, the system dynamics could be approximated as the state difference over the time step:

$$\dot{s}^i = \frac{s^i_{t+1} - s^i_t}{t} \tag{7.22}$$

To ensure the states stay in the safety set, the following equations needs to be satisfied:

$$\begin{aligned} \frac{d}{dt} h\left(s^i, s^{-k}\right) &\geq -\omega\left(h\left(s^i, s^{-k}\right) - \epsilon\right) \\ \Rightarrow \nabla_{s^i} h \cdot \frac{s^i_{t+1} - s^i_t}{t} &+ \omega\left(h\left(s^i, s^{-k}\right) - \epsilon\right) \geq 0 \end{aligned} \tag{7.23}$$

A linear $\omega > 0$ is used to make sure $h(s^i, s^{-k})$ can be updated to satisfy the criteria. For the bounded worst-case perturbations, let g be the attacker, the L_2 norm of $(g(s^i, \pi) - s^i)$ is bounded by μ.

Then the objective function of our proposed robust cooperative MARL is developed. Combining maximising the long-term expected rewards, minimising the regulariser, subjecting to the constraints, the following optimisation problem is presented:

$$\begin{aligned} \underset{\pi}{min} &\left\{ \frac{1}{T}\frac{1}{N} \sum_{t=0}^{T} \sum_{i=0}^{N} \left(\mathcal{L}^i_{fi}(\pi) + \mathcal{L}^i_{reg,t}\right) \right\} \\ subject \quad to \qquad & \mu - L_2\left(f\left(s^i_t, \pi\right), s^i_t\right) \geq 0 \\ \nabla_{s^i} h \cdot \frac{s^i_{t+1} - s^i_t}{t} &+ \omega\left(h\left(s^i_t, s^{-k}_t\right) - \epsilon\right) \geq 0 \end{aligned} \tag{7.24}$$

Where t and γ denote the time step and discount factor respectively. And $\mathcal{L}^i_{fi}(\pi)$ is the enhanced PPO objective for multi-agent systems:

$$\mathcal{L}^i_{fi}(\pi) = min\left(\left(\frac{\pi\left(a^i|s^i\right)}{\pi'\left(a^i|s^i\right)}, clip\left(\frac{\pi\left(a^i|s^i\right)}{\pi'\left(a^i|s^i\right)}, 1-\varepsilon, 1+\varepsilon\right)\right)\hat{A}^i_{fi,t}\right) \tag{7.25}$$

Based on the Lagrange multiplier technique, the generalised Lagrange function is obtained:

$$\mathcal{L}(\pi, \alpha, \beta) = \frac{1}{T}\frac{1}{N} \sum_{t=0}^{T} \sum_{i=0}^{N} \left(\mathcal{L}^i_{fi}(\pi) - \mathcal{L}^i_{reg,t} - \alpha C_1 - \beta C_2\right) \tag{7.26}$$

where $\alpha \geq 0, \beta \geq 0$ are the Lagrange multipliers, and C_1, C_2 are the first and the second constraints respectively. Now a new function $\theta_P(\alpha, \beta)$ with respect to α, β is

defined as:

$$\theta_P(\alpha, \beta) = \max_{\alpha \geq 0, \beta \geq 0} \mathcal{L}(\pi, \alpha, \beta) \tag{7.27}$$

If π satisfies the constrained condition in Equation (7.24), then:

$$\theta_P(\alpha, \beta) = \frac{1}{T}\frac{1}{N}\sum_{t=0}^{T}\sum_{i=0}^{N}(\mathcal{L}_{fi}^{i}(\pi) + \mathcal{L}_{reg,t}^{i}) \tag{7.28}$$

otherwise $\theta_P(\alpha, \beta) = +\infty$. This indicates that minimising the newly defined function $\min_{\pi}\theta_P(\alpha, \beta)$ is equivalent to solving the primal problem in Equation (7.24). If the constraints are not satisfied, the penalty becomes extremely large, which forces the policy π to comply with the constraints. The primal optimisation problem can be further transferred as:

$$\min_{\pi}\theta_P(\pi) = \min_{\pi}\max_{\alpha \geq 0, \beta \geq 0} \mathcal{L}(\pi, \alpha, \beta) \tag{7.29}$$

Furthermore, based on equation Lagrange duality, the dual problem of Equation (7.29) will always have smaller value than the primal problem, which leads to smaller loss and better performance. With the generalised Lagrange function $\mathcal{L}$, $\forall \pi, \alpha, \beta$:

$$\begin{aligned} \min_{\pi}\mathcal{L}(\pi, \alpha, \beta) \leq \mathcal{L}(\pi, \alpha, \beta) &\leq \max_{\alpha \geq 0, \beta \geq 0} \mathcal{L}(\pi, \alpha, \beta) \\ \Rightarrow \min_{\pi}\mathcal{L}(\pi, \alpha, \beta) &\leq \max_{\alpha \geq 0, \beta \geq 0} \mathcal{L}(\pi, \alpha, \beta) \\ \Rightarrow \max_{\alpha \geq 0, \beta \geq 0}\min_{\pi}\mathcal{L}(\pi, \alpha, \beta) &\leq \min_{\pi}\max_{\alpha \geq 0, \beta \geq 0} \mathcal{L}(\pi, \alpha, \beta) \end{aligned} \tag{7.30}$$

Therefore, it is easier to solve the dual problem than the primal problem. This optimisation is detailed by the two following steps:

$$\begin{aligned} &1. \quad \min_{\pi}\mathcal{L}(\pi, \alpha, \beta) \\ &2. \quad \max_{\alpha \geq 0, \beta \geq 0} \mathcal{L}(\pi, \alpha, \beta) \end{aligned} \tag{7.31}$$

First, the Lagrange multipliers α, β are fixed and the universal policy π is updated

Algorithm 2 R-CCMARL

Initialise the universal policy network parameter θ_0, the state-value critic network parameter ϕ_0, the state-risk network parameter ψ_0, the replay buffer $\mathcal{D}$;
Initialise the Lagrange multiplier α;
for $j = 0, 1, 2, ...$ **do**
  for $t = 0, 1, 2, ...$ **do**
    For each agent, share its non-perturbed observation and receive non-perturbed observations from all the neighbors.
    Sample action $a_t^i \sim \pi(\theta_j)(a_t^i|s_t^i, s_t^{-k})$ based on the policy $\pi(\theta_j)$ and state (s_t^i, s_t^{-k})
    Generate optimal adversary $s_{A,t}^i$ through equation 6.
    Obtain transitions $\{s_{t+1}^i, r_t^i, P_{tot,t}^i\}$ by executing a_t^i.
    Store the transitions $\{s_t^i, a_t^i, r_t^i, P_{tot,t}^i, s_{t+1}^i, s_{A,t}^i\}$ in $\mathcal{D}_j$.
  end for
  Calculate the reward-to-go $\hat{R}_t^i$.
  Calculate the safety-to-go $\hat{S}_t^i$.
  Calculate estimated advantage $\hat{A}_{fi,t}^i$ through equation 14.
  Calculate the regulariser $\mathcal{L}_{reg,t}^i$.
  Update the universal policy θ by equation 33.
  Update the Lagrange multiplier α and β by equation 34.
  Update the state-value critic network parameter ϕ.
  Update the state-risk network parameter ψ by equation 13.
end for

based on step 1, which is represented as minimising the $\mathcal{L}(\pi, \alpha, \beta)$:

$$\min_{\pi} \left\{ \frac{1}{T} \frac{1}{N} \sum_{t=0}^{T} \sum_{i=0}^{N} \left(\mathcal{L}_{fi}^i(\pi) + \mathcal{L}_{reg,t}^i - \alpha C_1 - \beta C_2 \right) \right\} \tag{7.32}$$

Then conversely, the updated policy π are fixed for the Lagrange multiplier α and β to be updated based on step 2:

$$\begin{aligned} \max_{\alpha \geq 0, \beta \geq 0} \Bigg\{ \frac{1}{T} \frac{1}{N} \sum_{t=0}^{T} \sum_{i=0}^{N} (\Big(& -\alpha(\mu - L_2 \left(f \left(s_t^i, \pi \right), s_t^i \right) \Big) \\ & -\beta \left(\nabla_{s^i} h \cdot \frac{s_{t+1}^i - s_t^i}{t} + \omega \left(h \left(s_t^i, s_t^{-k} \right) - \epsilon \right) \right)) \Bigg\} \end{aligned} \tag{7.33}$$

Finally, the original constrained minimisation problem becomes a maxmin problem without constraints. Gradient descent is used to find the optimal robust universal policy π and the optimal Lagrange multipliers α, β repeatedly through Equation (7.32) and Equation (7.33) in each batch update. The objective function is a jointly

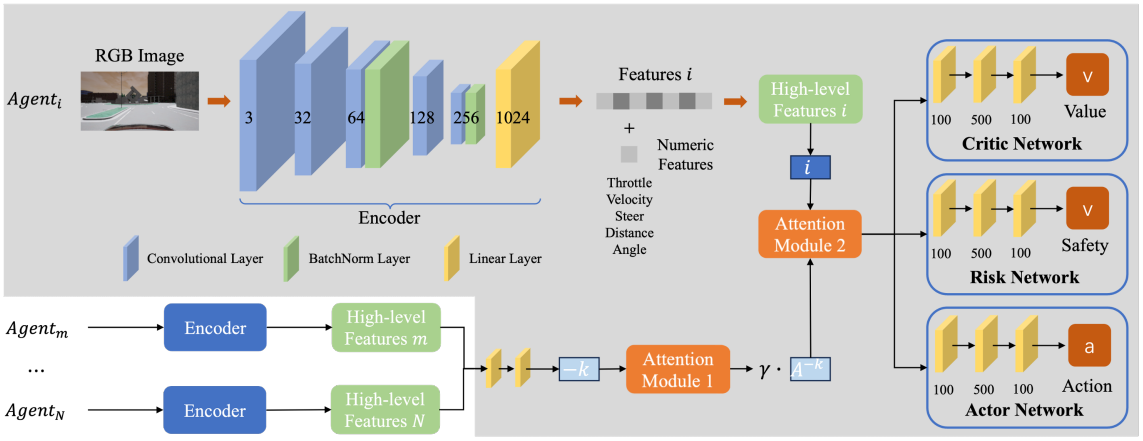

Figure 7.3 Network structure of the proposed robust cooperative communicated multi-agent DRL model. A universal driving model is applied, consisted of a feature extract encoder, an actor, critic, risk network and two fully-connected layers and two attention modules. The grey area indicates all the components in one agent. The three sub-task networks are fed with the enhanced features, which involves the local information the shared information from the neighbour agents. [Wang et al.]

optimisation problem, where the policy network balances the performance under normal observation and the adversarial defence capability. In the experiments, the policy will learn to satisfy the constraints in Equation (7.24) and resulting two constraint functions to decrease during the gradient descent process, and α, β in step 2 will gradually approach zero. Consequently, step 1 will find the minimum after α and β becomes stable. The method R-CCMARL is detailed in Algorithm 1.

7.3.6 Network Structure

The proposed robust cooperative communicated multi-agent reinforcement learning network structure is illustrated in Fig. 7.3. At each time step, agent i takes a monocular RGB image as input, which is passed through an encoder to extract essential visual features. These features are then concatenated with five additional numeric features that represent other important state information, including throttle, velocity, steer, distance to the road centre, angle between the vehicle vector and the waypoint vector. This concatenation of data results in a high-level feature map i. Simultaneously, each neighbouring agent extracts high-level features using the same encoder with identical parameters. The high-level features of all neighbours are concatenated and passed through two fully-connected layers, producing features

$-k$ of the same shape as feature map i. Next, the local features of agent i and the shared features of its neighbours are fused by two attention modules to create a more comprehensive representation of the environment.

Specifically, considering the ease of implementation, both attention modules uses the standard non-local block as previously mentioned in chapter 3. For attention module 1, it takes the mean observation $-k$ as input, and generates the learnable parameter weight attention map $\gamma \cdot A^{-k}$. This module omits the element-wise summation between $-k$ and A^{-k}. Next, local high-level features i is concatenated with $\gamma \cdot A^{-k}$ and passed to attention module 2, where the summation operation is retained.

Finally, after processing through the two attention modules, the final enhanced feature map is generated, which is then passed to the three separate sub-networks. These sub-networks are responsible for predicting three key outputs for agent i: the long-term expected returns, the long-term expected safety values, and the optimal actions. These outputs guide the agent in making safe and effective decisions within the multi-agent environment, ensuring robustness against state perturbations.

7.4 Experiments

The training and testing of the proposed framework are implemented in CARLA simulator. CARLA is an open-source platform for development, training, and validation of autonomous driving systems. It has a rich library of vehicle models and realistic urban road modelling, hence being near ideal to urban driving simulation. The experiment scene is set up in Town 02 within the CARLA map library.

7.4.1 Implementation Details

In the experiment, three vehicles are included in the multi-agent system to complete the intersection-passing task. As shown in Fig. 7.4, the three vehicles are spawned in different directions at a T-shaped intersection, starting randomly in the designated spawning areas (green areas) to increase uncertainty. Agent 1 aims to go straight, while both Agent 2 and Agent 3 make left turns, creating a highly collision-possible

Figure 7.4 Experiment scenarios with two different intersections. Green area are the starts, and the vehicles are meant to follow the blue trajectories. [Wang et al.]

situation. The experiments are conducted at two different intersections to introduce randomness. State-of-the-art multi-agent method is compared with the proposed algorithm and a variant method, referred as the risk-only model, in which the constraint for robust cooperative MARL is not implemented.

Each agent takes a 160*80 RGB image from a monocular camera and 5 numeric features as fusion inputs: throttle, velocity, steer, the distance between the vehicle and the road centres, and the angle between the vehicle forward vector and the tangent to the road as shown in the network structure in Fig. 7.3. For all the agents, the policy networks, value networks and the proposed additional risk networks share

Hyper-parameter	Value
discount factor γ	0.99
RL network learning rate	$1e^{-4} \sim 1e^{-6}$
Lagrange multiplier learning rate	$1e^{-2} \sim 1e^{-4}$
initial Lagrange multiplier α	0.01
initial Lagrange multiplier β	0.01
memory size	5000
clipping ratio	0.2
constraint threshold μ	0.001

Table 7.1 Hyper-parameters for the experiments. [Wang et al.]

the parameters in the feature extractor, where the high-level information is shared to benefit all tasks during the training. For the constraint threshold μ, a suitable value is selected manually from candidate values through experimentation. This threshold serves to restrict the gradient changes of the policy network's output in response to adversarial attacks, and is sensitive to the scenario while significantly affecting the stability and performance of the training process. The main hyper-parameters used are shown in Table 7.1. To achieve the non-task-oriented policy network in multi-agent self-driving, all the agents share the same model structure and hyper-parameters. For adversarial attack generations, $\varepsilon = 0.1$ and iterations of 20 are used. In the robust MARL evaluation section, the performance of the proposed method is compared with the variant risk-only-model and existing method MAPPO [205]. The R-CCMARL system is set to run at 10 fps in CARLA on a single RTX 4090 graphic card. The methods employed for comparisons follow the same settings.

7.4.2 Results

In this section, experiments are carried out for the robust cooperative MARL (R-CCMARL) guidance policy evaluation. To realise these tests, first the MAPPO, the proposed robust algorithm R-CCMARL, and its variant risk-only model are trained. To make fair comparison, all algorithms are trained under the same applicable hyper-parameter settings and environment conditions.

Robustness MARL Evaluation

To evaluate the performance of the proposed R-CCMARL under observation perturbations, a strong attack configuration which is difficult to mitigate is chosen, $\varepsilon = 0.1, iteration = 20$ and multiple attack strategies are applied. As mentioned before, one of the challenges in robust MARL system against adversarial attacks is the number of agents attacked is unknown. Therefore, four attack strategies are conducted: attack on each single agent and attack on all agents, and no attack to make sure the system works well in normal situation as well. For each category all models are evaluated for 50 episodes.

The MAPPO, the only-risk model where the constraint for robust cooperative MARL is not implemented, and the full proposed algorithm are evaluated in five evaluation metrics. The metrics are defined as following:

1. Average Team Reward: the average accumulative reward of the MARL system per episode.
2. Average Individual Reward: the average accumulative reward of each agent per episode.
3. Average Individual Success Rate: current success times to reach the goal position over current episodes.
4. Average Team Risk: the average accumulative safety of the MARL system per episode.
5. Average Individual Risk: the average accumulative safety of each agent per episode.

Note that despite the concerns about bias in evaluation due to incorporating extra, non-attacked numeric data as inputs to enhance stability and performance, the results show substantial impacts on the MAPPO model which also incorporates the these extra data, resulting in low success rate. The proposed method effectively handles strong adversarial attacks under the same conditions, indicating that the additional inputs do not compromise evaluation outcomes. In Table 7.2, the full

		Attack on Agent 1			Attack on Agent 2			Attack on Agent 3			Attack on All		
Method	**Metrics**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 1**	**Agent 2**	**Agent 3**
MAPPO	Success Rate	0.80	0.50	0.60	0.80	0.00	0.40	0.60	0.10	0.10	0.40	0.00	0.00
	Return	26.30	42.63	37.73	25.68	34.78	37.46	28.38	47.70	37.01	26.06	29.41	34.59
	Risk	-10.26	-7.16	-1.12	-6.37	-4.10	-1.16	-8.26	-4.57	-5.75	-9.89	-4.94	-2.31
Risk-Only	Success Rate	0.88	0.70	0.82	0.84	0.24	0.78	0.82	0.60	0.74	0.78	0.08	0.70
	Return	34.65	51.26	49.75	34.37	37.11	51.10	36.09	59.05	43.83	32.77	34.21	40.62
	Risk	-6.10	-4.14	-0.34	-3.53	-2.61	-0.37	-6.64	-3.94	-4.54	-8.37	-3.94	-1.79
R-CCMARL	Success Rate	0.98	0.92	0.94	0.98	0.82	0.90	0.96	0.88	0.88	0.90	0.78	0.84
	Return	36.37	63.84	52.13	34.98	56.81	50.05	40.53	69.74	56.07	37.59	45.21	52.77
	Risk	-5.61	-3.85	-0.41	-3.51	-1.70	-0.42	-5.77	-3.38	-4.39	-6.97	-4.05	-1.45

Table 7.2 Detailed performance comparison in various attack conditions. The evaluation metrics include average success rate, average long-term returns, the average long-term risks. The attacks are first implemented on every single agent, and attacks are applied on all the agents. The performance of risk-only variant of R-CCMARL is also presented for ablation study. [Wang et al.]

Method	Metrics	Agent 1	Agent 2	Agent 3
MAPPO	Success Rate	1.00	0.66	0.78
	Return	27.71	57.89	47.34
	Risk	-3.75	-5.66	-1.79
Risk-Only	Success Rate	1.00	0.92	0.98
	Return	38.48	61.29	57.30
	Risk	-1.98	-4.33	-1.12
R-CCMARL	Success Rate	1.00	0.94	1.00
	Return	36.34	95.02	70.53
	Risk	-1.91	-1.85	-0.43

Table 7.3 Detailed performance metrics in normal observation condition. [Wang et al.]

method R-CCMARL leads all the categories in success rate, long-term return, and long-term risk under every attack strategy. In terms of success rate, MAPPO delivers ok performance when agent 1 is attacked. Specifically, under this setting, agent 1 reaches 80% of success, while agent 2 and agent 3 remain 50% and 60%. However, in other situations, where agent 2 is attacked, agent 3 is attacked and all agents are attacked, MAPPO shows weak robustness. Agent 2 is not able to finish the intersection passing task, leaving only 10% of success rate at most. Agent 3 is a bit better than agent 2 but gets 40% at most. Lacking adversarial training and risk awareness, MAPPO shows no defence to strong adversarial perturbations. The risk-only method, though is not involved in adversarial training, shows certain improvement over MAPPO, the performance is far from satisfaction, especially the attack is implemented on agent 2. In contrast, the full method, benefiting from the multi-agent interaction modelling (mean-field information sharing), long-term risk minimisation, and the constrained adversarial optimisation with CBF, remain resilient to the strong perturbations, delivering robust performance even in the Attack on All condition. For robust MARL algorithms, it is also important to maintain performance in normal conditions while capable of mitigating the strong attacks. Table 7.3 shows the performance comparisons in the non-attacked environments. For success rate, the full method R-CCMARL achieves perfect scores for Agents 1 and 3, and nearly perfect for Agent 2, while MAPPO performs worst, particularly for Agent 2. In terms of return, R-CCMARL leads with higher returns, especially for Agents 2 and 3, indicating better performance compared to the other methods. Additionally,

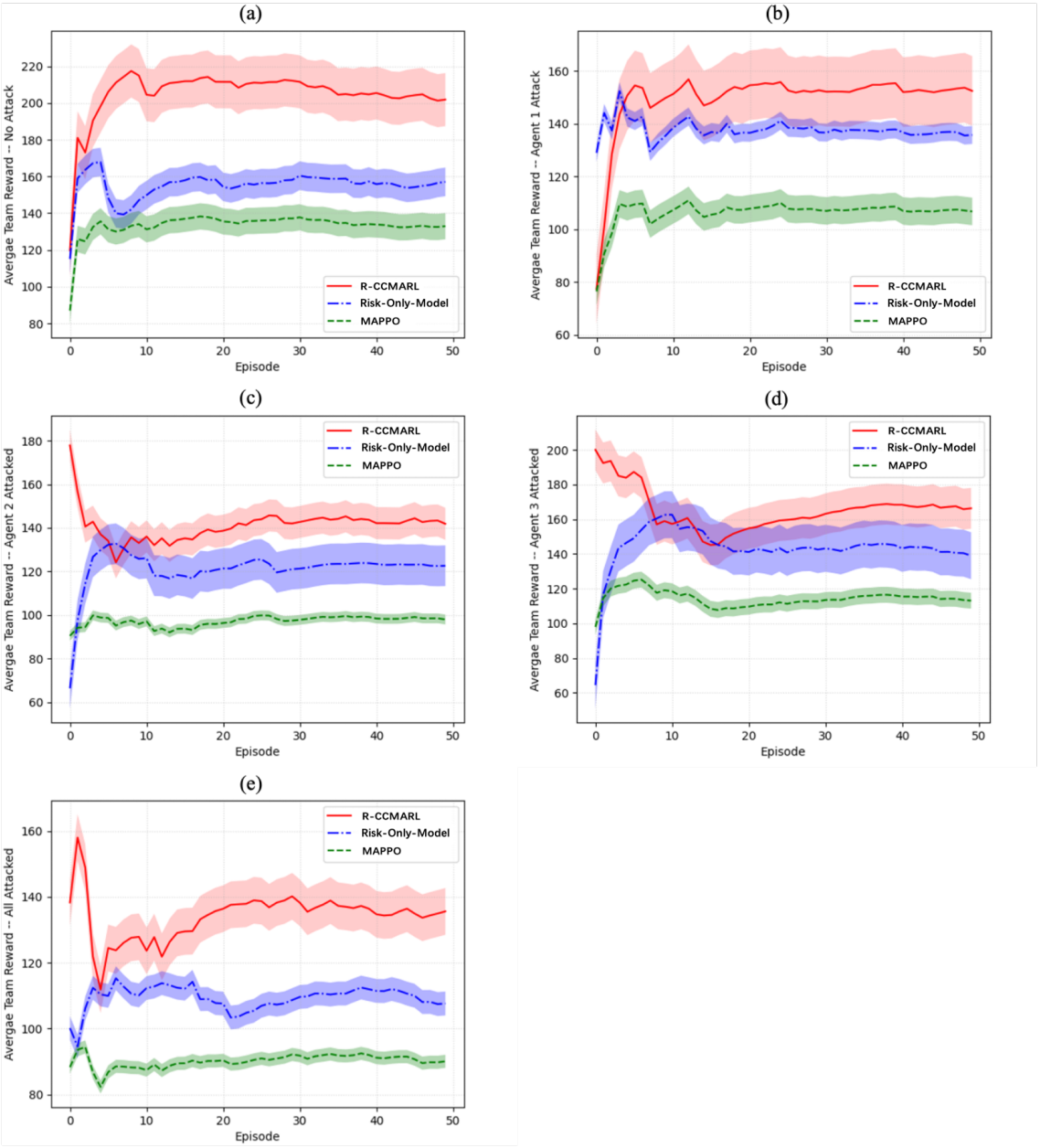

Figure 7.5 Detailed comparisons between the proposed method R-CCMARL, the variant and the MAPPO. The multi-agent systems are attacked with different agents and the same strength $\varepsilon = 0.1$, $iteration = 20$. The curve line is the average team reward while transparent area indicates standard deviation. [Wang et al.]

R-CCMARL consistently demonstrates lower risk values, particularly for Agents 2 and 3, suggesting that it not only yields high returns but does so more safely than MAPPO and Risk-Only, making it the most robust method overall.

Fig. 7.5 illustrates the performance of these methods under the five attack strategies in a more comprehensive way. The thin line curve is the mean value of each

category and the standard deviation is visualised around the mean value to show stability. Overall as it can be seen, in terms of average accumulative rewards, the risk-only outperform MAPPO in each attack situations, indicating the effeteness of the risk assessment function and the additional long-term risk minimisation. The full algorithm further boost performance over MAPPO and risk-only model. Specifically, in normal condition showed in Fig. 7.5 (a), MAPPO is at 136.15 per episode. In contrast, the full algorithm R-CCMARL achieves around 203.04 accumulative reward an episode, which represents 49.13% gain from the MAPPO. Comparing to the risk-only model, R-CCMARL achieve 28.5% improvement. Fig. 7.5 (a) states that the proposed robust cooperative MARL improves the guidance performance even without considering adversarial conditions. Fig. 7.5 (b) (c) (d) show the performance when different individual agent is under attacks. It is evident that in a MARL system, attacking different agents has varying effects, revealing the different levels of importance each agent holds in the system. In Fig. 7.5 (b), when agent 1 is attacked, the average team reward of the three methods drop by 25.2%, 13.2%, and 23.1%, resulting in rewards of 152.2, 137.1, 104.9. When agent 2 is attacked, the performance further reduced. In Fig. 7.5 (c) MAPPO achieve 99.2 in the long-term reward, and the risk-only model achieves 122.4. The full algorithm leads the board at 142.3 marking 43.4% and 16.3% improvement over MAPPO and risk-only model. In Fig. 7.5 (d), agent 3 is attacked. The full method obtain 168.6 of accumulative reward while the risk-only model and MAPPO are at 140.1 and 116.4, showing 20.3% and 44.8% of improvement respectively. Among the individual agent perturbations, attacks on agent 2 have the most significant impact on the entire system, making agent 2 the most vulnerable. The level of attacks is further increased by applying perturbations on every agent. In Fig. 7.5 (e), the full algorithm maintains the reward of 136.9. The risk-only model and MAPPO, both lacking constrained adversarial training, are not able to handle the situations, reaching only 106.2 and 91.7. These results demonstrate the effectiveness of the proposed constrained objective function in enhancing the robustness of the MARL system.

Risk Minimisation Analysis

The long-term risk minimisation with risk assessment and CBF constrained minimisation together with robust MARL shows significant robustness to perturbations in comparison with the non-robust MAPPO. Focusing solely on the risk metric, R-CCMARL and its variant risk-only model consistently demonstrates superior performance in mitigating risk compared to MAPPO across all attack scenarios. As demonstrated in Table 7.2, in the "Attack on Agent 1" scenario, MAPPO shows the highest risk at -10.26, while the risk-only model reduces it to -6.10, and the proposed method achieves the lowest risk at -5.61. Similarly, in the more challenging "Attack on All" scenario, the risk level of MAPPO remains high with values like -9.89 for Agent 1, while the risk-only model shows improvement, reducing it to -8.37. However, the full approach significantly outperforms both by further reducing the risk to -6.97. Combining the results shown in Table 7.2 and Table 7.3, across all situations, whether under heavy attacks or in normal conditions, R-CCMARL consistently shows lower risk values, indicating its greater resilience to adversarial conditions and ability to minimise performance degradation more effectively than the other models. This highlights the strength of the proposed constrained objective function in enhancing the robustness of the multi-agent reinforcement learning system under adversarial attacks. When considering the combined performance across all three metrics—success rate, return, and risk, it is evident that the proposed method R-CCMARL offers the most robust results. The lower risk values, paired with higher success rates and long-term returns, demonstrate the effectiveness of the risk assessment approach that the formulated risks are related directly to the overall driving performance. Notably, the risk is positively correlated with both the success rate and long-term return: as the risk decreases, the success rate and returns improve. This relationship highlights how well R-CCMARL handles adversarial conditions, with reduced risks translating into better overall performance.

7.5 Summary

In this chapter, a novel approach is introduced for deep-reinforcement-learning-based robust cooperative multi-agent autonomous driving against observation perturbations. These DRL-based autonomous driving models face safety challenges from sensor failures and domain transitions, resembling adversarial attacks that can compromise decision-making in critical situations, as discussed in previous chapter. And the challenge further elevates when switching from single-agent systems to multi-agent systems. Thanks to the innovations in this chapter, the problems for robust MARL are solved.

To more efficiently attack the MARL agent, similar attacking method is used. Iterative steps are employed on FGSM to help reduce the visibility of the adversarial examples and maintain stealthier attacks. Once the perturbed observation is obtained, Stochastic Games with perturbation are utilised to model the multi-agent intersection passing scenario for autonomous driving, where agents must cooperate to navigate complex intersections. A mean-field theory-supported information-sharing structure is developed to enhance global state and interaction awareness, allowing agents to better coordinate their actions under perturbations. Though the mean-field formulated communication helps the cooperative tasks, it also bring difficulties by sharing the perturbed high-level information.

An efficient risk assessment framework is introduced, focusing on long-term risk minimisation and integrated as a control barrier function. This framework not only helps tolerate bounded perturbations but also provides safety rewards as feedback to the multi-agent reinforcement learning (MARL) system, guiding agents toward safer policy learning. To further enhance the robustness of the system, a divergence-based regulariser term is incorporated to quantify and minimise the performance gap between non-adversarial and adversarial states, promoting smoother transitions and more resilient decision-making. A constrained objective function, inspired from chapter 6, is proposed with the new introduced CBF and adaption to the MARL

system. These proposed components solve the mitigation problems of MARL systems under adversarial attacks, and the problems brought by the communications.

Experimental results validate the effectiveness of the proposed method, showcasing its superiority in minimising risk and handling perturbations during multi-agent intersection passing tasks. The inclusion of a risk minimisation module and constrained optimisation proves advantageous in resolving the inherent challenges of multi-agent autonomous driving, particularly in safety-critical environments. This robust framework not only improves the overall performance of the agents in adversarial scenarios but also contributes to enhancing the safety and reliability of autonomous driving systems in real-world applications.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In Chapter 1, the objectives of this thesis were established based on the potential and challenges within autonomous driving systems. The objectives are divided into two primary categories. The first objective is to use deep-learning techniques to enhance scene understanding performance in autonomous driving systems, with lightweight solutions also considered for the system efficiency and mobility needs. The second objective focuses on designing deep-learning-based algorithms for decision-making guidance in both single-agent and multi-agent frameworks for self-driving vehicles. These algorithms must be robust and secure against observation shifts, such as adversarial attacks, while maintaining reliability in normal conditions. In this chapter, each contribution will be summarised, with a discussion on the novel elements and future work.

This thesis begins with a background review of deep learning techniques, including Convolutional Neural Networks (CNNs) and Deep Reinforcement Learning (DRL). It introduces the basic concept of CNNs, then explores advancements in CNN architectures and their impact on perception within autonomous driving systems. DRL is presented as a foundation for decision-making in autonomous driving, divided into on-policy and off-policy categories, offering a range of options for implementing self-driving algorithms. Key datasets and simulation tools are also discussed, highlighting their role in training and evaluating the proposed algorithms.

The methodology presented in Chapter 3 explores a deep-learning-based semantic segmentation technique specifically designed for urban driving scenarios, using a monocular camera as the primary vision sensor. This approach is critical for autonomous vehicle perception, as it provides pixel-level scene information that enhances understanding of the composition of the driving environment. Unlike raw RGB images, semantic segmentation assigns each pixel a category, enabling the autonomous driving system to interpret complex urban landscapes with diverse objects, ultimately aiding in accurate scene comprehension. The novel dual-attention mechanism introduced here decomposes high-level features into spatial and channel

information, effectively capturing contextual dependencies from both dimensions, which enhances segmentation accuracy. To account for the unique structures and nature of spatial and channel features, the model incorporates a pyramid pooling method for spatial attention and channel splitting for channel attention. This method ensures performance improvements without compromising computational efficiency. The integration of dual-attention with a fusion technique balances rich feature extraction with a lightweight architecture, providing an effective solution for real-time autonomous driving tasks.

In addressing 3D object detection in Chapter 4, the method leverages single-camera input and introduces a feature-enhancing pyramid module to boost feature representation. As verified in chapter 3 that a better representation in extracted high-level features benefit the accuracy in model outputs, this enhancement enables the model to retain detailed scene information across layers, benefiting the object regression networks responsible for constructing 3D bounding boxes around detected objects. To overcome the inherent limitations of RGB images in depth perception, an adaptive depth supervision signal and auxiliary depth estimator are included. This addition effectively enriches the model with missing 3D data, capturing accurate spatial information while keeping computational demands low. The technique provides a practical solution for 3D detection in cost-sensitive AD applications, where LiDAR or other depth sensors may be unavailable.

Additionally, while both methods in Chapters 3 and 4 benefit from enhanced feature representation, they retain a black-box quality due to the lack of mathematical interpretability, which is an inherent limitation in many deep learning models. To understand how these representations contribute to final predictions, an alternative explainable method Grad-CAM is utilised. Visual comparisons between enhanced and original features demonstrate that the improved features help the model focus more precisely on intended objects, filtering out background noise and thereby improving detection performance. Grad-CAM visualisations further verify that high-level features, which are more abstract, emphasise larger, nearby objects, while low-level features contribute detail to smaller or distant objects. This layered focus allows

for a combination of high-level and low-level features that enhances the capability of the model to detect objects throughout the entire scene. The explainable visual comparison not only clarifies feature contributions but also illustrates how feature hierarchy supports comprehensive scene understanding.

Chapter 5 shifts focus to a DRL-based guidance system for ground vehicles, while still investigates perception module by prediction depth images as guidance input to improve navigation and obstacle avoidance capabilities. This system operates in a two-stage process, combining imitation learning to establish baseline driving behaviours with reinforcement learning to optimise decision-making through trial and error. In theory, pretraining with imitation learning allows the DRL model to bypass early exploration, resulting in faster convergence. However, existing approaches often face challenges in transitioning IL-pretrained models into the RL phase. The proposed method identifies these issues and introduces a pretraining technique for both the Q-network and value network, which facilitates a smoother transfer between IL and RL, thereby enhancing training efficiency and robustness. To train depth perception module, a synthetic depth completion dataset is collected via simulation, addressing the shortcoming of incomplete depth ground truth of open datasets. The depth information derived from a binocular camera setup, combines supervised and stereo camera reconstruction based self-supervised techniques to generate a more comprehensive view of the environment, enhancing the system responsiveness to spatial details. Empirical results demonstrate that depth data substantially boosts the overall performance of the guidance network, making it well-suited for real-world navigation where precise environmental interaction is essential.

In Chapter 6, the robustness of single-agent AD systems is investigated through a mathematical defence algorithm that mitigates the impact of state perturbations, safeguarding performance in high-risk scenarios. These state perturbations are generated by an optimal adversarial attacker, induce directional interferences aimed at misleading the agent. Adversarial attacks mimic real-world uncertainties, such as sensor noise or environmental distortions, thus becoming ideal for testing the resilience of the agent decision-making capabilities. The defence algorithm highlights

a constrained optimisation objective solved by Lagrange multiplier technique and gradient descent. A regulariser is also introduced to stabilise agent behaviour across both normal and adversarial conditions, thereby enhancing the robustness of the agent. Additionally, an explainable attack detection mechanism is incorporated to identify and interpret these adversarial attacks, offering insights into the responses of the AD system and improving model reliability. This explainable detection not only strengthens system reliability but also illuminates the decision-making processes, helping users pinpoint areas for further improvement. Experimental results highlight the robustness of the proposed methodology against severe adversarial attacks and challenging, unseen weather conditions, demonstrating the effectiveness and adaptability of the approach.

Building on the foundation established in Chapter 6, Chapter 7 adapts the constrained objective function to a MARL context, creating a connected, cooperative framework suited for urban driving environments with multiple interacting vehicles. In the multi-agent setting, the complexity of dynamics and interactions introduces unique challenges that are aggravated under adversarial attacks. The proposed system models the interactions among agents by introducing the idea of using a universal policy for all agents, and utilising mean-field theory for effective information sharing. The universal policy facilitates non-task-oriented training, making it well-suited for applications like autonomous driving, where hierarchical levels for self-driving cars are unnecessary. This approach simplifies the training process, allowing the system to learn from a broader range of scenarios without the constraints of specific tasks. As a result, the model can adapt more effectively to varying driving conditions and environments, promoting enhanced versatility and performance in real-world situations. To establish robustness in MARL, a risk formulation is introduced. Inspired by the value network, a specialised risk network is introduced for the minimisation of long-term expected risks. The risk network is further utilised as the CBF, forming a constrained optimisation objective within the MARL framework and establishing safety criteria guarantees to manage adversarial attacks on the MARL system. Experimental results demonstrate that the connected MARL system effectively addresses robust MARL challenges, maintaining stability and efficiency in intersection negotiation

scenarios, highlighting the potential of cooperative frameworks for achieving reliable performance in complex urban traffic scenarios.

8.2 Future Work

This thesis proposed several deep-learning based models for AD perception modules, as well as robust single-agent and multi-agent decision making guidance algorithms. However, few potential future avenues for investigation can be explored in the future work.

8.2.1 Perturbation Denoising

Chapter 6 and 7 investigate the defence algorithm in the presence of adversarial attack by tolerant these attacks. Another research direction for mitigating the attacks focus on preprocessing input data to remove adversarial perturbations to effectively improve model performance in the face of misleading. Recent developments include autoencoder-based methods like Defense-GAN [206] and APE-GAN [207] that learn to reconstruct clean images. However, significant challenges remain, including computational overhead, limited transferability across different types of attacks, and the fundamental trade-off between robustness and accuracy. Future research directions include developing adaptive denoising methods that can automatically adjust to different attack types, creating more efficient architectures for resource-constrained environments, and establishing stronger theoretical foundations for understanding the relationship between model architecture and robustness.

8.2.2 Sim-to-Real

A limitation of Chapter 6 and 7 is the experiments are only conducted in the simulator environment. Sim-to-real transfer is essential for ensuring that models trained in simulations perform well when exposed to real-world variables such as lighting changes, weather conditions, and dynamic obstacles. Techniques that enhance this transfer often involve augmenting simulated data with real-world samples or using domain randomisation, where the parameters of the simulator are varied extensively

to cover a wide range of scenarios [208]. Another approach to sim-to-real transfer is the use of life-long learning that integrate both simulated and real data during training. These systems allow for continuous learning and adaptation, enabling the model to refine its performance based on real-world feedback [209]. This iterative approach is particularly valuable in autonomous driving, where the ability to adapt to new environments can significantly impact safety and efficiency. Moreover, recent advancements in multi-modal sensing, where various types of sensors (e.g., cameras, LiDAR, radar) are combined, also contribute to effective domain adaptation. By processing complementary information from different sensors, models can achieve a more comprehensive understanding of their environment, enhancing their robustness against domain shifts [210].

8.2.3 Complete AD Systems

The work in Chapters 5, 6, and 7 primarily focuses on the decision-making modules of AD systems, examining how DRL can enhance guidance and robustness in decision-making processes. However, to achieve the goal of full automation in real-world scenarios, it is essential to consider other critical components of an AD system, such as localisation, mapping, and integration of various perception modules. Localisation involves determining the precise position of the vehicle within a given environment, which is crucial for safe and efficient operation. This task typically employs a variety of sensors, including GPS, LiDAR, and cameras, to create a comprehensive understanding of the vehicle surroundings [132]. Accurate localisation enables the vehicle to understand its relationship to the road network and other dynamic entities, thereby ensuring safe navigation and obstacle avoidance. Mapping, on the other hand, involves constructing detailed representations of the environment. High-definition maps provide essential information, such as lane boundaries, traffic signals, and other critical features that inform the vehicle decision-making. Advanced mapping techniques often leverage machine learning and computer vision to enhance map accuracy and update in real-time, thus accommodating changes in the environment due to construction, traffic flow, or other factors [211]. Integrating these components with the decision-making modules explored in the previous chapters creates a holistic

AD system capable of achieving full autonomy. The synergy between decision-making, localisation, mapping, and perception is vital for handling the complexities of real-world driving situations.

References

[1] M. Channon, "Automated and electric vehicles act 2018: An evaluation in light of proactive law and regulatory disconnect," *European Journal of Law and Technology*, vol. 10, no. 2, 2019.

[2] H. Liu, M. Yang, C. Guan, Y. S. Chen, M. Keith, M. You, and M. Menendez, "Urban infrastructure design principles for connected and autonomous vehicles: a case study of oxford, uk," *Computational Urban Science*, vol. 3, no. 1, p. 34, 2023.

[3] J. Borenstein, J. Herkert, and K. Miller, "Autonomous vehicles and the ethical tension between occupant and non-occupant safety," *The Journal of Sociotechnical Critique*, vol. 1, no. 1, p. 6, 2020.

[4] E. Papadimitriou, H. Farah, G. van de Kaa, F. S. De Sio, M. Hagenzieker, and P. van Gelder, "Towards common ethical and safe 'behaviour'standards for automated vehicles," *Accident Analysis & Prevention*, vol. 174, p. 106724, 2022.

[5] N. J. Goodall, "Machine ethics and automated vehicles," *Road vehicle automation*, pp. 93–102, 2014.

[6] A. Rezaei and B. Caulfield, "Safety of autonomous vehicles: what are the insights from experienced industry professionals?" *Transportation research part F: traffic psychology and behaviour*, vol. 81, pp. 472–489, 2021.

[7] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[8] F. Dellaert, S. M. Seitz, C. E. Thorpe, and S. Thrun, "Structure from motion without correspondence," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, vol. 2. IEEE, 2000, pp. 557–564.

[9] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[10] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2650–2658.

[11] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 270–279.

[12] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4796–4803.

[13] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[15] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1991–1999.

[16] B. H. Macedo, G. F. Araujo, G. S. Silva, M. C. Crestani, Y. B. Galli, and G. N. Ramos, "Evolving finite-state machines controllers for the simulated car racing championship," in *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 2015, pp. 160–172.

[17] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on control systems technology*, vol. 15, no. 3, pp. 566–580, 2007.

[18] J. Zhao, W. Zhao, B. Deng, Z. Wang, F. Zhang, W. Zheng, W. Cao, J. Nan, Y. Lian, and A. F. Burke, "Autonomous driving system: A comprehensive survey," *Expert Systems with Applications*, p. 122836, 2023.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] M. Bojarski, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[22] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," *IEEE Transactions on Intelligent Vehicles*, 2023.

[23] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[30] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[33] Y. Lee and J. Park, "Centermask: Real-time anchor-free instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 13 906–13 915.

[34] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[35] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[36] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[37] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, UK, 1994, vol. 37.

[38] J. Schulman, "Trust region policy optimization," *arXiv preprint arXiv:1502.05477*, 2015.

[39] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[40] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[41] T. Lillicrap, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[42] T. Lin, "Focal loss for dense object detection," *arXiv preprint arXiv:1708.02002*, 2017.

[43] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 492–518.

[44] R. Roriz, J. Cabral, and T. Gomes, "Automotive lidar technology: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6282–6297, 2021.

[45] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[46] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[47] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer vision and image understanding*, vol. 81, no. 3, pp. 231–268, 2001.

[48] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2007.

[49] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[50] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs. sourceforge. net*, vol. 4, no. 6, p. 2, 2000.

[51] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang *et al.*, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1497–1504.

[52] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[53] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[54] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[55] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[56] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, pp. 98–136, 2015.

[57] L.-C. Chen, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[58] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1925–1934.

[59] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[60] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[61] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters–improve semantic segmentation by global convolutional network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4353–4361.

[62] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, "Psanet: Point-wise spatial attention network for scene parsing," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 267–283.

[63] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[64] J. Cheng, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[65] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 603–612.

[66] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 472–480.

[67] V. John, K. Yoneda, Z. Liu, and S. Mita, "Saliency map generation by the convolutional neural network for real-time traffic light detection using template matching," *IEEE transactions on computational imaging*, vol. 1, no. 3, pp. 159–173, 2015.

[68] B. Cheng, L.-C. Chen, Y. Wei, Y. Zhu, Z. Huang, J. Xiong, T. S. Huang, W.-M. Hwu, and H. Shi, "Spgnet: Semantic prediction guidance for scene parsing," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5218–5228.

[69] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a discriminative feature network for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1857–1866.

[70] L.-C. Chen, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[71] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[72] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 510–519.

[73] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[74] Y. Yuan, L. Huang, J. Guo, C. Zhang, X. Chen, and J. Wang, "Ocnet: Object context network for scene parsing," *arXiv preprint arXiv:1809.00916*, 2018.

[75] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3146–3154.

[76] H. Zhang, K. Zu, J. Lu, Y. Zou, and D. Meng, "Epsanet: An efficient pyramid squeeze attention block on convolutional neural network," in *Proceedings of the asian conference on computer vision*, 2022, pp. 1161–1177.

[77] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186.

[78] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7151–7160.

[79] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[80] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," *Advances in neural information processing systems*, vol. 29, 2016.

[81] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5162–5170.

[82] C. Wang and N. Aouf, "Explainable deep adversarial reinforcement learning approach for robust autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2024.

[83] M. Shah and N. Aouf, "3d cooperative pythagorean hodograph path planning and obstacle avoidance for multiple uavs," in *2010 IEEE 9th International Conference on Cyberntic Intelligent Systems*. IEEE, 2010, pp. 1–6.

[84] L. He, N. Aouf, J. F. Whidborne, and B. Song, "Integrated moment-based lgmd and deep reinforcement learning for uav obstacle avoidance," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7491–7497.

[85] H. Kanchwala, I. Bezerra Viana, and N. Aouf, "Cooperative path-planning and tracking controller evaluation using vehicle models of varying complexities," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 235, no. 16, pp. 2877–2896, 2021.

[86] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[87] Z. Tian, X. Chu, X. Wang, X. Wei, and C. Shen, "Fully convolutional one-stage 3d object detection on lidar range images," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 899–34 911, 2022.

[88] Z. Wieszok, N. Aouf, O. Kechagias-Stamatis, and L. Chermak, "Stixel based scene understanding for autonomous vehicles," in *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2017, pp. 43–48.

[89] J.-W. Ma, M. Liang, S.-L. Chen, F. Chen, S. Tian, J. Qin, and X.-C. Yin, "Depth-guided progressive network for object detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19 523–19 533, 2022.

[90] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[91] Y. Tian, W. Song, L. Chen, S. Fong, Y. Sung, and J. Kwak, "A 3d object recognition method from lidar point cloud based on usae-bls," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 267–15 277, 2022.

[92] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3d object detection," *arXiv preprint arXiv:1908.09492*, 2019.

[93] P. Li, H. Zhao, P. Liu, and F. Cao, "Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving," in *European Conference on Computer Vision*. Springer, 2020, pp. 644–660.

[94] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6851–6860.

[95] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[96] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[97] T. Zhang, X. Zhang, J. Shi, S. Wei, J. Wang, and J. Li, "Balanced feature pyramid network for ship detection in synthetic aperture radar images," in *2020 IEEE Radar Conference (RadarConf20)*. IEEE, 2020, pp. 1–5.

[98] G. Wang, J. Wu, B. Tian, S. Teng, L. Chen, and D. Cao, "Centernet3d: An anchor free object detector for point cloud," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 953–12 965, 2021.

[99] Y. Liu, Y. Yixuan, and M. Liu, "Ground-aware monocular 3d object detection for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 919–926, 2021.

[100] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2040–2049.

[101] J. A. Ansari, S. Sharma, A. Majumdar, J. K. Murthy, and K. M. Krishna, "The earth ain't flat: Monocular reconstruction of vehicles on steep and graded roads from a moving camera," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 8404–8410.

[102] A. Kundu, Y. Li, and J. M. Rehg, "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3559–3568.

[103] Z. Qin, J. Wang, and Y. Lu, "Monogrnet: A geometric reasoning network for monocular 3d object localization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 8851–8858.

[104] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2147–2156.

[105] Z. Qin, J. Wang, and Y. Lu, "Triangulation learning network: from monocular to stereo 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 7615–7623.

[106] A. Kumar, G. Brazil, and X. Liu, "Groomed-nms: Grouped mathematically differentiable nms for monocular 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 8973–8983.

[107] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6569–6578.

[108] Y. Chen, L. Tai, K. Sun, and M. Li, "Monopair: Monocular 3d object detection using pairwise spatial relationships," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 093–12 102.

[109] P. Li, X. Chen, and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7644–7652.

[110] B. Xu and Z. Chen, "Multi-level fusion based 3d object detection from monocular images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2345–2353.

[111] M. A. Haq, S.-J. Ruan, M.-E. Shao, Q. M. U. Haq, P.-J. Liang, and D.-Q. Gao, "One stage monocular 3d object detection utilizing discrete depth and orientation representation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 630–21 640, 2022.

[112] L. Wang, L. Du, X. Ye, Y. Fu, G. Guo, X. Xue, J. Feng, and L. Zhang, "Depth-conditioned dynamic message propagation for monocular 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 454–463.

[113] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 519–534.

[114] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta, "Beyond skip connections: Top-down modulation for object detection," *arXiv preprint arXiv:1612.06851*, 2016.

[115] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[116] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[117] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016, pp. 354–370.

[118] Z. Qin, Z. Li, Z. Zhang, Y. Bao, G. Yu, Y. Peng, and J. Sun, "Thundernet: Towards real-time generic object detection on mobile devices," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6718–6727.

[119] C. Guo, B. Fan, Q. Zhang, S. Xiang, and C. Pan, "Augfpn: Improving multi-scale feature learning for object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 595–12 604.

[120] C. Wang and N. Aouf, "Fusion attention network for autonomous cars semantic segmentation," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 1525–1530.

[121] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[122] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological cybernetics*, vol. 20, no. 3, pp. 121–136, 1975.

[123] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

[124] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 421–436.

[125] T. Wang, X. Zhu, J. Pang, and D. Lin, "Fcos3d: Fully convolutional one-stage monocular 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 913–922.

[126] T. Wang, Z. Xinge, J. Pang, and D. Lin, "Probabilistic and geometric depth: Detecting objects in perspective," in *Conference on Robot Learning*. PMLR, 2022, pp. 1475–1485.

[127] D. Park, J. Li, D. Chen, V. Guizilini, and A. Gaidon, "Depth is all you need for monocular 3d detection," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7024–7031.

[128] J. Gu, Z. Xiang, P. Zhao, T. Bai, L. Wang, X. Zhao, and Z. Zhang, "Cvfnet: Real-time 3d object detection by learning cross view features," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 568–574.

[129] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[130] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[131] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2020.

[132] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE intelligent vehicles symposium (IV)*. IEEE, 2011, pp. 163–168.

[133] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 33–47, 2020.

[134] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[135] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems*, vol. 27, 2014.

[136] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2800–2809.

[137] A. Eldesokey, M. Felsberg, and F. S. Khan, "Confidence propagation through cnns for guided sparse depth regression," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2423–2436, 2019.

[138] L. He, N. Aouf, J. F. Whidborne, and B. Song, "Deep reinforcement learning based local planner for uav obstacle avoidance using demonstration data," *arXiv preprint arXiv:2008.02521*, 2020.

[139] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[140] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[141] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 584–599.

[142] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *2017 international conference on 3D Vision (3DV)*. IEEE, 2017, pp. 11–20.

[143] N. Schneider, L. Schneider, P. Pinggera, U. Franke, M. Pollefeys, and C. Stiller, "Semantically guided depth upsampling," in *Pattern Recognition: 38th German Conference, GCPR 2016, Hannover, Germany, September 12-15, 2016, Proceedings 38*. Springer, 2016, pp. 37–48.

[144] A. Valada, R. Mohan, and W. Burgard, "Self-supervised model adaptation for multimodal semantic segmentation," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1239–1285, 2020.

[145] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with cnns: Depth completion and semantic segmentation," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 52–60.

[146] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM journal on control and optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[147] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5695–5703.

[148] A. E. Orhan and X. Pitkow, "Skip connections eliminate singularities," *arXiv preprint arXiv:1701.09175*, 2017.

[149] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3288–3295.

[150] L. Yan, K. Liu, and E. Belyaev, "Revisiting sparsity invariant convolution: A network for image guided depth completion," *IEEE Access*, vol. 8, pp. 126 323–126 332, 2020.

[151] C. Wang and N. Aouf, "Deep reinforcement learning based planning for urban self-driving with demonstration and depth completion," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2021, pp. 962–967.

[152] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5068–5078, 2021.

[153] C. Park, G. S. Kim, S. Park, S. Jung, and J. Kim, "Multi-agent reinforcement learning for cooperative air transportation services in city-wide autonomous urban air mobility," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4016–4030, 2023.

[154] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[155] A. Madry, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[156] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International conference on machine learning*. PMLR, 2019, pp. 7472–7482.

[157] V. Behzadan and A. Munir, "Whatever does not kill deep reinforcement learning, makes it stronger," *arXiv preprint arXiv:1712.09344*, 2017.

[158] M. Fischer, M. Mirman, S. Stalder, and M. Vechev, "Online robustness training for deep reinforcement learning," *arXiv preprint arXiv:1911.00887*, 2019.

[159] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[160] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *arXiv preprint arXiv:1704.02532*, 2017.

[161] P. Ye, H. Qi, F. Zhu, and Y. Lv, "Counterfactual evolutionary reasoning for virtual driver reinforcement learning in safe driving," *IEEE Transactions on Intelligent Vehicles*, 2023.

[162] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98, 2019.

[163] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102662, 2020.

[164] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 7391–7403, 2022.

[165] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking vision-based perception in end-to-end autonomous driving models," *Journal of Systems Architecture*, vol. 110, p. 101766, 2020.

[166] X. He, H. Yang, Z. Hu, and C. Lv, "Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 184–193, 2022.

[167] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *Machine Learning and Data Mining in Pattern Recognition: 13th International Conference, MLDM 2017, New York, NY, USA, July 15-20, 2017, Proceedings 13*. Springer, 2017, pp. 262–275.

[168] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[169] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[170] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," *arXiv preprint arXiv:1705.06452*, 2017.

[171] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," *arXiv preprint arXiv:1703.06748*, 2017.

[172] S. Lundberg, "A unified approach to interpreting model predictions," *arXiv preprint arXiv:1705.07874*, 2017.

[173] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[174] T. N. Mundhenk, B. Y. Chen, and G. Friedland, "Efficient saliency maps for explainable ai," *arXiv preprint arXiv:1911.11293*, 2019.

[175] E. Aksoy, A. Yazıcı, and M. Kasap, "See, attend and brake: An attention-based saliency map prediction model for end-to-end driving," *arXiv preprint arXiv:2002.11020*, 2020.

[176] J. Kim and M. Bansal, "Attentional bottleneck: Towards an interpretable deep driving network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 322–323.

[177] K. Zhang and L. Li, "Explainable multimodal trajectory prediction using attention models," *Transportation Research Part C: Emerging Technologies*, vol. 143, p. 103829, 2022.

[178] L. He, N. Aouf, and B. Song, "Explainable deep reinforcement learning for uav autonomous path planning," *Aerospace science and technology*, vol. 118, p. 107052, 2021.

[179] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[180] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[181] Z. Zhou, G. Liu, and M. Zhou, "A robust mean-field actor-critic reinforcement learning against adversarial perturbations on agent states," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[182] I. Csiszár and J. Körner, *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.

[183] I. Csiszár, "I-divergence geometry of probability distributions and minimization problems," *The annals of probability*, pp. 146–158, 1975.

[184] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[185] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[186] C. Wu, A. R. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: A modular learning framework for mixed autonomy traffic," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1270–1286, 2021.

[187] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2681–2690.

[188] J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang, "Exploration in deep reinforcement learning: From single-agent to multiagent domain," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[189] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[190] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[191] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 2961–2970.

[192] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7211–7218.

[193] Y. Yang, J. Hao, G. Chen, H. Tang, Y. Chen, Y. Hu, C. Fan, and Z. Wei, "Q-value path decomposition for deep multiagent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 706–10 715.

[194] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "Qplex: Duplex dueling multi-agent q-learning," *arXiv preprint arXiv:2008.01062*, 2020.

[195] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 5571–5580.

[196] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[197] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 024–21 037, 2020.

[198] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," *arXiv preprint arXiv:2101.08452*, 2021.

[199] T. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T.-W. Weng, "Robust deep reinforcement learning through adversarial loss," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 156–26 167, 2021.

[200] A. Kumar, A. Levine, and S. Feizi, "Policy smoothing for provably robust reinforcement learning," *arXiv preprint arXiv:2106.11420*, 2021.

[201] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," in *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 62–68.

[202] L. S. Shapley, "Stochastic games," *Proceedings of the national academy of sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[203] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[204] A. Bukharin, Y. Li, Y. Yu, Q. Zhang, Z. Chen, S. Zuo, C. Zhang, S. Zhang, and T. Zhao, "Robust multi-agent reinforcement learning via adversarial regularization: Theoretical foundation and stable algorithms," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[205] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[206] P. Samangouei, "Defense-gan: protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[207] S. Shen, G. Jin, K. Gao, and Y. Zhang, "Ape-gan: Adversarial perturbation elimination with gan," *arXiv preprint arXiv:1707.05474*, 2017.

[208] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[209] X. Yu, J. P. Queralta, and T. Westerlund, "Towards lifelong federated learning in autonomous mobile robots with continuous sim-to-real transfer," *Procedia Computer Science*, vol. 210, pp. 86–93, 2022.

[210] K. Huang, B. Shi, X. Li, X. Li, S. Huang, and Y. Li, "Multi-modal sensor fusion for auto driving perception: A survey," *arXiv preprint arXiv:2202.02703*, 2022.

[211] B. Wijaya, K. Jiang, M. Yang, T. Wen, Y. Wang, X. Tang, Z. Fu, T. Zhou, and D. Yang, "High definition map mapping and update: A general overview and future directions," *arXiv preprint arXiv:2409.09726*, 2024.