Daniel Filipe da Silveira Coelho

Aprendizagem por Reforço de Ponta a Ponta para Condução Autónoma em Ambientes Urbanos

End-to-End Reinforcement Learning for Autonomous Driving in Urban Environments

Daniel Filipe da Silveira Coelho

Aprendizagem por Reforço de Ponta a Ponta para Condução Autónoma em Ambientes Urbanos

End-to-End Reinforcement Learning for Autonomous Driving in Urban Environments

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Mecânica, realizada sob a orientação científica do Doutor Miguel Armando Riem de Oliveira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e do Doutor Vítor Manuel Ferreira dos Santos, Professor Associado com Agregação do Departamento de Engenharia Mecânica.

This work has been supported by FCT - Foundation for Science and Technology, in the context of Ph.D. scholarship 2022.10977.BD.

Fundação para a Ciência e a Tecnologia

universidade de aveiro
theoria poiesis praxis

o júri / the jury

presidente / president

Doutor Vitor Brás de Sequeira Amaral
Professor Catedrático, Universidade de Aveiro

vogais / examiners committee

Doutor Miguel Ángel Sotelo Vásquez
Professor Catedrático, Universidade de Alcalá

Doutor Luís Paulo Gonçalves dos Reis
Professor Associado, Universidade do Porto

Doutor Cristiano Premebida
Professor Auxiliar, Universidade de Coimbra

Doutora Petia Georgieva Georgieva
Professora Associada com Agregação, Universidade de Aveiro

Doutor Miguel Armando Riem de Oliveira
Professor Auxiliar, Universidade de Aveiro (orientador)

acknowledgments

I would like to express my deepest gratitude to my supervisor, Prof. Dr. Miguel Oliveira, for his unwavering support and guidance throughout my Ph.D. journey and even before it began. His passion for programming and autonomous driving inspired me to embark on this project, and for that, I am profoundly grateful. I extend my heartfelt thanks to my co-supervisor, Prof. Dr. Vítor Santos, whose invaluable experience and wisdom were important in shaping this work. Additionally, I am deeply thankful to Prof. Dr. Antonio López for his warm reception at CVC and his ongoing support. I am also indebted to German Ros for his assistance during the final stages of this thesis.

Special thanks go to my close friends at LAR—Joel, Lucas, and Manuel. The shared moments and camaraderie over these years were indispensable and made this journey memorable. I am equally grateful to Alex, whose companionship in Barcelona added a layer of joy and unforgettable experiences to my life.

Importantly, I owe a tremendous debt of gratitude to my family. Their unwavering belief in me and steadfast support have been my bedrock.

Finally, I wish to acknowledge the Foundation for Science and Technology (FCT) for funding the research presented in this thesis through the PhD scholarship 2022.10977.BD.

Palavras-chave

Condução Autônoma, Aprendizado por Reforço, Aprendizado Profundo, Sistemas de Ponta a Ponta

Resumo

Esta tese avança o campo da Condução Autónoma (AD) de ponta a ponta em ambientes urbanos, focando-se principalmente em técnicas de Aprendizagem por Reforço (RL) para resolver limitações existentes. A pesquisa começa com uma revisão abrangente dos sistemas de AD de ponta a ponta atuais, destacando os pontos fortes e fracos das abordagens de Aprendizagem por Imitação (IL) e RL, e identificando áreas críticas para melhoria e caminhos promissores para futuras pesquisas. Para enfrentar estes desafios, introduzimos o RLAD, o primeiro método de Aprendizagem por Reforço a partir de Pixels (RLfP) para AD urbana. Foram introduzidas várias técnicas para melhorar o desempenho dos métodos do estado da arte. Em primeiro lugar, desenvolvemos um codificador de imagens que utiliza tanto aumentações de imagens como camadas de Mistura de Sinal Local Adaptativa (A-LIX). Adicionalmente, introduzimos o WayConv1D, um codificador de waypoints que capta a informação geométrica 2D dos waypoints utilizando convoluções 1D. Além disso, desenvolvemos uma função de custo auxiliar para enfatizar a importância dos semáforos na representação latente do ambiente. RLAD demonstrou um desempenho positivo no benchmark NoCrash, mas necessitava da integração de demonstrações para igualar os sistemas de AD do estado de arte. Consequentemente, desenvolvemos o RLfOLD (Aprendizagem por Reforço a partir de demonstrações online), que combina IL e RL ao incorporar demonstrações online no treino de RL. Propusemos uma rede de políticas que gera dois desvios padrão, permitindo um controlo adaptativo para exploração e treino de IL enquanto considera a incerteza em ambos os domínios. Adicionalmente, incorporámos uma técnica baseada em incerteza orientada por um especialista online para melhorar o processo de exploração. O RLfOLD atinge resultados de estado de arte no benchmark NoCrash com maior eficiência e menor utilização de recursos. Abordámos ainda os desafios do benchmark CARLA Leaderboard 2.0 desenvolvendo o PRIBOOT, um agente especialista que aproveita informação privilegiada e dados limitados de condução humana para navegar em cenários exigentes. As técnicas inovadoras do PRIBOOT, como a representação de visão de pássaro (BEV) e o processamento do BEV como uma imagem RGB em vez de um conjunto de máscaras, melhoraram significativamente o desempenho. Esta abordagem forneceu um especialista capaz de navegar neste benchmark desafiador, permitindo aos investigadores gerar extensos conjuntos de dados e potencialmente resolver os problemas de disponibilidade de dados que têm dificultado o progresso. Coletivamente, o nosso trabalho apresenta contribuições significativas para o campo da AD, oferecendo insights e ferramentas que pavimentam o caminho para sistemas autónomos mais seguros e fiáveis em ambientes urbanos.

Keywords Autonomous Driving, Reinforcement Learning, Deep Learning, End-to-End Systems

Abstract This thesis advances the field of end-to-end Autonomous Driving (AD) in urban environments, focusing primarily on Reinforcement Learning (RL) techniques to address existing limitations. The research begins with a comprehensive review of current end-to-end AD systems, highlighting the strengths and weaknesses of Imitation Learning (IL) and RL approaches, and identifying critical areas for improvement and promising avenues for future research. To address these challenges, we introduced RLAD, the first Reinforcement Learning from Pixels (RLfP) method for urban AD. Several techniques were introduced to enhance the performance of state-of-the-art methods. First, we developed an image encoder that utilizes both image augmentations and Adaptive Local Signal Mixing (A-LIX) layers. Additionally, we introduced WayConv1D, a waypoint encoder that captures the 2D geometrical information of waypoints using 1D convolutions. Furthermore, we designed an auxiliary loss function to emphasize the significance of traffic lights in the latent representation of the environment. RLAD demonstrated good performance on the NoCrash benchmark but required further integration of demonstrations to match state-of-the-art AD systems. Consequently, we developed RLfOLD (Reinforcement Learning from Online Demonstrations), which combines IL and RL by incorporating online demonstrations into RL training. We proposed a policy network that outputs two standard deviations, enabling adaptive control for exploration and IL training while considering uncertainty in both domains. Additionally, we incorporated an uncertainty-based technique guided by an online expert to enhance the exploration process. RLfOLD achieves state-of-the-art results on the NoCrash benchmark with enhanced efficiency and resource utilization. We further tackled the CARLA Leaderboard 2.0 benchmark's challenges by developing PRIBOOT, an expert agent leveraging privileged information and limited human driving logs to navigate demanding scenarios. PRIBOOT's novel techniques, such as the bird's-eye view (BEV) representation and processing the BEV as an RGB image instead of a set of masks, significantly improved performance. This approach provided an expert capable of navigating this challenging benchmark, enabling researchers to generate extensive datasets and potentially resolving the data availability issues that have hindered progress. Collectively, our work presents significant contributions to the field of AD, offering insights and tools that pave the way for safer and more reliable autonomous systems in urban environments.

Table of contents

List of figures

List of tables

List of abbreviations

A3C . . . . . Asynchronous Advantage Actor Critic

A-LIX . . . . . Adaptive Local Signal Mixing

AD . . . . . Autonomous Driving

ADAS . . . . . Advanced Driver-Assistance System

AI . . . . . Artificial Intelligence

BEV . . . . . Bird's Eye View

CNN . . . . . Convolution Neural Network

DDPG . . . . . Deep Deterministic Policy Gradient

DL . . . . . Deep Learning

DS . . . . . Driving Score

EMA . . . . . Exponential Moving Average

IL . . . . . Imitation Learning

IP . . . . . Infraction Penalty

IRS . . . . . Infraction Rate Score

LLMs . . . . . Large Language Models

MSE . . . . . Mean Squared Error

POMDP . . . . . Partially Observable Markov Decision Process

PPO . . . . . Proximal Policy Optimization

PRIBOOT . . . . . Privileged Information Bootstrapping

RC . . . . . Route Completion

RL . . . . . Reinforcement Learning

RLAD . . . . . Reinforcement Learning from Pixels for Autonomous Driving

RLfD . . . . . Reinforcement Learning from Demonstrations

RLfOLD . . . . . Reinforcement Learning from Online Demonstrations

RLfP . . . . . Reinforcement Learning from Pixels

SAC . . . . . Soft Actor-Critic

SL . . . . . Supervised Learning

TD . . . . . Temporal Differences

Chapter 1

Introduction

1.1 Context

Autonomous Driving (AD) refers to the capability of a vehicle to operate and navigate without human intervention, leveraging advanced technologies to perceive the environment and make driving decisions [1]. In recent years, AD has experienced significant growth due to advancements in sensor technology, sensor fusion, computer vision, and AI, garnering widespread attention in both academia [2, 3] and industry [4, 5]. This technology offers numerous benefits. Primarily, it promises substantial improvements in road safety by eliminating human error, the leading cause of traffic accidents [6]. Autonomous vehicles can consistently adhere to traffic laws and react to their surroundings more swiftly than human drivers [7]. Furthermore, AD enhances mobility for individuals unable to drive, such as the elderly and disabled [8]. Additionally, it has the potential to increase traffic efficiency and reduce congestion through optimized routing and vehicle coordination [9]. In addition to fully AD, the development of Advanced Driver-Assistance System (ADAS) has significantly contributed to improving vehicle safety and driving comfort. ADAS encompasses a range of systems designed to assist or monitor driver actions, thereby enhancing safety and driving efficiency. These systems include features such as adaptive cruise control [10], lane-keeping assistance [11], and automatic emergency braking [12]. By providing real-time support to drivers, ADAS helps prevent accidents and reduces driver fatigue, thus serving as a crucial stepping stone towards the realization of fully autonomous vehicles [13].

In general terms, AD involves tasks that fall into two main categories: perception and decision-making [14, 15]. First, the autonomous agent must derive a useful representation of the environment from sensor data. This perception component is responsible for several tasks, including sensor fusion, object detection, tracking, and localization [16]. Once the environment is accurately perceived, the decision-making component takes over. This component utilizes the processed information to compute the control commands necessary for safe and efficient navigation. It encompasses tasks such as path planning, where the optimal route is determined, and the use of controllers to execute driving maneuvers [17]. The decision-making process must consider real-time environmental changes, traffic laws, and safety constraints to ensure the vehicle navigates effectively and securely.

The conventional approach to tackling the diversity of problems in AD involves dividing the driving task into standard modules, such as object detection, localization, path planning, and others, and then building rule-based methods to connect the different modules [18]. This approach, commonly known as modular, is widely used in the industry [19, 20]. A notable example is Waymo, a subsidiary of Alphabet Inc. Waymo's system employs a well-defined modular architecture, where each module is developed and optimized separately before integration into the overall system [21]. The modular approach offers several advantages. Firstly, it allows for specialization, where each module can be developed and optimized by experts in that particular domain, leading to high performance and reliability for individual

components. Secondly, it provides flexibility by enabling developers to upgrade or replace individual modules without, in theory, affecting the entire system. However, the modular approach also has limitations. One major challenge is the integration of different modules, as inconsistencies or delays in information transfer between modules can affect the overall system performance. Additionally, the sequential nature of the modular pipeline can introduce latency, as each module must wait for the previous one to complete its task before processing can continue [22, 23].

A more recent and promising approach is end-to-end AD. Unlike the modular approach, the end-to-end approach uses a single model to learn the entire driving process from sensory input to control output. This method leverages Deep Learning (DL) techniques, where a neural network is trained to map raw sensor data directly to driving actions [24]. One notable example of this approach is Wayve, a company that focuses on developing end-to-end AD technology. Wayve's system utilizes neural networks that can learn to drive from data, allowing the vehicle to interpret complex driving scenarios and make decisions autonomously. By training their models on vast amounts of driving data, Wayve aims to create a flexible and scalable solution that can adapt to diverse environments [25]. The end-to-end approach offers several advantages. Firstly, it simplifies the system architecture by eliminating the need for separate modules, which can reduce latency and improve real-time performance. Secondly, it allows for holistic optimization, where the entire driving process can be fine-tuned to achieve better overall performance. Thirdly, this approach has the potential to generalize better across different driving conditions, as the model learns to understand driving scenarios directly from data [26]. However, the end-to-end approach also has its limitations. One significant challenge is the requirement for large amounts of high-quality labeled data to train the models effectively. Additionally, the interpretability of these models can be limited, making it difficult to understand and debug the decision-making process. Furthermore, ensuring safety and reliability in diverse and unpredictable real-world conditions remains a critical concern [27]. End-to-end AD has gained more supporters over time, with many companies gradually transitioning to this approach. For instance, Tesla has increasingly incorporated end-to-end techniques into its Full Self-Driving (FSD) software [28]. Similarly, Comma AI focuses on an end-to-end approach with its OpenPilot software, which is designed to control vehicles directly from camera inputs, showcasing the potential of end-to-end learning in creating effective and adaptive AD systems [29].

There are primarily two approaches for end-to-end AD: Imitation Learning (IL) and Reinforcement Learning (RL). IL involves training a model to mimic the behavior of expert drivers. This approach uses Supervised Learning (SL), where the model is trained on a dataset of driving demonstrations provided by expert drivers. The neural network learns to map sensory inputs directly to driving actions [30]. The advantages of IL include its simplicity, data efficiency, and rapid development. Since the training data typically comes from human drivers,

it captures realistic driving behaviors. However, the performance of these models heavily depends on the quality and diversity of the driving data, as well as the performance of the expert demonstrators. Insufficient or biased data can lead to poor generalization [4]. Additionally, IL can suffer from a distribution gap between the data encountered by the model during deployment and the training data. This discrepancy occurs because the agent's actions can lead to states that the demonstrations did not cover, resulting in a divergence from the training distribution [31].

RL, on the other hand, involves training a model through trial and error, where the model learns to make driving decisions by receiving rewards or penalties based on its actions. In this approach, the autonomous agent explores the driving environment and learns an optimal policy that maximizes cumulative rewards over time [32]. The advantages of RL include its adaptability, exploration capabilities, and robustness. RL models can adapt to a wide range of driving scenarios, including those that were not explicitly included in the training data. However, RL models often require extensive training time and computational resources due to the need for exploration and the large state-action space in driving environments [33]. Additionally, designing an appropriate reward function that accurately reflects safe and efficient driving behavior is critical and complex [34]. Furthermore, applying RL in the real-world is often impractical because learning by trial and error can be unsafe, necessitating the use of realistic simulators or precise world models for training. Current approaches using RL in AD, particularly for complex tasks such as urban environments, focus on decoupling the perception training from the policy-driving training due to the challenges of training large neural networks with RL. Typically, the encoder is first trained using SL techniques, and then the policy driving network is trained separately with RL. Although this approach adds stability and speeds up the training process, it can lead to suboptimal policies because the encoder may not be fully aligned with the downstream task [35].

More recently, Reinforcement Learning from Demonstrations (RLfD) has emerged as an approach to combine the benefits of IL and RL by incorporating expert demonstrations into RL training. These demonstrations aim to enhance the sample efficiency of RL training by providing valuable insights, enabling the agent to explore the state-action space more effectively [36]. However, balancing the contributions of SL from demonstrations with the exploration-driven learning of RL requires sophisticated techniques to ensure the model benefits from both sources without overfitting to the demonstrations or becoming unstable during exploration [37]. Additionally, since the expert demonstrations come from pre-collected datasets, there is a potential distribution gap between the demonstrations and the training environment, similar to the challenges faced in IL techniques [38].

While end-to-end AD is highly promising, it still faces significant challenges. This Ph.D. work aimed to address some of these challenges and contribute to the advancement of effective and reliable AD systems.

1.2 Simulation Framework

Conducting real-world research in AD is often expensive, risky, and filled with ethical dilemmas. Evaluating and testing AD systems in real-world conditions requires extensive resources and infrastructure and can expose human participants and property to danger. These constraints make it impractical to rely solely on real-world testing for the development and validation of AD technologies.

Simulations offer a viable and effective alternative for AD research. They provide a controlled, safe, and cost-effective environment for testing various scenarios, algorithms, and system responses. By using simulations, researchers can create diverse driving situations that would be difficult, expensive, or unsafe to reproduce in the real world. Furthermore, simulations allow for repeatable and scalable experiments, facilitating thorough testing and validation of autonomous systems.

In this research, we utilized CARLA (Car Learning to Act) [39], a state-of-the-art open-source simulator for AD research. CARLA stands out as one of the best available open-source simulators, offering a comprehensive and versatile platform for developing, training, and evaluating AD systems [40, 41].

CARLA is designed to simulate urban driving environments with high fidelity, providing realistic physics and sensor simulations, including RGB cameras, LiDAR, Radar, GPS, among others. The simulator supports flexible configuration of weather conditions, traffic scenarios, and pedestrian behaviors, allowing researchers to test autonomous systems under a wide range of conditions.

Despite its many advantages, CARLA does have some limitations. For example, the complexity and computational requirements of running high-fidelity simulations can be demanding, requiring powerful hardware to achieve real-time performance. Additionally, while CARLA provides a rich set of features, there may still be discrepancies between simulated and real-world environments that need to be considered when transferring simulation results to practical applications.

To foster innovation, CARLA includes several benchmarks that are widely used in the AD research community to evaluate the performance of driving systems. These benchmarks provide standardized tasks and evaluation metrics, enabling fair comparisons between different approaches. Among several benchmarks, we highlight the following:

- **CoRL2017:** The original CARLA benchmark. It includes tasks such as navigation in urban environments and handling dynamic obstacles.
- **NoCrash:** A benchmark designed to assess the robustness of AD systems in challenging scenarios, including various weather conditions and dense traffic.
- **Leaderboard 1.0:** A public leaderboard that evaluates AD agents based on a set

of predefined routes and tasks, providing a competitive platform for researchers to showcase their systems.

- **Leaderboard 2.0:** An updated version of the Leaderboard 1.0 with more complex driving scenarios, pushing the boundaries of AD research.

All benchmarks have already seen competitive results, demonstrating the effectiveness and reliability of autonomous systems developed and tested using CARLA. However, Leaderboard 2.0 remains a notable exception, where the maximum route completion rate currently stands at only 15%. This indicates the higher level of difficulty set by Leaderboard 2.0, presenting a significant challenge and opportunity for further advancements in AD research.

In summary, CARLA provides a powerful and flexible simulation framework for AD research. Its combination of high-fidelity simulations, extensive configurability, and established benchmarks makes it an invaluable tool for developing and evaluating AD technologies.

1.3 Research Objectives

This Ph.D. has three key research objectives:

1. Development of End-to-End RL Architectures for AD Systems in Urban Environments: Current methods that employ RL in urban AD often separate the training of the perception network from the policy driving network. Typically, the perception network is first trained using SL techniques, and only then is the policy driving network trained using RL. This approach stems from challenges such as sample inefficiency when training large neural networks with RL. However, this decoupling can lead to representations misaligned with the downstream task, resulting in suboptimal performance. Therefore, our goal is to develop an end-to-end system trained with RL, employing techniques to mitigate these issues and achieve superior driving performance.

2. Integration of expert demonstrations in an end-to-end RL architecture for AD systems: Expert demonstrations play a crucial role in enhancing the performance of AD systems. When combined with RL, expert demonstrations can significantly improve sample efficiency, reduce the learning curve, and help overcome the challenges associated with complex decision-making scenarios. By integrating expert demonstrations into the end-to-end RL architecture developed in the first objective, we aim to leverage these benefits to further improve the driving performance and robustness of our AD system.

3. Development of a Data-Driven Expert Agent for Improved Driving Simulations: Expert agents are vital in advancing AD research, particularly in simulation, by

providing ground-truth data for training student models. The benchmark CARLA Leaderboard 1.0 led to numerous publications showcasing innovative approaches, enabled by expert agents like CARLA Autopilot and Roach [42], which offered near-perfect demonstrations for benchmark completion. However, with the release of CARLA Leaderboard 2.0, current methods face difficulties in achieving satisfactory results due to the absence of ground-truth data. Thus, our goal is to develop a data-driven expert agent capable of achieving high performance on CARLA Leaderboard 2.0. This agent will provide reliable, high-quality demonstrations to support and advance subsequent AD research efforts, ensuring that future methods have the necessary data to succeed in more challenging simulation environments.

1.4 Thesis Organization

This document is organized as follows:

- **Chapter 2:** This chapter presents a review of end-to-end AD in urban environments. This work serves as the description of several end-to-end AD methods proposed in the CARLA simulator. This chapter corresponds to a scientific article published in IEEE Access [43].
- **Chapter 3:** This chapter presents RLAD, the first AD system trained end-to-end with RL in urban environments. This work addresses the first objective defined in the previous section. This chapter corresponds to a scientific article published in IEEE Transactions on Automation Science and Engineering [44].
- **Chapter 4:** This chapter presents RLfOLD, which results from the integration of online demonstrations with RLAD. This work addresses the second objective defined in the previous section. This chapter corresponds to a scientific article published in the Proceedings of the AAAI Conference on Artificial Intelligence [45].
- **Chapter 5:** This chapter presents PRIBOOT, the first expert agent capable of successfully navigating the complex scenarios posed by the CARLA Leaderboard 2.0. This work addresses the third objective defined in the previous section. This chapter corresponds to a scientific article submitted in IEEE Transactions on Automation Science and Engineering.

Chapter 2

A Review of End-to-End Autonomous Driving in Urban Environments

Coelho, Daniel, and Miguel Oliveira. "A review of end-to-end autonomous driving in urban environments." IEEE Access 10 (2022): 75296-75311, doi: 10.1109/ACCESS.2022.3192019.

Abstract: Autonomous Driving (AD) in urban environments requires intelligent systems that are able to deal with complex and unpredictable scenarios. Traditional modular approaches focus on dividing the driving task into standard modules, and then use rule-based methods to connect those different modules. As such, these approaches require a significant effort to design architectures that combine all system components, and are often prone to error propagation throughout the pipeline. Recently, end-to-end autonomous driving systems have formulated the autonomous driving problem as an end-to-end learning process, with the goal of developing a policy that transforms sensory data into vehicle control commands. Despite promising results, the majority of end-to-end works in autonomous driving focus on simple driving tasks, such as lane-following, which do not fully capture the intricacies of driving in urban environments. The main contribution of this paper is to provide a detailed comparison between end-to-end autonomous driving systems that tackle urban environments. This analysis comprises two stages: a) a description of the main characteristics of the successful end-to-end approaches in urban environments; b) a quantitative comparison based on two CARLA simulator benchmarks (*CoRL2017* and *NoCrash*). Beyond providing a detailed overview of the existent approaches, we conclude this work with the most promising aspects of end-to-end autonomous driving approaches suitable for urban environments.

2.1 Introduction

In the last decades, the field of Autonomous Driving (AD) has received a massive amount of interest, both in academia [46–51] and in industry [4, 52–54]. The principal factor that triggered this interest concerns safety issues [55]. National Highway Traffic Safety Administration (NHTSA) reported that 94% of accidents are caused by drivers [56]. Another key factor is related to the traffic flow. Replacing humans by AD systems result in an optimized traffic flow, offering both financial and environmental benefits [57]. The benefits of fully AD appear to be considerable, which is why the research on autonomous driving remains an active area [24]. One of the most difficult challenges in this field concerns AD in urban environments [58]. Compared with highway driving or lane following, urban environments pose additional obstacles due to the unpredictability and variety of agents present in the scene, as well as complex and uncertain situations, such as pedestrians crossing lanes, traffic-lights, intersections, among others. In 2007, during the DARPA Urban Challenge [59], several researchers around the world tested their AD systems in a controllable urban environment and only six teams were able to complete the event [60]. The environment used in DARPA still lacked certain aspects of the real world, such as pedestrians and cyclists. Nevertheless, the fact that six teams were able to complete the event was extraordinary, especially at that time. Despite all the impressive research in this area, fully AD systems capable of driving in complex and unknown urban environments are still years away and the main reason for this

is the arduous task of generalization to unpredictable situations in a short period of time [61].

AD systems are complex systems that integrate many technologies, from sensors, processing units, software, among others. Therefore, AD systems need to deal with a wide range of problems: sensor inaccuracies, hardware reliability, object detection, localization, etc. The conventional approach to tackle this diversity of problems consists of dividing the driving task into standard modules such as object detection, localization, path planning, etc. and then build rule-based methods to connect the different modules [17, 18, 62] (see Figure 2.1). This approach is commonly called modular, and is widely used in the industry [4]. The interconnectivity between different modules in a system is a problem extensively investigated in the robotics field [63]. For example, this interconnectivity between modules led to the creation of frameworks, such as Robot Operating System (ROS) [33, 64]. The modular architecture enables the development of each module in an independent fashion facilitating the collaboration between all elements of the engineering team. In addition, the development of individual and specific modules divides the autonomous driving task into a set of narrow problems widely investigated in the literature, as is the case of localization, computer vision, motion planning, among others. Finally, interpretability constitutes one of the great advantages of these modular approaches: as the entire system is divided into modules, the source of a malfunction can more easily be tracked to the responsible module.

The major disadvantage of modular systems is the arduous task of developing and maintaining the interconnection between all modules in the system. For example, different scenarios may require different connections between modules [22], which compromises the modularity paradigm. The modular architecture is also prone to error propagation [23], in which a minor error in one module can produce catastrophic results in another, for example, a misclassification of a traffic-light can influence the decision-making process to generate a path planning that leads to a collision. Additionally, as the modules are task-specialized, they may fail to generalize to unusual conditions and unexpected situations.

More recently, end-to-end approaches emerged as an alternative technique to tackle the AD problem. The end-to-end approach formulates the AD task as an end-to-end learning process, in which the objective is to learn a policy capable of transforming sensor data into control commands [24]. In general, end-to-end architectures are simpler and have fewer components than modular architectures (see Figure 2.1). Unlike the modular paradigm, the end-to-end paradigm also captures the human driving essence: a simultaneous perception and action [65]. The downside of end-to-end systems is the lack of interpretability [33]. It is difficult to track down the source errors or to explain certain decisions taken by the model [22]. Over the years, the interest in the end-to-end approach for AD was scarce, especially compared with the amount of research done in modular approaches. However, due to the rise of Artificial Intelligence [66–68] in the past years, and due to the recent developments of Deep Reinforcement Learning (Deep RL) [69–71] by Deep Mind [72, 73], end-to-end approaches have begun

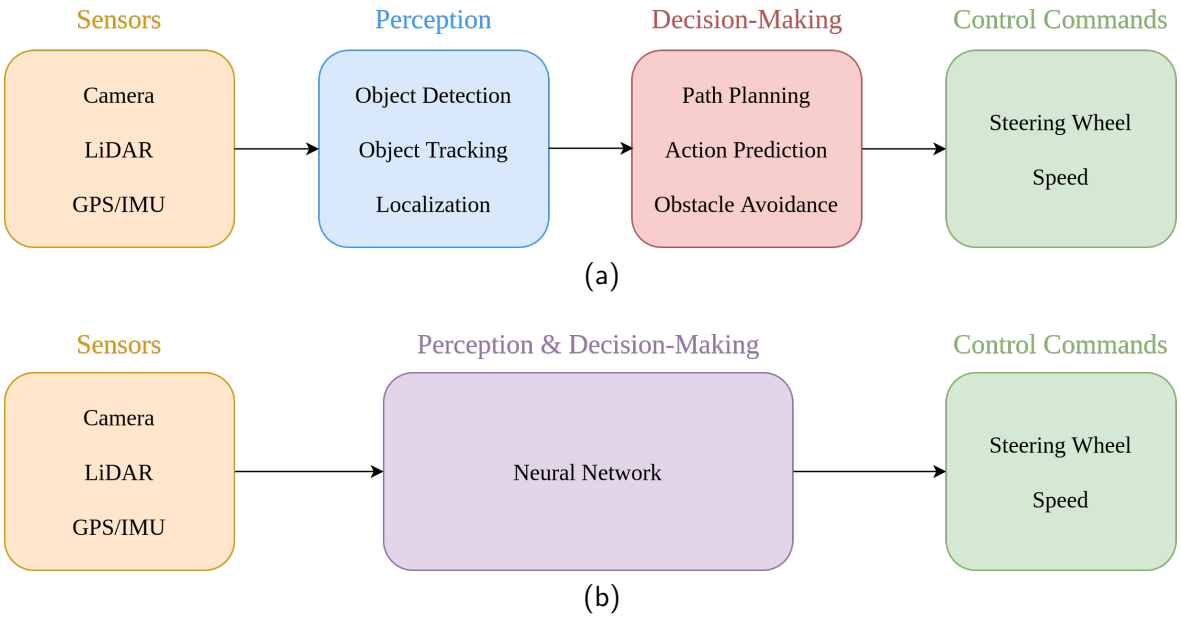


Figure 2.1: Architecture of: (a) a modular approach [76], and (b) an end-to-end approach. The modular architecture consists of several interconnected modules, whereas in the end-to-end architecture those different modules are replaced by a single, learning-based module.

to show promising results [74, 75].

In end-to-end approaches, there are two different learning methodologies: Imitation Learning (IL) and Reinforcement Learning (RL). IL aims to learn a policy by observing the actions performed by humans. It is a supervised learning approach, in which the model tries to mimic human behavior [77]. NVIDIA achieved excellent results using this methodology by training a convolution neural network (CNN) to predict the steering angle of a vehicle [26]. One advantage of the IL methodology is that it can use solely Deep Learning (DL) and merely optimize the parameters of the model to reduce the difference between the model behavior and human behavior. However, the process of scaling an AD system based on IL is a challenging task due to the impossibility of covering all possible scenarios in the training phase [4].

RL methodologies aim to learn a policy that maximizes the cumulative rewards received by an automatic system, as it interacts with the environment [78, 79]. One variant of RL is Deep RL, which combines DL with RL [80]. Although there are some technical differences between RL and Deep RL, for the purposes of this review, which is to differentiate IL and RL, we will use the terms RL and Deep RL interchangeably. In the case of RL, there is no need to collect data from human driving, because as the agent interacts with the environment, it learns how to behave in order to maximize the reward. As the training of RL models occurs online, it is possible to explore the environment and train simultaneously, which is a great advantage compared with the IL models. The downside is that RL is less data-efficient in the training stage [33]. Liang et al. combined the advantages of IL and RL by creating an IL

model based on labeled data, and then optimizing the policy using an online RL-based policy tuning [81]. One crucial element of RL algorithms is the definition of rewards. As the agent tries to maximize the reward, the definition of the reward function directly influences the behavior learned by the agent. One common example is to reward the movement towards the goal [4,81], or to punish whenever a collision occurs [82]. In 2018, Kendall et al. demonstrated the first real-life application of RL in AD, in which they were able to train a driving policy capable of learning how to follow a lane in less than 30 minutes [4].

The application of end-to-end methodologies in AD is relatively recent: the first use case was in 2016 [26]. In the three subsequent years, several approaches were proposed focusing on simplified versions of AD, such as lane following [83–88]. As expected, the application of end-to-end AD in urban environments is even more recent: 65% of the works found on this topic are from 2020 or 2021. As these works are recent, the majority of reviews of AD do not include their findings [17, 89, 90]. The only review solely focused on end-to-end was proposed by Tampuu et al., where the authors performed a thorough analysis of the different architectures and training methods of end-to-end approaches applied in AD [33]. However, their work was not focused especially in urban environments, and therefore, some of the findings should not be generalized to more complex environments. This paper presents the first review targeting **end-to-end autonomous driving in urban environments**, which is an emerging topic in the literature. The key contributions of this paper are to provide:

- a description of the main differences between the successful end-to-end approaches in urban environments;
- a quantitative analysis based on two CARLA simulator benchmarks (*CoRL2017* [39] and *NoCrash* [91]).

This paper considers only systems trained and tested in the CARLA simulator for two reasons: a) CARLA is considered the state-of-the-art open-source simulator for self-driving cars [41]; b) to ensure a fair comparison between all approaches. In [41], the authors performed an extensive comparative analysis of six simulators based on features like perception, path planning, 3D virtual environments, traffic-scenario, scalability, etc. and concluded that CARLA outperforms all other simulators.

The remainder of this document is organized as follows: in Section 2.2, we present a thorough analysis of the most successful end-end AD systems in urban environments; Section 2.3 evaluates the detailed approaches based on quantitative metrics; Section 2.4 presents the conclusions.

2.2 Discussion

In this section, we perform a thorough comparison of several end-to-end AD approaches in urban environments. This comparison is based on three main points: architectures, input sensor modalities, and output modalities. As discussed before, the targeted end-to-end AD approaches discussed in this work are the ones that used CARLA as the environment to train and test the models.

2.2.1 Architectures

Due to the complexity of urban environments, it is common the usage of low dimensional intermediate representation of the environment instead of parsing the raw data from the scene. One of the options for this low dimensional intermediate representation is called affordances [92]. Sauer et al. proposed an AD system based on affordances, especially designed for urban environments [93]. Examples of these affordances include: presence of hazard stop, red traffic-light, speed sign, distance to vehicle, relative angle, and distance to center line. These affordances are predicted by neural networks that receive both RGB images and a navigation command, e.g., "go straight", "turn left", or "turn right". The affordances are then processed by controllers in order to produce the control commands. Figure 2.2 depicts a simplified version of the system proposed by Sauer et al.

Mehta et al. also used affordances to aid the AD task [94]. The affordances allow to infuse human knowledge into the system instead of expecting the network to learn all relevant features for driving from scratch. Unlike Sauer et al., in this case the affordances are learned by the network simultaneously with the main task of driving. The authors demonstrated that the joint learning of the auxiliary tasks and the usage of the predicted affordances in the final control commands prediction increases the performance of learning. Furthermore, the authors also claim that the usage of affordances significantly increases the level of interpretability of the system, which is, as explained in Section 2.1, one of the shortcomings of end-to-end systems.

Chen et al., in [74], instead of using human defined labels, used an algorithm to provide the true labels. First, a privileged agent is trained with access to ground-truth data to imitate an expert autopilot. Then, the authors used the trained privileged agent to train another agent with only visual input. The results showed that the usage of an agent with privileged information significantly improves the vision-only driving agent.

In [95], Prakash et al. proposed an architecture that comprises two main blocks: a *Multi-Modal Fusion Transformer* (TransFuser) and a waypoint prediction network. The TransFuser receives data from different sensor modalities as input, and it produces a compact representation of the environment as output. This process is carried out by using the self-attention mechanisms of transformers [96] to incorporate the information between the different modal-

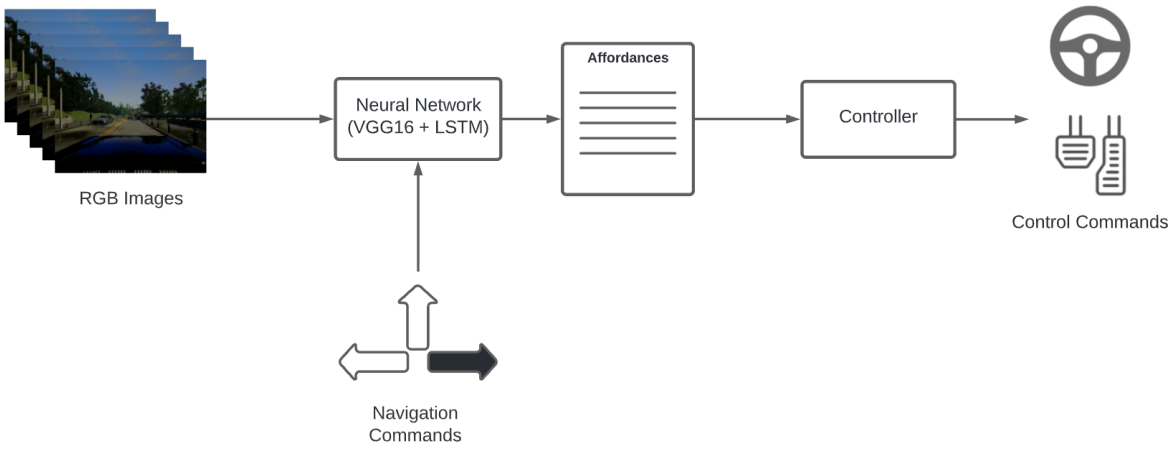


Figure 2.2: Simplified version of the system proposed by Sauer et al. [93]. The system receives as input RGB images and a navigation command provided by CARLA. The features are then extracted from the images using convolutional layers of a VGG16 neural network. Based on the navigational command received, the features are processed by a different block to produce the affordances. Finally, the predicted affordances are used by a controller to produce the control commands.

ities. The compact representation of the environment is then passed into an encoder neural network to reduce its dimensionality, and therefore increase computational efficiency. The waypoint prediction network receives both the encoded version of the environment and the desired trajectory provided by a global planner. The network consists of several Gated Recurrent Units (GRUs) [97,98], that outputs the predicted trajectory, under the form of waypoints (more details in Section 2.2.3). A simplified version of the system proposed by Prakash et al. is depicted in Figure 2.3.

One of the major problems in applying RL in the field of AD consists of the high-dimensional sensor inputs, as is the case of RGB images. For this reason, most of the applications of RL in AD have focused on simple driving tasks, such as lane following [4, 23]. However, in the last two years, some methods have been proposed that address this problem [24, 99–101]. For example, in [24], Agarwal et al. presented a framework that creates a low-dimensional state representation that comprises a stack of bird's eye-view semantic segmented images, desired trajectory, kinematics features, and traffic-light states. Then, the low-dimensional state representation is conveyed to the RL algorithm (Proximal Policy Optimization (PPO) [102, 103]). A simplified version of the architecture of the solution is illustrated in Figure 2.4.

Chen et al., in [99], proposed a system that encodes RGB images and global path trajectories using a CNN [104] and an LSTM (Long Short-Term Memory) [105] to extract both spatial and temporal features. Then, the policy network receives the encoded data and outputs control commands after the defuzzification procedure. The defuzzification procedure is

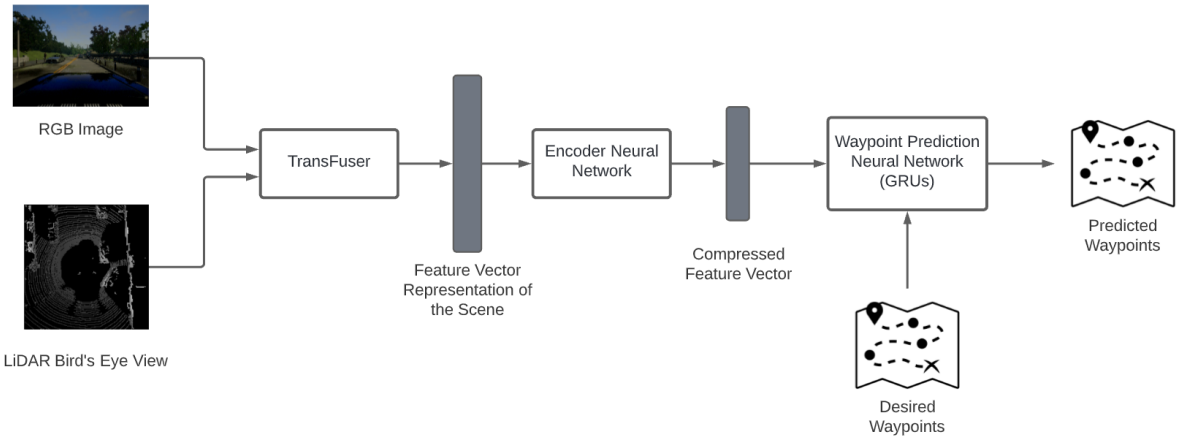


Figure 2.3: Simplified version of the system proposed by Prakash et al. [95]. The system receives a RGB image and a LiDAR bird's eye view image as input of the *Multi-Modal Fusion Transformer* (TransFuser). The fusion process is attained by using several transformer modules to combine the intermediate feature maps between both modalities. The output of the TransFuser constitutes a compact representation of the environment that comprises the global context of the scene. This compact representation it then conveyed into an encoder neural network to reduce computational efforts. The waypoint prediction neural network consists of several GRUs that receive both the output of the encoder neural network (compressed feature vector) and the desired trajectory, provided by a global planner, and outputs the future waypoints. The authors reported that vehicle measurements are also used in the TransFuser, but for clarity reasons, that information is omitted.

responsible for transforming the output of the policy network into control commands.

Chen et al., in [100], proposed the combination of the modular framework and the RL framework. As input, the system receives data from two sensors: a RGB camera and a LiDAR. During training, a semantic mask is obtained using some components in the modularized framework, such as object detection, mapping, and localization, and then the mask enters the system as labeled data. The policy network receives RGB images and LiDAR data and produces the control commands together with the semantic mask. The semantic mask produced by the policy network provides an interpretable explanation of how the agent understands the world that surrounds him.

The previous approaches based on RL did not have the ability to foresee the future, which is a feature that we, humans, have inherited from years of experience [106]. C. Huang et al., in [101], focuses on building a RL agent capable of predicting new observations. The first layer of the system consists of a Semantic Encoding Mapping (SEM) [107] that learns a semantic representation from raw images. This representation is then sent to a Deep RL algorithm (Deep Deterministic Policy Gradient (DDPG) [108, 109]). The core element of this Deep RL is a deductive reasoner that enables policy to be learned in a model-based manner. This way, it can predict the next state and reward based on the current state and reward,

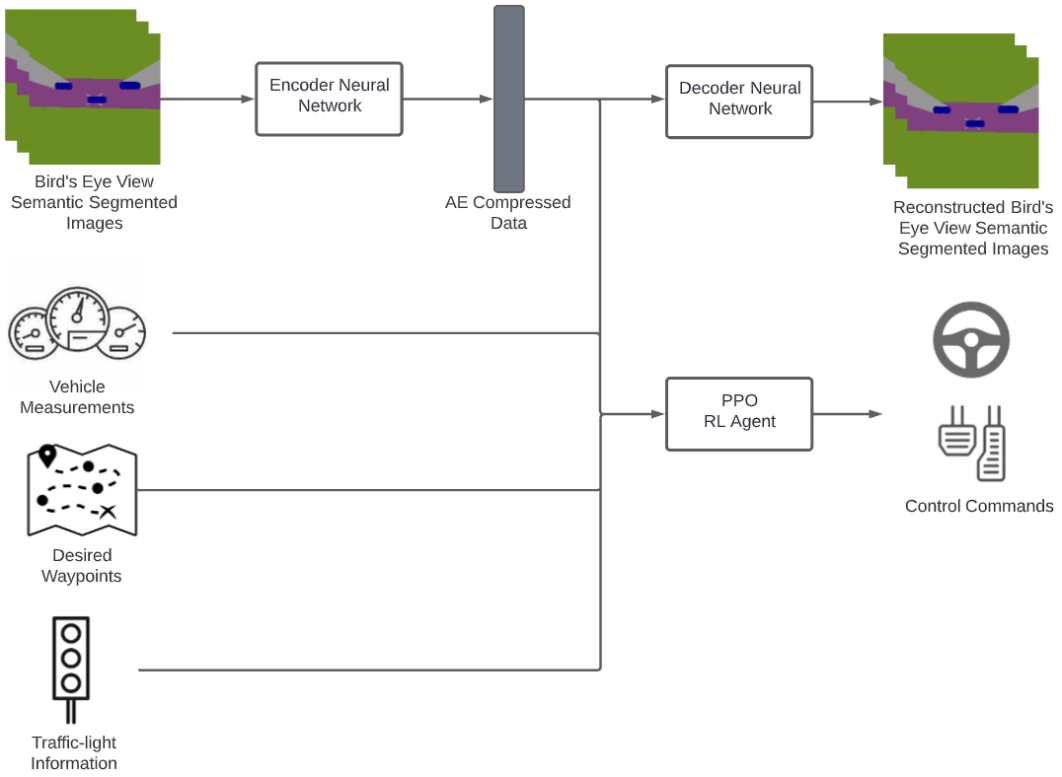


Figure 2.4: Simplified version of the system proposed by Agarwal et al. [24]. The system receives as inputs three bird's eye view semantic segmented images, vehicle measurements, navigation commands, desired waypoints and traffic-light information. Each input is provided directly by CARLA. The images are processed by an autoencoder (AE) [110] that produces a compressed version of the images (AE Compressed Data). The compressed data is combined with the remaining inputs to form the RL state representation. The RL agent it then responsible to produce the control commands. In addition to the control commands, the system also outputs reconstructed bird's eye view semantic segmented images through the decode of the compressed data.

which produces a more reliable driving policy.

In [81], Liang et al. proposed a system called CIRL (*Controllable Imitative Reinforcement Learning*) that aims to combine the advantages of IL and RL. First a supervised network is pretrained based on human labeled data. Then a Deep RL model (DDPG) is initialized with the pretrained weights of the supervised network. The authors claim that the usage of human driving demonstrations for the initialization of the RL model can significantly reduce the sample complexity, and therefore, saving innumerous hours of exploration with the environment.

More recently, some authors have also used the aforementioned affordances to tackle the high-dimensional data issue of RL. Ahmed et al., proposed an end-to-end AD system comprised of two major components: supervised network and a Deep RL agent (DDPG) [75]. The

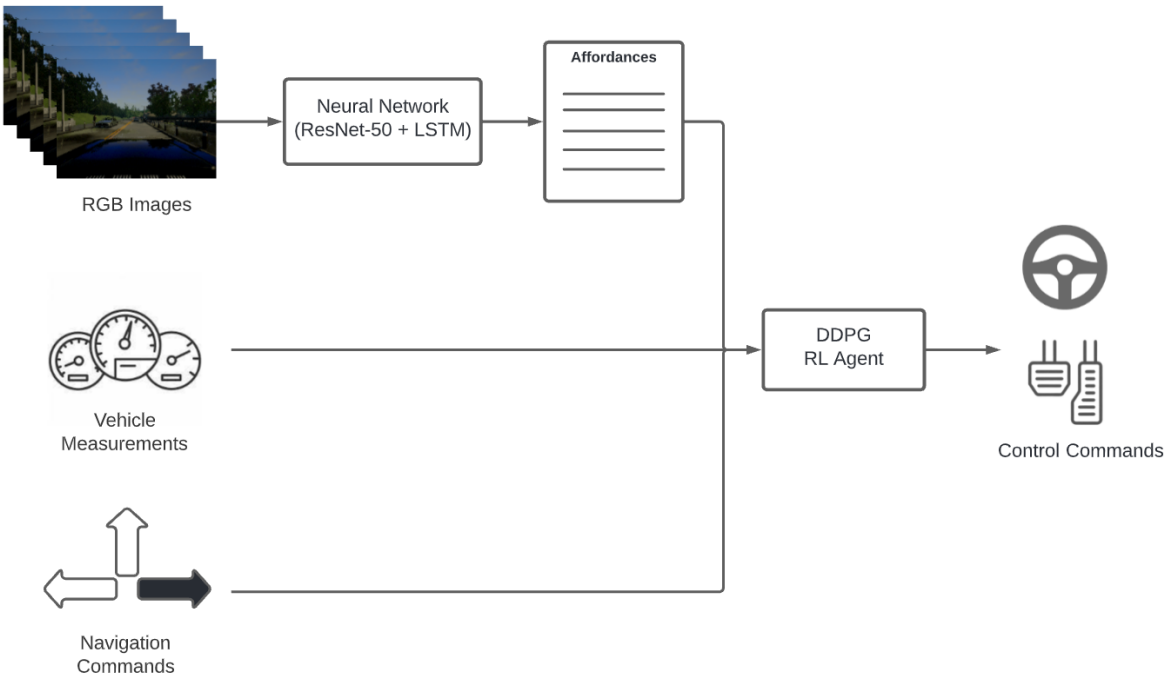


Figure 2.5: Simplified version of the system proposed by Ahmed et al. [75]. The system receives as input a stack of RGB images, vehicle measurements and navigational commands. Firstly, a residual network (ResNet-50 [112]) is used to extract the features of the images, and then LSTM units are employed to model the dependencies between successive frames. At the end of this processing, the affordances are predicted. The DDPG RL agent receives a vector that comprises the affordances, vehicle measurements and a navigation command, and produces as output the control commands.

supervised network encodes RGB images into a set of affordances. Subsequently, the Deep RL transforms the affordances, vehicle measurements and a navigation command into control commands. A simplified version of the architecture of this solution is depicted in Figure 2.5.

Toromanoff et al. also used affordances in a RL pipeline [111]. The first component of the system is an encoder trained to predict affordances such as distance to centerline and traffic-lights. Then, the output features of the encoder are conveyed into the RL, instead of the affordances. The authors have named this approach *implicit affordances*, since it uses the information that predicted the affordances and not the affordances itself.

There are mainly two differences between the approaches described in the last paragraphs: how to encode the raw data from the sensors? how to learn a driving policy based on the encoded data? Whether using encoder neural networks [24, 95, 99, 100] or affordances [93, 94, 111], the goal is to produce a low dimensional intermediate representation of the environment to simplify further processing. In this aspect, we believe that encoder neural networks are preferable to affordances. In encoder neural networks, the relevant features are learned by the model, whilst in affordances, the relevant features to be learned are user-

defined, which can introduce human bias into the system. Furthermore, it is questionable whether an approach based on affordances should be considered end-to-end or not, because it presents the same problems of modular approaches: human definition of affordances; diverse ways to integrate the affordances in a learning method; error propagation due to incorrect prediction of the affordances. Regarding the learning of the driving policy, there are two distinct approaches: IL and RL. Based on the ratio of IL/RL in recent approaches, it is inconclusive to assess what is the leading learning method in urban environments. As will be discussed in Section 2.3, both approaches achieve satisfactory results and therefore further research is required to investigate which learning method is more suitable for AD in urban environments.

2.2.2 Input Sensor Modalities

The majority of end-to-end systems rely only on vision [77, 81, 111], using only one camera to predict the control commands. However, in urban environments the single modality configuration is usually insufficient to produce a robust and reliable AD system [95]. Furthermore, AD in urban environments requires navigation from one point to another, and therefore, additional navigation inputs are often mandatory. Xiao et al., in [65], performed a comparison between single and multimodal end-to-end AD systems. They used RGB images and depth information as the sensor modalities and demonstrated that multimodality is beneficial to end-to-end systems, outperforming single modality configurations. Regarding the fusion scheme, the authors concluded that the early fusion, i.e. increasing the number of channels from three (RGB) to four (RGBD), was the one that achieved the best results.

Regarding the navigation inputs, Codevilla et al. performed a study about the incorporation of navigation commands into the AD system [77]. Navigation commands are referred to as an indication about the future action taken by the agent, such as "go right" or "turn left on the intersection." These commands can be generated by high-level route planners [113, 114] or by humans. Codevilla et al. implemented two different architectures: *command input* (see Figure 2.6) and *branched* (see Figure 2.7). In *command input*, the network takes the navigation commands as input, together with the raw images and some vehicle measurements. These three inputs are processed independently, and then the combination of the three results are delivered to a control model to produce the control commands. On the other hand, in the *branched* architecture only the image and the measurements are conveyed into the network as inputs. In this case, the control module is replaced by a set of branches. The role of the navigation command is to select which branch should be active, and therefore the navigation command can be seen as a switch. Results demonstrated that the *branched* architecture performed significantly better than the command input approach and other baseline approaches.

Huang et al., in [101], proposed a multimodal system that receives RGB images and depth

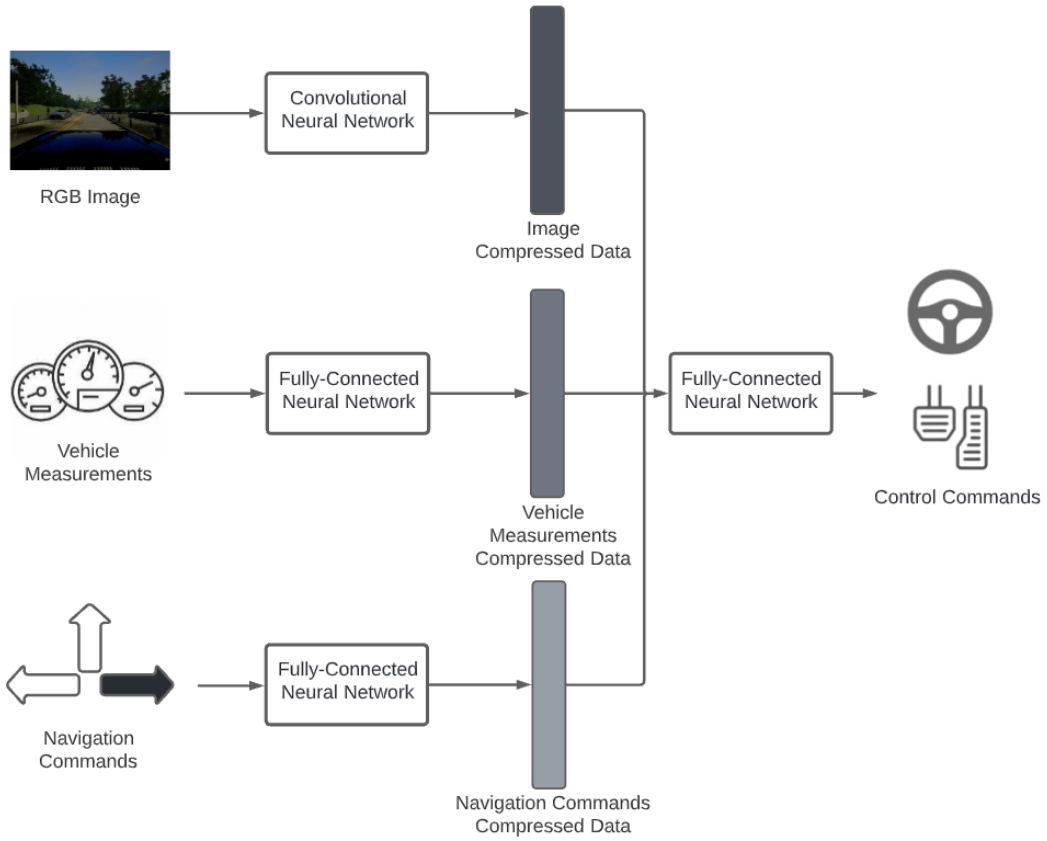


Figure 2.6: Simplified version of the *command input* architecture proposed by Codevilla et al. [77]. The system takes a RGB image as an input, alongside with vehicle measurements and navigation commands. These inputs are processed independently by three neural networks: a convolutional neural network and two fully-connected neural networks, respectively. The outputs of these neural networks are then concatenated to form the input of the control module, which is a fully-connected neural network. At the end, the control module produces the control commands.

information as input. This information is encoded by a neural network and is processed by a conditional driving policy. The conditional driving policy is a branched fully connected network, and in addition to receiving the encoded data, also receives a navigation command. The navigation command, as in [77], activates the corresponding branch, and each branch is a neural network that produces the control commands.

In [93], Sauer et al. also used the concept of specialized neural networks, but instead of predicting conditional control commands, the system predicts conditional affordances. Their architecture, Figure 2.2, receives a navigation command that selects a specific neural network to predict the affordances. The authors reported that training specialized neural networks for each navigation command leads to better performance than training neural networks that use navigation commands as inputs.

As an alternative to navigation commands, some authors have used the desired trajectory,

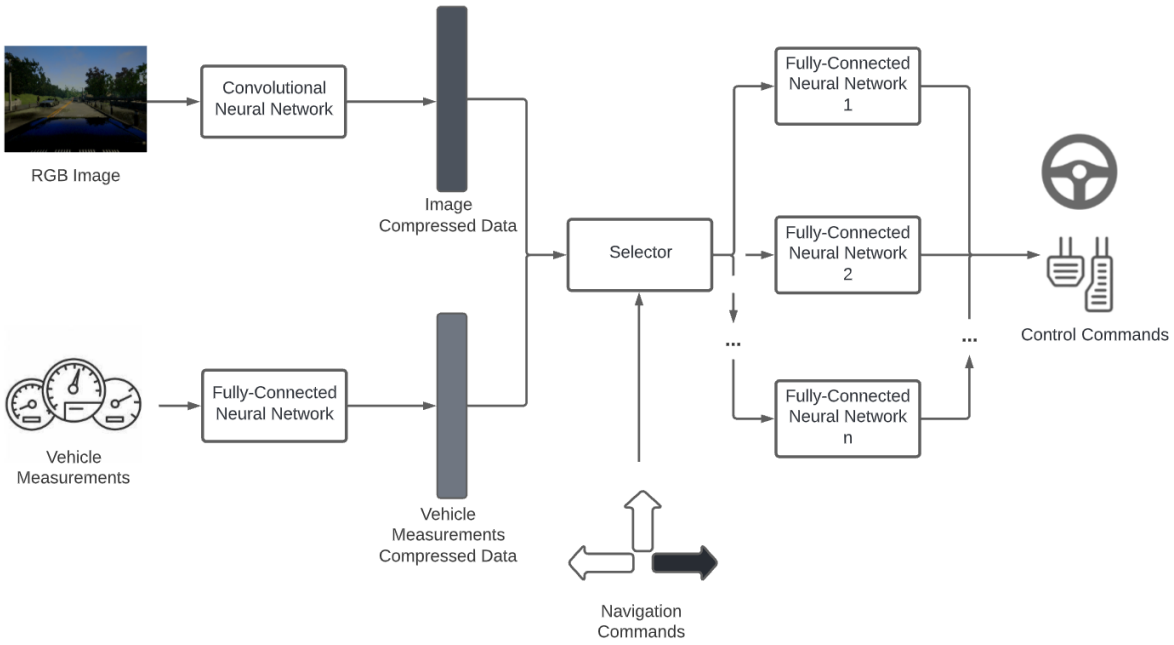


Figure 2.7: Simplified version of the *branched* architecture proposed by Codevilla et al. [77]. The system receives as inputs a RGB image, vehicle measurements and a navigation command. The RGB image and the vehicle measurements are processed independently by a convolutional neural network and a fully-connected neural network, respectively. The outputs of these neural networks are concatenated to form the input of the next stage. The navigation command is used as a switch that selects which fully-connected neural network should process the concatenated data and therefore produce the control commands.

provided by a global planner, as input [24, 95, 115]. Usually the trajectory is conveyed into the system under the form of waypoints. For instance, in [95], Prakash et al. used GPS coordinates provided by CARLA as input, to predict local waypoints (see Figure 2.3). The GPS coordinates provided by CARLA are relatively sparse and can be spaced hundreds of meters apart. Conversely, the waypoints predicted by the neural network refer to the trajectory that the agent should follow in subsequent timestamps. Cai et al., in [115], instead of using the global planner from CARLA, implemented the A^* [116,117] algorithm to plan the coarse route from the initial point to the destination point based on static maps. The waypoints provided by global planners do not consider dynamic objects nor information regarding traffic-lights. Its only purpose is to provide a global trajectory based on static elements of the environment.

In the last few years, several authors suggested that, for urban environments, the integration of RGB cameras and LiDAR is essential [95, 100, 115]. These modalities are often seen as complementary, where the RGB cameras provide information about the road and visual aspects of the scene, such as traffic-lights, and the LiDAR provides accurate spatial information in 360 degrees [100]. Prakash et al., in [95], as discussed above, proposed the usage of the attention mechanism of transformers to integrate RGB images and LiDAR data.

Authors argued that it is a robust and flexible way of integrate different modalities of sensors in general, and not only the RGB camera and LiDAR (see Figure 2.3). Chen et al. also used RGB images and LiDAR data as inputs. Instead of using the raw point clouds from the LiDAR as input, they performed a prepossessing step, where the raw point clouds are converted into a 2D LiDAR bird's eye image, which is then conveyed into the network [100].

Cai et al. in [115] explored even further the combination of multiple modalities, where they proposed an end-to-end AD system that receives RGB camera, LiDAR and RADAR data as input. This multimodal information is processed by uniform alignment and projection onto the image plane. In addition to cameras, LiDARs and RADARs, some authors also use HD maps as inputs. For example, Zeng et al. proposed a system that takes LiDAR data and HD maps as inputs of the network [22].

Multiple authors have also used high-level measurements about the state of the vehicle, such as current velocity or acceleration [75, 77, 81]. When the model only considers one frame to make a decision, the usage of the current velocity can be highly useful [33]. However, in IL approaches, if the model receives the current speed and predicts the speed for the next timestamp as one of the outputs, there is high changes of causing the inertia problem. In most cases, the current speed and the speed in the next timestamp are highly correlated, which can induce the model to only consider the current speed to predict the speed in the next timestamp. This can seriously hamper the effectiveness of the agent, because it can lead to agents that are reluctant to change the velocity.

Although most of end-to-end approaches use a RGB camera as the only input modality, several works reported that multimodality outperforms single modality configurations in urban environments [77, 95, 100, 115]. Based on latest end-to-end AD papers, it is evident that researchers are moving from single modality to multimodal configurations (see Table 2.1). From all the available sensors, LiDAR appears to be the one that can add more valuable information to the RGB camera. Furthermore, as AD in urban environments deals with the problem of navigating from one location to another, navigation commands or desired trajectories, are often used.

2.2.3 Output Modalities

The majority of the end-to-end AD systems produce the steering angle and the speed for the next timestamp as outputs [65, 93, 99, 111]. These properties are easily obtained from a vehicle and, therefore, can be used as labeled data for IL approaches. Usually, traditional PID controllers [118] are required to convert the steering angle and speed outputted by the network into acceleration/brake and steering torque of the vehicle [24, 99, 101]. The downside of these approaches is that it is very difficult to comprehend the decisions taken by the model. In other words, when the model produces an incorrect driving decision, it is not possible to understand the reason for that decision.

In recent years, some authors have explored the usage of trajectories (waypoints) as the output modality of the system [22, 95]. For instance, Prakash et al. proposed a system that predicts waypoints for the future 4 timestamps [95] (see Figure 2.3). Zeng et al., in [22], proposed a system that produces 3D detections as well as their future trajectories, and then uses a planner to choose from the set of possible trajectories the one that minimizes a predefined cost. In this output modality, it is also necessary to implement a controller (usually a PID) that generates low-level steering and acceleration/braking commands to reach the desired trajectory [95, 101]. As an alternative to PID controllers, Gutiérrez et al. proposed a modular and scalable waypoint tracking controller, fully integrated in ROS [119]. One advantage of outputting waypoints, instead of the steering angle and velocity commands, is that the model is required to plan the action for the future timestamps, and not only for the next one. This long-term planning increases the robustness of the driving policy because it converts a reactive agent into a planning agent. Another advantage concerns interpretability: it is much easier to interpret and analyze waypoints rather than momentary steering and velocity commands. Using waypoints, it is possible to convey the intentions of the system.

In order to further increase the interpretability of the systems, some authors have also used additional outputs. For example, in [100], as discussed above, the system predicts a semantic mask of the scene, and in [101] the system reconstructs semantic images using a scene understanding decoder. Usually, these additional outputs are computed by a separate branch of the network based on an intermediate layer of the network [33]. These outputs are not directly related to the main output of the network, but they provide information about the internal representation of the network, which is immensely useful to comprehend the driving decisions and to explain the failures. The authors, in [94], reported that the joint learning of the main and additional outputs leads to more robust and effective driving policies.

Although the majority of end-to-end AD produces steering angle, throttle, and braking as final output of the network, the most promising output modality is waypoints. The long-term planning and interpretability make waypoints more suitable for AD, especially for urban environments. To further increase the level of interpretability, the usage of additional outputs, learned in a joint learning mechanism, is highly recommended.

2.3 Evaluation

This section focuses on quantitatively evaluating the results from the approaches described throughout the document. However, to additionally provide a comparison between end-to-end approaches and modular approaches, we have also included a modular approach, proposed by Dosovitskiy et al. [39], in this section. This approach divides the driving task into three modules: perception, planning, and continuous control. The perception module uses semantic segmentation to estimate lanes, dynamic objects, and other hazards. The planning module

consists of a rule-based state machine that implements a driving policy specially designed for urban environments. Finally, the continuous control module is a PID controller that produces the steering, throttle, and brake commands.

Table 2.1 depicts the summary of the aforementioned approaches divided by the three points of discussion: architecture, inputs and outputs. This table follows a chronological order to facilitate the extraction of possible trends. Regarding the architectures, there is not a clear trend. Thus, it is not possible to claim which approach, IL or RL, is the most suitable for urban environments. Concerning the inputs, in 2018, none of the approaches used LiDAR as data source, while in 2021, four out of seven used LiDAR, which clearly shows the applicability and usefulness of LiDAR in urban environments. Finally, regarding the outputs, in 2018, the only output modality was steering angle, throttle and braking, while in 2021, some authors have used future waypoints as the output of the models. As stated before, around 65% of the approaches studied are from 2020 or 2021, which indicates that, more than ever, researchers are focused on applying end-to-end AD system in urban environments.

Given that this paper focuses on evaluating end-to-end AD systems tested on CARLA, we used two benchmarks (*CoRL2017* and *NoCrash*) of the simulator to accurately compare the performance of the approaches. An additional advantage of comparing all approaches within the same benchmarks is that all approaches use the same sensors, and therefore share the technical specifications. For example, for all approaches, the camera sensor provides images with 800x600 pixels and has a horizontal field of view of 90 degrees. The LiDAR sensor is a Velodyne 64, with a range of 10 meters, and covers a horizontal field of view of 360 degrees. Lastly, the RaDAR sensor has a range of 100 meters and a horizontal field of view of 30 degrees.

Our goal was to compare all algorithms listed in Table 2.1; however, some works were evaluated in different benchmarks, reason why they are not considered in the evaluations. Notwithstanding, the majority of the approaches were evaluated in the aforementioned benchmarks. In both benchmarks, there are two towns: Town 01 and Town 02.

Town 01 (see Figure 2.8) consists of 2.9 Km of road with 11 intersections, and it is used to train the models. We will refer to this as the training conditions. Town 02 (see Figure 2.9) consists of 1.4 Km of road with 8 intersections, and it is used to test the models under different weather conditions when comparing with the ones used in the training conditions. We will refer to this as the test conditions. It is worth noting that Town 01 and Town 02 are simplified versions of urban environments. Their layout consists of several T-junctions with traffic lights. There are more realistic scenarios in CARLA, as is the case of Town 3, which contains 5-lane junctions, a roundabout, and a tunnel. However, only a few approaches have tested their algorithms under such conditions and therefore it is unsuitable for comparisons, at least for now.

The *CoRL2017* benchmark is the original CARLA benchmark, and the goal is to navigate

Table 2.1: Contributions of AD systems in urban environments, described in terms of: architecture, inputs and outputs. The table follows a chronological order of the papers.

Approach	Architecture	Inputs	Outputs
Dosovitskiy et al. 2017 [39]	modular architecture: perception, planning, and continuous control	RGB image navigation command	steering angle throttle brake
Sauer et al. 2018 [93]	conditional affordances prediction followed by a controller	RGB image navigation command	steering angle throttle brake
Mehta et al. 2018 [94]	IL with affordances prediction	RGB images navigation command	steering angle throttle brake
Liang et al. 2018 [81]	IL followed by RL	RGB image vehicle measurements navigation command	steering angle throttle brake
Codevilla et al. 2018 [77]	IL with navigational command as input (command input)	RGB image vehicle measurements navigation command	steering acceleration
Codevilla et al. 2018 [77]	conditional IL (branched)	RGB image vehicle measurements navigation command	steering acceleration
Chen et al. 2019 [74]	privileged IL	RGB image vehicle measurements navigation command	steering angle throttle brake
Toromanoff et al. 2020 [111]	RL with implicit affordances	RGB images navigation command	steering angle throttle
Xiao et al. 2020 [65]	multimodal conditional IL	RGB image depth information/LiDAR vehicle measurements navigation command	steering angle throttle brake
Cai et al. 2020 [115]	probabilistic motion planning using multimodal information	RGB image LiDAR RADAR vehicle measurements desired trajectory	steering angle throttle brake
Chen et al. 2020 [99]	conditional DQN	RGB image desired trajectory	steering angle acceleration
Chen et al. 2021 [100]	latent RL	RGB image LiDAR desired trajectory	steering angle throttle brake
Zeng et al. 2021 [22]	interpretable IL	LiDAR HD map	waypoints
Huang et al. 2021 [101]	IL with scene understanding	RGB image depth information/LiDAR navigation command	steering angle speed
Ahmed et al. 2021 [75]	affordances prediction followed by RL	RGB image vehicle measurements navigation command	steering angle acceleration brake
Agarwal et al. 2021 [24]	combination of modular and RL	segmented bird's eye view traffic-light information vehicle measurements desired trajectory	steering angle speed
Prakash et al. 2021 [95]	IL with multimodal fusion transformer	RGB image LiDAR vehicle measurements desired trajectory	waypoints
C. Huang et al. 2021 [101]	deductive RL	RGB image navigation command	steering angle acceleration brake

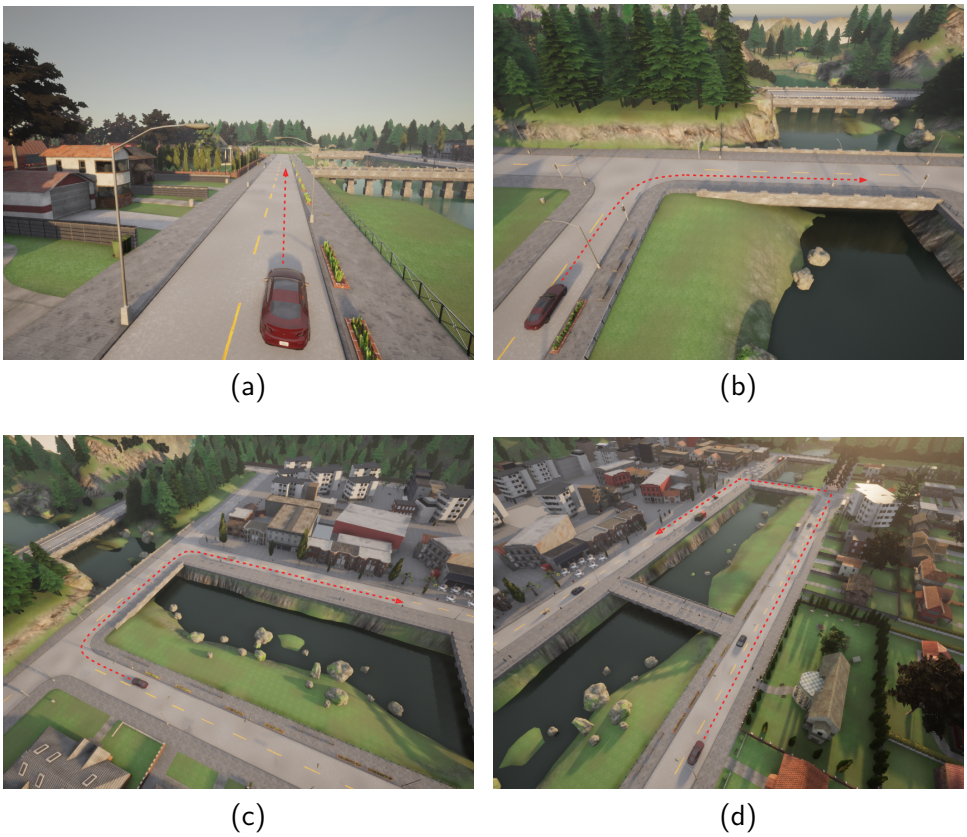


Figure 2.8: Illustration of the four driving tasks of *CoRL2017* benchmark in Town 01. (a), (b), (c), and (d) represent the Straight task, One Turn task, Navigation task, and Navigation with Dynamic Obstacles task, respectively.

from a starting point to a destination point using a route planner [39]. There are four driving tasks, with increasing difficulty levels: Straight, One Turn, Navigation and Navigation with Dynamic Obstacles (see Figure 2.8). In the Straight task, the destination is straight ahead of the starting point, and there are no dynamic obstacles in the environment. In the One Turn task, the destination is one turn away from the starting point and the environment contains no dynamic obstacles. In the Navigation task there is no restriction on the location of the destination and starting point, and once again, there are no dynamic obstacles. Finally, in the Navigation With Dynamic Obstacles, the scheme is the same as the previous task, but with dynamic obstacles (cars and pedestrians) introduced in the scene. For each task, for an episode to be considered successful the agent must reach the destination within a time limit, defined as the time required to reach the destination along the optimal path at a speed of 10 km/h [39]. Driving infractions, such as collisions or driving on the sidewalk do not lead to the termination of the episode, which means that the primary objective of this benchmark is to evaluate skills such as lane following and performing 90 degrees turns [91].

Table 2.2 depicts, for the *CoRL2017* benchmark, the percentage of successfully completed

Table 2.2: Results of *CoRL2017* benchmark. Each value corresponds to the percentage of successfully completed episodes, for each task in **training conditions**. Each column corresponds to an approach. Best scores are highlighted in bold.

Task	Dosovitskiy et al. 2017	Liang et al. 2018	Sauer et al. 2018	Chen et al. 2019	Toromanoff et al. 2020	Xiao et al. 2020	Huang et al. 2021	Ahmed et al. 2021	Agarwal et al. 2021	C. Huang et al. 2021
Straight	98	98	**100**	**100**	**100**	98	**100**	**100**	99	**100**
One Turn	82	97	97	**100**	**100**	99	**100**	98	**100**	98
Navigation	80	93	92	**100**	**100**	93	**100**	93	**100**	93
Nav. dynamic	77	82	83	**100**	**100**	89	98	94	**100**	75

episodes out of 25, for each task in training conditions, using 10 approaches. In general, the results of each approach are worsening with the increase in difficulty of the task. However, both Chen et al. [74] and Toromanoff et al. [111] achieved the maximum score in all tasks. Huang et al. [101] and Agarwal et al. [24] also achieved excellent results. As expected, the overall results are very satisfactory, because the agents are being tested under the same conditions in which they were trained (same town and same weather).

Table 2.3 has the same structure as Table 2.2, but refer to the testing conditions. Chen et al. [74] kept the maximum score, while Toromanoff et al. [111] suffered a slightly decreased in the score. Huang et al. [101] also achieve very satisfactory results. As expected, the scores in training conditions are much better than in testing conditions. Nevertheless, the overall results under testing conditions are surprisingly good, suggesting that the models were able to generalize the driving policy to unknown scenarios.

The main difference between the system proposed by Chen et al. and the others is that the system uses a teacher that has access to privileged information to train a vision-based agent. Based on the results described above, there are strong indications to conclude that this learning method is immensely effective considering the evaluation metrics of the *CoRL2017* benchmark.

Based on Table 2.2 and Table 2.3, it is also possible to unmask some of the limitations of modular approaches, namely poor generalization, and error propagation. The testing results were considerably worse than the training conditions because the perception module fails systematically under complex and unseen conditions. When the perception module fails, the planning module is not able to produce a reliable path and therefore the continuous control module is unable to produce accurate control commands. However, we want to stress that these results do not allow us to conclude that current end-to-end approaches are superior to modular approaches. The modular approach considered is from 2017, and it is expected that novel and better ones have been developed in the meantime. Furthermore, such bold claim would require an extensive comparison between modular and end-to-end approaches, which is out of the scope of this paper.

The *NoCrash* benchmark comprises three tasks with distinct levels of difficulties: Empty, Regular, and Dense (see Figure 2.9) [91]. The Empty task corresponds to an uninhibited town with no dynamic obstacles. The Regular task consists of a town with a moderate number of

Table 2.3: Results of *CoRL2017* benchmark. Each value corresponds to the percentage of successfully completed episodes for each task in **testing conditions**. Each column corresponds to an approach. Best scores are highlighted in bold.

Task	Dosovitskiy et al. 2017	Liang et al. 2018	Sauer et al. 2018	Chen et al. 2019	Toromanoff et al. 2020	Xiao et al. 2020	Huang et al. 2021	Ahmed et al. 2021	Agarwal et al. 2021	C. Huang et al. 2021
Straight	50	98	94	**100**	**100**	97	**100**	97	**100**	**100**
One Turn	50	82	72	**100**	**100**	83	**100**	95	98	82
Navigation	47	68	68	**100**	**100**	93	**100**	92	**100**	68
Nav. dynamic	44	62	64	**100**	98	94	94	91	99	60

vehicles and pedestrians. Finally, the Dense task corresponds to a town with many vehicles and pedestrians. This benchmark is more recent than *CoRL2017*, so only a few approaches have assessed their algorithms in these tasks. The core idea of this benchmark is to introduce a new aspect in the evaluation of the agent: the response to dynamic objects. Measuring an agent solely based on whether it navigates to the destination point without considering what happened in the meantime is a limited judgement of its driving capabilities. As such, in the *NoCrash* benchmark, for an episode to be considered successful, the agent must reach the destination under the time limit with the additional constraint of not colliding with any object. This benchmark is a more complete assessment of the driving performance, although there are several other aspects of urban driving which are yet to be considered. Some examples are respecting traffic lights and abiding to speed limits.

Table 2.4 contains the percentage of successfully completed episodes out of 25, for each task in training conditions, for 5 approaches. The overall scores, in this table, clearly indicate that this benchmark is more difficult than the *CoRL2017* benchmark. Once again, Chen et al. [74] achieved the best results in all tasks. Without considering Chen et al. [74], all other approaches presented relatively low scores in the Dense task, even though the conditions were exactly the same as in training.

Table 2.5 contains the results for the testing conditions, and here, Ahmed et al. [75] achieved the best results, surpassing Chen et al. [74] in two tasks. Agarwal et al. [24] also achieved excellent scores in these conditions, achieving the same score as Ahmed et al. [75] in the Regular and Dense tasks. In the Empty and Regular tasks, all approaches have achieved good results, always above 80%. The poor results showed in the Dense task in training conditions were amplified in test conditions. Toromanoff et al. [111] and Huang et al. [101] achieved a score of less than 50% successful episodes, which clearly proves that current end-to-end AD systems have a considerable difficulty in dealing with dense urban environments.

Results from the quantitative evaluation are inconclusive in what concerns the best architecture for urban environments. For the *CoRL2017* benchmark, the most effective approach was IL-based architecture (Chen et al. [74]), while for the *NoCrash* benchmark the most effective approach was RL-based architecture (Ahmed et al. [75]). Table 2.2 and Table 2.3 suggest that current end-to-end AD systems are already very effective at tackling simple driving tasks, such as Straight and One Turn. These approaches also appear to be relatively well prepared

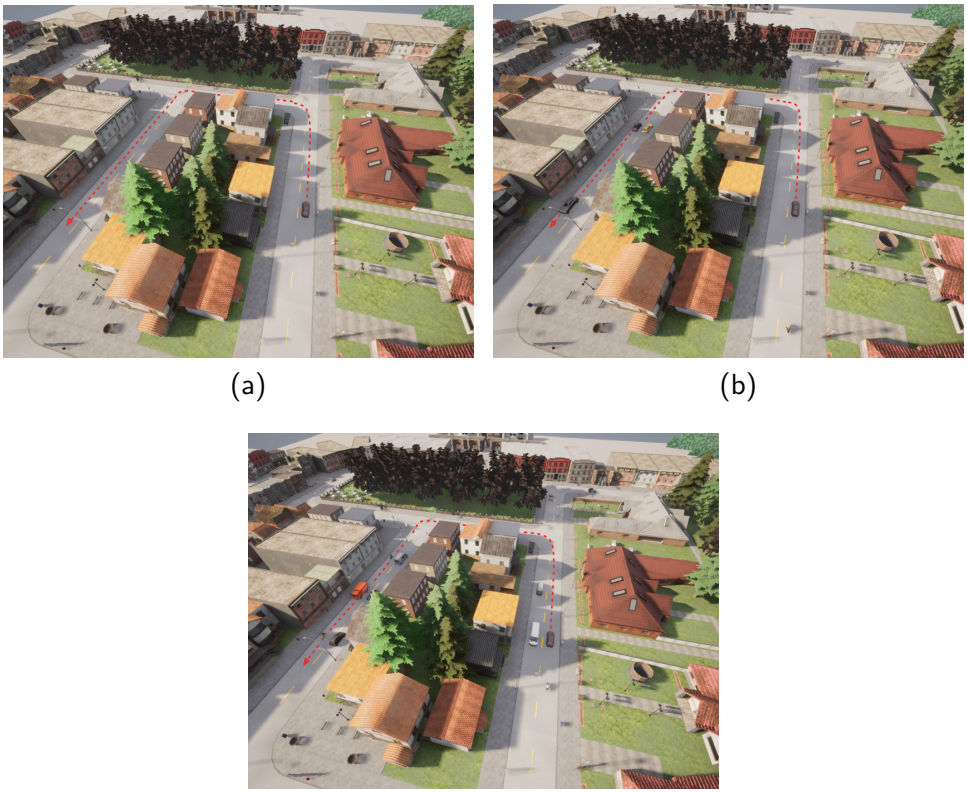


Figure 2.9: Illustration of the three driving tasks of *NoCrash* benchmark in Town 02. (a), (b), and (c) represents the Empty task, Regular task, and Dense task, respectively.

for Navigation tasks. Conversely, Table 2.4 and Table 2.5 suggest that current end-to-end AD systems are not yet prepared to cope with dense traffic situations, characteristic of many real-world cities. The intricate problem of dealing with multiple agents and their unpredictability appears to be most challenging problem for end-to-end AD systems in urban environments. These results clearly indicate that further research is required in this area to tackle the dense urban environments.

2.4 Conclusions

This paper is the first evaluation of end-to-end AD systems in urban environments. We performed a detailed analysis of 17 approaches, based on three key points: architecture, inputs and outputs. Two CARLA benchmarks were used to quantitatively compare the approaches: *CoRL2017* and *NoCrash*. For the *CoRL2017* benchmark, we compared 10 approaches both in training and testing conditions, where we show that the solution proposed by Chen et al. [74] achieved an excellent score of 100% in all tasks considered. Furthermore, results also

Table 2.4: Results of *NoCrash* benchmark. Each value corresponds to the percentage of successfully completed episodes for each task in **training conditions**. Each column corresponds to an approach. Best scores are highlighted in bold.

Task	Chen et al. 2019	Toromanoff et al. 2020	Huang et al. 2021	Ahmed et al. 2021	Agarwal et al. 2021
Empty	**100**	**100**	**100**	**100**	**100**
Regular	**99**	96	91	97	90
Dense	**95**	70	61	70	87

Table 2.5: Results of *NoCrash* benchmark. Each value corresponds to the percentage of successfully completed episodes for each task in **testing conditions**. Each column corresponds to an approach. Best scores are highlighted in bold.

Task	Chen et al. 2019	Toromanoff et al. 2020	Huang et al. 2021	Ahmed et al. 2021	Agarwal et al. 2021
Empty	**100**	99	**100**	**100**	**100**
Regular	94	87	82	**96**	**96**
Dense	85	42	43	**87**	82

suggest that modular approaches suffer from poor generalization and error propagation. For the *NoCrash* benchmark, we compared 5 approaches both in training and testing conditions. In the training conditions Chen et al. [74], once again, achieved the best results, while in the testing conditions the approach proposed by Ahmed et al. [75] achieved the best scores. From the analyses of Table 2.4 and Table 2.5, we can conclude that the current end-to-end AD systems are not prepared to deal with dense traffic, suggesting that additional research is required.

The use of simulators, such as CARLA, clearly plays a key role in the training of AD systems. Learning to drive in the simulation domain presents innumerous advantages: avoiding human casualties and expensive crashes, changing lightning and weather conditions, and reshaping structural elements of the scenes. It is also possible to reconstruct rare and dangerous scenarios that foster the learning of a robust and safer driving policy [120]. Furthermore, in simulators, we can exploit privileged information, such as the pose of the vehicle and semantic information, that would otherwise not be possible to have. However, the carry over from simulation to reality poses significant problems, mainly due to the simulation-reality gap [121]. The process of transferring a model trained in simulation to the real world is referred to as transfer learning and, in the past years, several approaches have been proposed to tackle this issue [120–123]. Due to the novelty of end-to-end AD systems, none of the works addressed in this paper have been validated in real-sized vehicles in the real world. This is an important area to address in the near future.

The next paragraphs of this section offer a critical analysis of end-to-end AD, focused on the three points considered in this paper: architecture, inputs and outputs.

IL is the most dominant strategy for end-to-end AD systems. However, there are some fundamental limitations that should be highlighted. First, IL is limited to the average of the training data, i.e., the model will learn the most repeated features in the data, ignoring the rare cases. In driving, a rare case might be a child running towards a ball in the middle of the road, and that is not a case that we are willing to ignore. Second, and from our point of view, the most limiting factor of IL concerns the limitation of the teachers. Since the goal of AD is to obtain systems that can drive better than humans, we cannot be limited by demonstrations from humans, otherwise, the best we can hope to achieve is the same driving performance of humans, and as we have noted, it is not enough. For those reasons, we believe that the most promising architecture is **RL-based**. In recent years, agents trained with RL techniques have already achieved super-human performance, as in the case of game playing [124] and robotics [125]. An agent trained with RL can explore various possible cases, including dangerous and rare cases, and then learn based on those cases.

The most dominant strategy for input sensor modalities, in end-to-end AD, is to use RGB cameras. Most proponents of those configurations claim that it is the most affordable way to deploy AD systems. However, based on the works described above, **multimodality** appears to be much more effective than the single modality RGB cameras. Furthermore, as demonstrated by Chen et al. [100], RGB images and LiDAR information are complementary, providing a more detailed understanding of the scene. Concerning navigation information, we believe that momentary indications, such as "turn right", can induce ambiguity in a scene where multiple roads may lead to the right. Based on that, it seems that the model should receive the **desired trajectory**, provided by a global planner, as input. As it is widely used, **vehicle measurements** should also be taken into consideration.

The most common outputs in end-to-end AD systems are steering angle, throttle and braking. At first glance, this is the logic approach since we are dealing with end-to-end systems. However, we believe that it is not the most promising approach. Momentary control commands are very difficult to interpret, and to explain the decisions taken by the model. On the other hand, as used in [22, 95], **waypoints** are better to convey the intentions of the system. Furthermore, predicting waypoints forces the model to plan the long-term trajectory instead of reacting to momentary inputs. When using waypoints, both the lateral and longitudinal movement are explained: the lateral movement is explicit in the waypoints and the longitudinal movement can be easily explained by the distance between successive points. To further increase the visualization of the intentions of the model, and to increase the effectiveness of the driving policy, **additional outputs**, learned in a joint mechanism, are highly recommended.

Chapter 3

RLAD: Reinforcement Learning from Pixels for Autonomous Driving in Urban Environments

Coelho, Daniel, Miguel Oliveira, and Vitor Santos. "RLAD: Reinforcement Learning From Pixels for Autonomous Driving in Urban Environments." IEEE Transactions on Automation Science and Engineering (2023), doi: 10.1109/TASE.2023.3342419.

Abstract: Current approaches of Reinforcement Learning (RL) applied in urban Autonomous Driving (AD) focus on decoupling the perception training from the driving policy training. The main reason is to avoid training a convolution encoder alongside a policy network, which is known to have issues related to sample efficiency, degenerated feature representations, and catastrophic self-overfitting. However, this paradigm can lead to representations of the environment that are not aligned with the downstream task, which may result in suboptimal performances. To address this limitation, this paper proposes RLAD, the first Reinforcement Learning from Pixels (RLfP) method applied in the urban AD domain. We propose several techniques to enhance the performance of an RLfP algorithm in this domain, including: i) an image encoder that leverages both image augmentations and Adaptive Local Signal Mixing (A-LIX) layers; ii) WayConv1D, which is a waypoint encoder that harnesses the 2D geometrical information of the waypoints using 1D convolutions; and iii) an auxiliary loss to increase the significance of the traffic lights in the latent representation of the environment. Experimental results show that RLAD significantly outperforms all state-of-the-art RLfP methods on the NoCrash benchmark. We also present an infraction analysis on the NoCrash-regular benchmark, which indicates that RLAD performs better than all other methods in terms of both collision rate and red light infractions. The source code of RLAD is available at https://github.com/DanielCoelho112/rlad.

3.1 Introduction

In recent years, Autonomous Driving (AD) has experienced significant growth due to advancements in artificial intelligence and information sensing, which have received widespread attention in both academia and industry [43]. In general terms, AD involves tasks that fall into two main categories: environment perception and driving policy [14, 15, 126]. First, the autonomous agent must derive a useful representation of the environment from sensor data, and then generate the appropriate control commands based on the driving policy in order to keep the vehicle on a safe route.

Urban driving is one of the most challenging environments for autonomous vehicles, mainly due to the unpredictability and diversity of agents present in the environment, as well as complex situations, such as pedestrians crossing lanes, traffic lights, intersections, among others [43, 111]. Due to its complexity, researchers have shifted their focus to end-to-end methods (*e.g.*, Imitation Learning (IL) and Reinforcement Learning (RL)), instead of modular pipelines [75].

Imitation Learning (IL) learns a driving policy from a dataset of expert demonstrations using supervised learning techniques [24], in which the goal is to create an agent that behaves as similarly as possible to the expert. The major limitations of this method are that the driving policy is limited to the performance of the experts, and it is practically inconceivable

to collect expert data covering all possible driving situations [43]. IL algorithms also suffer from a distribution mismatch in the training data, *i.e.*the algorithm will never encounter failing situations, and therefore, will not react properly in those conditions [111, 127].

Conversely, Reinforcement Learning (RL) learns a driving policy by interacting directly with an environment and collecting rewards that assess the suitability of an action taken in a given state [38]. Usually, the goal of the agent is to maximize the cumulative rewards. As in this case the agent is interacting directly with the environment, it does not suffer from a distribution mismatch and is also not limited to the performance of an expert. However, due to the extensive exploration of the environment during the training stage, RL is known to have a poor sample efficiency, requiring an order of magnitude more data than IL to converge [38].

Reinforcement Learning from Pixels (RLfP) is a type of RL that directly maps the image data into actions. This requires simultaneously training a convolution encoder alongside a policy network, which is a challenging task due to the sample efficiency problem [128]. Additionally, it is known that performing Temporal Differences (TD) learning with a convolutional encoder leads to unstable training and premature convergence, which eventually results in degenerated feature representations [129].

Existing RLfP approaches have been applied on Atari games [130] and MuJoCo [131] tasks, which present significantly fewer challenges in terms of environment perception when compared to AD. For instance, in Atari and MuJoCo, practically any change in the observation space is task-relevant, whereas in AD the observation space contains predominately task-irrelevant information, as is the case of clouds and architectural details [132]. To bypass this problem, current RL approaches applied in urban AD focus on decoupling the perception training from the driving policy training [24, 38, 75, 101, 111, 133]. The idea is to train an encoder using supervised or unsupervised techniques to derive a latent representation from the sensor data, and then train an RL algorithm that maps the latent representation into actions. This adds stability to the optimization by circumventing dueling training objectives. However, it leads to suboptimal policies because the encoder may not be aligned with the downstream task [35]. Since the objective is to maximize the cumulative rewards, it is beneficial to use them to improve simultaneously the feature representation of the sensor data and the driving policy network [132].

This paper proposes RLAD, a **R**einforcement **L**earning from Pixels **A**utonomous **D**riving agent, capable of driving under complex urban environments. This is the first approach to carry out a successful simultaneous training of the encoder and policy network using RL in the domain of vision-based urban AD. We leverage the latest advancements in RLfP that have been achieved by Meta AI[1] and propose techniques to integrate those advancements in the urban AD domain. Overall, we summarize our main contributions as follows:

- We propose RLAD, the first method that learns simultaneously the encoder and the

[1] https://ai.facebook.com/

driving policy network using RL in the domain of vision-based urban AD. We also show that RLAD significantly outperforms all state-of-the-art RLfP methods in this domain;

- We introduce an image encoder that leverages both image augmentations and Adaptive Local Signal Mixing (A-LIX) layers to minimize the catastrophic self-overfitting of the encoder;
- We propose WayConv1D, a waypoint encoder that leverages the 2D geometrical information of the waypoints using 1D convolutions with a 2×2 kernel, which significantly improves the stability of the driving;
- We perform a comparative analysis of the state-of-the-art RLfP in the domain of vision-based urban AD, where we show that one of the main challenges is obeying traffic lights. To address this limitation, we incorporate an auxiliary loss that specifically targets the traffic light information in the latent representation of the image, thereby enhancing its significance.

3.2 Related Work

3.2.1 Reinforcement Learning for Autonomous Driving

RL has been used in AD to overcome the limitations of IL, however, vision-based RL, or more precisely RLfP, comes with several drawbacks [38]. Camera images are of high dimensions, thus requiring larger RL networks and optimizing dueling training objectives: the image encoder and the policy network [35]. To overcome these limitations, the common approach is to disentangle the perception network from the policy network and perform a two-stage training [24, 38, 75, 101, 111, 133]. The first stage consists of encoding the sensor data in a latent representation by pretraining a visual encoder on visual tasks, such as classification and segmentation [38]. Then, the latent representations are received by an RL algorithm to train the driving policy network. Following this line, Toromanoff *et al.* [111] proposed a method called *Implicit Affordances*. First, a visual encoder is trained using auxiliary tasks, such as traffic light state and distance, road type, semantic segmentation classification, among others. Then the visual encoder is frozen, and an RL algorithm (Rainbow-IQN Ape-X [134]) is used to train the policy network on the latent representation. Ahmed *et al.* [75] also used the concept of affordances, but went even further and used the affordances themselves as the input of the RL algorithm. More recently, Zhao *et al.* [133] proposed CADRE, a cascade RL framework for vision-based urban autonomous driving. Their method first trains offline a co-attention perception module to learn relationships between the input images and the corresponding command controls from a driving dataset. This module is then frozen and is used as the input of an efficient distributed Proximal Policy Optimization (PPO) [135] that learns the driving policy network online [133]. The usage of the two-stage training allowed

these approaches to use large image encoders to derive a more complex representation of the environment from the sensor data. However, one can argue that the obtained representation may not be totally aligned with the downstream task since it was not trained jointly with the driving policy network. RLfP aims to fix this limitation by updating the image encoder alongside the driving policy network. It is a method that is receiving massive attention in recent years and can offer numerous benefits to vision-based urban AD.

3.2.2 Reinforcement Learning from Pixels

Sample-efficient RL algorithms capable of training directly from pixel observations could open up a multitude of real-world applications [128]. However, simultaneously training an image encoder and a policy network is a challenging problem due to the strong correlation between samples, sparse reward signal, and degenerated feature representations [128,129,136]. Naive approaches that use a large image encoder result in severe overfitting, and a smaller image encoder usually produces impoverished representations which limit the performance of the agent [128]. One way of addressing this problem is to employ auxiliary losses. Shelhamer *et al.* [136] proposed to use several auxiliary losses to enhance the performance of Asynchronous Advantage Actor Critic (A3C) [137]. Zhang *et al.* [132] predicted the rewards and dynamics of the environment as auxiliary losses. Yarats *et al.* [35] proposed SAC+AE, where the authors demonstrated that combining the off-policy RL algorithm Soft Actor-Critic (SAC) [138] with pixel reconstruction is vital for learning good representations. Following this line, Srinivas *et al.* [139] proposed CURL – Contrastive Unsupervised Representations for Reinforcement Learning. CURL uses contrastive learning to maximize agreement between an augmented version of the same observation, and to minimize agreement between different observations [139]. The authors showed that this method significantly improves the sample efficiency of the algorithm. A different line of research was proposed by Kostrikov *et al.* [128], where the authors proposed DrQ. This work demonstrated how image augmentations can be applied in the context of model-free off-policy RL algorithms. The authors proved that using image augmentations leads to better results than using auxiliary losses [128]. Finally, Yarats *et al.* [140] proposed an improved version of DrQ, named DrQ-V2. This version is the result of several algorithmic changes: (i) changing from SAC to Deep Deterministic Policy Gradient (DDPG) [141], (ii) incorporating multi-step return, (iii) improving the data augmentation technique, (iv) introducing an exploration schedule, (v) selecting better hyper-parameters [140].

3.3 Method

RLAD is the first RLfP method applied to the domain of urban AD. Its main purpose is to derive a feature representation from the sensor data that is fully aligned with the driving task

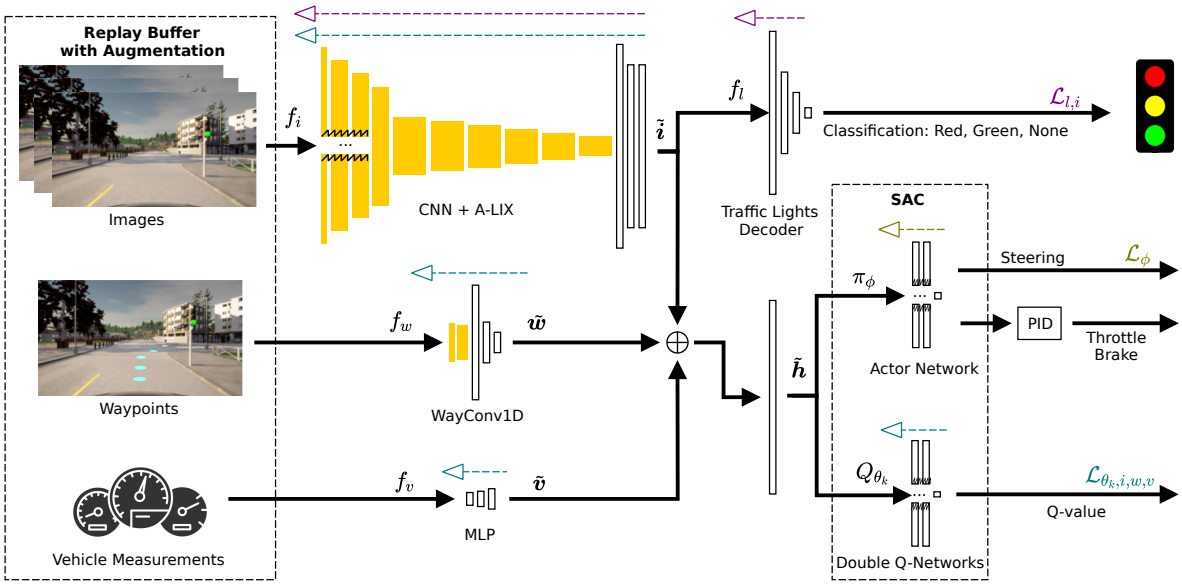


Figure 3.1: Architecture of RLAD. As input, the system receives K consecutive central images, N waypoints computed using a global planner, and measurements from the vehicle from the last K steps. Each input is processed independently by a different encoder. The latent representations of each input are then concatenated, forming the input of the SAC algorithm ($\tilde{\boldsymbol{h}} = \begin{bmatrix}\tilde{\boldsymbol{i}} & \tilde{\boldsymbol{w}} & \tilde{\boldsymbol{v}}\end{bmatrix}$). The actor network of the SAC, along with a PID, is responsible for outputting the command controls, while the Q-networks are responsible for outputting the value function. To guide $\tilde{\boldsymbol{i}}$ to contain information about the traffic lights, we add an auxiliary branch to perform traffic light classification. All elements of the neural networks are represented at scale. The dashed arrows provide a visual representation of how each loss function affects the parameters of the system in the backpropagation stage.

while learning the driving policy simultaneously. The core of RLAD is built upon DrQ [128], but with several modifications. First, in addition to the image augmentations, we also append at the end of each convolutional layer of the image encoder, a regularization layer called Adaptive Local Signal Mixing (A-LIX) [129] (more details in Section 3.3.2), which significantly improves the stability and efficiency of the training. Second, we performed an extensive study of the best hyperparameters, where we realized that some hyperparameters of DrQ weren't optimal for the AD domain. Finally, we use an additional loss for traffic light classification in order to guide the latent representation of the image ($\tilde{\boldsymbol{i}}$) to contain information about the traffic lights.

3.3.1 Learning Environment

The learning environment is defined as a Partially Observable Markov Decision Process (POMDP). The environment was built by using the CARLA driving simulator (version 0.9.10.1) [39].

State space $\mathcal{S}$ is defined by CARLA, containing the ground truth about the world. The agent has no access to the state of the environment.

Observation space At each step the state $s_t \in \mathcal{S}$ generates the corresponding observation $o_t \in \mathcal{O}$, which is conveyed to the agent. An observation is a stack of 3 sets of tensors from the last K timesteps ($K = 3$). Specifically, $o_t = \{(\boldsymbol{I}, \boldsymbol{W}, \boldsymbol{V})_k\}_{k=0}^{2}$, where: $\boldsymbol{I}$ is a $3\times256\times256$ image, $\boldsymbol{W}$ corresponds to the 2D coordinates related to the vehicle, for the next $N = 10$ waypoints provided by the global planner from CARLA, and $\boldsymbol{V}$ corresponds to a two-dimensional vector containing the current speed and steering of the vehicle.

Action Space $\mathcal{A}$ is composed of three continuous actions: throttle, which ranges from 0 to 1; brake, which ranges from 0 to 1; and steering, which ranges from -1 to 1.

Reward function We used the reward function defined in [42] because it has been shown to accurately guide the AD training.

Training We used CARLA at 10 FPS. Similarly to [42], at the beginning of the episode, the start and target locations are randomly generated and the desired route is computed using the global planner. When the target location is reached, a new random target location is computed. The episode is terminated if one of the following conditions is met: collision, running a red traffic light, blocked, or if a predefined timeout is reached.

3.3.2 Agent Architecture

The architecture of RLAD is depicted in Figure 3.1. In general, our system has three main components: an encoder (Section 3.3.2), an RL algorithm (Section 3.3.2), and an auxiliary loss (Section 3.3.2). To simplify the longitudinal control and ensure smooth control, we reparameterize the throttle and brake commands using a target speed. As such, a PID controller is appended at the end of the actor network that produces the corresponding throttle and brake commands that match the predicted target speed.

Encoder

The encoder is responsible for transforming the data from the sensors (o_t) into a low-dimension feature vector ($\tilde{\boldsymbol{h}}_t$) to be processed by the RL algorithm.

Image Encoder As demonstrated in [128], the size of the image encoder is a critical element in an RLfP method. Due to the weak signal of the RL loss, encoders commonly used in AD methods, such as ResNet50 [112] ($\sim$ 25M parameters) or Inception V3 [142] ($\sim$ 27M parameters), are impracticable. On the other side, small encoders, designed for scenarios of

smaller complexity, such as IMPALA [143] ($\sim$ 200k parameters), cannot produce an adequate representation of the environment, which limits the performance of the driving agent. A comparison of different encoder sizes and their performance in terms of return can be seen in Figure 3.5. Our findings suggest that the optimal configuration entails a trade-off between larger networks that are unsuitable for training with RL and smaller networks that cannot accurately perceive the environment. The architecture of the proposed image encoder is shown in Table 3.1, containing around 1M parameters. Similarly to DrQ and DrQ-V2, we leverage simple image augmentations to regularize the value function [128,140]. First, we apply padding to each side of the 256×256 image by repeating the 8 boundary pixels and then selecting a random crop of 256×256. As in [140], we found it useful to apply bilinear interpolation on the cropped images. In addition to the image augmentations, we also found that appending an A-LIX layer [129] at the end of each convolution layer improves the performance of the agent, possibly by preventing a phenomenon called catastrophic self-overfitting (spatially inconsistent feature maps that lead to discontinuous gradients in the backpropagation). A-LIX is applied on the features produced by the convolution layers $a \in \mathbb{R}^{C \times H \times W}$, by randomly mixing each component a_{cij} with its neighbors belonging to the same feature map. Consequently, the output of A-LIX is of the same dimensionality as the input, but with the difference that the computation graph minimally disrupts the information of each feature a_{cij}, while smoothing discontinuous component of the gradients signal during backpropagation [129]. Hence, this technique works by enforcing the image encoder to produce feature maps that are spatially consistent and thus minimizing the effect of the catastrophic self-overfitting phenomenon. This process can be succinctly summarized as $\tilde{\boldsymbol{i}}_t = f_i(\text{aug}([\{\boldsymbol{I}_{t-k}\}_{k=0}^{2}]))$, where f_i is the image encoder, aug corresponds to the data augmentation applied, and $\tilde{\boldsymbol{i}}_t$ corresponds to the latent representation of the stack of three consecutive images $([\{\boldsymbol{I}_{t-k}\}_{k=0}^{2}])$.

Waypoint Encoder Usually, the waypoint encoder consists of using the mean orientation between the current pose of the agent and the next N waypoints [24] or flattening the waypoints' 2D coordinates into a vector and then applying an MLP [115]. In our point of view, both approaches have serious limitations. The former approach clearly oversimplifies the problem by encoding all waypoint coordinates into a single value. This method only works for small values of N, because as N increases, the waypoints become more scattered, and thus the mean orientation ceases to be a reliable indicator. Although the latter approach works for all values of N, by flattening the 2D waypoint coordinates into a vector, the 2D geometrical information is not being used. To overcome both limitations, we propose WayConv1D, a waypoint encoder that leverages the 2D geometrical structure of the input by applying 1D convolutions with a 2×2 kernel over the 2D coordinates of the next N waypoints. The output of the 1D convolution is then flattened and processed by an MLP. This process can be summarized as $\tilde{\boldsymbol{w}}_t = f_w(\boldsymbol{W}_t)$, where f_w corresponds to the WayConv1D, and $\tilde{\boldsymbol{w}}_t$ corresponds to the latent representation of the waypoints for the current step ($\boldsymbol{W}_t$). We found that with

Table 3.1: Architecture of the proposed image encoder. After each convolution layer, we applied the ReLU function [144] and the A-LIX regularizer layer [129].

type	kernel/stride	input size
conv	3×3/2	9×256×256
conv	3×3/2	32×127×127
conv	3×3/2	32×63×63
conv	3×3/2	32×31×31
conv	3×3/1	64×15×15
conv	3×3/1	64×13×13
conv	3×3/1	64×11×11
conv	3×3/1	64×9×9
conv	3×3/1	64×7×7
flatten	-	64×5×5
linear	-	1600
layernorm	-	256
tanh	-	256

WayConv1D, the agent learns more efficiently to follow the trajectory without oscillating near the center of the lane. This is a common issue encountered when utilizing RL in the urban AD domain, as documented in previous studies [42, 111].

Vehicle Measurements Encoder Similarly to [115], we apply an MLP to the vehicle measurements: $\tilde{\boldsymbol{v}}_t = f_v([\{\boldsymbol{V}_{t-k}\}_{k=0}^{2}])$, where f_v is the MLP, and $\tilde{\boldsymbol{v}}_t$ corresponds to the latent representation of the concatenation of the vehicle measurements across three steps $([\{\boldsymbol{V}_{t-k}\}_{k=0}^{2}])$.

RL Algorithm

As the RL algorithm, we use the SAC [138], which is a model-free off-policy actor-critic algorithm that learns two Q-functions Q_{θ_1}, Q_{θ_2}, a stochastic policy π_ϕ, and a temperature α to find an optimal policy by optimizing a γ-discounted maximum-entropy objective [128, 145]. The actor policy $\pi_\phi(a_t \mid \tilde{\boldsymbol{h}}_t)$ is a parametric tanh-Gaussian that given $\tilde{\boldsymbol{h}}_t = \begin{bmatrix}\tilde{\boldsymbol{i}}_t & \tilde{\boldsymbol{w}}_t & \tilde{\boldsymbol{v}}_t\end{bmatrix}$, samples $a_t = \tanh\left(\mu_\phi(\tilde{\boldsymbol{h}}_t) + \sigma_\phi(\tilde{\boldsymbol{h}}_t)\epsilon\right)$, where $\epsilon \sim \mathcal{N}(0,1)$, and μ_ϕ and σ_ϕ are the parametric mean and standard deviation. The double Q-networks are learned by optimizing a single step of the soft Bellman residual:

$$\mathcal{L}_{\theta_k,i,w,v} = \mathbb{E}_{\substack{o_t,a_t,o_{t+1}\sim\mathcal{D} \\ a'_{t+1}\sim\pi_\phi(\cdot|\tilde{\boldsymbol{h}}_{t+1})}} \left[\left(Q_{\theta_k}\left(\tilde{\boldsymbol{h}}_t, a_t\right) - y\right)^2\right], \forall k \in \{1,2\}, \tag{3.1}$$

with the TD target y defined as:

$$y = r_t + \gamma\left(\min_{k=1,2} Q_{\bar{\theta}_k}\left(\tilde{\boldsymbol{h}}_{t+1}, a'_{t+1}\right) - \alpha \log \pi_\phi\left(a'_{t+1} \mid \tilde{\boldsymbol{h}}_{t+1}\right)\right), \tag{3.2}$$

where $\mathcal{D}$ represents the replay buffer, r_t is the reward received, γ is the discount factor, and $Q_{\bar{\theta}_1}$ and $Q_{\bar{\theta}_2}$ denote the Exponential Moving Average (EMA) of the parameters of Q_{θ_1} and Q_{θ_2}, respectively. Similarly to DrQ-V2, we found it useful to use a single encoder, rather than a main encoder and an EMA of the main encoder. The policy is updated to maximize the expected future return plus the expected future entropy:

$$\mathcal{L}_\phi = -\mathbb{E}_{\substack{o_t \sim \mathcal{D} \\ a_t \sim \pi_\phi(\cdot|\tilde{\boldsymbol{h}}_t)}}\left[\min_{k=1,2} Q_{\theta_k}\left(\tilde{\boldsymbol{h}}_t, a_t\right) - \alpha \log \pi\left(a_t \mid \tilde{\boldsymbol{h}}_t\right)\right]. \tag{3.3}$$

Finally, the temperature α is learned using the loss proposed by Haarnoja *et al.* [146].

As noted in Equation 3.1 and 3.3, not all losses are propagated to the encoders. Following [35], we block the actor's gradients from propagating to the encoder. In contrast with DrQ-V2, we found that using a learning rate of 10^{-3}, instead of 10^{-4}, results in a faster and more stable training. One intuition to explain this improvement is related to the observation space. DrQ-V2 was evaluated in tasks where the observation space is likely task-relevant, whereas in AD the observation space contains task-irrelevant information, such as clouds and buildings. Thus, with a larger learning rate, the encoder will learn faster to distinguish the task-relevant objects from the non-relevant, which prevents the agent from exploring using unreliable representations of the environment.

Auxiliary Loss

Based on initial experiments, we observed that the agent struggled to associate the color of the traffic light with the negative reward incurred when passing through a red light. This is understandable, particularly when we take into account that the traffic light color occupies only a small fraction of the entire image. To address this issue, we implemented an auxiliary loss that strengthens the significance of traffic light information in the latent representation of the image ($\tilde{\boldsymbol{i}}$). As such, we added a traffic light decoder (f_l) to the end of the image encoder and performed traffic light classification using three classes ($C = 3$): *None*, *Red*, and *Green*. *None* signifies that there is no traffic light within the vicinity of the agent, *Red* indicates the presence of a red or yellow traffic light near the agent, and finally, *Green* denotes a green traffic light near the agent. A traffic light is considered relevant if it is within a distance of 18 meters from the agent. This threshold ensures that the agent reacts in a timely manner to the traffic light's state, enabling safe and compliant navigation through intersections. Every time we sample a batch of transitions from the replay buffer, we perform traffic light classification

using the cross-entropy loss:

$$\mathcal{L}_{l,i} = -\sum_{b=1}^{B}\sum_{c=1}^{C} \log\left(\frac{e^{(x_{b,c})}}{\sum_{i=1}^{C} e^{(x_{b,i})}}\right) y_{b,c}, \tag{3.4}$$

where B is the batch size, x corresponds to the logits outputted by f_l, and y corresponds to the ground truth class. In the backpropagation stage, this loss updates both the parameters of the traffic light decoder and the parameters of the image encoder.

3.4 Experiments

In this section, we compare RLAD with the state-of-the-art RLfP methods, applied to the domain of urban AD. First, we define the setup of the experiments, and then compare RLAD with the state-of-the-art methods both in terms of expected return and using specific metrics related to urban AD. Finally, we present an ablation study that guided the development of RLAD.

3.4.1 Setup

Benchmark The methods are evaluated on the NoCrash benchmark [91]. This benchmark considers generalization from Town 1, a town composed of one-lane roads and T-junctions with traffic lights, to Town 2, which is a scaled-down version of Town 1 with different textures. The training is performed using four training weather types, and the testing uses two different weather types. This benchmark has three levels of traffic density (empty, regular, and dense) according to the number of vehicles and pedestrians. The results are reported in terms of success rate, which is the percentage of routes completed without collision. Additionally, we also report information related to the percentage of route completion, red light infractions, collisions with vehicles, pedestrians, and layout, and blockages per kilometer.

Training Details All algorithms are trained on the same hardware: a single NVIDIA RTX 2080 Ti. The algorithms are trained for 10^6 environment steps and are evaluated every 20 000 environment steps. Each evaluation query averages episode returns over 10 episodes. The Deep Learning library used was PyTorch [147]. The list with the main hyperparameters used is present in Table 3.2.

Baselines Given that we are the first to propose an RLfP method applied in urban AD, we compare our method with the state-of-the-art RLfP methods: SAC+AE [35], CURL [139], DrQ [128], and DrQ-V2 [140]. The official implementation of these methods only uses images as input, so we added two additional encoders: a waypoint encoder similar to [115], and a vehicle measurement encoder similar to ours. These encoders were selected because they

Table 3.2: List of the hyperparameters used by RLAD.

Parameter	Value
Replay Buffer capacity	100000
Batch size	256
Action repeat	2
Discount factor γ	0.99
Optimizer	Adam [148]
Learning rate	10^{-3}
Critic target update frequency	1
Critic Q-function soft update rate	0.01
Critic update frequency	1
Actor update frequency	1
Auxiliary loss update frequency	1
$\dim(\tilde{\boldsymbol{i}})$	256
$\dim(\tilde{\boldsymbol{w}})$	32
$\dim(\tilde{\boldsymbol{v}})$	16
SAC networks size	1024
Actor log stddev bounds	[-10,2]
Init temperature	0.1

are the most commonly utilized in the end-to-end AD field. For a fair comparison, we also reparameterize the throttle and brake commands using a PID controller.

3.4.2 Comparison with Baselines

Figure 3.2 depicts the average return for each method during the NoCrash benchmark's training process. It is evident that RLAD significantly outperforms all state-of-the-art methods. By the end of the training, RLAD manages to attain an average return that is roughly three times greater than all other methods.

Table 3.3 shows the performance of the algorithms in terms of success rate for every task of the NoCrash benchmark under testing conditions. In the empty task, all algorithms perform reasonably well, with the exception of DrQ-V2. However, as the difficulty of the task increases, the difference between the performance of RLAD and the other methods becomes more evident. Although RLAD performs equally to DrQ in the empty task, it outperforms the second best method by 50 % in the regular task and by 220 % in the dense task. One important observation arising from the table is the consistent decline in performance across all methods as we transition from empty to dense traffic scenarios. A significant factor contributing to

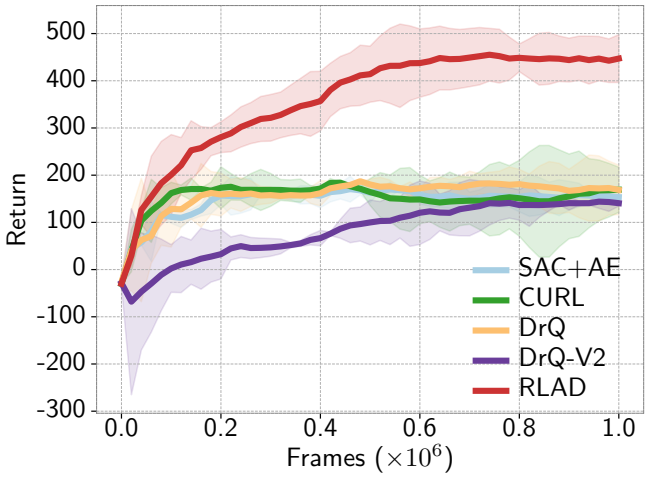


Figure 3.2: Comparison of RLAD with state-of-the-art RLfP methods in terms of average return per episode on the NoCrash benchmark. The solid lines represent the mean performance over 3 seeds, and the shaded regions represent 90 % confidence intervals.

this trend is inherent to the RLfP paradigm. As we train the convolutional encoder in tandem with the driving policy network, the process becomes notably intricate. The complexities of blending RL losses with the task of training a convolutional encoder lead us to deploy smaller encoders to manage training stability. While these encoders suffice in empty traffic scenarios, they struggle with the elevated demands of regular and dense traffic environments, causing difficulties in capturing the intricacies of the environment.

Table 3.3: Success rate (%) on NoCrash benchmark for each task in testing conditions (Town 2 with new weather). The results for each method correspond to the best seed considering the average episode return (Figure 3.2). Best scores for each task highlighted in bold.

	Empty	Regular	Dense
SAC+AE	82	42	6
CURL	74	30	2
DrQ	**94**	42	10
DrQ-V2	10	8	0
RLAD	**94**	**62**	**32**

Table 3.4 provides a detailed analysis of the performance of the methods with respect to AD-related metrics, specifically through an infraction analysis conducted on the regular task of the NoCrash benchmark. With the exception of DrQ-V2, all state-of-the-art RLfP methods achieve a route completion over 90 %, but achieve a success rate of less than 50 %.

This clearly shows that the state-of-the-art RLfP methods are very good at following the trajectory generated by the global planner, but struggle to deal with dynamic obstacles, as is the case of vehicles and pedestrians. In contrast, RLAD is capable of dealing with dynamic obstacles, achieving the best score for all metrics related to collisions. The scores related to the red light infractions clearly demonstrate that obeying the traffic lights is a challenging task for RLfP algorithms. RLAD with the auxiliary loss is able to perform 17 % better when compared with the second best, but still very poorly when compared with state-of-the-art methods that use IL [2,95,149]. This limitation arises from the impracticability of using large image encoders in RLfP, which makes it challenging to create representations of the environment that include small yet important features, such as the color of traffic lights. Among all methods, DrQ-V2 is the one that performs worse in virtually all metrics. Internal investigations showed that the primary reason for this performance was the RL algorithm used - DDPG. Using our training conditions, DDPG quickly converges to a suboptimal policy, where the agent tends to remain still in various situations. This problem can be easily identified in the column related to the blockages of DrQ-V2: 22.14 blockages per kilometer.

Although RLAD outperforms all RLfP methods in the urban AD domain, it is not yet competitive with state-of-the-art RL methods that decouple the training of encoder and the policy network [75,111,133]. However, in the field of continuous control tasks in the MuJoCo simulator [150], RLfP methods have already surpassed those that decouple the encoder from the policy network, which suggests that the same pattern may occur in the urban AD domain as well. Furthermore, the image encoder of state-of-the-art RL methods that decouple the encoder from the policy network contains around 25 times more parameters than the image encoder of RLAD [75,111,133], requiring more computation power and resources, which may compromise their application in real-time settings.

Table 3.4: Driving performance and infraction analysis on the NoCrash benchmark, using the regular task in testing conditions. The results for each method correspond to the best seed considering the average episode return (Figure 3.2). Best scores are highlighted in bold.

	Success rate %, ↑	Route completion %,↑	Collision pedestrian #/Km, ↓	Collision vehicle #/Km, ↓	Collision layout #/Km, ↓	Red light infraction #/Km ↓	Agent blocked #/Km, ↓
SAC+AE	42	98	0.60	1.71	0.99	6.16	0.23
CURL	30	99	0.97	1.91	1.68	6.81	0.20
DrQ	42	**100**	0.77	1.51	0.19	7.35	**0.00**
DrQ-V2	8	53	0.58	1.68	1.63	7.04	22.14
RLAD	**62**	94	**0.41**	**0.71**	**0.16**	**5.10**	0.84

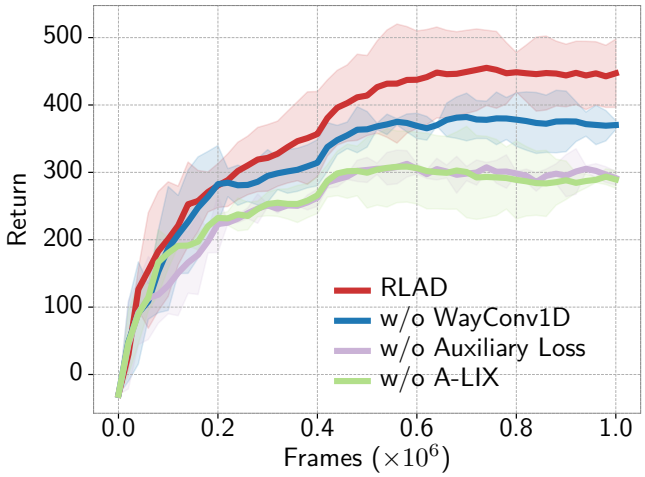


Figure 3.3: Ablation study in terms of average return per episode. The solid lines represent the mean performance over 3 seeds, and the shaded regions represent 90 % confidence intervals.

3.4.3 Ablation Study

We performed three experiments: using the waypoint encoder of [115] instead of WayConv1D; removal of the auxiliary loss; and removal of the A-LIX layers. Figure 3.3 shows the influence of each of these components in terms of the average return per episode. The most impactful components are the auxiliary loss and the A-LIX layers. Removing them results in a performance decrease of 33 % in terms of return. Replacing the WayConv1D by the waypoint encoder of [115] results in a performance decrease of 15 %. While the effect of WayConv1D on return might seem less pronounced, its role is vital for the overall driving experience. Figure 3.3 illustrates the horizontal distances to the center lane for both RLAD and RLAD without WayConv1D. With RLAD, the distribution is narrower, suggesting that the vehicle maintains a consistent position closer to the center of the lane. In contrast, the wider distribution for RLAD without WayConv1D indicates the vehicle experiences more lateral oscillations around the center of the lane.

Table 3.5 provides an in-depth look at driving performance and infractions based on the ablation study, offering insights into the direct impact of each component. In line with Figure 3.3, RLAD without WayConv1D experienced the slightest decline in performance. Notably, the removal of the auxiliary loss in the RLAD method resulted in a significant dip in performance. This was particularly evident with a 1.85 increase in red light infractions per kilometer. This outcome aligns with expectations since the auxiliary loss is designed to enhance the significance of traffic light information in the latent representation. Meanwhile, excluding A-LIX from RLAD also led to a noticeable drop in performance, primarily evidenced by a rise in collisions per kilometer.

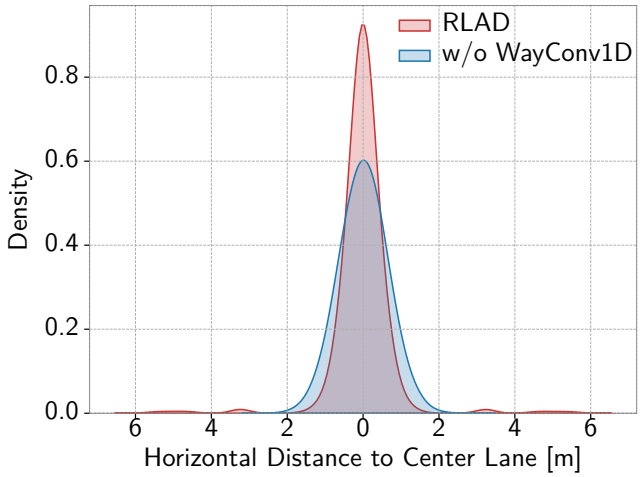


Figure 3.4: Distribution of horizontal distances to the center lane using the best seed considering the average episode return (Figure 3.3). Distributions were computed using the regular task in testing conditions.

Table 3.5: Ablation study: driving performance and infraction analysis on the NoCrash benchmark, using the regular task in testing conditions.

	Success rate %, ↑	Route completion %,↑	Collision pedestrian #/Km, ↓	Collision vehicle #/Km, ↓	Collision layout #/Km, ↓	Red light infraction #/Km ↓	Agent blocked #/Km, ↓
w/o WayConv1D	58	90	0.46	1.01	0.17	**5.04**	1.20
w/o Auxiliary Loss	49	92	0.56	1.25	0.20	6.95	1.43
w/o A-LIX	51	**94**	0.59	1.43	0.24	5.12	0.90
RLAD	**62**	**94**	**0.41**	**0.71**	**0.16**	5.10	**0.84**

3.5 Conclusion

This paper introduced RLAD, the first algorithm that learns simultaneously the encoder and the driving policy network using Reinforcement Learning (RL) in the domain of vision-based urban Autonomous Driving (AD). Our method significantly outperforms all RLfP state-of-the-art methods in this domain. Although our method is not yet competitive with the state-of-the-art methods in end-to-end urban AD, we believe that RLAD can foster the interest in applying RLfP to the domain of urban AD. Methods that learn simultaneously the encoder and the policy network have demonstrated better performance in the continuous control tasks in the MuJoCo simulator, compared to those that decouple the encoder from the policy network. Based on this, we have grounds to expect that a comparable pattern will emerge in

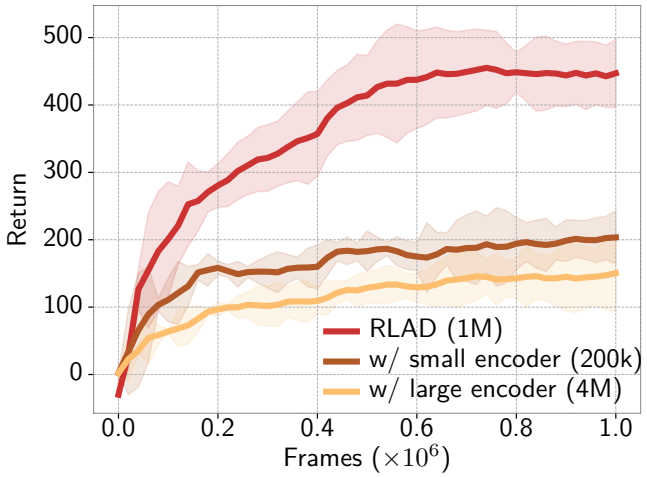


Figure 3.5: Comparison of different sizes of image encoders in terms of average return per episode on the NoCrash benchmark. The solid lines represent the mean performance over 3 seeds, and the shaded regions represent 90 % confidence intervals.

the realm of urban AD, and we believe that RLAD constitutes the first step toward this end.

Chapter 4

RLfOLD: Reinforcement Learning from Online Demonstrations in Urban Autonomous Driving

Daniel Coelho, Miguel Oliveira, and Vitor Santos. "RLfOLD: Reinforcement Learning from Online Demonstrations in Urban Autonomous Driving." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 38. No. 10. 2024, doi: 10.1609/aaai.v38i10.29049.

Abstract: Reinforcement Learning from Demonstrations (RLfD) has emerged as an effective method by fusing expert demonstrations into Reinforcement Learning (RL) training, harnessing the strengths of both Imitation Learning (IL) and RL. However, existing algorithms rely on offline demonstrations, which can introduce a distribution gap between the demonstrations and the actual training environment, limiting their performance. In this paper, we propose a novel approach, Reinforcement Learning from Online Demonstrations (RLfOLD), that leverages online demonstrations to address this limitation, ensuring the agent learns from relevant and up-to-date scenarios, thus effectively bridging the distribution gap. Unlike conventional policy networks used in typical actor-critic algorithms, RLfOLD introduces a policy network that outputs two standard deviations: one for exploration and the other for IL training. This novel design allows the agent to adapt to varying levels of uncertainty inherent in both RL and IL. Furthermore, we introduce an exploration process guided by an online expert, incorporating an uncertainty-based technique. Our experiments on the CARLA NoCrash benchmark demonstrate the effectiveness and efficiency of RLfOLD. Notably, even with a significantly smaller encoder and a single-camera setup, RLfOLD surpasses state-of-the-art methods in this evaluation. These results, achieved with limited resources, highlight RLfOLD as a highly promising solution for real-world applications.

4.1 Introduction

Urban Autonomous Driving (AD) is considered a challenging and critical task. In order to navigate effectively, agents must analyze a highly intricate environment and continuously make real-time decisions to adhere to driving regulations, while also interacting with other dynamic agents like drivers and pedestrians [43]. Consequently, researchers have been redirecting their efforts from rule-based methods to end-to-end learning approaches.

End-to-end learning methods can be divided into two categories: Imitation Learning (IL) and Reinforcement Learning (RL). In IL, an agent learns a task by imitating an expert's behavior, leveraging expert demonstrations as ground truth. The main advantage of IL is the ability to rapidly learn from expert knowledge, accelerating the learning process and acquiring safe and efficient behaviors [151]. However, agents trained with IL may face challenges in generalizing to unseen scenarios, as they tend to be biased towards the demonstrated behavior [38]. On the other hand, RL involves learning through trial and error, where an agent explores the environment, receives feedback in the form of rewards, and gradually improves its policy. A key advantage of RL is its ability to handle unknown situations. However, RL often requires extensive exploration and can be sample-inefficient, requiring significant time and resources for learning [152].

Reinforcement Learning from Demonstrations (RLfD) seeks to harness the benefits of both IL and RL by integrating expert demonstrations into the RL training. Thus offering a

significant boost in sample efficiency compared to standalone RL [151]. This enhancement stems from the ability of expert demonstrations to minimize the required interactions with the environment for learning the desired behavior. Moreover, the expert demonstrations provide valuable insights that enable the agent to explore the state-action space more effectively [36].

Despite the recent advancements of RLfD [153,154], the conventional approach of collecting a demonstration dataset has inherent limitations. One major drawback is the requirement of pre-collecting a dataset, which can be a laborious and time-consuming process. In complex domains, like urban AD, it can be particularly arduous to ensure the dataset's diversity and coverage of various scenarios and edge cases. Moreover, the reliance on an offline dataset introduces a potential distribution gap between the demonstrations and the training environment, hindering the agent's ability to generalize effectively [38]. One known strategy to mitigate the distribution mismatch between offline datasets and the training environment is the DAGGER algorithm [31], which iteratively refines policies by aggregating training data across a mixture of expert and learner-induced distributions. However, while DAGGER reduces the distribution gap, it does not inherently account for the uncertainty in decision-making, which can be critical in dynamic and unpredictable urban driving scenarios.

To tackle the limitations of traditional RLfD, we introduce RLfOLD. RLfOLD utilizes online demonstrations, collected using privileged information from the simulator, which circumvents the need for a pre-collected dataset. These demonstrations are seamlessly integrated into the agent's replay buffer, ensuring that the agent learns from up-to-date and pertinent scenarios, thereby effectively bridging the distribution gap. Furthermore, RLfOLD innovates by merging IL with RL through a dual standard deviation policy network. By employing different standard deviations, the algorithm can adapt to the varying levels of uncertainty inherent in RL and IL. Inspired by recent works [155–159], our approach further refines the exploration process. It empowers the agent with the ability to selectively invoke expert guidance when faced with high uncertainty, enhancing the decision-making process and potentially leading to more effective learning.

Overall, we summarize our main contributions as follows:

- Introduce RLfOLD, a novel approach that seamlessly integrates IL and RL by leveraging online demonstrations, effectively bridging the distribution gap between demonstration and training environments;
- Propose a policy network that outputs two standard deviations, enabling adaptive control for exploration and IL training while considering uncertainty in both domains;
- Incorporate an uncertainty-based technique guided by an online expert to enhance the exploration process;
- Conduct extensive experiments on the NoCrash benchmark, which demonstrate the

superior effectiveness and efficiency of RLfOLD, surpassing state-of-the-art methods even with reduced resources.

The source code of RLfOLD is available at https://github.com/DanielCoelho112/rlfold.

4.2 Related Work

While RLfOLD is applicable to various tasks, our focus is on testing RLfOLD in the context of urban AD using the CARLA simulator [39]. As such, this section is focused on the application of IL, RL, and RLfD methods within the CARLA environment.

4.2.1 Imitation Learning

IL methods aim to learn from an expert using offline demonstrations. In the domain of AD, various IL approaches have been proposed, and they have demonstrated significant success. Notably, IL-based methods have consistently achieved top performance in the CARLA Leaderboard [160], showcasing their effectiveness in tackling complex driving tasks. Early works include CIL [77] and CILRS [91] that employ a conditional architecture to activate different policies based on the navigation command received. LBC [74] and Roach [42] use privilege experts to provide knowledge to student models. Transfuser [95, 161] designs a multimodal transformer that fuses the front camera image and LiDAR data, and then a simple GRU to auto-regress navigation waypoints. LAV [149] trains on data from experiences collected not just from the ego-vehicle, but also from all surrounding vehicles. This is accomplished by learning a viewpoint-invariant spatial intermediate representation. TCP [3] proposes two branches that generate the planned trajectory and the multi-step control commands, respectively. Then the outputs of both branches are fused to achieve complementary advantages. Finally, InterFuser [2] uses a transformer to fuse multi-view sensors to encourage global contextual perception. Despite the remarkable achievements of IL approaches, a significant challenge in their deployment lies in addressing the distribution gap between the demonstration dataset and the environment in which the agent interacts.

4.2.2 Reinforcement Learning

RL has been used in AD to overcome the shortcomings of IL, however, vision-based RL presents several challenges. One such challenge is the training of a convolution encoder alongside a policy network, which often leads to catastrophic self-overfitting [44]. To address this issue, RLAD [44] proposes an image encoder that leverages both Adaptive Local Signal Mixing (A-LIX) [129] layers and image augmentations. While RLAD represents a significant advancement in vision-based RL for urban AD, its performance still falls short of the current state-of-the-art methods. The most successful RL algorithms applied in urban AD disentangle

the perception network from the policy network by performing two-stage training [44]. The first stage consists of encoding the sensor data in a latent representation by pretraining a large encoder on visual tasks, such as classification and segmentation [38]. Then, the latent representation is processed by an RL algorithm to train the policy network. Following this line, IAs [111] proposes an algorithm composed of two subsystems. First, the encoder is trained using auxiliary tasks. Then the encoder is frozen and an RL algorithm is trained on the encoder latent space. Another example of this disentanglement is CADRE [133]. This method first trains offline a co-attention perception module to learn the relationships between the input and the corresponding control commands from a dataset. Then, this module is frozen and is used to feed an efficient distributed Proximal Policy Optimization (PPO) that learns the driving policy. While RL overcomes the distribution gap limitation of IL, it often suffers from sample-inefficiency, requiring significant time and resources for learning.

4.2.3 Reinforcement Learning from Demonstrations

As stated in Section 4.1, both IL and RL have inherent limitations, which have led to the growing interest in the concept of RLfD over the years [151, 162, 163]. The main objective of RLfD is to combine the sample-efficiency of IL with the exploration capability of RL. For instance, CIRL [81] adopts a two-stage training approach. Initially, the agent is trained using IL with human demonstrations, followed by fine-tuning using an RL algorithm. BC-SAC [164] and SAC-IL [151] propose methodologies that integrate the Soft Actor-Critic (SAC) algorithm with the IL loss. GRIAD [38] combines IL and RL under the assumption that all expert demonstrations are optimal and therefore assigned with maximum rewards. With this assumption, they process the expert demonstrations indistinguishable from the experiences of the RL exploration agent. While this assumption is overly optimistic, GRIAD is able to achieve very satisfactory results in both the CARLA Leaderboard and the NoCrash Benchmark [91]. WOR [165] assumes the world to be on rails, meaning that the actions of the agent affect only its own state and do not influence the environment. With this assumption, they convert the problem into a tabular model-based RL setup and supervise the policy learning with offline demonstrations. While these approaches have shown promising results, they share a common limitation: the use of offline demonstrations, which can introduce a distribution gap between the demonstrations and the training environment. To address this limitation, we propose a novel approach called Reinforcement Learning from Online Demonstrations (RLfOLD). Our method leverages online demonstrations, obtained during the agent's exploration, to bridge the distribution gap and to guide the exploration of the agent.

4.3 Method

4.3.1 Learning Framework

The learning process follows a Partially Observable Markov Decision Process (POMDP). The environment was built using the CARLA driving simulator (version 0.9.10.1). At every timestep t, the environment generates an observation o_t, which is passed to the agent. An observation is defined as a stack of three sets of tensors from the last $K = 2$ timesteps. Specifically, $o_t = \{(\boldsymbol{I}, \boldsymbol{W}, \boldsymbol{V})_k\}_{k=0}^{1}$, where $\boldsymbol{I}$ represents a 3×256×256 image, $\boldsymbol{W}$ corresponds to the 2D coordinates with respect to the vehicle for the next $N = 10$ waypoints provided by the global planner from CARLA, and $\boldsymbol{V}$ is a two-dimensional vector containing the current speed and steering of the vehicle. The agent processes o_t and executes an action a_t according to its policy. Finally, the environment returns a reward r_t and the next observation o_{t+1}. The action a_t is composed of three continuous values: throttle and brake, which range from 0 to 1, and steering, which ranges from -1 to 1. Similar to [44], we parameterize the throttle and brake commands using a target speed. Specifically, we append a PID controlled at the end of the policy network to generate the throttle and brake commands that correspond to the predicted target speed.

Figure 4.1 illustrates the architecture of RLfOLD. At a high level, the system can be divided into three main parts: encoder, actor-critic algorithm with IL, and online expert. Additionally, an important part of this work consists of using the online expert to guide the exploration. In general, RLfD algorithms use two replay buffers: one for the exploration agent and one for the demonstration agent [38, 151, 164]. However, in our approach, we take advantage of having an online expert and create a single replay buffer, denoted as $\mathcal{D}$, to integrate information from both the exploration agent and the online expert. This replay buffer contains transitions in the form of $\{(o_t, a_t, a^*{}_t, r_t, o_{t+1})\}$, where a_t corresponds to the action executed by the agent, and a_t^* corresponds to action generated by the expert policy (π^*).

4.3.2 Encoder

As shown in Figure 4.1, RLfOLD trains simultaneously the encoder and the policy network. The reason is to ensure that the latent representations produced by the encoder are fully aligned with the driving task. However, as several studies have reported, performing Temporal Differences (TD) learning with a convolution encoder may lead to unstable training, premature convergence, and catastrophic self-overfitting [128, 129]. In light of these limitations, we employ the encoder proposed in RLAD, which incorporates techniques to mitigate these problems.

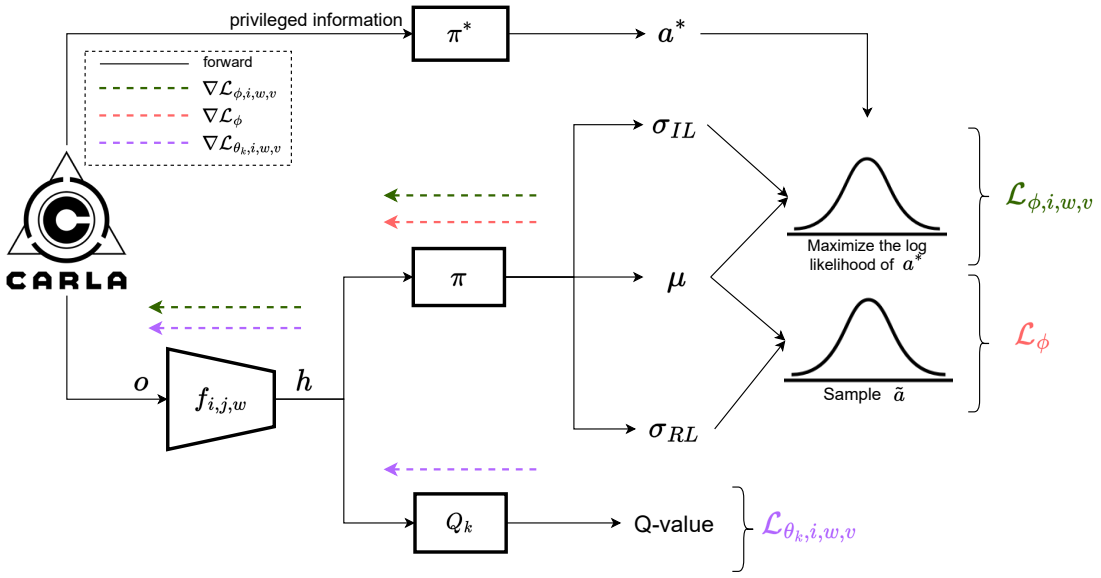


Figure 4.1: RLfOLD leverages online demonstrations through an expert policy (π^*) with access to privileged information. The encoder ($f_{i,j,w}$) converts the observation (o) into a latent representation ($\boldsymbol{h}$), which serves as input for a modified SAC. The policy (π) outputs the mean (μ) and two standard deviations: σ_{IL} and σ_{RL}. σ_{IL} is used to maximize the log-likelihood of the expert action (a^*), while σ_{RL} is employed to explore the environment.

Image Encoder

The image encoder is a convolution neural network consisting of approximately 0.65M parameters, which is significantly smaller in size compared to state-of-the-art methods in urban AD (see Table 4.2). We leverage image augmentations to regularize the value function and to increase the generalization [140]. Specifically, we apply color jittering, Gaussian blur, and random crop. At the end of each convolution encoder, we append an Adaptive Local Signal Mixing (A-LIX) layer [129] to mitigate the catastrophic self-overfitting phenomenon. For the convolutional layers, we employed the Delta-Orthogonal initialization technique [166], and for the linear layers, we employed the Orthogonal initialization technique [167].

The image encoder, f_i, can be formalized as $\boldsymbol{i}_t = f_i(\text{aug}([\{\boldsymbol{I}_{t-k}\}_{k=0}^{1}]))$, where aug corresponds to the image augmentation applied, and $\boldsymbol{i}_t$ corresponds to the latent representation of the stack of two consecutive images ($[\{\boldsymbol{I}_{t-k}\}_{k=0}^{1}]$).

Waypoint Encoder

To encode the waypoints we use WayConv1D [44]. This method leverages the 2D geometrical structure of the waypoints by applying 1D convolutions with a 2×2 kernel over the 2D coordinates of the next N waypoints. The process can be described as $\boldsymbol{w}_t = f_w(\boldsymbol{W}_t)$, where f_w corresponds to the WayConv1D, and $\boldsymbol{w}_t$ corresponds to the latent representation of the

waypoints ($\boldsymbol{W}_t$).

Vehicle Measurements Encoder

We apply directly an MLP to the vehicle measurements: $\boldsymbol{v}_t = f_v([\{\boldsymbol{V}_{t-k}\}_{k=0}^{1}])$, where f_v is the MLP, and $\boldsymbol{v}_t$ corresponds to the latent representation of the concatenation of the vehicle measurements across two steps ($[\{\boldsymbol{V}_{t-k}\}_{k=0}^{1}]$).

The latent representation of all the inputs ($\boldsymbol{h_t}$) is then obtained by concatenating the latent representation of each input: $\boldsymbol{h}_t = \begin{bmatrix} \boldsymbol{i}_t & \boldsymbol{w}_t & \boldsymbol{v}_t \end{bmatrix}$. Throughout the document, to simplify the notation, we will refer to all encoders f_i, f_w, and f_v as $f_{i,w,v}$:

$$\boldsymbol{h}_t = f_{i,w,v}\left(o_t\right). \tag{4.1}$$

4.3.3 Soft Actor-Critic with Imitation Learning

We use the SAC algorithm with a one-step return as the base algorithm. SAC is a model-free off-policy actor-critic algorithm that learns two Q-functions Q_{θ_1}, Q_{θ_2}, a stochastic policy π_ϕ, and a temperature α to find an optimal policy by optimizing a γ-discounted maximum-entropy objective [128, 145]. The actor policy $\pi_\phi(\tilde{a}_t \mid \boldsymbol{h}_t)$ is a parametric tanh-Gaussian that given $\boldsymbol{h}_t$, samples $\tilde{a}_t = \tanh\left(\mu(\boldsymbol{h}_t) + \sigma_{RL}(\boldsymbol{h}_t)\epsilon\right)$, where $\epsilon \sim \mathcal{N}(0,1)$, and μ and σ_{RL} are the parametric mean and standard deviation. For the target speed, we apply a post-processing transformation to scale the tanh output to the desired range.

In contrast to the original SAC algorithm, our adapted actor policy produces three values: μ, σ_{RL}, and σ_{IL}. This modification proved to be more adequate, as it enables us to utilize distinct standard deviations for the various losses functions (more details provided below).

The double Q-networks are learned by optimizing a one-step of the soft Bellman residual:

$$\mathcal{L}_{\theta_k,i,w,v} = \mathbb{E}_{\substack{o_t,a_t,o_{t+1}\sim\mathcal{D} \\ \tilde{a}_{t+1}\sim\pi_\phi(\cdot|\boldsymbol{h}_{t+1})}} \left[\left(Q_{\theta_k}\left(\boldsymbol{h}_t, a_t\right) - y\right)^2\right], \forall k \in \{1,2\}. \tag{4.2}$$

with the TD target y defined as:

$$y = r_t + \gamma\left(\min_{k=1,2} Q_{\bar{\theta}_k}\left(\boldsymbol{h}_{t+1}, \tilde{a}_{t+1}\right) - \alpha \log \pi_\phi\left(\tilde{a}_{t+1} \mid \boldsymbol{h}_{t+1}\right)\right), \tag{4.3}$$

where γ is the discount factor, and $Q_{\bar{\theta}_1}$ and $Q_{\bar{\theta}_2}$ denote the target parameters of Q_{θ_1} and Q_{θ_2}, respectively. The policy is updated to maximize the expected future return plus the expected future entropy:

$$\mathcal{L}_\phi = -\mathbb{E}_{\substack{o_t\sim\mathcal{D} \\ \tilde{a}_t\sim\pi_\phi(\cdot|\boldsymbol{h}_t)}} \left[\min_{k=1,2} Q_{\theta_k}\left(\boldsymbol{h}_t, \tilde{a}_t\right) - \alpha \log \pi\left(\tilde{a}_t \mid \boldsymbol{h}_t\right)\right]. \tag{4.4}$$

Finally, the parameter α is automatically tuned over the training according to [146].

To incorporate IL, we utilize the same batch of transitions used by RL and create a Gaussian distribution (p_ϕ) using the parameters generated by π, namely μ and σ_{IL}. Subsequently, this distribution is employed to maximize the log-likelihood of the action produced by the online expert (a^*):

$$\mathcal{L}_{\phi,i,w,v} = -\mathbb{E}_{o_t,a_t^* \sim \mathcal{D}}\Big[\log p_\phi\left(a_t^* \mid \boldsymbol{h_t}\right)\Big]. \tag{4.5}$$

By using different standard deviations, the algorithm can adapt to the varying levels of uncertainty in RL and IL. It allows the RL component to explore the state-action space more broadly (with a larger standard deviation), while the IL component can focus on imitating the expert's behavior more closely (with a smaller standard deviation). This adaptability to uncertainty can lead to a better balance between exploration and exploitation, and thus a seamless integration of RL and IL.

As illustrated in Figure 4.1, each loss updates specific parameters, and this document follows a nomenclature that associates the indexes of the loss function with the corresponding updated parameters.

4.3.4 Online Expert

The online expert has access to privileged information from the simulator, enabling it to generate expert actions. These actions serve two distinct purposes: assisting the exploration process and contributing to Equation 4.5. The online expert can take the form of a neural network or a set of heuristics. For this study, we have chosen to implement the online expert as a set of simple heuristics, with future plans to transition to a neural network-based approach.

As previously mentioned, the action is parameterized using target speed and steering. Inspired by [111], the target speed is dynamically computed based on the agent's surroundings. As the distance to the front vehicle decreases, the target speed linearly reduces to 0, and conversely, as the distance increases, the target speed increases accordingly. The same principle applies when approaching obstacles, pedestrians, or traffic lights. For all other situations, the target speed remains set at a constant maximum speed. Regarding the steering, we conducted experiments using different heuristics that considered the agent's position and orientation relative to the waypoints. However, given the limited complexity of the online expert, we found that utilizing the steering from the RL policy (π) produced superior results. Consequently, in this work, we solely rely on the expert action to determine the target speed.

The efficacy of RLfOLD is substantially influenced by the quality of the online expert. The expert's input is crucial, providing accurate ground truth actions for IL and assisting in decision-making when the policy's uncertainty is high. While our dual standard deviation ap-

proach is designed to leverage this expert guidance effectively, it is important to acknowledge that the overall performance may vary with the expert's proficiency.

4.3.5 Expert-guided Exploration based on Uncertainty

In RLfD algorithms, the expert actions are only used in the loss functions to update the model parameters. However, by leveraging the online nature of the expert, we extended the usage of the expert actions to the exploration. The idea is to use the σ_{RL} as the uncertainty of the decision taken by the current policy (π). This uncertainty quantifies the confidence level of the policy, allowing us to gauge its competence in exploring the environment effectively. When the policy's uncertainty falls below a predefined threshold (u), the agent executes the action recommended by the policy, fostering efficient exploitation of its learned knowledge. On the other hand, if the policy's uncertainty exceeds the threshold, the agent seeks the guidance of the online expert to make informed decisions in uncertain situations. This decision-making process can be described as follows:

$$a = \begin{cases} \tilde{a} \text{ if } \sigma_{RL} < u \\ a^* \text{ otherwise} \end{cases} . \tag{4.6}$$

This method establishes a dynamic learning relationship between the agent and the online expert. Similar to a student seeking guidance from a teacher, the agent autonomously explores when confident, and seeks assistance from the expert when uncertain. This adaptive approach promotes efficient learning, safer exploration, and the potential for rapid skill acquisition in complex environments.

For a more comprehensive understanding of our learning framework, we provide the pseudocode implementation in Algorithm 1.

4.4 Experiments

4.4.1 Setup

Benchmark

The algorithms are evaluated on the NoCrash benchmark. This benchmark examines the ability to generalize from Town 1, characterized by one-lane roads and T-junctions with traffic lights, to Town 2, a scaled-down version of Town 1 with different textures. The training process involves four distinct weather types, while the testing phase employs two different weather types. Within this benchmark, three levels of traffic density (empty, regular, and dense) are considered based on the number of vehicles and pedestrians present. The evaluation results are presented in terms of the success rate, representing the percentage of completed routes without any collisions. Additionally, for the ablation study, we also provide information

Algorithm 1 Reinforcement Learning from Online Demonstrations (RLfOLD)

Input: initial encoder parameters $f_{i,w,v}$, Q-function parameters Q_{θ_1}, Q_{θ_2}, policy parameters π_ϕ, entropy parameter α, empty replay buffer $\mathcal{D}$

$Q_{\bar{\theta}_k} \leftarrow Q_{\theta_k}$, for k = 1,2
repeat
    Get observation o_t
    Compute expert action a_t^* using π^*
    Encode o_t into $\boldsymbol{h_t}$ using Equation 4.1
    Sample policy action $\tilde{a}_t \sim \pi_\phi(\cdot \mid \boldsymbol{h_t})$
    Execute a_t according to Equation 4.6
    Get next observation o_{t+1} and reward r_t
    Store transition $(o_t, a_t, a_t^*, r_t, o_{t+1})$ in $\mathcal{D}$
    if o_{t+1} is terminal **then**
        Reset environment state
    end if
    if time to update **then**
        Randomly sample a batch of transitions, $\mathcal{B} = \{(o_t, a_t, a^*{}_t, r_t, o_{t+1})\}$ from $\mathcal{D}$
        Update Q_{θ_1}, Q_{θ_2} and $f_{i,w,v}$ using Equation 4.2
        Update π_ϕ using Equation 4.4
        Update π_ϕ, and $f_{i,w,v}$ using Equation 4.5
        Update α according to [146]
        Update $Q_{\bar{\theta}_k}$ with $Q_{\bar{\theta}_k} \leftarrow (1-\rho)\, Q_{\bar{\theta}_k} + \rho Q_{\theta_k}$, for k = 1,2
    end if
until convergence

regarding the percentage of route completion, collisions with vehicles, pedestrians, and layout, as well as the number of blockages per kilometer.

Training Details

All algorithms are trained on the same hardware, specifically a single NVIDIA RTX 2080 Ti. The training process spans 10^6 environment timesteps, with evaluations conducted every 20k environment timesteps. During each evaluation, episode returns are averaged over 10 episodes. Each experiment was conducted with three different seeds to account for the high variability in RL training. We use the reward function defined in [42]. The Deep Learning library used was PyTorch [147]. Table 4.1 contains the main hyperparameters used by RLfOLD.

State-of-the-art Algorithms

We compare RLfOLD with the state-of-the-art methods that reported their results on the NoCrash benchmark. The comparison includes algorithms of the three types (RL, IL, and RLfD):

Table 4.1: List of the hyperparameters used by RLfOLD.

Parameter	Value
Replay Buffer capacity	100000
Batch size	128
Action repeat	2
Discount factor (γ)	0.85
Optimizer	Adam
Learning rate	10^{-3}
Target Q-network update rate (ρ)	0.01
dim($\boldsymbol{i}$)	256
dim($\boldsymbol{w}$)	32
dim($\boldsymbol{v}$)	16
SAC networks size	1024
Init entropy parameter (α)	0.2
Uncertainty threshold (u)	0.8

- **RL**: IAs [111], CADRE [133];
- **IL**: CILRS [91], LBC [74];
- **RLfD**: GRIAD [38], WOR [165];

4.4.2 Comparative Analysis

The number of parameters of a model is considered an essential metric to gauge computational requirements and memory consumption. However, obtaining this value can be challenging as it is often not reported in many studies. To address this, we use the size of the image encoder as a representative proxy (see Table 4.2). Since state-of-the-art methods typically employ very large image encoders, this component accounts for a substantial portion of the model's parameters. As reported in Table 4.2, RLfOLD utilized a significantly smaller encoder when compared to the state-of-the-art methods: approximately 3% of the average encoder size found in those methods. Table 4.2 also reports the number of cameras used, where all methods used only one camera, with the exception of GRIAD and WOR, which used three cameras.

Table 4.3 shows the comparative results in terms of the success rate on the NoCrash benchmark. The success rate values for the methods IAs, LBC, and WOR were obtained from [165], the values of CADRE were taken from [133], the values of CILRS were taken from [168], and finally, the values of GRIAD were taken from [38]. The proposed method

Table 4.2: Comparison of the number of parameters in image encoders and the number of cameras used by the state-of-the-art methods.

	# of parameters	# of cameras
IAs	∼30M	1
CADRE	∼25M	1
CILRS	∼22M	1
LBC	∼22M	1
GRIAD	∼14M	3
WOR	∼22M	3
RLfOLD	∼0.65M	1

outperforms all single-camera approaches across various tasks, showcasing its superior performance despite employing a significantly smaller encoder. Among the single-camera methods, CADRE emerges as the closest competitor, albeit with a notable 9% performance loss of the average score compared to RLfOLD. Even when compared against three-camera methods, all of which are RLfD algorithms, RLfOLD demonstrates its superiority in performance. With an average score exceeding GRIAD by 6% and WOR by 2%, RLfOLD outperforms its multi-camera counterparts. RLfOLD secures the top rank in more tasks than any other method, outperforming all competitors in seven distinct tasks. These results underscore the effectiveness and efficiency of RLfOLD, solidifying its position as the top-performing approach in the evaluation, even when employing a significantly smaller encoder and a single camera setup.

4.4.3 Ablation Study

To gain deeper insights into the strengths of RLfOLD, we conducted an ablation study examining its main components. Firstly, we established a RL baseline version without demonstrations (referred to as RL baseline). Next, we evaluated the significance of the two standard deviations by testing a variant of RLfOLD that generates only one standard deviation (σ_{RL}) and employs Mean Squared Error (MSE) loss for the IL training (RLfOLD w/o two SDs). Furthermore, to assess the impact of expert-guided exploration based on uncertainty, we experimented with two versions of RLfOLD: one that excludes expert guidance during exploration (RLfOLD w/o unc. (p=0.0)) and another that incorporates expert guidance with a fixed probability of 0.3 for each action taken (RLfOLD w/o unc. (p=0.3)). The results of the ablation study, as shown in Table 4.4, provide insights into the role of different components within RLfOLD. The RL baseline, which lacks the integration of demonstrations, achieved a success rate of 52%, which stands for a marginal loss of 34% considering the original version of RLfOLD. This difference clearly indicates the challenges of learning complex driving tasks using RL from scratch. The variant RLfOLD w/o two SDs demonstrates the importance of

Table 4.3: Comparison of the success rate (%) on NoCrash benchmark using the state-of-the-art methods. The method IAs was not evaluated under testing weather conditions.

			RL		IL		RLfD		
Task	Town	Weather	IAs	CADRE	CILRS	LBC	GRIAD*	WOR*	RLfOLD
Empty	train	train	85	95	97	89	98	98	**100**
Regular			85	92	83	87	98	**100**	94
Dense			63	82	42	75	94	**96**	90
Empty	test	train	77	92	66	86	94	94	**100**
Regular			66	78	49	79	**93**	89	92
Dense			33	61	23	53	78	74	**80**
Empty	train	test	-	94	**96**	60	83	90	**96**
Regular			-	86	77	60	87	**90**	84
Dense			-	76	39	54	83	**84**	74
Empty	test	test	-	78	66	36	69	78	**100**
Regular			-	72	56	36	63	82	**86**
Dense			-	52	24	12	52	**66**	**66**
Average	-	-	68	80	60	60	83	87	**89**

* Used 3 cameras as input.

the two standard deviations. This variant achieved a success rate of 64%, which is significantly inferior to the one achieved using the two standard deviations - 86%. Furthermore, the integration of expert-guided exploration based on uncertainty proves to be highly beneficial. When we exclude expert guidance during exploration (RLfOLD w/o unc. (p=0.0)), the success rate drops to 72%. This indicates that the online expert provides valuable insights to guide the agent's exploration. Moreover, incorporating the online expert with a fixed probability for each action (RLfOLD w/o unc. (p=0.3)) achieves a success rate of 80%, which is 8% better than not using the online expert, but is 6% worse than using the expert-guided exploration based on uncertainty. In conclusion, the ablation study demonstrates the crucial role of each component in RLfOLD, emphasizing the significance of leveraging online demonstrations, the separate standard deviations output, and the expert-guided exploration based on uncertainty.

4.5 Conclusion

In this paper, we have presented RLfOLD, a novel and effective RLfD algorithm. Our method introduces a seamless integration of IL and RL by leveraging online demonstrations

Table 4.4: Ablation study evaluating the success rate and infraction analysis on the regular task under testing conditions (town and weather). Mean and standard deviation over 3 seeds.

	Success rate %, ↑	Route completion %,↑	Collision pedestrian #/Km, ↓	Collision vehicle #/Km, ↓	Collision layout #/Km, ↓	Agent blocked #/Km, ↓
RL baseline	52±4	98±3	1.03±0.34	1.40±0.11	0.26±0.05	0.36±0.13
RLfOLD w/o two SDs	64±10	90±6	0.33±0.13	0.53±0.09	0.15±0.09	4.45±1.43
RLfOLD w/o unc. (p=0.0)	72±2	96±3	0.14±0.04	0.48±0.03	0.12±0.03	3.99±0.47
RLfOLD w/o unc. (p=0.3)	80±3	91±1	0.30±0.04	0.45±0.06	**0.00**±0.00	2.76±0.91
RLfOLD	**86**±4	**99**±2	**0.09**±0.03	**0.32**±0.04	0.09±0.03	**0.15**±0.08

to bridge the distribution gap between the demonstration and the training environment. Unlike conventional policy networks used in actor-critic algorithms, RLfOLD adopts a policy network that outputs two standard deviations: one for exploration and another for IL training. Additionally, we utilize the online expert to guide the exploration process, incorporating an uncertainty-based technique. The results obtained on the NoCrash benchmark underscore the superior effectiveness and efficiency of RLfOLD. Notably, even with a significantly smaller encoder and a single-camera setup, RLfOLD surpasses all tested state-of-the-art methods. The ability to achieve such results with limited resources makes RLfOLD a highly promising solution for real-world applications. In the future, we aim to enhance the complexity of the online expert by transitioning from a rule-based approach to a more advanced neural network-based approach and to test this algorithm in other applications.

Chapter 5

PRIBOOT: A New Data-Driven Expert for Improved Driving Simulations

Paper submitted at: IEEE Transactions on Automation Science and Engineering
Authors: Daniel Coelho, Miguel Oliveira, Vítor Santos, and Antonio M. López

Abstract: The development of Autonomous Driving (AD) systems in simulated environments like CARLA is crucial for advancing real-world automotive technologies. To drive innovation, CARLA introduced Leaderboard 2.0, significantly more challenging than its predecessor. However, current AD methods have struggled to achieve satisfactory outcomes due to a lack of sufficient ground truth data. Human driving logs provided by CARLA are insufficient, and previously successful expert agents like Autopilot and Roach, used for collecting datasets, have seen reduced effectiveness under these more demanding conditions. To overcome these data limitations, we introduce PRIBOOT, an expert agent that leverages limited human logs with privileged information. We have developed a novel BEV representation specifically tailored to meet the demands of this new benchmark and processed it as an RGB image to facilitate the application of transfer learning techniques, instead of using a set of masks. Additionally, we propose the Infraction Rate Score (IRS), a new evaluation metric designed to provide a more balanced assessment of driving performance over extended routes. PRIBOOT is the first model to achieve a Route Completion (RC) of 75% in Leaderboard 2.0, along with a Driving Score (DS) and IRS of 20% and 45%, respectively. With PRIBOOT, researchers can now generate extensive datasets, potentially solving the data availability issues that have hindered progress in this benchmark.

5.1 Introduction

Autonomous Driving (AD) is a key technological advancement with the potential to transform transportation, improve road safety, and redefine urban environments [43, 57]. Despite its potential, developing fully autonomous vehicles involves significant challenges. These include integrating diverse sensors, processing complex data, making real-time decisions, and addressing ethical issues. Such vehicles must operate reliably in unpredictable conditions, requiring advanced systems capable of handling a wide range of scenarios [39]. Real-world testing of autonomous vehicles, while necessary, is often expensive, risky, and encumbered by ethical dilemmas.

Simulations serve as a critical complement to real-world testing, providing a safe and controlled environment that replicates complex real-world scenarios without the associated costs and risks [39, 40]. This enhances the development of autonomous driving technologies by allowing preliminary testing and refinement in simulations, reserving real-world trials for the final stages of development. Moreover, in these simulated environments, it is possible to leverage privileged information, otherwise not available in the real-world, to create expert systems that can provide demonstrations, further enriching the development process.

Among various open-source AD simulators, CARLA [39] is often listed as the premier choice [40, 41]. CARLA offers a suite of essential features for realistic and effective simulation of driving scenarios. These include comprehensive environmental conditions, detailed vehicle

models, and a wide array of sensors, making it an ideal platform for advanced AD research and development.

To accelerate innovation, CARLA introduced the CARLA Leaderboard 1.0[1] benchmark, designed to assess the driving proficiency of autonomous agents within realistic traffic scenarios. Despite the complex scenarios presented in the benchmark, various methods such as ReasonNet [169], InterFuser [2], and TCP [3] have consistently reported high performance over the years. Notably, the CARLA Autopilot, a rule-based agent, achieved near-perfect performance. This underscores the benchmark's capacity to be effectively mastered using current technologies. Building on this foundation, CARLA Leaderboard 2.0[2] introduces even more complex and challenging scenarios, such as obstacles in the lane and parking exits. These novel scenarios significantly increase the difficulty level, challenging the limits of existing autonomous driving systems. To this date, all approaches tested on the Leaderboard 2.0 benchmark have shown very poor performance, with the highest Route Completion (RC) reaching 15%, and the highest DS reaching 1% [170]. We believe that the primary reason for this notable decline in performance can be attributed to the insufficiency of available training data. CARLA provides a set of human driving logs from a few route scenarios, but these are insufficient for training models that rely on sensor information. In Leaderboard 1.0, researchers could leverage online experts like CARLA Autopilot or Roach [42] to generate demonstrations. However, in Leaderboard 2.0, these experts are either markedly less effective or completely ineffective, as we will show in Section 5.2.

This paper presents a method to address the challenges posed by the limited training data availability. The driving logs from CARLA, while insufficient alone for models requiring sensor inputs, become significantly more useful when combined with privileged information from the simulator, specifically Bird's Eye View (BEV). This integration effectively simplifies the complexity of the benchmark. Employing this strategy, we apply Imitation Learning techniques to develop PRIBOOT (**Pri**vileged **I**nformation **Boot**strapping), an expert agent capable of navigating the demanding scenarios presented in Leaderboard 2.0. PRIBOOT utilizes privileged information to master these scenarios, subsequently enabling the generation of extensive datasets or providing online demonstrations. Although PRIBOOT was designed to address the challenges of Leaderboard 2.0, it is also applicable to any other CARLA benchmark.

Overall, we summarize our main contributions as follows:

- Introduce PRIBOOT, an expert agent that effectively leverages privileged information and limited data for model training, marking the first instance of achieving significant performance milestones on the CARLA Leaderboard 2.0;
- Develop a tailored Bird's Eye View (BEV) representation to effectively address the

[1] https://leaderboard.carla.org/#leaderboard-10
[2] https://leaderboard.carla.org/

complex driving scenarios encountered in CARLA Leaderboard 2.0.

- Process the BEV as an RGB image rather than a set of masks. This facilitates the application of transfer learning techniques, which significantly enhance model performance and efficiency, particularly in the context of limited data availability;
- Introduce Infraction Rate Score (IRS), a novel evaluation metric that considers infractions per kilometer rather than the total number of infractions. This metric is designed to complement the Driving Score (DS) by providing a more detailed assessment of driving behavior over long routes.

The source code of PRIBOOT is available at https://github.com/DanielCoelho112/priboot.

5.2 Related Work

This section is divided into two topics: the application of expert agents in AD, and an overview of all expert agents utilized in CARLA.

5.2.1 Application of Experts in Autonomous Driving

In recent years, the field of AD has seen significant advancements through the application of online experts [3, 42, 171, 172]. A notable example of this is showcased in [171], where the utility of online experts is demonstrated in real-world, high-speed off-road driving scenarios. In their approach, an initial expert system equipped with expensive sensors is developed using a combination of hand-engineered components. This expert system then serves as a reference model, providing high-quality driving demonstrations to train a student model, which operates using more affordable sensors.

Building on the foundational use of online experts, the transfer of knowledge from the expert to the student model can be accomplished through various methodologies. One prevalent method involves the creation of offline datasets, which are subsequently employed for offline Imitation Learning (IL) [91,169] or Reinforcement Learning from Demonstrations (RLfD) [38]. These approaches are particularly valuable in scenarios where direct interaction with the environment is either too costly or filled with risks. However, a significant challenge with using offline datasets is the potential for a distribution shift [31]. To address the issue of distribution shift, an alternative strategy is online IL [74], where the student actively explores the environment while the teacher provides on-demand supervision. This method helps to align the student's learning experience more closely with the actual operational environment, thereby reducing the impact of distribution shift. Nevertheless, this approach still relies heavily on the quality of the data provided by the expert. If the expert's behavior is not optimal, the student is likely to inherit these imperfections [42]. To further refine this process and overcome the limitations of potentially suboptimal expert data, another innovative approach is

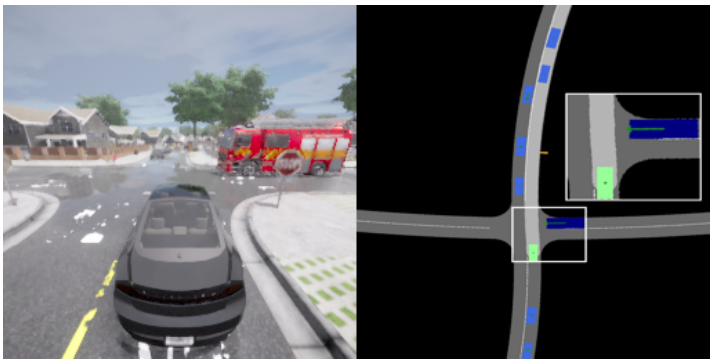
Reinforcement Learning from Online Demonstrations (RLfOLD) [45]. This technique merges the benefits of Online IL with the principles of RL, tackling both the issue of distribution shift and the challenge of learning from a suboptimal expert.

5.2.2 Experts in CARLA

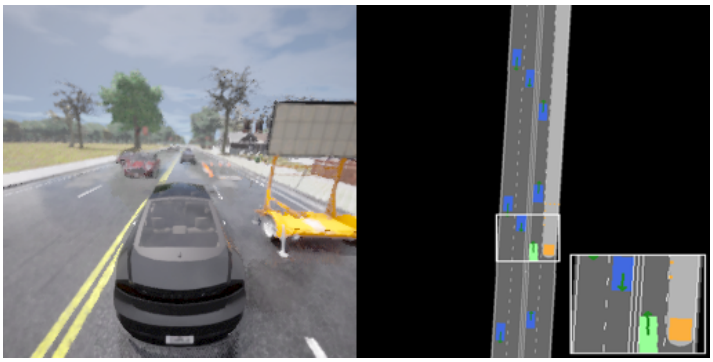
CARLA incorporates a built-in expert system known as Autopilot, which relies on a series of handcrafted rules that utilize the internal state of the simulator for navigation [39]. In Leaderboard 1.0, Autopilot demonstrated commendable performance, contributing significantly to data collection for top-ranked methods such as ReasonNet [169], which relies on datasets generated by this expert. However, the transition to Leaderboard 2.0 reveals a stark contrast in the efficacy of the Autopilot system. As detailed in Section 5.4, the performance of Autopilot is markedly diminished in the more demanding scenarios of this updated benchmark. The primary challenge lies in the inherent limitations of a rule-based navigation framework, which struggles to adapt to the complex and dynamic driving conditions presented in Leaderboard 2.0, such as yielding to emergency vehicles or overtaking obstacles in the lane.

While Autopilot has shown competent performance in earlier benchmarks, the adoption of learning-based experts presents distinct advantages [42, 74, 168]. These methods usually decouple the perception from planning, which simplifies the training process. Typically, such methods leverage privileged information from the simulator to bypass the need for complex perception systems, focusing instead on training the planning module directly. For example, LBC [74] and SAM [168] replace the perception module with simulator-derived privileged information, and then train the planning component using IL based on demonstrations provided by the Autopilot. To ensure effective knowledge transfer from the expert to the student, LBC aims to minimize the output differences between them, whereas SAM focuses on aligning the latent representations of both models. These learning-based experts have been assessed in straightforward benchmarks, such as the NoCrash benchmark [91]. This benchmark, with its relatively simple navigation and collision avoidance tasks, represents a significantly lesser challenge than even the earlier Leaderboard 1.0, and far less demanding than the complexities encountered in Leaderboard 2.0.

More recently, the Roach expert [42] was introduced and has since become the most utilized expert in Leaderboard 1.0 [3, 173]. Roach processes inputs using a Bird's Eye View (BEV) image that encapsulates roads, lanes, routes, vehicles, pedestrians, traffic lights, and stop signs. This information is then processed using a model-free Reinforcement Learning (RL) algorithm to generate vehicle control commands. While Roach has demonstrated impressive results in Leaderboard 1.0, its applicability to Leaderboard 2.0 is questionable without significant modifications. Several challenges hinder the transition of Roach to the more demanding Leaderboard 2.0. Firstly, their BEV implementation struggles with scalability issues



(a)



(b)

Figure 5.1: BEV used in PRIBOOT. This representation was built upon Roach and LBC, with critical adaptations to tailor it for the complexities of Leaderboard 2.0. (a) Differentiates emergency vehicles (dark blue) from regular vehicles (blue). (b) Introduces an additional class for construction objects, illustrated in orange. Additionally, both images depict a simple method to represent motion with directional arrows, illustrated in green. The images with a white frame provide a zoomed-in view to highlight specific details.

in the larger towns of Leaderboard 2.0, primarily due to memory constraints when computing the cache for the roads and lanes. Secondly, the existing classes in the BEV representation fall short in capturing complex new scenarios introduced in the updated leaderboard, such as construction zones or the presence of emergency vehicles. Lastly, there is uncertainty regarding the effectiveness of Roach's model-free RL approach when faced with the heightened complexity and dynamic requirements of Leaderboard 2.0. It is important to note that Roach was trained for about one week on an Nvidia RTX 2080 Ti to achieve its results on Leaderboard 1.0. Considering the increased difficulty and complexity of scenarios in Leaderboard 2.0, adapting and retraining Roach could potentially require significantly more time, further complicating its deployment in this new benchmark.

Recognizing the limitations of existing expert agents for Leaderboard 2.0, CARLA has made available a set of driving logs that showcase human-driven routes in various scenarios. However, these logs alone do not suffice to train a system capable of processing sensor inputs

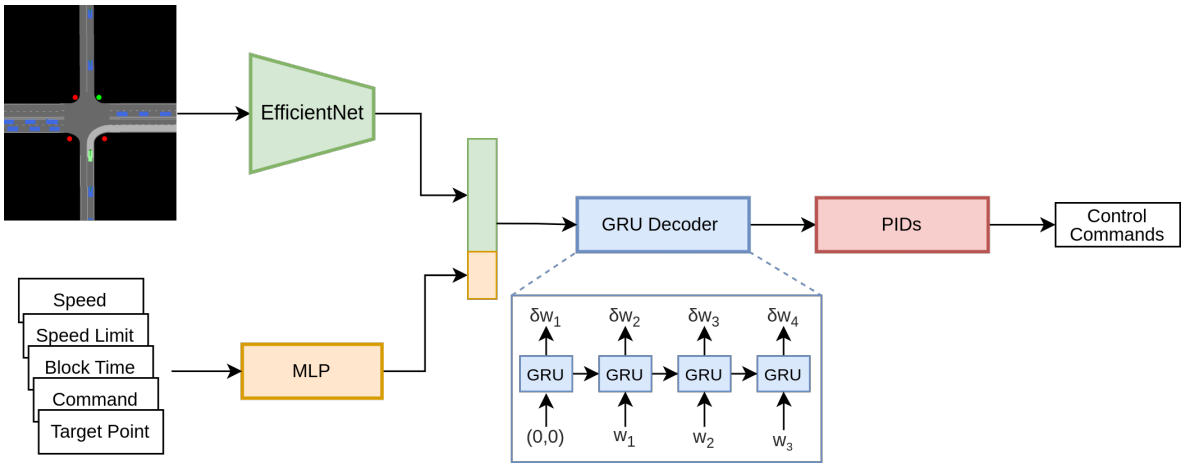


Figure 5.2: Architecture of PRIBOOT. The system receives two types of inputs: a BEV image and a vector of vehicle measurements. These inputs are processed independently— the BEV through a pretrained EfficientNet model and the vehicle measurements via a MLP. The resultant feature vectors from both models are concatenated to form a comprehensive feature vector, which is then fed into a GRU-based waypoint decoder, similar to the approach used by Transfuser [95]. The final stage involves processing the waypoints through both longitudinal and lateral PID controllers to generate the vehicle control commands.

and generating control commands. In response, and inspired by the approaches of LBC and Roach, we propose PRIBOOT, a method that simplifies the perception component by employing a Bird's Eye View (BEV) as the primary input. However, instead of using BEV as independent mask channels for training a CNN from scratch, PRIBOOT converts the mask into an RGB image and leverages transfer learning techniques using pre-trained networks from the ImageNet dataset [174]. This adaptation is crucial, particularly given the limited data available.

5.3 Method

PRIBOOT (**Pr**ivileged **I**nformation **Boot**strapping) leverages the limited logs available in Leaderboard 2.0 to establish the first expert agent capable of achieving satisfactory results within this demanding benchmark, as we show in Section 5.4. The development of PRIBOOT was structured in two phases: First, we focused on generating the most effective input representation tailored to the unique challenges of Leaderboard 2.0, as detailed in Section 5.3.1. Following this, we designed and implemented a neural network architecture that is specifically optimized for handling the constraints of limited data, described in Section 5.3.2.

5.3.1 Generation of Bird's Eye View

Building on the approach used by Roach [42] and LBC [74], we employ a BEV to model the environment. However, adaptations were necessary to tailor it for the complexities of Leaderboard 2.0. Roach's and LBC's BEV include various classes such as roads, desired routes, lane boundaries, vehicles, pedestrians, traffic lights, and stop signs. While these classes were adequate for Leaderboard 1.0, they proved insufficient for the expanded scope of Leaderboard 2.0. Our enhancements to the BEV are outlined below:

Scalable Cache Current BEV approaches utilize a caching mechanism to store the roads and lanes for the entire town, a process completed once per town to facilitate real-time BEV generation. However, applying this method to the larger towns in Leaderboard 2.0 caused memory overflows due to the use of Pygame. We addressed this by adopting a more efficient caching technique inspired by deepsense.ai[3], implementing the cache with NumPy for enhanced performance.

Decomposition of the Vehicles Class In current BEV representations, all vehicle types are aggregated under a single class. We refined this by segmenting the Vehicles class into three distinct categories: Bikes, Emergency Vehicles, and Regular Vehicles. This differentiation is crucial as the driving behavior varies significantly based on the type of nearby vehicle, especially in emergency situations (see Figure 5.1a).

Simplified Motion Representation Roach's BEV uses multiple temporal masks to capture movement, which increases significantly the computational load. Instead, we introduced a single additional mask featuring an arrow for each actor, as illustrated with green arrows in Figure 5.1. This arrow indicates both the direction (orientation) and the speed (length) of the actor, simplifying the representation while reducing memory and computational demands.

Incorporation of a New Class Leaderboard 2.0 introduces scenarios requiring interaction with new environmental elements not covered by existing classes. For instance, construction zones that necessitate slight route deviations were not previously accounted for. To accommodate this, we added a new class named Construction, which encompasses all pertinent elements like traffic cones and street barriers, represented in orange in 5.1b.

RGB Format Instead of Masks Roach and LBC process the BEV using independent mask channels, requiring the training of a CNN from scratch. Given the limited data in Leaderboard 2.0, we found that converting these masks into an RGB format and utilizing pre-

[3] https://github.com/deepsense-ai/carla-birdeye-view

trained visual encoders not only saves training time but also enhances the model's performance and efficiency.

5.3.2 Architecture

The architecture of PRIBOOT is depicted in Figure 2. Our system takes as input a BEV image and a vector of vehicle measurements. The vehicle measurement vector encompasses several key parameters: current speed and the road speed limit, block time, a target point, and a navigation command. The "block time" parameter denotes the duration during which the vehicle has been stationary, aiding the system in determining whether to overtake or maintain its position due to typical traffic conditions. The "target point" is a waypoint located 30 meters ahead on the desired trajectory provided by a global planner, and the "navigation command" provides high-level directional indication from the global planner, encoded as a one-hot vector.

We utilize an EfficientNet [175] for processing the BEV image and an MLP for handling vehicle measurements. Given the constraint of limited available data, we adopt transfer learning by employing the pretraining of EfficientNet with the ImageNet dataset. Subsequently, the feature vectors extracted from both the EfficientNet and the MLP are merged and inputted into an autoregressive GRU decoder. This decoder is tasked with predicting the subsequent T=4 waypoints $\{w_t\}_{t=1}^{T}$ within the ego-vehicle coordinate framework, drawing inspiration from the methodology applied in Transfuser [95].

To convert the predicted waypoints into control commands, we employ two PID controllers—one for lateral control and another for longitudinal control—following methodologies from [74, 95]. The longitudinal controller uses the magnitude of the average vector between consecutive waypoints, while the lateral controller relies on their orientation. Additionally, similar to Transfuser, we integrate a creeping behavior and a safety heuristic mechanism utilizing information from the simulator.

This system is trained end-to-end using an L_1 loss between the predicted waypoints and the ground truth waypoints from the human logs. Let w_t^* represent the ground truth waypoint at timestep t, the the loss function is defined as:

$$\mathcal{L} = \sum_{t=1}^{T} \|w_t - w_t^*\|_1. \tag{5.1}$$

The human demonstrations typically exhibit minimal deviation from the center of the lane, resulting in noise-free data. However, this adherence to the centerline causes a distribution shift between the training distribution and inference distribution. During inference, due to planning or controller inaccuracies, the agent may find itself in scenarios that deviate from the center of the lane. These instances are encountered as out-of-distribution events, presenting difficulties for the agent to navigate. To address this issue, inspired by the LBC

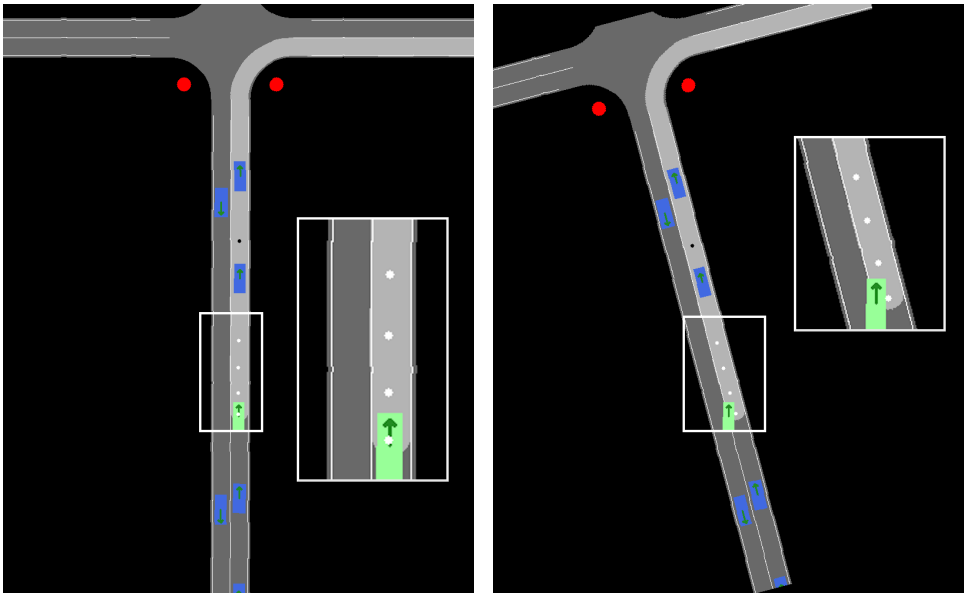


Figure 5.3: Data augmentation techniques used to expose the agent to a broader range of driving scenarios. In these illustrations, white dots indicate the future waypoints that were followed by the human driver, and the black dot represents the target point. (a) Displays a sample with no augmentation, showing the standard scenario. (b) Shows a sample where both translation and rotation augmentations have been applied to the ego vehicle, illustrating a situation where the agent needs to recover to the center of the lane.

approach, we use data augmentation techniques regarding the position and orientation of the vehicle. By varying the position and orientation of the vehicle, we expose the agent to diverse configurations, enabling it to learn effective recovery strategies. Figure 5.3 provides a visual representation of this augmentation process.

5.4 Experiments

This section starts with an overview of the setup used for collecting the experiments, followed by a comparative analysis of expert agents. It concludes with a presentation of an ablation study.

5.4.1 Setup

Benchmark

CARLA Leaderboard 2.0 builds upon CARLA Leaderboard 1.0, increasing the complexity of the benchmark in three distinct ways: a) by extending the route lengths approximately tenfold, b) by incorporating a new set of intricate driving scenarios derived from the NHTSA

Table 5.1: Comparison of run time inference using the expert agents.

	Run Time ↓ s
Autopilot	**0.007**
PRIBOOT	0.011

typology [176], and c) by increasing the frequency of these scenarios along each route. Additionally, this new benchmark introduces larger and more complex environments, as exemplified by Town12 and Town13. Town 12 is a 10×10 km^2 map that features a mix of urban, residential, and rural areas, offering varied types of challenges. Town13, while sharing many characteristics with Town12, distinguishes itself with different architectural styles, road and pavement textures, and vegetation types. These enhancements aim to rigorously test the adaptability and resilience of autonomous driving systems under varied and challenging conditions.

The benchmark uses different metrics to assess different aspects of driving performance. The Route Completion (RC) indicates the percentage of the route completed by the agent. The Infraction Penalty (IP) quantifies the severity of infractions and is calculated using the following formula:

$$IP = \prod_{i=1}^{q} p_i^{n_i}, \tag{5.2}$$

where q denotes the total number of different infraction types, p_i is the penalty associated with the infraction type i, and n_i is the number of infractions of type i. The main metric of the benchmark, Driving Score (DS), is calculated by multiplying RC and IP, providing a composite score that reflects both route completion success and adherence to driving regulations.

Infraction Rate Score

While DS provides valuable insights into agent performance, it inherently biases against longer routes due to its cumulative penalty for infractions, which are statistically more likely to occur over extended distances. To address this discrepancy and promote fairness, we introduce the Infraction Rate Score (IRS). This metric accounts for the infraction rate per kilometer, adjusting for route length and providing a balanced evaluation across varying driving conditions. The IRS is defined as:

$$IRS = RC \cdot \prod_{i=1}^{q} e^{-\lambda \cdot \frac{n_i}{L} \cdot (1-p_i)}, \tag{5.3}$$

Table 5.2: Abbrevation and the corresponding full name of the metrics used in Leaderboard 2.0.

Abbreviation	Full Name
DS	Driving Score
IRS	Infraction Rate Score
RC	Route Completion
IP	Infraction Penalty
C.P	Collisions Pedestrians
C.V	Collisions Vehicles
C.L	Collisions Layout
R.L	Red Light Infractions
Stop	Stop Sign Infractions
O.R	Off-road Infractions
R.D	Route Deviation
Block	Agent Blocked
Y.E	Yield Emergency Infractions
S.T	Scenario Timeouts
M.S	Min Speed Infractions

where L represents the length of the route in kilometers, and λ is a tunable exponent set to 4 based on empirical testing to optimize the metric's sensitivity to infractions per distance traveled.

Training Details

We utilized the human driving logs provided by CARLA to train PRIBOOT. These logs correspond to 10 routes in Town12 and 10 routes in Town13 and amount to approximately 700,000 samples collected at a frequency of 20Hz. Each sample contains all the information required at each training step, including the BEV image, the vector of vehicle measurements, and the global location of the agent on the map. For our experiments, we used CARLA version 0.9.15. PRIBOOT was trained on a single NVIDIA A40 GPU. During the training phase, we used a batch size of 256 and the Adam optimizer [148] with a learning rate of 0.0001.

5.4.2 Comparative Analysis

This section outlines a comparative analysis conducted on Leaderboard 2.0, focusing exclusively on two expert agents: Autopilot and PRIBOOT. An attempt was made to adapt the Roach system to this benchmark; however, it was unsuccessful. The benchmark currently

Table 5.3: Driving performance and infraction analysis of expert agents on CARLA Leaderboard 2.0 in Town12 and Town13.

		DS ↑ %	**IRS ↑** %	**RC ↑** %	**IP ↑** %	**C.P ↓** #/Km	**C.V ↓** #/Km	**C.L ↓** #/Km	**R.L ↓** #/Km	**Stop ↓** #/Km	**O.R ↓** #/Km	**R.D ↓** #/Km	**Block ↓** #/Km	**Y.E ↓** #/Km	**S.T ↓** #/Km	**M.S ↓** #/Km
Town12	Autopilot	1.22	0.51	5.97	0.26	1.26	4.59	0.58	0.11	1.84	0.62	0.66	1.26	**0.00**	0.34	**0.00**
	PRIBOOT	**22.80**	**42.75**	**76.46**	**0.30**	**0.00**	**0.31**	**0.06**	**0.01**	**0.02**	**0.05**	**0.00**	**0.06**	0.04	**0.03**	0.11
Town13	Autopilot	0.99	0.22	5.55	0.20	0.83	3.06	0.83	**0.00**	0.02	0.35	0.69	0.69	**0.00**	0.10	**0.00**
	PRIBOOT	**18.84**	**46.97**	**74.29**	**0.24**	**0.01**	**0.34**	**0.05**	**0.00**	**0.01**	**0.05**	**0.00**	**0.04**	0.02	**0.02**	0.06

cannot support running a RL algorithm like Roach due to memory leaks that prevent the execution of millions of steps without causing server crashes.

Table 5.1 provides a comparison of the runtime between Autopilot and PRIBOOT. In this evaluation, Autopilot achieves a runtime of 0.007 seconds, while PRIBOOT records a runtime of 0.011 seconds. This difference in performance is expected, given that Autopilot operates based on a predefined set of rules, whereas PRIBOOT processes high-dimensional inputs.

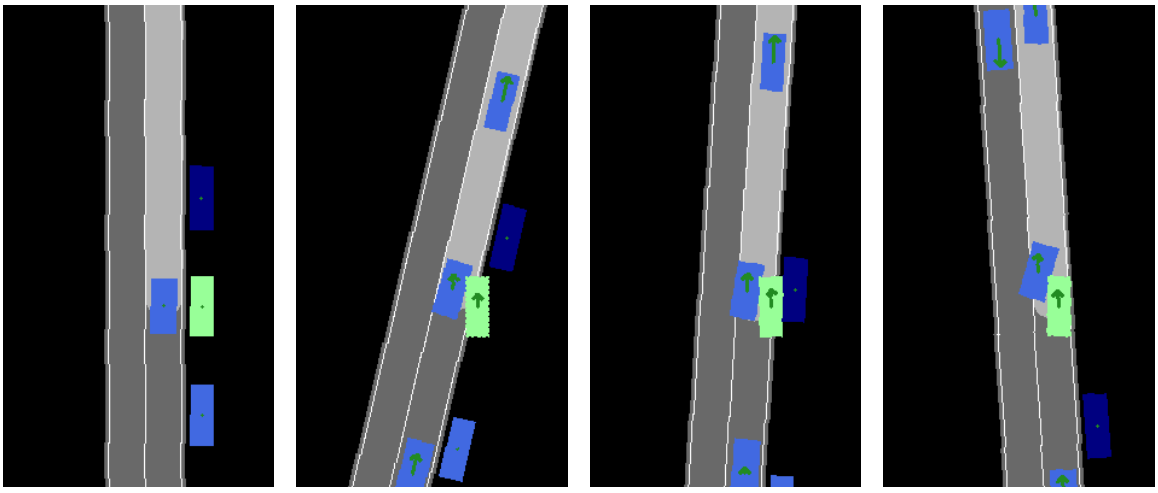
For the Leaderboard 2.0 results, 15 metrics were utilized to assess the performance of the models. These metrics are detailed in Table 5.2, where each abbreviation is associated with its full metric name.

As demonstrated in Table 5.3, we conducted performance comparisons of the agents in two distinct Towns: Town12 and Town13. The evaluations are based on averages derived from 90 routes in Town12 and 20 routes in Town13, as stipulated in Leaderboard 2.0. PRIBOOT consistently outperformed Autopilot across nearly all metrics in both towns, often by substantial margins. In Town12, for instance, PRIBOOT's DS was approximately 19 times higher than that of Autopilot, and its IRS was 84 times better. Similar trends were observed in Town13, with PRIBOOT achieving 19 times higher DS and 214 times higher IRS than Autopilot. Notably, PRIBOOT recorded zero collisions with pedestrians per kilometer in Town12 and only 0.01 collisions per kilometer in Town13, underscoring its effectiveness in minimizing accidents involving pedestrians.

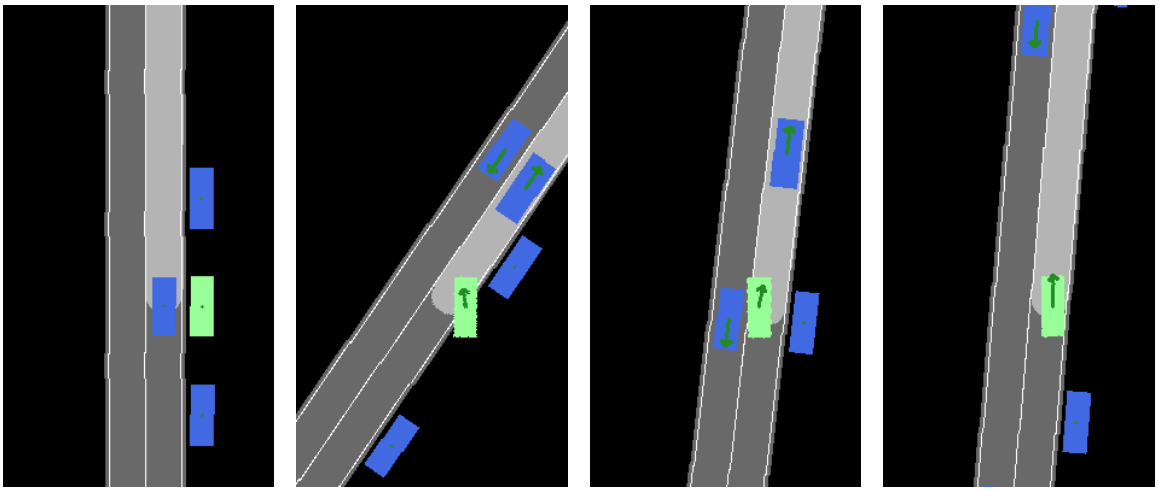
In contrast, Autopilot demonstrated superior performance in two specific metrics: yielding to emergency vehicles and maintaining minimum speed. The former was due to its lower RC score, which resulted in zero scenarios requiring yielding to an emergency vehicle. The latter stems from Autopilot operating at a fixed target speed consistently above the minimum speed requirement for the roads where the agent drove.

PRIBOOT stands out as the first agent to achieve a RC of approximately 75% in both towns, coupled with a satisfactory DS and IRS. This marks a significant milestone, positioning PRIBOOT as a pioneering agent capable of navigating the complexities of the benchmark, which can be used for data collection or online demonstrations.

To enhance the quantitative comparison presented earlier, we also include a qualitative evaluation. Our analysis of all routes in the benchmark reveals that Autopilot struggles

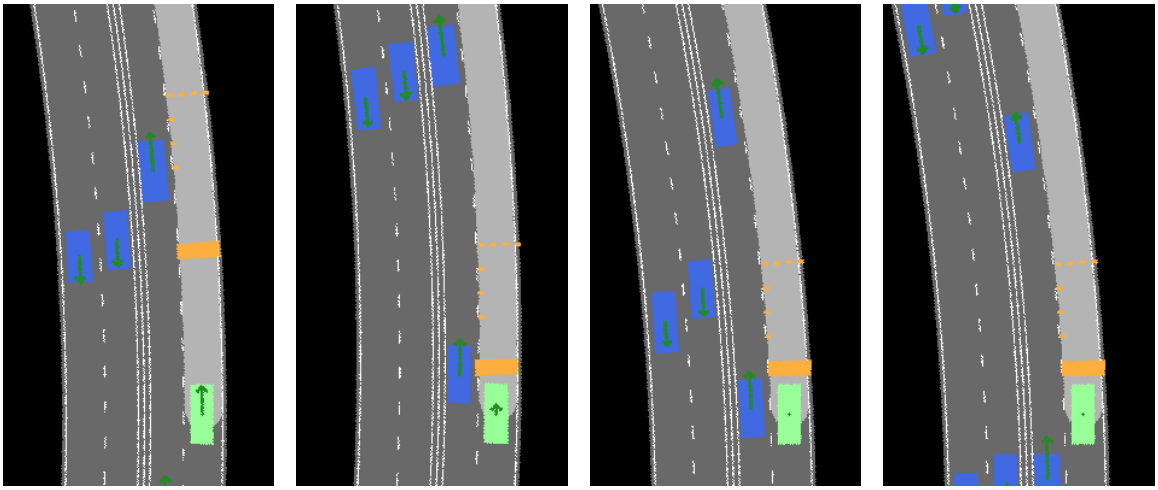


(a) Autopilot: t=0s (b) Autopilot: t=2s (c) Autopilot: t=3s (d) Autopilot: t=4s

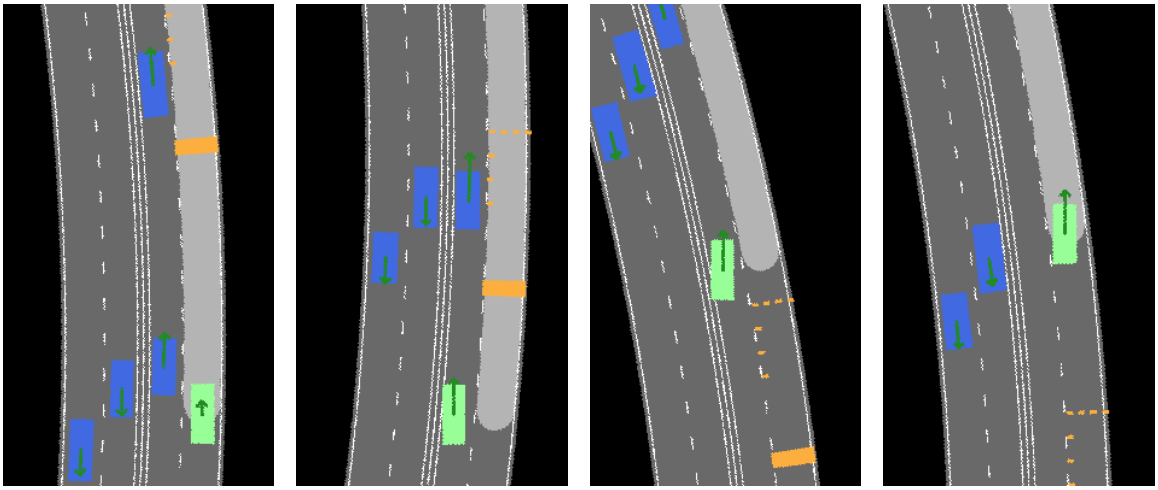


(e) PRIBOOT: t=0s (f) PRIBOOT: t=8s (g) PRIBOOT: t=9s (h) PRIBOOT: t=10s

Figure 5.4: Qualitative comparison in a parking exit scenario between Autopilot and PRIBOOT. The first row depicts a sequence of keyframes from Autopilot, while the second row shows the keyframes from PRIBOOT.



(a) Autopilot: $t = 0$s (b) Autopilot: $t = 2$s (c) Autopilot: $t = 4$s (d) Autopilot: $t = 6$s



(e) PRIBOOT: $t = 0$s (f) PRIBOOT: $t = 2$s (g) PRIBOOT: $t = 4$s (h) PRIBOOT: $t = 6$s

Figure 5.5: Qualitative comparison in a lane obstacle scenario between Autopilot and PRIBOOT. The first row depicts a sequence of keyframes from Autopilot, while the second row shows the keyframes from PRIBOOT.

with the novel scenarios introduced by Leaderboard 2.0, particularly those requiring slight deviations from the global planner's trajectory. These scenarios include instances like parking exits and lane obstacles. Figure 5.4 illustrates a sequence of keyframes in a parking exit scenario, first showing Autopilot's performance and then PRIBOOT's. As shown, Autopilot immediately exits the park without considering the vehicles in the lane, leading to a collision. Conversely, PRIBOOT waits for a moment when the lane is clear before exiting, as expected. Figure 5.5 also shows a sequence of keyframes, this time involving an obstacle in the lane. Here, Autopilot approaches the obstacle and then stops, whereas PRIBOOT slightly deviates from the trajectory to avoid the obstacle and returns to the original path once it is clear.

Table 5.4: Ablation Study: Driving performance and infraction analysis of PRIBOOT variants on CARLA Leaderboard 2.0 in Town13.

		DS ↑ %	**IRS ↑** %	**RC ↑** %	**IP ↑** %	**C.P ↓** #/Km	**C.V ↓** #/Km	**C.L ↓** #/Km	**R.L ↓** #/Km	**Stop ↓** #/Km	**O.R ↓** #/Km	**R.D ↓** #/Km	**Block ↓** #/Km	**Y.E ↓** #/Km	**S.T ↓** #/Km	**M.S ↓** #/Km
Town13	w/o aug	2.66	5.22	15.92	**0.29**	**0.01**	1.22	0.35	**0.00**	0.02	0.45	0.02	0.40	**0.00**	0.17	**0.02**
	w/ masks	5.55	21.08	54.02	0.18	0.05	0.52	0.23	0.01	**0.00**	0.08	0.03	0.06	0.05	0.11	0.14
	PRIBOOT	**18.84**	**46.97**	**74.29**	0.24	**0.01**	**0.34**	**0.05**	**0.00**	0.01	**0.05**	**0.00**	**0.04**	0.02	**0.02**	0.06

These qualitative comparisons demonstrate that PRIBOOT is better equipped to handle the challenging new driving scenarios introduced by Leaderboard 2.0.

Additionally, we provide access to a series of demonstration videos that illustrate the performance of PRIBOOT on Leaderboard 2.0. These can be accessed here.

5.4.3 Ablation Study

To explore the individual contributions of key components within PRIBOOT, particularly under conditions of limited data, we performed an ablation study focusing on two crucial elements: data augmentation and the utilization of RGB BEV in conjunction with transfer learning. This study involved training two variants of PRIBOOT: the first variant (referred to as "w/o aug") was developed without the data augmentations depicted in Figure 5.3b, and the second variant (referred to as "w/ masks") employed the BEV as a set of masks and training a CNN from scratch, consistent with methodologies reported in the literature [42].

The comparative analysis of driving performance and infractions for these variants is presented in Table 5.4. The w/o aug variant exhibited a significant decline in performance, as evidenced by a RC of approximately 16%, which adversely affected all other performance metrics. The reason for this is that due to planning or control inaccuracies, the agent encounters situations where it deviates from the center of the lane and lacks the capability to effectively recover. On the other hand, the w/ masks variant demonstrated improved results compared to the w/o aug variant. However, it still fell short of the full PRIBOOT system's capabilities. Specifically, the w/ masks variant scored three times lower in terms of DS and two times lower in terms of IRS.

Figure 5.6 illustrates the validation loss across epochs for the ablations considered. While the w/o aug variant achieves results similar to PRIBOOT during training, it recurrently faces out-of-distribution events, as detailed in Table 5.4. In contrast, the w/ masks variant displays a distinct pattern in validation loss: it requires 20% more epochs to converge and converges at a loss value that is twice that of PRIBOOT. This performance deficit underscores the critical role of utilizing RGB BEV and transfer learning techniques in cases where data availability is limited.

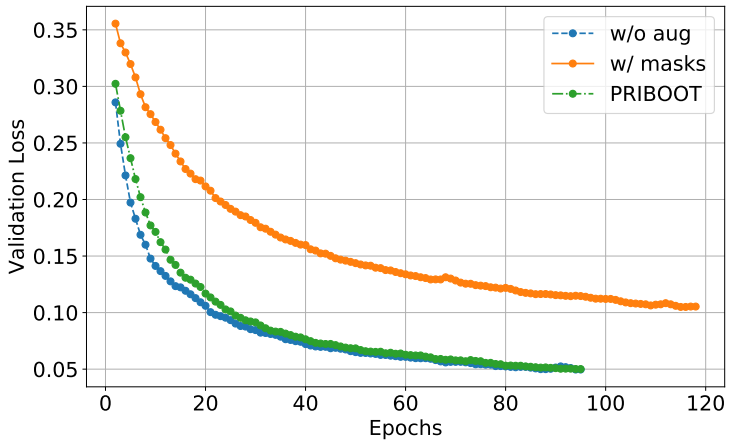


Figure 5.6: Validation loss across epochs of PRIBOOT variants.

5.5 Conclusion

In this paper, we introduced PRIBOOT, a system that utilizes privileged information alongside limited human driving logs to establish the first expert with satisfactory driving performance on the CARLA Leaderboard 2.0. Our results demonstrate that PRIBOOT significantly outperforms Autopilot across nearly all benchmark metrics, highlighting its superior capability in complex and challenging autonomous driving scenarios. Additionally, we presented an ablation study that evaluates the impact of using augmentations to aid recovery processes. Furthermore, we demonstrated the benefits of employing RGB BEV images with transfer learning, which proved more efficient in terms of training speed and performance than using masks and training a CNN from scratch. While our work has focused on the CARLA simulator, it is important to note that the idea behind PRIBOOT can eventually be applied in other simulators. In the future, we plan to employ PRIBOOT to generate large datasets that can be instrumental in training student models that receive sensor information as input.

Chapter 6

Discussion and Concluding Remarks

6.1 Discussion

In this section, we provide a detailed discussion of each chapter, clearly outlining their contributions and addressing any limitations encountered. Furthermore, we establish a cohesive flow and interconnection between the chapters to present a comprehensive overview of the conducted research.

Chapter 2 presents a review of end-to-end AD in urban environments. This work lays the foundation of end-to-end versus modular approaches, detailing the first end-to-end models proposed in the CARLA simulator. We address the general architectures of state-of-the-art end-to-end systems and discuss the various input and output modalities employed by these systems. Methods are categorized into IL and RL, with no single method proving superior in our studies. Over time, researchers have focused on incorporating additional modalities beyond camera images, with LiDAR emerging as a prevalent choice due to its rich spatial information. In terms of output modalities, most systems utilize steering angle, throttle, and braking, though some employ waypoints as the output of the learning system, subsequently relying on PID controllers to generate final control commands. We also provided a quantitative comparison of several end-to-end systems on the NoCrash benchmark, revealing that current methods struggle with dense traffic, indicating the need for further research. A critical analysis was presented, highlighting the most promising avenues in end-to-end research. IL, while instrumental in the initial development of AD systems, faces several limitations. Firstly, IL models tend to learn from the most frequent patterns in the training data, often neglecting rare but critical scenarios. This bias towards the average cases can lead to suboptimal performance in edge cases, such as rare traffic situations or unexpected obstacles. RL, on the other hand, offers significant advantages. RL enables the system to learn optimal policies through trial and error, exploring a wider range of scenarios and learning from interactions with the environment. This capacity for exploration allows RL models to handle rare and complex situations more effectively than IL. Based on these insights, advancing research in end-to-end AD with a focus on RL appears promising and formed the main topic of this Ph.D. research.

In Chapter 3, we explored the application of RLfP in AD. We observed that no prior approaches had successfully implemented this due to challenges such as sample inefficiency, degenerated feature representations, and catastrophic self-overfitting. Instead, most researchers adopted a two-stage process: first, training large visual encoders using SL techniques, followed by training the policy network with RL. However, this paradigm often results in environmental representations that are misaligned with the downstream task, leading to suboptimal performance. To address these limitations, we proposed RLAD, the first RLfP method applied in the urban AD domain. We introduced several techniques to enhance the performance, including: 1) an image encoder that utilizes both image augmentations and Adaptive Local Signal Mixing (A-LIX) layers; 2) WayConv1D, a waypoint encoder that captures the 2D ge-

ometrical information of waypoints using 1D convolutions; and 3) an auxiliary loss function to emphasize the significance of traffic lights in the latent representation of the environment. Experimental results demonstrate that RLAD significantly outperforms all state-of-the-art RLfP methods on the NoCrash benchmark. However, RLAD is not yet competitive with state-of-the-art AD systems that incorporate demonstrations. This led to the next chapter of this Ph.D. research: integrating demonstrations into RLAD.

In Chapter 4, we explored the integration of expert demonstrations into RL training, specifically within the RLAD framework. RLfD has emerged as a powerful technique that combines the strengths of both IL and RL. However, two significant challenges remain: 1) achieving the optimal balance between the contributions of IL and RL is complex, and 2) addressing the potential distribution gap between the demonstrations and the training environment. To address these challenges, we introduced RLfOLD, which stands for Reinforcement Learning from Online Demonstrations. RLfOLD integrates IL and RL by utilizing online demonstrations, effectively bridging the distribution gap between the demonstrations and training environment. We proposed a policy network that outputs two standard deviations, allowing for adaptive control during exploration and IL training while accounting for uncertainty in both domains. Additionally, we incorporated an uncertainty-based technique, guided by an online expert, to enhance the exploration process. Our results on the NoCrash benchmark demonstrate the superior effectiveness and efficiency of RLfOLD, outperforming state-of-the-art methods even with reduced resources.

After achieving top performance on the NoCrash benchmark, we advanced to the more recent and challenging CARLA benchmark: Leaderboard 2.0. To this date, all approaches tested on the Leaderboard 2.0 benchmark have shown very poor performance, with the highest RC reaching only 15% and the highest DS just 1%. This notable poor performance is primarily due to the insufficiency of available training data. CARLA provides a limited set of human driving logs from a few route scenarios, which are inadequate for training models that rely heavily on sensor information. While researchers could previously leverage online experts like CARLA Autopilot or Roach to generate demonstrations in Leaderboard 1.0, these experts are either markedly less effective or completely ineffective in Leaderboard 2.0. To foster innovation in this benchmark, we proposed PRIBOOT, an expert agent capable of navigating the demanding scenarios presented in Leaderboard 2.0. PRIBOOT leverages privileged information and limited human driving logs, marking the first instance of achieving significant performance milestones on the CARLA Leaderboard 2.0. A key component of PRIBOOT is the development of a BEV representation specifically tailored to address the complex demands of Leaderboard 2.0. By processing this BEV as an RGB image rather than a set of masks, we facilitate the application of transfer learning techniques, significantly enhancing model performance and efficiency, especially in the context of limited data availability. Additionally, we introduced the IRS, a novel evaluation metric that considers infractions per kilometer rather

than just the number of infractions. This metric is designed to complement the DS by providing a more detailed assessment of driving behavior over long routes. PRIBOOT stands as the best expert agent capable of navigating this challenging benchmark, enabling researchers to generate extensive datasets and potentially resolving the data availability issues that have hindered progress in this benchmark.

6.2 Conclusion

In this thesis, we have made significant advancements in the field of end-to-end AD, particularly within urban environments. Our research began with a comprehensive review of existing end-to-end AD systems, establishing a foundational understanding of the strengths and limitations of both IL and RL approaches. We identified critical areas for improvement, particularly the need for more robust handling of complex and rare driving scenarios.

Building on this foundation, we introduced RLAD, the first RLfP method applied to urban AD. RLAD demonstrated superior performance on the NoCrash benchmark, although further integration of demonstrations was necessary to match state-of-the-art AD systems. To address this, we developed RLfOLD, which effectively combines IL and RL by incorporating online demonstrations, achieving state-of-the-art results with improved efficiency and resource utilization.

Finally, we tackled the significant challenge posed by the CARLA Leaderboard 2.0 benchmark with PRIBOOT, an expert agent that leverages privileged information and limited human driving logs to navigate demanding scenarios. PRIBOOT can now be used to generate extensive datasets, potentially solving the data availability issues that have hindered progress in this benchmark.

Collectively, our work not only advances the state of the art in end-to-end AD but also provides valuable methodologies and insights for future research in this rapidly evolving field.

6.3 Contributions

In this section, the contributions made in this Ph.D. thesis are outlined, each corresponding to a specific chapter, which comprises the following scientific articles:

- **A Review of End-to-End Autonomous Driving in Urban Environments** [43]

  Authors: Daniel Coelho and Miguel Oliveira

  Journal: IEEE Access

  Year: 2022

- **RLAD: Reinforcement Learning From Pixels for Autonomous Driving in Urban Environments** [44]

Authors: Daniel Coelho, Miguel Oliveira, and Vítor Santos

Journal: IEEE Transactions on Automation Science and Engineering

Year: 2023

Code: https://github.com/DanielCoelho112/rlad

- **RLfOLD: Reinforcement Learning from Online Demonstrations in Urban Autonomous Driving** [45]

  Authors: Daniel Coelho, Miguel Oliveira, and Vítor Santos

  Conference: Proceedings of the AAAI Conference on Artificial Intelligence

  Year: 2024

  Code: https://github.com/DanielCoelho112/rlfold

- **PRIBOOT: A New Data-Driven Expert for Improved Driving Simulations**

  Authors: Daniel Coelho, Miguel Oliveira, Vítor Santos, and Antonio M. López

  Journal: Submitted at IEEE Transactions on Automation Science and Engineering

  Code: https://github.com/DanielCoelho112/priboot

6.4 Future Directions

The research presented in this thesis opens several promising avenues for future exploration in the realm of end-to-end AD. Despite the advancements achieved, there remain numerous challenges and opportunities for further enhancement and innovation.

Firstly, enhancing RLfOLD is a promising direction for future research. In the current work, RLfOLD utilized a rule-based expert to provide demonstrations, which, while effective, is suboptimal compared to more sophisticated methods. A promising future avenue could involve replacing the rule-based expert with a learning-based expert, with PRIBOOT serving as an optimal choice for this enhancement. This could potentially lead to higher-quality demonstrations and improved training efficiency, leveraging the advanced capabilities of learning-based systems to provide more nuanced and contextually aware guidance during the training process.

Another exciting direction involves the integration of Large Language Models (LLMs) into AD systems, particularly in conjunction with RL. LLMs can be leveraged to enhance RL by providing richer contextual understanding and more sophisticated decision-making capabilities. In terms of interpretability, LLMs can also facilitate more understandable and transparent decision-making processes within AD systems. By integrating LLMs, autonomous

vehicles can explain their actions in human-readable terms, providing insights into the reasoning behind specific maneuvers or decisions. This can significantly enhance trust and reliability in AD technologies, making their operations more transparent and easier to scrutinize.

Lastly, transfer learning from simulation to reality remains a crucial task. While simulation provides a controlled environment for training and testing AD systems, the transition to real-world scenarios poses significant challenges due to the discrepancies between simulated and real environments. Future research should focus on developing advanced transfer learning techniques to bridge this gap effectively. This includes refining simulation environments to better mimic real-world conditions and developing algorithms capable of adapting knowledge gained in simulations to real-world driving situations seamlessly. This will be instrumental in ensuring that the advancements achieved in simulated environments can be reliably translated into practical, real-world applications, thereby accelerating the deployment of robust and safe autonomous vehicles.

Overall, the future of end-to-end AD research is filled with opportunities to improve system robustness, adaptability, and efficiency, ultimately leading to safer and more reliable autonomous vehicles.

References

[1] D. Parekh, N. Poddar, A. Rajpurkar, M. Chahal, N. Kumar, G. P. Joshi, and W. Cho, "A review on autonomous vehicles: Progress, methods and challenges," *Electronics*, vol. 11, no. 14, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/14/2162

[2] H.-C. Shao, L. Wang, R. Chen, H. Li, and Y. T. Liu, "Safety-enhanced autonomous driving using interpretable sensor fusion transformer," *ArXiv*, vol. abs/2207.14024, 2022.

[3] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided Control Prediction for End-to-end Autonomous Driving: A Simple yet Strong Baseline," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 6119–6132. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/286a371d8a0a559281f682f8fbf89834-Paper-Conference.pdf

[4] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. M. Allen, V. D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 8248–8254, 2019.

[5] A. Hu, L. Russell, H. Yeo, Z. Murez, G. Fedoseev, A. Kendall, J. Shotton, and G. Corrado, "Gaia-1: A generative world model for autonomous driving," *arXiv preprint arXiv:2309.17080*, 2023.

[6] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[7] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous vehicles: disengagements, accidents and reaction times," *PLoS one*, vol. 11, no. 12, p. e0168054, 2016.

[8] C. D. Harper, C. T. Hendrickson, S. Mangones, and C. Samaras, "Estimating potential increases in travel with autonomous vehicles for the non-driving, elderly and people with travel-restrictive medical conditions," *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 1–9, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X16301590

[9] T. Litman, "Autonomous vehicle implementation predictions: Implications for transport planning," 2020.

[10] Y. He, B. Ciuffo, Q. Zhou, M. Makridis, K. Mattas, J. Li, Z. Li, F. Yan, and H. Xu, "Adaptive cruise control strategies implemented on experimental vehicles: A review,"

IFAC-PapersOnLine, vol. 52, no. 5, pp. 21–27, 2019, 9th IFAC Symposium on Advances in Automotive Control AAC 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319306238

[11] Y. Jeong, "Interactive lane keeping system for autonomous vehicles using lstm-rnn considering driving environments," *Sensors*, vol. 22, no. 24, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/24/9889

[12] J. B. Cicchino, "Effects of automatic emergency braking systems on pedestrian crash risk," *Accident Analysis Prevention*, vol. 172, p. 106686, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457522001221

[13] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, pp. 18–25, 2018.

[14] Y. Wu, S. Liao, X. Liu, Z. Li, and R. Lu, "Deep Reinforcement Learning on Autonomous Driving Policy With Auxiliary Critic Network," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2021.

[15] Z. Huang, S. Sun, J. Zhao, and L. Mao, "Multi-modal policy fusion for end-to-end autonomous driving," *Information Fusion*, vol. 98, p. 101834, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253523001501

[16] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742030628X

[17] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[18] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 163–168, 2011.

[19] 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.

[20] G. Lan and Q. Hao, "End-to-end planning of autonomous driving in industry and academia: 2022-2023," *arXiv e-prints*, pp. arXiv–2401, 2023.

[21] L. Waymo, "On the road to fully self-driving," *Waymo Safety Report*, pp. 1–43, 2017.

[22] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 8652–8661, 2019.

[23] B. Peng, Q. Sun, S. E. Li, D. Kum, Y. Yin, J. Wei, and T. Gu, "End-to-End Autonomous Driving Through Dueling Double Deep Q-Network," *Automotive Innovation*, vol. 4, no. 3, pp. 328–337, 2021. [Online]. Available: https://doi.org/10.1007/s42154-021-00151-3

[24] T. Agarwal, H. Arora, and J. Schneider, "Learning Urban Driving Policies using Deep Reinforcement Learning," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2021-Septe, pp. 607–614, 2021.

[25] A. Hu, G. Corrado, N. Griffiths, Z. Murez, C. Gurau, H. Yeo, A. Kendall, R. Cipolla, and J. Shotton, "Model-based imitation learning for urban driving," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20 703–20 716, 2022.

[26] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016. [Online]. Available: https://arxiv.org/abs/1604.07316

[27] G. Varisteas, R. Frank, S. A. S. Alamdari, H. Voos, and R. State, "Evaluation of end-to-end learning for autonomous driving: The good, the bad and the ugly," *2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS)*, pp. 110–117, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:145967509

[28] Tesla, "Tesla ai day," 2021, online; accessed 17-May-2024. [Online]. Available: https://www.tesla.com/ai

[29] C. AI, "Comma ai openpilot," 2021, online; accessed 17-May-2024. [Online]. Available: https://comma.ai/

[30] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[31] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[33] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A Survey of End-to-End Driving: Architectures and Training Methods," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[34] 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.

[35] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *AAAI Conference on Artificial Intelligence*, 2019.

[36] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Efficiently Combining Human Demonstrations and Interventions for Safe Training of Autonomous Systems in Real-Time," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 2462–2470, 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4091

[37] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[38] R. Chekroun, M. Toromanoff, S. Hornauer, and F. Moutarde, "Gri: General reinforced imitation and its application to vision-based autonomous driving," *ArXiv*, vol. abs/2111.08575, 2021.

[39] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," no. CoRL, pp. 1–16, 2017. [Online]. Available: http://arxiv.org/abs/1711.03938

[40] Y. Li, W. Yuan, S. Zhang, W. Yan, Q. Shen, C. Wang, and M. Yang, "Choose your simulator wisely: A review on open-source simulators for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2024.

[41] P. Kaur, S. Taghavi, Z. Tian, and W. Shi, "A Survey on Simulators for Testing Self-Driving Cars," *Proceedings - 2021 4th International Conference on Connected and Autonomous Driving, MetroCAD 2021*, pp. 62–70, 2021.

[42] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. V. Gool, "End-to-end urban driving by imitating a reinforcement learning coach," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15 202–15 212, 2021.

[43] D. Coelho and M. Oliveira, "A review of end-to-end autonomous driving in urban environments," *IEEE Access*, vol. 10, pp. 75 296–75 311, 2022.

[44] D. Coelho, M. Oliveira, and V. Santos, "Rlad: Reinforcement learning from pixels for autonomous driving in urban environments," *IEEE Transactions on Automation Science and Engineering*, 2023.

[45] ——, "Rlfold: Reinforcement learning from online demonstrations in urban autonomous driving," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 10, 2024, pp. 11 660–11 668.

[46] J. Laconte, A. Kasmi, R. Aufrère, M. Vaidis, and R. Chapuis, "A Survey of Localization Methods for Autonomous Vehicles in Highway Scenarios," *Sensors*, vol. 22, no. 1, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/1/247

[47] E. Horváth, C. Pozna, and M. Unger, "Real-Time LIDAR-Based Urban Road and Sidewalk Detection for Autonomous Vehicles," *Sensors*, vol. 22, no. 1, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/1/194

[48] Y. B. Chang, C. Tsai, C. H. Lin, and P. Chen, "Real-time semantic segmentation with dual encoder and self-attention mechanism for autonomous driving," *Sensors*, vol. 21, no. 23, 2021.

[49] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, A. Patsekin, J. Kindelsberger, L. Ding, S. Seaman, A. Mehler, A. Sipperley, A. Pettinato, B. D. Seppelt, L. Angell, B. Mehler, and B. Reimer, "Mit advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation," *IEEE Access*, vol. 7, pp. 102 021–102 038, 2019.

[50] Y. Zhang, H. Chen, S. L. Waslander, J. Gong, G. Xiong, T. Yang, and K. Liu, "Hybrid trajectory planning for autonomous driving in highly constrained environments," *IEEE Access*, vol. 6, pp. 32 800–32 819, 2018.

[51] C. Sun, X. Zhang, Q. Zhou, and Y. Tian, "A model predictive controller with switched tracking error for autonomous vehicle path tracking," *IEEE Access*, vol. 7, pp. 53 103–53 114, 2019.

[52] M. Bansal, A. Krizhevsky, and A. S. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *ArXiv*, vol. abs/1812.03079, 2019.

[53] E. Santana and G. Hotz, "Learning a Driving Simulator," pp. 1–8, 2016. [Online]. Available: http://arxiv.org/abs/1608.01230

[54] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall, "FIERY: Future instance segmentation in bird's-eye view from surround monocular cameras," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[55] T. Arakawa, "Trends and Future Prospects of the Drowsiness Detection and Estimation Technology," *Sensors*, vol. 21, no. 23, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/23/7921

[56] Singh, S. (2018, March). Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey. (Traffic Safety Facts Crash • Stats. Report No. DOT HS 812 506). Washington, DC: National Highway Traffic Safety Administration.

[57] P. Andersson and P. Ivehammar, "Benefits and Costs of Autonomous Trucks and Cars," *Journal of Transportation Technologies*, vol. 09, no. 02, pp. 121–145, 2019.

[58] S. Arshad, M. Sualeh, D. Kim, D. V. Nam, and G.-W. Kim, "Clothoid: An Integrated Hierarchical Framework for Autonomous Driving in a Dynamic Urban Environment," *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5053

[59] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[60] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[61] C. Gómez-Huélamo, J. Del Egido, L. M. Bergasa, R. Barea, E. López-Guillén, F. Arango, J. Araluce, and J. López, "Train here, drive there: ROS based end-to-end autonomous-driving pipeline validation in CARLA simulator using the NHTSA typology," *Multimedia Tools and Applications*, no. 0123456789, 2021. [Online]. Available: https://doi.org/10.1007/s11042-021-11681-7

[62] C. Urmson, J. Anhalt, J. A. Bagnell, C. R. Baker, R. Bittner, M. N. Clark, J. M. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. M. Peterson, B. Pilnick, R. R. Rajkumar, P. E. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. M. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. N. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," in *The DARPA Urban Challenge*, 2009.

[63] S. Yang, X. Mao, S. Yang, Z. Liu, G. Chen, S. Wang, J. Xue, and Z. Xu, "Towards a robust software architecture for autonomous robot software," *2017 7th International Workshop on Computer Science and Engineering, WCSE 2017*, no. April, pp. 1197–1207, 2017.

[64] M. Quigley, "Ros: an open-source robot operating system," in *ICRA 2009*, 2009.

[65] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. Lopez, "Multimodal End-to-End Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 537–547, 2020.

[66] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[67] X.-W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[68] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[69] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[70] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39 974–39 982, 2019.

[71] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Li, "Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach," *IEEE Access*, vol. 6, pp. 25 463–25 473, 2018.

[72] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[73] O.-E. L. Team, A. Stooke, A. Mahajan, C. Barros, C. Deck, J. Bauer, J. Sygnowski, M. Trebacz, M. Jaderberg, M. Mathieu, N. McAleese, N. Bradley-Schmieg, N. Wong, N. Porcel, R. Raileanu, S. Hughes-Fitt, V. Dalibard, and W. M. Czarnecki, "Open-Ended Learning Leads to Generally Capable Agents," 2021. [Online]. Available: http://arxiv.org/abs/2107.12808

[74] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by Cheating," *Conference on Robot Learning (CoRL) 2019*, no. CoRL.

[75] M. Ahmed, A. Abobakr, C. P. Lim, and S. Nahavandi, "Policy-Based Reinforcement Learning for Training Autonomous Driving Agents in Urban Areas With Affordance Learning," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2021.

[76] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[77] F. Codevilla, M. Miiller, A. Lopez, V. Koltun, and A. Dosovitskiy, "End-to-End Driving Via Conditional Imitation Learning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4693–4700, 2018.

[78] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning," *Sensors*, vol. 21, no. 4, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1278

[79] W. Zu, H. Yang, R. Liu, and Y. Ji, "A Multi-Dimensional Goal Aircraft Guidance Approach Based on Reinforcement Learning with a Reward Shaping Algorithm," *Sensors*, vol. 21, no. 16, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5643

[80] H. Hu, Z. Lu, Q. Wang, and C. Zheng, "End-to-end automated lane-change maneuvering considering driving style using a deep deterministic policy gradient algorithm," *Sensors (Switzerland)*, vol. 20, no. 18, pp. 1–22, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5443

[81] X. Liang, T. Wang, L. Yang, and E. Xing, "CIRL: Controllable imitative reinforcement learning for vision-based self-driving," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11211 LNCS, pp. 604–620, 2018.

[82] H. Yi, E. Park, and S. Kim, "Multi-agent Deep Reinforcement Learning for Autonomous Driving," *KIISE Transactions on Computing Practices*, vol. 24, no. 12, pp. 670–674, 2018.

[83] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car," pp. 1–8, 2017. [Online]. Available: http://arxiv.org/abs/1704.07911

[84] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," *Proceedings of the American Control Conference*, pp. 4914–4919, 2017.

[85] Z. Chen and X. Huang, "End-To-end learning for lane keeping of self-driving cars," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1856–1860, 2017.

[86] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies," *Conference: 31st Conference on Neural Information Processing Systems (NIPS 2017), MLITS workshop*, no. December, 2017. [Online]. Available: http://arxiv.org/abs/1710.03804

[87] S. K. Kwon, J. H. Seo, J. W. Lee, and K. D. Kim, "An Approach for Reliable End-to-End Autonomous Driving Based on the Simplex Architecture," *2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018*, pp. 1851–1856, 2018.

[88] Y. Wang, D. Liu, H. Jeon, Z. Chu, and E. T. Matson, "End-to-end learning approach for autonomous driving: A convolutional neural network model," *ICAART 2019 - Proceedings of the 11th International Conference on Agents and Artificial Intelligence*, vol. 2, no. Icaart, pp. 833–839, 2019.

[89] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: Approaches, lessons and challenges," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4649–4672, 2010.

[90] M. Aria, "A Survey of Self-driving Urban Vehicles Development," *IOP Conference Series: Materials Science and Engineering*, vol. 662, no. 4, 2019.

[91] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9328–9337, 2019.

[92] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, no. Figure 1, pp. 2722–2730, 2015.

[93] A. Sauer, N. Savinov, and A. Geiger, "Conditional Affordance Learning for Driving in Urban Environments," *Conference on Robot Learning (CoRL)*, no. June, 2018. [Online]. Available: http://arxiv.org/abs/1806.06498

[94] A. Mehta, A. Subramanian, and A. Subramanian, "Learning End-to-end Autonomous Driving using Guided Auxiliary Supervision," *ICVGIP 2018: 11th Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 1–8, 2018.

[95] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2022.

[96] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-December, no. Nips, pp. 5999–6009, 2017.

[97] Y. Gao and D. Glowacka, "Deep gate recurrent neural network," *Journal of Machine Learning Research*, vol. 63, pp. 350–365, 2016.

[98] G. Dai, C. Ma, and X. Xu, "Short-term traffic flow prediction method for urban road sections based on space–time analysis and gru," *IEEE Access*, vol. 7, pp. 143 025–143 035, 2019.

[99] L. Chen, X. Hu, B. Tang, and Y. Cheng, "Conditional DQN-Based Motion Planning With Fuzzy Logic for Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020.

[100] 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*, pp. 1–11, 2021.

[101] C. Huang, R. Zhang, M. Ouyang, P. Wei, J. Lin, J. Su, and L. Lin, "Deductive Reinforcement Learning for Visual Autonomous Urban Driving Navigation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5379–5391, 2021.

[102] R. F. J. Dossa, S. Huang, S. Ontañón, and T. Matsubara, "An empirical investigation of early stopping optimizations in proximal policy optimization," *IEEE Access*, vol. 9, pp. 117 981–117 992, 2021.

[103] S. H. Silva, A. Alaeddini, and P. Najafirad, "Temporal graph traversals using reinforcement learning with proximal policy optimization," *IEEE Access*, vol. 8, pp. 63 910–63 922, 2020.

[104] U. M. Gidado, H. Chiroma, N. Aljojo, S. Abubakar, S. I. Popoola, and M. A. Al-Garadi, "A survey on deep learning for steering angle prediction in autonomous vehicles," *IEEE Access*, vol. 8, pp. 163 797–163 817, 2020.

[105] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–1780, 1997.

[106] E. Jo, M. Sunwoo, and M. Lee, "Vehicle Trajectory Prediction Using Hierarchical Graph Neural Network for Considering Interaction among Multimodal Maneuvers," *Sensors*, vol. 21, no. 16, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5354

[107] C. E. Van Uden, S. A. Nastase, A. C. Connolly, M. Feilong, I. Hansen, M. I. Gobbini, and J. V. Haxby, "Modeling Semantic Encoding in a Common Neural Representational Space," *Frontiers in Neuroscience*, vol. 12, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00437

[108] S. Zheng and H. Liu, "Improved multi-agent deep deterministic policy gradient for path planning-based crowd simulation," *IEEE Access*, vol. 7, pp. 147 755–147 770, 2019.

[109] Y.-H. Xu, C.-C. Yang, M. Hua, and W. Zhou, "Deep deterministic policy gradient (ddpg)-based resource allocation scheme for noma vehicular communications," *IEEE Access*, vol. 8, pp. 18 797–18 807, 2020.

[110] Y. Qi, C. Shen, D. Wang, J. Shi, X. Jiang, and Z. Zhu, "Stacked sparse autoencoder-based deep network for fault diagnosis of rotating machinery," *IEEE Access*, vol. 5, pp. 15 066–15 079, 2017.

[111] M. Toromanoff, É. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7151–7160, 2019.

[112] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[113] J. R. Sánchez-Ibáñez, C. J. Pérez-del Pulgar, and A. García-Cerezo, "Path Planning for Autonomous Mobile Robots: A Review," *Sensors*, vol. 21, no. 23, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/23/7898

[114] M. Alharbi and H. A. Karimi, "A Global Path Planner for Safe Navigation of Autonomous Vehicles in Uncertain Environments," *Sensors*, vol. 20, no. 21, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6103

[115] P. Cai, S. Wang, Y. Sun, and M. Liu, "Probabilistic End-to-End Vehicle Navigation in Complex Dynamic Environments with Multimodal Sensor Fusion," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4218–4224, 2020.

[116] A. K. Guruji, H. Agarwal, and D. Parsediya, "Time-efficient A* Algorithm for Robot Path Planning," *Procedia Technology*, vol. 23, pp. 144–149, 2016.

[117] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, "An improved A-Star based path planning algorithm for autonomous land vehicles," *International Journal of Advanced Robotic Systems*, vol. 17, no. 5, p. 1729881420962263, 2020. [Online]. Available: https://doi.org/10.1177/1729881420962263

[118] B. Hekimoğlu, "Optimal tuning of fractional order pid controller for dc motor speed control via chaotic atom search optimization algorithm," *IEEE Access*, vol. 7, pp. 38 100–38 114, 2019.

[119] R. Gutiérrez, E. López-Guillén, L. M. Bergasa, R. Barea, Ó. Pérez, C. Gómez-Huélamo, F. Arango, J. Del Egido, and J. López-Fernández, "A waypoint tracking controller for autonomous road vehicles using ROS framework," *Sensors (Switzerland)*, vol. 20, no. 14, 2020.

[120] A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V. D. Lam, and A. Kendall, "Learning to drive from simulation without real world labels," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 4818–4824, 2019.

[121] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving Policy Transfer via Modularity and Abstraction," no. CoRL, 2018. [Online]. Available: http://arxiv.org/abs/1804.09364

[122] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 23–30, 2017.

[123] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *British Machine Vision Conference 2017, BMVC 2017*, 2017.

[124] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[125] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *ArXiv*, vol. abs/1806.10293, 2018.

[126] K. Chitta, A. Prakash, and A. Geiger, "Neat: Neural attention fields for end-to-end autonomous driving," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15 773–15 783, 2021.

[127] J. Wu, Z. Huang, Z. Hu, and C. Lv, "Toward human-in-the-loop ai: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving," *Engineering*, vol. 21, pp. 75–91, 2023.

[128] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *ArXiv*, vol. abs/2004.13649, 2020.

[129] E. Cetin, P. J. Ball, S. Roberts, and O. Çeliktutan, "Stabilizing off-policy deep reinforcement learning from pixels," in *International Conference on Machine Learning*, 2022.

[130] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *ArXiv*, vol. abs/1312.5602, 2013.

[131] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[132] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," *ArXiv*, vol. abs/2006.10742, 2020.

[133] Y. Zhao, K. Wu, Z. Xu, Z. Che, Q. Lu, J. Tang, and C. H. Liu, "Cadre: A cascade deep reinforcement learning framework for vision-based autonomous urban driving," in *AAAI Conference on Artificial Intelligence*, 2022.

[134] M. Hessel, J. Modayil, H. V. Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," *ArXiv*, vol. abs/1710.02298, 2017.

[135] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[136] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, "Loss is its own reward: Self-supervision for reinforcement learning," *ArXiv*, vol. abs/1612.07307, 2016.

[137] V. Mnih, A. P. Badia, L. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *33rd International Conference on Machine Learning, ICML 2016*, vol. 4, pp. 2850–2869, 2016.

[138] 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*, 2018.

[139] A. Srinivas, M. Laskin, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *International Conference on Machine Learning*, 2020.

[140] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Mastering visual continuous control: Improved data-augmented reinforcement learning," *ArXiv*, vol. abs/2107.09645, 2021.

[141] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[142] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2015.

[143] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *ArXiv*, vol. abs/1802.01561, 2018.

[144] A. F. Agarap, "Deep learning using rectified linear units (relu)," *ArXiv*, vol. abs/1803.08375, 2018.

[145] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2008.

[146] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *ArXiv*, vol. abs/1812.05905, 2018.

[147] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Neural Information Processing Systems*, 2019.

[148] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[149] D. Chen and P. Krähenbühl, "Learning from all vehicles," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17 201–17 210, 2022.

[150] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. P. Lillicrap, and M. A. Riedmiller, "Deepmind control suite," *ArXiv*, vol. abs/1801.00690, 2018.

[151] H. Liu, Z. Huang, J. Wu, and C. Lv, "Improved deep reinforcement learning with expert demonstrations for urban autonomous driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 921–928.

[152] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[153] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[154] N. Hansen, Y. Lin, H. Su, X. Wang, V. Kumar, and A. Rajeswaran, "Modem: Accelerating visual model-based reinforcement learning with demonstrations," in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022. [Online]. Available: https://openreview.net/forum?id=HSgg4RZ9qz

[155] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "Ensembledagger: A bayesian approach to safe imitation learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5041–5048.

[156] Z. Peng, Q. Li, C. Liu, and B. Zhou, "Safe driving via expert guided policy optimization," in *Conference on Robot Learning*. PMLR, 2022, pp. 1554–1563.

[157] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8077–8083.

[158] Q. Li, Z. Peng, and B. Zhou, "Efficient learning of safe driving policy via human-ai copilot optimization," *arXiv preprint arXiv:2202.10341*, 2022.

[159] S. Dey, S. Pendurkar, G. Sharon, and J. P. Hanna, "A joint imitation-reinforcement learning framework for reduced baseline regret," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3485–3491.

[160] CARLA, "CARLA Autonomous Driving Leaderboard," 2020. [Online]. Available: https://leaderboard.carla.org/

[161] A. Prakash, K. Chitta, and A. Geiger, "Multi-Modal Fusion Transformer for End-to-End Autonomous Driving," *2021 Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 7073–7083, 2021.

[162] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. P. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," in *AAAI Conference on Artificial Intelligence*, 2017.

[163] 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.

[164] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, B. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson *et al.*, "Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios," *arXiv preprint arXiv:2212.11419*, 2022.

[165] D. Chen, V. Koltun, and P. Krähenbühl, "Learning to drive from a world on rails," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 590–15 599.

[166] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5393–5402.

[167] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.

[168] A. Zhao, T. He, Y. Liang, H. Huang, G. V. d. Broeck, and S. Soatto, "Sam: Squeeze-and-mimic networks for conditional visual driving policy learning," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 156–175. [Online]. Available: https://proceedings.mlr.press/v155/zhao21a.html

[169] H. Shao, L. Wang, R. Chen, S. L. Waslander, H. Li, and Y. Liu, "Reasonnet: End-to-end driving with temporal and global reasoning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 13 723–13 733.

[170] W. Zhang, M. Elmahgiubi, K. Rezaee, B. Khamidehi, H. Mirkhani, F. Arasteh, C. Li, M. A. Kaleem, E. R. Corral-Soto, D. Sharma *et al.*, "Analysis of a modular autonomous driving architecture: The top submission to carla leaderboard 2.0 challenge," *arXiv preprint arXiv:2405.01394*, 2024.

[171] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *arXiv preprint arXiv:1709.07174*, 2017.

[172] X. Jia, Y. Gao, L. Chen, J. Yan, P. L. Liu, and H. Li, "Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7953–7963.

[173] Y. Xiao, F. Codevilla, D. Porres, and A. M. López, "Scaling vision-based end-to-end autonomous driving with multi-view attention learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1586–1593.

[174] 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*, 2009, pp. 248–255.

[175] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[176] W. G. Najm, J. D. Smith, and M. Yanagisawa, "Pre-crash scenario typology for crash avoidance research," John A. Volpe National Transportation Systems Center (U.S.), Technical Report DOT-VNTSC-NHTSA-06-02; DOT HS 810 767, 4 2007. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/6281