Investigating the Missing Pieces of Sensorimotor Reinforcement Learning Agents for Autonomous Driving

by

Ege Onat Özsüer

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

September 5, 2023

Investigating the Missing Pieces of Sensorimotor Reinforcement Learning Agents for Autonomous Driving

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Ege Onat Özsüer

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Asst. Prof. Fatma Güney

_____

Asst. Prof. Barış Akgün

_____

Assoc. Prof. Nazım Kemal Üre

_____

Prof. Yücel Yemez

Date: _____

ABSTRACT

Investigating the Missing Pieces of Sensorimotor Reinforcement Learning Agents for Autonomous Driving
Ege Onat Özsüer
Master of Science in Computer Science and Engineering
September 5, 2023

Reinforcement Learning (RL) has the potential to surpass human capabilities in self-driving without needing any expert supervision. Despite its promise, the state-of-the-art in sensorimotor self-driving is dominated by imitation learning methods due to the inherent challenges of RL algorithms. Nonetheless, RL agents are able to discover highly successful policies when provided with privileged ground truth representations of the environment. In this work, we investigate what separates privileged RL agents from sensorimotor agents for urban driving in order to bridge the gap between the two. We propose vision-based deep learning models to approximate the privileged representations from sensor data. In particular, we identify aspects of state representation that are crucial for the success of the RL agent such as desired route generation and traffic light prediction, and propose solutions to gradually remove privileged information for each with the existing computer vision approaches. Through rigorous evaluation on the CARLA simulation, we shed light on the significance of the state representation in RL for autonomous driving and outline unresolved challenges for future research.

ÖZETÇE

Sensöre Dayalı Pekiştirmeli Öğrenme ile Otonom Sürüş İçin Eksik Kalan Parçaların İncelenmesi

Ege Onat Özsüer

Bilgisayar Mühendisliği, Yüksek Lisans

5 Eylül 2023

Pekiştirmeli Öğrenme (PÖ), herhangi bir uzman denetimine ihtiyaç duymadan otonom sürüş konusunda insan yeteneklerini aşma potansiyeline sahiptir. Buna rağmen, sensöre dayalı otonom sürüşteki en son teknolojiler büyük çoğunlukla taklit ile öğrenme yöntemine dayanmaktadır. Bunun sebebi PÖ algoritmalarının doğasında olan zorluklardır. Bununla birlikte, PÖ modelleri, ortama ilişkin gerçek referans değerlerden oluşan ayrıcalıklı durum temsilleri ile eğitildiğinde son derece başarılı politikalar keşfedebilmektedir. Bu çalışmada, ayrıcalıklı durum temsili kullanan ve sensöre dayalı PÖ modelleri arasındaki performans farklarının sebepleri araştırılmaktadır. Sensör verilerinden ayrıcalıklı durum temsillerini tahmin etmek için görüşe dayalı derin öğrenme modellerini kullanan çözümler önerilmektedir. Özellikle, istenen rota oluşturma ve trafik ışığı tahmini gibi PÖ modelinin başarısı için hayati önem taşıyan durum temsilleri belirlenmektedir ve mevcut bilgisayarlı görme yaklaşımlarıyla her biri için ayrıcalıklı bilgilerin kademeli olarak kaldırılmasına yönelik çözümler önerilmektedir. CARLA simülasyonu üzerinde yapılan değerlendirme sayesinde, otonom sürüş için PÖ'de durum temsilinin önemine ışık tutulmuştur ve gelecekteki araştırmalar için çözülmemiş zorlukların ana hatları çizilmiştir.

ACKNOWLEDGMENTS

I would like to thank my parents for their endless support, my friends in the autonomous vision group for making our lab more fun, my advisors for encouraging me to believe in myself, and my aunt for buying me an espresso machine.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RL	Reinforcement Learning
IL	Imitation Learning
BEV	Bird's Eye View
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
PID	Proportional–Integral–Derivative
WOR	World on Rails
LBC	Learning by Cheating

Chapter 1

INTRODUCTION

1.1 Motivation

Traffic accidents result in the loss of a staggering 1.3 million lives every year, being the leading cause of death for people between ages 5-29, and costing most countries 3 percent of their gross domestic product[Organization, 2018]. From an analysis of 2,189,000 accidents that were filed to the United States Department of Transportation in a survey, 94 percent of accidents were determined to be caused by driver errors[Singh, 2015]. As human limitations play a pivotal role in these tragedies, the concept of autonomous driving presents itself as an attractive avenue for mitigating such errors and, consequently, substantially reducing traffic accidents.

The study of autonomous driving not only promises to significantly reduce these accidents, but it also could be used as a way to reduce traffic congestion and reduce carbon emissions[Fu et al., 2023]. A report by Lin et al.[Lin et al., 2021] shows that as little as 20 percent of the vehicles on the road being replaced with autonomous vehicles could make a noticeable positive difference in traffic flow.

1.2 Current State of Autonomous Driving

Supported by the rapid advances in deep learning, computer vision and robotics, self-driving vehicles are now closer than ever. In fact, vehicles with limited self-driving capacity are already available for purchase. However, full self-driving is a different problem with unique challenges. Despite self-driving vehicles being presented as being just around the corner for about a decade now[Korosec, 2015, Hart, 2023], it is still not possible to remove a human driver from a vehicle and guarantee safe

driving.

There are various technical reasons that make full self-driving difficult. An autonomous driving agent must be able to make critical decisions in milliseconds to ensure safe driving. The agent must also be capable of handling all the unexpected situations it can encounter in the real world, some of which are going to be absent from the agent's training data. Additionally, when a novel approach is proposed, safety concerns prevent easy testing and deployment of the agents in the real world, slowing down progress.

Despite all these challenges, the field of autonomous driving research is highly active. A critical contributing factor that allows research to continue without being hindered by the aforementioned economic and safety concerns is realistic driving simulators. The usage of simulation environments allows researchers to develop new solutions without the costs or safety risks of testing self-driving agents in the real world. In addition to lowering the barrier to entry of autonomous driving research, these simulators increase comparability between different methods by evaluating agents under the same driving scenarios.

The traditional approach to autonomous driving includes a modular pipeline, where there are separate modules for tasks such as localization, object detection, planning, control, etc. Each of these modules is generally optimized for a separate metric that is defined for its task. While this makes the design and testing of individual parts of the autonomous driving pipeline easier to manage, it also means the system as a whole is not optimized for the driving performance metrics we actually care about. An example of this could be a semantic segmentation network optimized to generate segmentation masks with a high intersection-over-union for pedestrians. In reality, we do not care about every pixel belonging to a pedestrian equally, a human driver would pay more attention to pedestrians that are closer, and pedestrians that might move towards the road. Learning such relations would require the segmentation network to be trained together with the control network, allowing the perception network to focus on parts of the sensor inputs that are important for generating suitable actions.

This realization, combined with the recent advances in machine learning meth-

ods, encouraged researchers to turn towards end-to-end solutions. End-to-end solutions learn to map raw sensor inputs directly to driving actions. This reduces the hand-tuning and engineering efforts associated with designing a modular pipeline immensely, however, this optimization process is also significantly harder. The model is expected to handle the tasks of perception, planning, and control implicitly in the layers of its neural network. However, this also means the entire driving model can be optimized for metrics we actually care about, such as safe and comfortable driving.

There are two major approaches among the state-of-the-art autonomous driving literature attempting to replace modular pipelines with deep learning methods that can be optimized jointly. The first approach is imitation learning methods, specifically approaches involving behavior cloning, a way of training neural networks to approximate the actions of an expert agent. Imitation learning methods are appealing because of the abundance of training data. By equipping everyday vehicles with sensors, it is possible to generate vast training datasets through humans acting as driving experts and collecting labeled data. However, imitation learning methods have a major limitation because of distribution shift. Distribution shift is a mismatch between the training dataset and the testing environment, which occurs naturally as imitation learning agents make minor mistakes that carry them into novel states not present in the training dataset. As the agent moves away from states the expert driver would visit, it starts making worse choices, which takes it into states that are even further from expert demonstrations, creating a loop of exponentially deteriorating driving decisions. While there are ways of reducing the severity of this problem, it remains a significant limitation.

The second approach is reinforcement learning (RL), where instead of expert imitation, an agent is trained for the discovery of an optimal policy that maximizes a cumulative sum of rewards of our choosing. There are different algorithms to make this possible, including offline RL methods which require an expert dataset similar to imitation learning, and online RL methods that require interaction with the driving environment. Online RL methods in particular have the advantage of completely removing the distribution shift problem, as the agent optimizes itself by making

continuous rollouts in the environment and generating its own data to learn from. However, RL methods have their own disadvantages compared to supervised learning approaches, having lower sample efficiency, worse convergence properties, and more complex training schemes that are brittle towards changes in hyperparameters.

The training and evaluation of most of these state-of-the-art methods are made possible by the CARLA open-source autonomous driving simulator[Dosovitskiy et al., 2017]. CARLA is an urban driving simulation designed to be a realistic training and testing ground for autonomous driving algorithms. The simulator allows the usage of sensors such as RGB cameras, LIDAR's, GNSS and IMU sensors, and more. CARLA also defines challenging scenarios that have a high probability of causing accidents, such as occluded pedestrians suddenly moving toward the vehicle's path, allowing autonomous agents to be evaluated in rare and critical situations without any safety risk.

In the last years, the CARLA simulator leaderboard has seen an influx of imitation learning methods that improve performance through the usage of various new architectures[Chitta et al., 2021], auxiliary supervision methods[Chitta et al., 2022] and training schemes [Chen and Krähenbühl, 2022]. On the other hand, RL methods have been on the decline, with no RL-based method present in the top 10 methods listed on the leaderboard at the time of writing. While this can be partially because training RL agents is a relatively high-effort task in comparison to imitation learning methods, we believe there to be some missing technical discovery holding back progress in the field.

1.3 Contributions

In this work, we investigate the specific problems holding back reinforcement learning agents from achieving superhuman driving performance. To that end, we turn our attention towards the only reinforcement learning agent that surpasses expert performance to our knowledge, the ROACH method by Zhang et al.[Zhang et al., 2021]. Zhang et al. train an RL agent in their work to act as an oracle in assisting the learning processes of other agents. This expert agent surpasses even the au-

topilot of the CARLA method in some of their benchmarks. While their method accesses internal simulator data to function and is therefore not comparable to realistic sensor-based methods, their RL training method consistently trains stable and successful RL agents. We conduct a comprehensive analysis of their methodology, with a particular focus on discerning the specific attributes that confer a competitive advantage over alternative approaches.

Our contributions are three-fold: We investigate how the reinforcement learning problem formulation of ROACH is different from standard sensor-based agents. We analyze which differences contribute to the success of ROACH and to what degree. Since our findings for these two research questions suggest the success of the agent is heavily dependent on the state representation of ROACH, we then focus on how we can replace the unrealistic state representations with sensor-based alternatives. Our final and most important contribution is investigating how computer vision models can be utilized to replace the state representation of ROACH, highlighting the missing pieces required to reach state-of-the-art.

We believe that our work provides an important starting point for future reinforcement learning research on the CARLA[Dosovitskiy et al., 2017] autonomous driving benchmark. We also highlight how computer vision methods can be utilized for autonomous driving, and where they fall short. Our hope is that by showing where existing models fall short, we aid computer vision researchers to align themselves with the needs of autonomous driving research.

Chapter 2

RELATED WORK

2.1 Imitation Learning

Imitation learning has been used for autonomous driving since before the prevalence of deep learning methods and has shown success in tasks like lane following [Pomerleau, 1988] and off-road obstacle avoidance [Muller et al., 2005]. However, these methods did not generalize to the challenging urban driving problem. Modern imitation learning methods on the CARLA[Dosovitskiy et al., 2017] simulator were initiated by the Conditional Imitation Learning (CIL)[Codevilla et al., 2018] architecture. In order to extend imitation learning to the urban navigation setting, Codevilla et al. propose a novel branched architecture that conditions actions on high-level route commands by using a different head to generate separate actions for each possible command. In follow-up work, they explore the limitations of their modified behavior cloning approach [Codevilla et al., 2019] and improve performance significantly by using a larger model and an auxiliary loss to predict the vehicle's speed.

A significant issue of the imitation learning family of methods is distribution shift, which causes the policy to suffer from compounding errors as it diverges from the states present in the expert demonstrations. In order to circumvent this distribution shift problem, Chen et al. [Chen et al., 2020] use ground truth bird's eye view semantic segmentation maps as input to train an expert agent, which can provide supervision to a sensorimotor agent as it is deployed in the simulation. The availability of expert supervision for on-policy data allows the usage of data aggregation techniques like DAGGER[Kumamoto et al., 1980] to mitigate distribution shift. An extension of this work called Learning from All Vehicles (LAV)[Chen and Krähenbühl, 2022] achieves much higher performance by using every agent in the

scene as a source of supervision.

An important aspect of autonomous driving is input representation, as it usually involves multiple sensor inputs such as RGB images, LIDAR point clouds, GNSS coordinates, and IMU readings. Which sensors to use while training an agent is an engineering decision, for example, while LIDAR can significantly improve safety while driving around vehicles and pedestrians[Chitta et al., 2022], it also increases computational complexity and results in a sensor setup that would be much more costly to build in the real world.

Another possibility regarding model inputs is the usage of intermediate representations. Intermediate representations are inputs for an autonomous driving system, usually generated through processing the sensor inputs with a deep learning model or by accessing normally hidden simulator variables. The expert model of LBC[Chen et al., 2020] is an example where a bird's eye view (BEV) semantic segmentation map is used as an intermediate representation. Behl et al. [Behl et al., 2020] investigate intermediate representations for autonomous driving, focusing on semantic segmentation and analyzing the task-relevant object classes, showing that a good intermediate representation can play a crucial role in driving performance.

The actions generated by autonomous driving agents can also be represented in multiple ways. The simplest representation is to predict the raw control values for steering and acceleration as is done in early work[Codevilla et al., 2018, Chen et al., 2020, Ohn-Bar et al., 2020]. Later state-of-art imitation learning papers [Chitta et al., 2022, Chitta et al., 2021] have shown that predicting multiple future waypoints instead of raw actions, and using a PID controller to convert these waypoints to raw actions can boost performance. Most recently, Wu et al. [Wu et al., 2022] show that both waypoints and raw action representations can be helpful in different situations, and dynamically weighting the results of the two action predictions can result in greater performance.

2.2 Bird's Eye View Perception

BEV perception models play a crucial role in enhancing the situational awareness and decision-making capabilities of autonomous driving systems. By providing a top-down view of the surrounding environment, BEV models offer a comprehensive understanding of the spatial relationships between the vehicle and its surroundings. These models often fuse information from various sensors such as LiDAR, cameras, and radar to generate a holistic representation of the scene. The BEV perspective provides a combined representation of surrounding vehicles, pedestrians, objects, lane markings, and road topology, which are essential for tasks like lane-keeping, object avoidance, and path planning. Furthermore, BEV perception models are instrumental in addressing challenges posed by occlusions, complex intersections, and urban environments where traditional sensor perspectives might fall short. As autonomous vehicles navigate diverse and dynamic environments, BEV perception models contribute to the robustness and safety of their operation, facilitating accurate decision-making and enhancing overall road safety.

BEV semantic segmentation models are closely related to our research in particular. The conventional approach of BEV semantic segmentation models is to encode the RGB image inputs with convolutional layers, project them to BEV coordinate space, and decode them with a semantic segmentation head. The BEV projection technique used by models plays a large role in their architecture and performance, which is a unique problem that makes this task more difficult in comparison to the conventional semantic segmentation task.

Among the state-of-the-art models, Lift-Splat-Shoot (LSS)[Philion and Fidler, 2020] uses a depth prediction module and projects encoded features from the camera space to the BEV space based on this depth estimation. The success of this method naturally depends on successfully predicting the depth values of the RGB image, however, if the depth model is perfect, the features are only projected to the surface of an object, which might make it difficult to segment them entirely. A later work called BEVFormer [Li et al., 2022] instead utilizes deformable attention to learn a mapping function that maps features in the image space to a feature grid

in the BEV space. BEVFormer achieves great performance in both the semantic segmentation task and the object detection task. However, their heavy transformer-based architecture combined with the learned projection method results in a very large model that is difficult to train or use. Finally, SimpleBEV[Harley et al., 2023] proposes a strikingly straightforward way of using simple parameter-free bilinear interpolation between RGB space and BEV space. This method performs surprisingly well while reducing hardware requirements and reducing training time.

2.3 Reinforcement Learning

Reinforcement learning agents for autonomous driving on the CARLA simulator generally fall short of imitation learning methods in performance. However, the promise of training agents not affected by distribution shift, or generating agents that can surpass human driving still act as powerful motivators for RL researchers.

The first deep RL method for autonomous driving on CARLA, proposed as a baseline in the CARLA challenge [Dosovitskiy et al., 2017] itself, uses the A3C algorithm with discrete actions [Mnih et al., 2015] but fails to be comparable to the performance of imitation learning. One of the first approaches that achieved state-of-the-art results by Liang et al. [Liang et al., 2018] uses DDPG with continuous actions [Lillicrap et al., 2016] by initializing the policy network from an imitation learning agent. Toromanoff et al. [Toromanoff et al., 2020] first train a network to predict affordances related to the environment along with semantic segmentation maps. They then freeze this network and use its bottleneck features to train an RL agent using the Rainbow-IQN algorithm [Hessel et al., 2018]. Chen et al. [Chen et al., 2021] follow a model-based approach by factorizing the driving state and the world model. The world is assumed to be independent of the agent's actions which allows training in pre-recorded driving logs. Despite this limiting assumption, which almost never holds, it outperforms model-free methods.

Chapter 3

PRELIMINARIES

In this chapter, we provide a primer by clarifying fundamental concepts, terminology, and methodological foundations that form the basis of our research. This knowledge equips readers with the understanding needed to grasp the following chapters of this thesis. The first section will go over the basics of reinforcement learning. The second section will explain the methodology of ROACH[Zhang et al., 2021]. The third and final section will go over the limitations of the privileged ROACH approach, and the aspects preventing it from being used as an unprivileged driving agent.

3.1 Reinforcement Learning

Reinforcement Learning is a branch of machine learning focused on training agents to make sequential decisions in dynamic environments, with the aim of maximizing a cumulative reward. At its core, reinforcement learning involves an agent interacting with an environment, taking actions, and receiving feedback through rewards. The agent's primary objective is to learn an optimal policy to optimize the expected cumulative reward over time. Figure 3.1 depicts a basic RL loop and the interaction between the agent and the environment.

3.1.1 Markov Decision Processes

Reinforcement learning problems are often formulated as Markov Decision Processes (MDPs). An MDP is formally defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where the elements of the tuple are defined as:

- $\mathcal{S}$: The space of all possible states
- $\mathcal{A}$: The space of all possible action

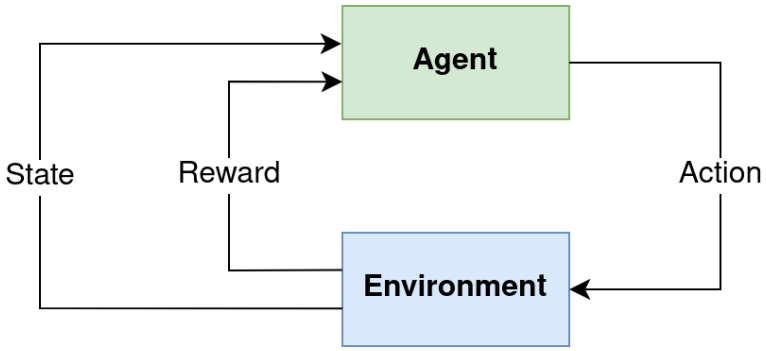

Figure 3.1: An overview of a standard single-agent reinforcement learning (RL) process. RL revolves around the interaction between an agent and an environment. The actor makes decisions by taking observations as input and generating actions. The environment responds by providing updated observations and reward signals.

- P: The transition probability function that defines the probability of transitioning from one state to another after taking a particular action.
- R: The reward function

3.1.2 State

A state $s \in \mathcal{S}$, where $\mathcal{S}$ is the state space, represents the current situation or observation of the environment. It serves as the context for the agent's decision-making process, containing essential information that influences its actions and strategies. The choice of how a state is represented plays a pivotal role in guiding the agent's learning and performance within the environment. The ideal state should be as informative as possible without incorporating data that is unrelated to the task, both to make the learning task easier and to increase efficiency. As the state space becomes high dimensional, exploring all possible state configurations quickly becomes harder, which can become a problem when the model faces new situations. Moreover, the state should ideally contain low amounts of noise, considering that most RL methods already suffer from sample efficiency and convergence issues that could be intensified by noisy inputs.

3.1.3 Action

An action $a \in \mathcal{A}$, where $\mathcal{A}$ is the action space, refers to the choices or decisions that an agent can make within a specific state. Actions are the agent's way of interacting with its environment, influencing future states and the rewards it receives. Actions can take different forms, including discrete choices or continuous values.

Discrete actions offer simplicity and computational efficiency, suitable for tasks with well-defined choices. However, they may lack the granularity needed for tasks demanding fine control. In contrast, continuous actions provide nuanced decision-making, vital for precise control tasks. Yet, managing continuous actions requires handling complexities in optimization and exploration strategies.

3.1.4 Policy

A policy π defines the agent's strategy for acting in the world based on the states it finds itself in. For a wide range of problems, policies are parametric. A policy can be a deterministic mapping from states to actions in the form of $a = \pi_\theta(s)$ or a distribution over actions given states such as $\pi_\theta(a|s)$. The goal of a policy-based RL agent is to select actions that will achieve its goals, which translates to optimizing the policy parameters θ to maximize the cumulative sum of rewards.

3.1.5 Reward Function

A reward function $\mathcal{R}(s, a)$ in reinforcement learning is a mathematical construct that quantifies the immediate desirability of an agent's actions within a specific state. It assigns numerical values to the outcomes of the agent's actions, providing a measure of the immediate benefit or cost associated with each action. The role of the reward function is critical, as it guides the agent's learning process by providing feedback on the quality of its decisions. An effective reward function should ideally offer dense and informative feedback to guide the agent's learning process effectively. Poorly designed reward functions can lead to an agent being unable to find good policies, or can inadvertently lead to undesired agent behavior.

The goal of an RL agent is to find an optimal policy, denoted as π^*, that max-

imizes the expected cumulative reward. This can be expressed with the following equation:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right] \tag{3.1}$$

where γ is the discount factor that quantifies the preference for immediate rewards over future rewards.

3.2 ROACH

The ROACH autonomous driving agent by Zhang et al.. [Zhang et al., 2021] is a privileged reinforcement learning agent that is designed to be used as an expert agent. A privileged method in the context of autonomous driving in the CARLA simulator [Dosovitskiy et al., 2017] is any method that relies on information not available through the sensor interface that is designed to emulate the sensors available to real-world self-driving vehicles. While an agent that relies on privileged data cannot be fairly compared to sensorimotor agents, they can still be utilized as experts. Expert agents are used to generate behavior cloning datasets, or even to provide direct feature supervision to assist the learning process of other agents [Wu et al., 2022, Zhang et al., 2021, Zhang et al., 2023]. This makes such privileged experts valuable for autonomous driving research despite their reliance on internal simulator representations, which would not be available to a self-driving vehicle in the wild.

The expert ROACH agent is the baseline for our experiments, and it acts as an upper bound for unprivileged sensorimotor RL agents. This section will provide a thorough overview of the problem formulation, and deep learning model architecture.

3.2.1 Problem Formulation

State Space

The state representation s of the ROACH expert is made up of two components, $s_{\mathbf{bev}}$ and $s_{\mathbf{mes}}$. The BEV representation $s_{\mathbf{bev}}$ replaces the role traditionally handled by RGB images in sensorimotor agents, consisting of BEV semantic segmentation

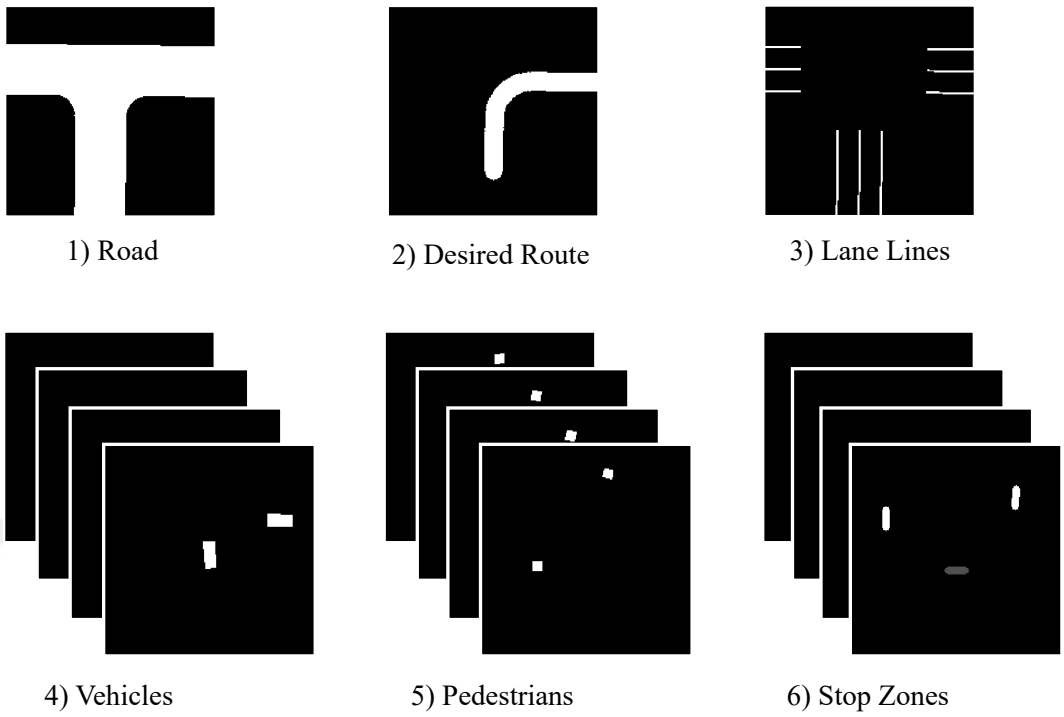

Figure 3.2: Visualization of $s_{\mathbf{bev}}$

maps $s_{\mathbf{bev}} \in [0,1]^{CxHxW}$. These BEV masks cover a square area with sides of 38.4 meters, where the ego-vehicle is aligned to be horizontally centered, and 8 meters up vertically from the bottom of the map area. The BEV masks are rendered in 192x192 resolution and contain 15 channels. There are 4 temporally stacked channels each for vehicles, pedestrians, and "stop zones". Stop zones are a construct utilized in the CARLA simulator to represent areas where a vehicle should not move, and they are all linked to a traffic light or stop sign that controls the stop zone's state. When a corresponding traffic light for a stop zone turns red, the stop zone's area is rendered in BEV frame's corresponding channel. There is additionally one channel each for roads, lane lines, and the "desired route". The desired route is the fine-grained path the agent should take to reach the next waypoint, calculated by accessing the road topology from the simulation internals and visualized in the BEV coordinate space. Visualizations of these BEV masks are given in 3.2.1.

In addition to the BEV masks, the state includes a vector of scalar measurements $s_{\mathbf{mes}} \in \mathbb{R}^t$, which is composed of the steering, throttle, brake, vehicle gear, lateral

and horizontal speed values observed in the last time step. These measurements are relatively easy to obtain and are not considered privileged.

Action Space

In the CARLA simulator, the raw actions required to control a vehicle are steering, throttle and brake. The steering action is in the range of $[-1, 1]$, with -1 being fully steering to the left and $+1$ being fully steering to the right. The throttle and brake actions are both in the range $[0, 1]$, with 0 being no action and 1 being full throttle or brake, respectively. However, the convention in the state-of-the-art is to combine the throttle and brake actions into a single acceleration action in the range $[-1, 1]$, with -1 being mapped to full braking and $+1$ being full throttle. This simplifies the action space to be in the range $\mathbf{a} \in [-1, 1]^2$ while preventing the agent from predicting actions that would apply throttle and brakes at the same time. ROACH follows this action representation without any discretization or modification.

Reward Function

The reward function used by the RL Expert is a reward function designed by Toromanoff et al.. [Toromanoff et al., 2020]. The reward function they propose consists of 3 main components:

Desired Speed Component: The speed component $r_{\mathbf{spd}}$ incentivizes the agent to maintain a speed appropriate for the current context. The reward is maximal (equal to 1) when the agent travels at the desired speed, linearly decreasing to 0 if the agent's speed deviates from this target. The desired speed adapts dynamically to the situation based on hand-tuned rules first proposed by Toromanoff et al.. [Toromanoff et al., 2020]. The desired speed decreases as the agent approaches obstacles, pedestrians, bicycles, vehicles, or red traffic lights. It returns to the maximum allowed speed of 6 m/s when conditions permit.

Desired Position Component: This component $r_{\mathbf{pos}}$ encourages the agent to remain within the lane corresponding to the desired route. It is inversely proportional to the distance between the agent's position and the middle of the desired

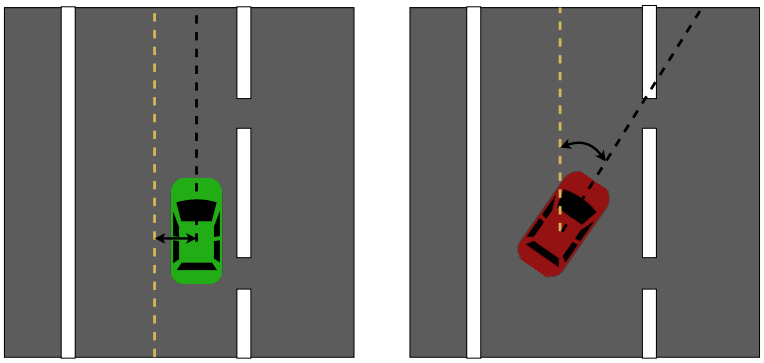

Figure 3.3: The green vehicle depicts the lateral distance from the road center used to calculate $r_{\mathbf{pos}}$. The red vehicle depicts the rotational difference between the vehicle and the correct heading used to calculate $r_{\mathbf{rot}}$. Figure adapted from Toromanoff et al. [Toromanoff et al., 2020]

lane, computed using the privileged waypoint information from the simulator. The reward reaches its maximum (0) when the agent is precisely in the middle of the lane and decreases linearly to -1 as the agent deviates further from the lane center. The episode terminates if the agent exceeds a predefined maximum distance of 3.5 meters from the lane center. Additionally, the episode can terminate for other reasons, such as collisions, running a red light, or becoming stuck, with the agent receiving a reward of -1 in such cases.

Desired Rotation Component: To mitigate oscillatory behaviors observed when relying solely on the previous two components, a third reward component, desired rotation $r_{\mathbf{rot}}$, is introduced. This component is inversely proportional to the angular difference between the agent's orientation and the orientation of the nearest waypoint on the optimal trajectory. This reward discourages the agent from deviating from the optimal path.

A visualization of the desired position and rotation components is given in Figure 3.2.1. The final reward can be summarized with the following formula as a weighted sum of the three components:

$$r = \alpha_{spd} r_{spd} + \alpha_{pos} r_{pos} + \alpha_{rot} r_{rot} \tag{3.2}$$

The advantage of this reward function is that it is very dense, giving direct supervision to the RL agent in every time step, which is a common requirement

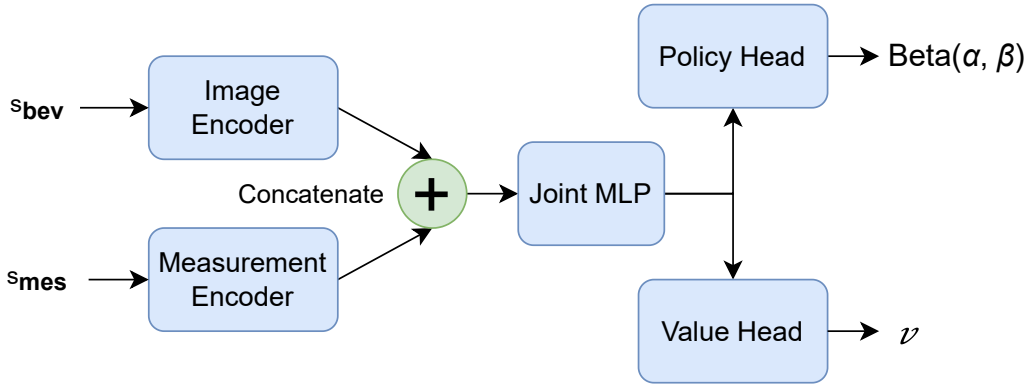

Figure 3.4: Visualization of ROACH[Zhang et al., 2021] RL Expert's model architecture.

for RL algorithms to perform well. The disadvantage is the amount of engineering tuning required to prevent unwanted behaviors from emerging. The reward function consists of three main components: desired speed, desired position, and desired rotation, each contributing to shaping the agent's behavior.

3.2.2 Model Architecture

The ROACH expert's network architecture, illustrated in 3.2.2, is designed to process the BEV representation of the environment and integrate crucial measurement vectors for autonomous driving to first generate a joint feature vector, and then process those features with two separate heads, tasked with generating action distributions and predicting the value of the current state. We describe the parts of the model below.

BEV Encoder: The BEV encoder consists of six convolutional layers responsible for encoding the BEV representation $s_{\mathbf{bev}}$. These convolutional layers are instrumental in extracting high-level features from the BEV, enabling the model to comprehend the spatial layout of the environment effectively.

Measurement Vector Encoder: To incorporate measurements $s_{\mathbf{mes}}$ that are difficult to represent in the BEV space, two fully connected layers dedicated to encoding the measurement vector are employed. This component ensures that critical real-world data, such as sensor measurements, are seamlessly integrated into the

network.

Joint Encoder: The outputs from the BEV encoder and the measurement vector encoder are concatenated to form a combined feature representation. This combined feature representation is subsequently processed by an additional two FC layers. This processing results in the generation of a combined latent feature denoted. This latent feature serves as the input for two distinct heads within the network:

Value Head: The value head comprises two FC hidden layers and is responsible for estimating the expected cumulative rewards, or the value, associated with a particular state. Accurate value estimation is critical for assessing the desirability of different states and actions during policy optimization.

Policy Head: The policy head, akin to the value head, consists of two FC hidden layers. Its primary role is to generate the probability distribution over available actions. This distribution guides the agent's actions as it navigates through the environment.

3.3 Comparison Between Privileged and Sensorimotor Agents

It is expected that privileged agents can achieve better driving performance in comparison to sensorimotor agents. However, we observe that the usage of privileged information creates an immense difference in the case of RL, while imitation learning agents cannot utilize the same privileged data as effectively. This claim is examined in Table 3.1, where two BEV semantic segmentation map-based privileged methods[Zhang et al., 2021, Chen et al., 2020] are compared against two sensorimotor methods that rely on images from regular RGB cameras. Despite being the best-performing sensorimotor RL method at the time of writing, the World On Rails method's driving score falls short of all other methods in the table. On the other hand, the similarly vision-based imitation learning method NEAT achieved equivalent performance to a privileged behavior cloning method. This motivates the idea that RL is still a promising approach to generating successful autonomous driving agents, and the problem of the state-of-the-art sensorimotor RL agents lies in their

Table 3.1: Comparison of the ROACH expert with state-of-the-art example imitation learning and RL methods on the Longest-6 benchmark of the CARLA Simulator. **RC** is the route completion percentage, **IS** is the infraction score that quantifies the penalty an agent accumulates from breaking traffic rules, and **DS** is the main metric of the CARLA benchmark, calculated by multiplying **RC** and **IS**.

Method	RL	Privileged	DS	RC	IS
ROACH Expert [Zhang et al., 2021]	+	+	60.14±2.40	85.83±0.60	0.69±0.03
World On Rails [Chen et al., 2021]	+	-	17.36±2.95	43.46±2.99	0.54±0.06
Learning by Cheating Expert [Chen et al., 2020]	-	+	24.08±2.83	73.36±1.08	0.31±0.06
NEAT [Chitta et al., 2021]	-	-	24.08±3.30	59.94±0.50	0.49±0.02

noisy, high-dimensional state representations.

This realization that a good input representation is the critical missing piece for successful RL agents in autonomous driving is our motivation for examining the privileged components of the state space of ROACH and how they could be replaced with sensor-based approaches to pave the way for RL agents that are on par with imitation learning agents. As the privileged information is entirely in the BEV representation, the next chapter focuses on the specific challenges of predicting each channel from sensor data and our proposed approaches to bridge the gap between privileged and vision-based RL for autonomous driving.

Chapter 4

TOWARDS UNPRIVILEGED REINFORCEMENT LEARNING

The privileged BEV semantic segmentation mask inputs to the ROACH[Zhang et al., 2021] architecture are critical to the success of the agent. In order to replicate the privileged expert RL agent's success with an unprivileged sensorimotor agent, we propose to use separate prediction models for each BEV channel in the privileged expert's state. While some channels are possibly predictable through the recent RGB-to-BEV methods, desired routes require additional information, such as GNSS and IMU readings, and predicting the locations of stop zones without accessing the simulator's internal representation presents additional challenges. This chapter describes the challenges associated with predicting each BEV channel of ROACH's BEV representation $s_{\mathbf{bev}}$ in detail, how we propose to replace them with predicted components, and our experimental results.

For all our experiments where an RL agent is trained, we use the same environment and the same training scheme as the RL expert of ROACH. We leave the deep learning architecture of the RL agent untouched except for the initial layers to accommodate different input sizes.

4.1 BEV Perception

In this section, we investigate which channels can be successfully predicted by the state-of-the-art BEV perception models, and how their replacement affects the performance of the RL agent. We consider three possible BEV perception models for the task, Lift-Splat-Shoot[Philion and Fidler, 2020], BEVFormer[Li et al., 2022] and SimpleBEV[Harley et al., 2023]. Out of these three, BEVFormer would be the most suited for the task at hand, as it can simultaneously segment road and lane lines

while predicting bounding boxes for vehicles and pedestrians. Unfortunately, this multitask performance comes at the cost of a much heavier model, which makes it infeasible to use in RL training. Since we need these perception systems to be active through millions of steps of RL training along with the already GPU-intensive CARLA simulators and our RL agent, hardware constraints and training time pose significant limitations. Lift-Splat-Shoot can be used with a smaller backbone, however, it relies on first solving a depth-prediction task to project features from RGB space to BEV space, increasing latency and memory constraints. We find SimpleBEV to perform better with fewer parameters and smaller latency, which fits their claims regarding the advantages of parameter-free projection of RGB features to the BEV coordinate space. We proceed to use their architecture with an EfficientNet-B0[Tan and Le, 2019] backbone for all of our final experiments.

Taking these BEV perception models optimized for other benchmarks such as NuScenes[Caesar et al., 2020], generating suitable datasets for their training, and finding the right configuration to achieve comparable results on CARLA is a non-trivial task. The following sub-sections will explain the critical modifications we made to the SimpleBEV model to make it work on CARLA [Dosovitskiy et al., 2017] benchmark and our RL training setup, our data collection strategies, and our experimental results.

4.1.1 Modifications to SimpleBEV

The SimpleBEV[Harley et al., 2023] architecture is designed and optimized for the NuScenes[Caesar et al., 2020] BEV segmentation task. Following the conventions of this task, they define the spatial range for the predicted BEV region as 100 meters by 100 meters, divided into a grid of 200 by 200 segments. These BEV maps are centered and aligned with respect to a reference camera, which is usually the front-facing camera except for data augmentation purposes. They also conform to the NuScenes sensor setup, getting RGB inputs from 6 cameras around the vehicle, which cover a 360-degree area with their combined fields of vision.

On the other hand, the CARLA benchmark allows the usage of a maximum of 4

RGB cameras to keep agents comparable. Because of this, state-of-the-art methods that make predictions in the BEV frame, such as Transfuser[Chitta et al., 2022] or NEAT[Chitta et al., 2021] use 3 cameras and make predictions for the area ahead of the vehicle. Our BEV maps also mainly consist of the area ahead of the vehicle so we follow their examples and use a similar setup using 3 cameras. The 3 cameras are positioned on the front of the vehicle, one of them facing the front and two of them facing 60 degrees to the left and right, respectively. Each camera has a field of view of 100 degrees and generates images of resolution 320x160. Our predicted BEV maps cover the same area as the ROACH[Zhang et al., 2021] BEV state $s_{\mathbf{bev}}$.

The RGB image resolution of 320x160 is lower than the best-performing resolutions reported by the original SimpleBEV paper. However, we also need to predict a smaller region; the farthest point on our BEV prediction grid is 35.95 meters away from our ego vehicle, while the original paper needs to segment areas up to 70.71 meters away. Since our initial experiments also confirmed our intuition that a higher resolution does not bring a significant gain in performance, we keep the resolution low to reduce memory consumption during RL training.

The dominant paradigm in state-of-the-art BEV segmentation models is to train separate models for each class that needs segmenting, and SimpleBEV follows that approach. However, while training and using separate models for each class improves performance, applying the same technique during reinforcement learning is infeasible. To keep memory and time constraints manageable, we modify the SimpleBEV architecture to simultaneously predict multiple binary segmentation masks. Only the final 1D convolution layer of the segmentation head is changed, such that the output consists of multiple channels of BEV masks rather than a single map.

4.1.2 Collecting Training Data

We collect 20 hours of driving data at 10 hertz using the CARLA[Dosovitskiy et al., 2017] autopilot on the ROACH[Zhang et al., 2021] RL environment. We collect data from towns 1 through 6 of the CARLA simulator, reserving 10 percent of the data for validation purposes. We apply triangular perturbations to the expert

agent's actions in order to reduce the effects of distribution shift, following the conventional approach to augment the expert trajectories first proposed by Codevilla et al.. [Codevilla et al., 2018]. The weather, time of day, and lighting conditions are randomly selected and continue to change dynamically during episodes.

4.1.3 Training

We trained all models for 50,000 steps using the 1cycle learning rate scheduler proposed by Smith et al..[Smith and Topin, 2019], taking the best-performing checkpoint. Unlike regular multi-class segmentation methods that use cross-entropy loss, we use binary cross-entropy loss for training. This is necessary since in our BEV segmentation task, a pixel of the BEV grid doesn't necessarily belong to a single class, instead, multiple channels can be active at once as in the case of a vehicle positioned on a road. We also apply positive weighting when predicting classes that are dominated by negative samples that usually cover a very small portion of the grid, such as pedestrians or lane lines. We found this to be critical, as otherwise the model tended to give much more false negatives than false positives, predicting empty masks whenever faced with a difficult sample. In autonomous driving, such false negatives are usually more dangerous than false positives, since they can cause crashes against vehicles and pedestrians. Positive weighting counteracts this problem by multiplying the loss from positive pixels in the label with some value to make them more significant than the dominating majority of negative pixels. We apply a positive weighting factor of 15 for pedestrians, 8 for lane lines, and 5 for vehicles.

4.1.4 Experimental Results

We experiment with predicting all but two classes with the SimpleBEV[Harley et al., 2023] model. We do not attempt to predict desired route channels in this manner since predicting it requires knowledge about the relative positions of future waypoint coordinates and is impossible to predict by only using RGB inputs. In the initial experiments, we quickly discovered that predicting stop zones with this approach is also not feasible. We reason that this is caused by the stop zones being spatially

Table 4.1: Class-wise IOU values on the validation set for the SimpleBEV model trained for simultaneous segmentation of 4 classes.

Pedestrian IOU	**Vehicle IOU**	**Lane Line IOU**	**Road IOU**
0.013	0.403	0.132	0.788

distant from the traffic lights, both in image space and in BEV space. This is especially true in US-style farside traffic lights . Learning the relation between a small red light in an RGB image and an invisible rectangular area in the distance successfully enough to segment it in the BEV grid is a challenging task, even for a method like SimpleBEV that uses attention mechanisms.

Excluding the desired route and stop zone channels leaves behind the task of predicting the road, lane line, vehicle, and pedestrian classes. We train and evaluate our modified model on the CARLA data we have collected. Our per-class intersection-over-union (IOU) scores are given in Table 4.1. The results, show that while the model can learn to segment the easier classes such as vehicles and roads with some degree of success, it can not segment the more difficult classes such as pedestrians and lane lines at the same time. While a lower IOU can be tolerated for the case of lane-lines where a slight misalignment of thin lines could result in a big drop in the evaluation metric, the amount of pedestrians missed by the perception model makes it impossible to drive safely. Our attempts at tuning the per-class losses failed to fix this problem, showing that these BEV perception models are not suited to learning the prediction of multiple difficult classes in this manner.

We believe that advances in computer vision techniques for BEV perception and 3D detection tasks can make it possible for vehicles and pedestrians to be more accurately perceived in the future, without imposing the heavy hardware requirements of current models like BEVFormer[Li et al., 2022]. We then shift our focus towards the segmentation of roads and lane-lines in particular, as all BEV perception models optimize for those two classes. The per-class IOU for the model trained for the segmentation of roads and lane lines is given in Table 4.2. This approach with only two predicted classes achieves significantly higher IOU metrics for both classes.

Table 4.2: Class-wise IOU values on the validation set for the SimpleBEV model trained for simultaneous segmentation of road and lane-line classes only.

Lane Line IOU	**Road IOU**
0.756	0.924

We then experiment with replacing the road and lane-line classes from already trained expert agents and evaluating the impact on performance. As RL agents can have considerable variations in performance depending on the random seed, we train 3 privileged agents from scratch and evaluate them. Then for all three agents, we take the best-performing checkpoints and evaluate them after replacing the ground truth road and lane-line masks with predicted roads and lane lines. We report the observed mean and standard deviation values in Table 4.3. Without requiring any fine-tuning, the results are surprisingly close to the privileged agent.

Inspecting the driving behavior and BEV prediction visualizations of the network provided us with two critical insights.

First, while BEV perception models can reach very high IOU scores in static datasets, like our generated dataset or the NuScenes[Caesar et al., 2020] dataset, that performance does not easily convert to successful predictions during RL training. As the RL agent needs to explore unusual state action pairs before achieving successful driving, the agent ends up visiting states that cause our BEV prediction model to fail much more severely than it did on the validation set. This problem persisted even when we applied the data augmentation methods commonly found in BEV segmentation literature. We believe it is important that future BEV Perception research for autonomous driving focuses on successfully predicting the vehicle's states in rarer scenarios that differ from the dataset's dominant paradigm of driving on a straight road.

The second critical insight we gain by observing the driving and BEV predictions of this agent is that even in situations where our road and lane-line predictions were unsuccessful, the agent is able to drive. We reason that this is caused by the ROACH[Zhang et al., 2021] architecture severely relying on the desired route

Table 4.3: Evaluation of trained privileged experts and their modified versions using SimpleBEV predictions to replace roads and lane lines. The evaluation is done on the randomly generated evaluation routes of the ROACH RL environment. The metrics DS, RC, and IS are the same as the previously explained CARLA benchmark metrics, with the only difference being the possibility of achieving route completion greater than 1.0. This is possible because the ROACH environment keeps extending the route randomly after an agent reaches the final target waypoint, creating endless trajectories to follow.

Model	**DS**	**RC**	**IS**
Privileged expert	0.778 ± 0.192	0.927 ± 0.159	0.785 ± 0.089
Predicted roads & lanes	0.729 ± 0.196	0.844 ± 0.228	0.813 ± 0.011

channel while ignoring road and lane line channels. This claim and its repercussions will be investigated in the next section.

4.2 Desired Route Generation

Before attempting to predict desired routes and replace their ground truth counterparts in our state representation, we do an experiment to confirm that the ROACH expert heavily relies on the desired route component while ignoring road and lane line information. In order to examine this claim, we train an RL expert that does not receive the two channels for roads and lane lines in the input. This agent would need to navigate the roads entirely depending on the desired route channel, not having access to any information regarding road topology, other lanes or intersections. We also attempt to replace the desired route channel with a less informative but non-privileged alternative. We accomplish this by projecting the GNSS coordinate of the next target waypoint of the current route trajectory our agent must follow and rendering it as a heatmap of the target location instead of the privileged route to that location. As this heatmap representation is considerably different than the desired route, we train three models from scratch to evaluate the effects of the switch. A figure of a target heatmap is shown in Figure 4.2.

The experimental results comparing these agents' performance to the privileged agent are given in Table 4.4. The experiment results certainly confirm our suspi-

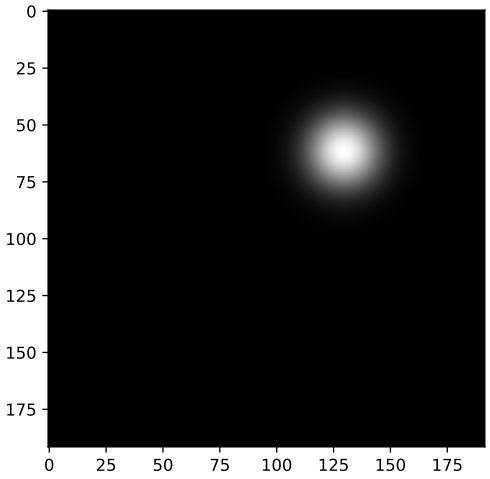

Figure 4.1: A visualization of our proposed less privileged representation of the current target waypoint the agent should drive towards. When this target is close to the ego vehicle, it is clear what the agent should do, but when it is further away, it can be ambiguous which lane should be followed.

cions regarding the importance of desired route channels, as the agent learns faster and achieves higher driving performance with less variance when it only sees the desired route. Moreover, we see that a less privileged route representation causes a catastrophic drop in agent performance.

This is not to say that road and lane line information is not needed. The desired route is in fact the shortest path towards the future GNSS coordinates of target waypoints, calculated with access to the internal road topology representation of

Table 4.4: Performance comparison of the baseline privileged agent compared with an agent that is trained to navigate solely based on the desired route, without having access to road or lane line channels

Model	**DS**	**RC**	**IS**
Privileged expert	0.778 ± 0.192	0.927 ± 0.159	0.785 ± 0.089
Privileged without roads or lanes	0.868 ± 0.119	1.109 ± 0.126	0.768 ± 0.070
Target heatmap	0.018 ± 0.012	0.027 ± 0.011	0.858 ± 0.041

CARLA. If access to the simulator internals is out of the question, as in the case of sensorimotor agents, generating this path becomes a novel problem. To our knowledge, the critical importance of this desired route representation and how it can be generated from sensor data has not been studied before. We describe our proposed approach to generating these desired route masks in the following sections.

4.2.1 Route Generation Model

In order to generate the desired route, we must design a neural network that combines information regarding the surrounding road topology, and the target waypoints we are tasked with visiting as we control the ego vehicle. We propose an architecture that encodes both of these inputs into features and fuses them using a cross-attention layer.

Data

The target waypoints in CARLA[Dosovitskiy et al., 2017] are provided to the ego-vehicle as a list of GNSS coordinates that are each a vector containing longitude, latitude, and altitude. Following the convention of other state-of-the-art methods, we first convert these GNSS coordinates into relative coordinates with respect to the ego vehicle's frame of reference. This is possible without doing any learning, as we have access to the sensor readings of the GNSS and IMU sensors on the vehicle. Once the coordinates are converted to be relative target vectors $\mathbf{t} \in \mathbb{R}^2$, we take the next five waypoints we need to visit, and use it as one of the inputs of the model.

For a genuinely sensor-based, unprivileged agent that generates desired routes, a new architecture needs to be designed that first extracts features regarding the road topology from raw RGB images and then combines them with the target waypoint information. We leave the design and training of this architecture to future work and study the more straightforward problem of generating routes while assuming access to the road and lane line masks of the surrounding area. Therefore we use the ground truth BEV masks of roads and lane lines as the second input to our model. In theory, combining such an architecture with a sufficiently accurate BEV

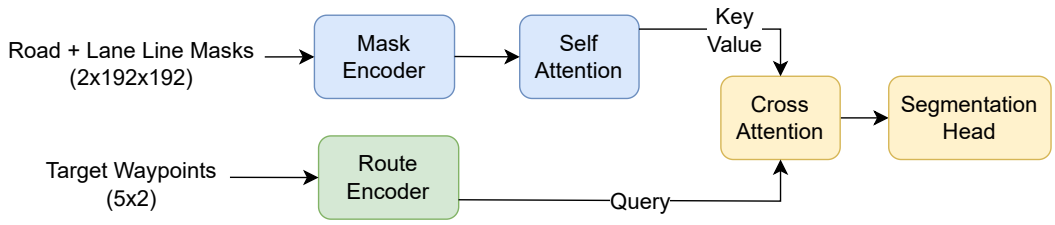

Figure 4.2: The model architecture we use to generate our route predictions from road and lane line maps in combination with five future target waypoint coordinates.

perception module could be used to generate routes from pure sensor data, but attempting this is beyond the scope of this work.

Model Architecture

Our proposed model encodes the road and lane line masks into mask features using convolutional layers and tokenizes the feature map following to apply self-attention, inspired by the strategy utilized by vision transformers[Carion et al., 2020]. Then, the target waypoints are encoded using a simple MLP, which is then used as the query inputs to a cross-attention layer to generate a BEV feature map that is conditioned on the route we must follow. Finally, the tokenized BEV representation is reshaped to create a two-dimensional feature map, which is processed by a segmentation head to predict the desired route mask. Figure 4.2.1 shows an overview of our model.

Training

We train the model on the same dataset we collect for our BEV Perception model. We train for 40 epochs, using binary cross-entropy loss with the Adam[Kingma and Ba, 2014] optimizer and a learning rate of 0.001.

Results

We replace the desired route channel of the RL agent with the predicted routes and train from scratch, training for a total of 20 million steps as opposed to the regular 10 million steps of ROACH. The model requiring a longer time to discover successful

Table 4.5: Performance comparison of the predicted route agent in comparison to the privileged agent and our alternative unprivileged agent using the target heatmap representation.

Model	**DS**	**RC**	**IS**
Privileged expert	0.778 ± 0.192	0.927 ± 0.159	0.785 ± 0.089
Target heatmap	0.018 ± 0.012	0.027 ± 0.011	0.858 ± 0.041
Predicted route	0.484	0.574	0.883

driving behavior is expected since our predictions introduce an additional source of noise. The results are explored in Table 4.5. While there is a drop in performance compared to the privileged agent, we believe this is a major step towards building RL systems that can learn to drive through the usage of informative representations. Note that as this predicted route method takes longer to train, we do not train multiple agents.

4.3 Red Light Prediction

The final component we seek to replace in the BEV representation is the traffic light stop zones. While existing object detection methods can be used to detect the presence of traffic lights and their positions, state-of-the-art autonomous driving methods usually do not need specific information, such as the location of the traffic light in the image space. Moreover, detecting a red light in the image does not mean it affects the ego-ehicle, as we might be viewing a light that is signaling other vehicles.

To circumvent this limitation, we propose to predict a binary variable that denotes whether the ego vehicle is currently inside an active stop zone. This is a much simpler task than predicting the segmentation mask of the stop-zones in BEV space, and this binary variable is enough to inform the agent when it must stop for red lights. The prediction of this binary variable then becomes a straightforward binary classification task. We employ a simple off-the-shelf EfficientNet-B0[Tan and Le, 2019] model with a binary classification head for this task. The model takes a single

RGB input from a frontal camera with a resolution of 160x320. It is trained on the same dataset as our previous models, optimizing for a binary cross-entropy loss with a positive weighting of 2.0 to factor in the scarcity of red light examples in the collected data. We observe that this simple approach can effectively directly predict whether the vehicle is currently affected by a red light for both the European and US-style traffic lights without needing to predict any intermediate stop zone information.

However, having a binary variable brings the question of how it should be integrated into the existing architecture. We experiment with multiple options. First, we remove the traffic light stop zone channel from the $s_{\mathbf{bev}}$ and add a binary variable to $s_{\mathbf{mes}}$. This is the most straightforward approach, using the vector of scalars that we already feed to the RL agent. However, it also performs surprisingly poorly. The agent fails to achieve a driving score over 0.05 in any of our experiments with this approach. We believe that the RL agent is ignoring the measurement vector $s_{\mathbf{mes}}$ and relying on $s_{\mathbf{bev}}$ for the most part. To test our claim, we experiment with replacing the stop zone channel with a new binary channel, which is set to all 1s when we predict that we are affected by a red light. The experiment results are shown in Table 4.6

One issue this experiment brings to mind is whether a change in performance should be attributed to the inaccuracies of the red light prediction model or the replacement of the more informative stop-zone representation. To understand the effects of both changes, we train an additional agent that replaces the stop zone channel with a binary channel, but instead of using predictions, it uses ground truth information from the simulator to determine if we are affected by a red light at each step. Observing the results, we see that replacing the stop zone channel with a ground truth binary channel results in only a modest drop in performance, showing that the privileged representation is not critical. The agent that relies on predictions of whether we are at a red light shows a drop in performance, however, it manages to achieve comparable results to the privileged expert.

Table 4.6: Comparison of driving performance in RL agents with different traffic light (TL) representations

Model	**DS**	**RC**	**IS**
Privileged expert	0.778 ± 0.192	0.927 ± 0.159	0.785 ± 0.089
GT binary TL channel	0.747 ± 0.073	0.908 ± 0.034	0.791 ± 0.069
Predicted binary TL channel	0.659 ± 0.141	0.779 ± 0.093	0.892 ± 0.071

Chapter 5

DISCUSSION

In this chapter, we discuss our findings, share our thoughts on how future work can help build better autonomous driving agents, and present the conclusion of our thesis.

5.1 Findings

In this work, we attempt to understand the reason behind the success of privileged RL experts, the challenges of replicating that success with a sensorimotor agent, and our proposed approaches to bridge the gap between the two.

We investigate whether the recent improvements in BEV perception literature[Harley et al., 2023, Li et al., 2022, Philion and Fidler, 2020] actually translate to usable models for autonomous driving agents, by attempting to replicate ground truth BEV channels with their predicted counterparts. What we observe is that the biggest issue with these models is their heavy hardware demands, making it difficult to use state-of-the-art BEV segmentation models in autonomous driving pipelines. This issue is compounded by the fact that models such as SimpleBEV[Harley et al., 2023] are optimized for the segmentation of one class at a time.

We propose possible changes to reduce the memory requirements of the SimpleBEV BEV perception model, train it to predict various classes, and report the driving performance of agents using this perception pipeline. We show that while roads and lane lines can be predicted by the model, predicting vehicles at the same time is challenging, and predicting pedestrians on top of everything is even more challenging. Moreover, we discover that the IOU values normally reported by BEV perception models do not directly translate to good driving behavior for autonomous driving agents in simulation. The mismatch in observed states between a driving

dataset like NuScenes, where accidents or wild maneuvers are rare and an RL agent exploring different state-action pairs is significant. This results in BEV predictions getting worse when during policy rollouts, the RL agent strays from paths available in the training dataset, making recoveries from mistakes much harder, similar to the distribution shift problem observed in imitation learning.

Another finding of our research is the critical importance of how the target route is represented and given to a self-driving RL agent. Although it is not analyzed within the ROACH[Zhang et al., 2021] paper, we claim that this representation of target trajectories is one of the key elements behind the immense success of their expert agent. We claim that the prediction of this route representation is an important step towards providing agents with a better understanding of the path they should follow to reach their goal. Our intuition is that by removing the responsibility of planning a short-term path from the RL agent, we allow it to use its capacity to solve other aspects of driving.

Finally, we investigate whether a privileged representation of the "stop zone" areas affected by traffic lights is necessary for the success of RL agents. We propose to replace the stop zone channel with binary predictions from a vision-based classification model that directly predicts if we are currently affected by a red light. We experiment with different ways of incorporating this information into the model inputs and discover that the RL agent can act upon the data given in the BEV frame much more easily compared to data given as a vector of measurements. When we give this binary red light representation as a channel in the BEV state, the agent is able to drive, showing that less privileged representations are still acceptable for driving.

5.2 Future Work

Our work does not present a state-of-the-art sensorimotor RL agent, but analyzes why such an agent has not been discovered yet and proposes a way towards RL agents that can compete with the dominant behavior cloning approach. Moreover, we pinpoint what exactly autonomous driving agents require from BEV perception

methods, and highlight areas that need improvement. We share our thoughts on required future research directions in this section.

Our findings from our experiments incorporating the Lift-Splat-Shoot[Philion and Fidler, 2020] and SimpleBEV[Harley et al., 2023] models into the autonomous driving pipeline show that there is a mismatch between what is being optimized in the BEV perception research community and what is required by autonomous driving agents. This is partially caused by the evaluation setups of the datasets and benchmarks used. Just as the CARLA benchmark uses challenging hand-designed rare scenarios to test autonomous driving agents, a challenging benchmark to test the robustness of these BEV perception models under unusual positions and environments could alleviate this misalignment issue.

In particular, we discover three major shortcomings of the existing segmentation-only approach of methods like Lift-Splat-Shoot[Philion and Fidler, 2020] and SimpleBEV[Harley et al., 2023]. They are unsuitable for the successful segmentation of the small and unpredictable class of pedestrians, their performance degrades rapidly when the observed states start to differ from the mainstream states of the training dataset. Moreover, we observe that there is a need for models that can segment multiple classes at once. Despite being a heavier model, we think the general idea behind BEVFormer [Li et al., 2022] is promising for the usage of one perception model to handle different segmentation and detection tasks with one architecture. We believe the discovery of a smaller version of such a model would facilitate different research directions for autonomous driving as well.

We also discover that the representation of the trajectory the ego-vehicle must follow is more critical than it appears, despite not being a major point of discussion in the ROACH[Zhang et al., 2021] paper. The performance of a widely used and successful RL expert can plummet when it is presented with a heatmap replacement instead of this dense route representation. We believe that similar to how the branched architecture idea by Codevilla et al. brought forward a wave of success for imitation learning agents, there is more success to be gained for RL agents in finding a better representation method for the state. Our discoveries regarding how the RL agent struggles to act upon information presented in $s_{\mathbf{mes}}$ also support this claim

that better representation methods are needed. We believe that the desired route prediction task is a good candidate for generating a better representation, and an architecture that generates such a short-term plan representation from sensor input could allow a breakthrough in sensorimotor RL agents.

Finally, we investigate how traffic light information can be predicted in the simplest way, and rather than doing difficult object detections or BEV segmentations, we show that simple binary classification models with a very small backbone can be an alternative. However, our hope is that future work can propose a better way of incorporating these predictions than our counter-intuitive approach of giving a binary variable as BEV map channel.

5.3 Conclusion

In this work, we analyze the uniquely successful expert RL agent of Zhang et al. in order to find what makes it different from sensorimotor agents. We pinpoint the parts of the approach that rely on privileged data, present how they can be approached through the usage of existing state-of-the-art models, and design our own architecture in the absence of preexisting solutions.

Although the state-of-the-art RL approaches fall behind the methods of the imitation learning paradigm in autonomous driving benchmarks, we believe there is still discoveries to be made. Although the end-to-end reinforcement learning of autonomous driving appears infeasible without access to vast resources, we find that the advances in computer vision can be used in combination with reinforcement learning to make the learning problem easier and more approachable. We believe that the generation of informative and accurate state representations through computer vision methods holds the key for the discovery of successful sensorimotor RL agents.

BIBLIOGRAPHY

[Behl et al., 2020] Behl, A., Chitta, K., Prakash, A., Ohn-Bar, E., and Geiger, A. (2020). Label efficient visual abstractions for autonomous driving. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2338–2345. IEEE.

[Caesar et al., 2020] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In *CVPR*.

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

[Chekroun et al., 2021] Chekroun, R., Toromanoff, M., Hornauer, S., and Moutarde, F. (2021). Griad: General reinforced imitation for autonomous driving. method ranked# 1 and# 4 in carla challenge 2021. In *NeurIPS 2021 Workshop on Machine Learning for Autonomous Driving*.

[Chen et al., 2021] Chen, D., Koltun, V., and Krähenbühl, P. (2021). Learning to drive from a world on rails. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15590–15599.

[Chen and Krähenbühl, 2022] Chen, D. and Krähenbühl, P. (2022). Learning from all vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17222–17231.

[Chen et al., 2020] Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2020). Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR.

[Chitta et al., 2021] Chitta, K., Prakash, A., and Geiger, A. (2021). Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15793–15803.

[Chitta et al., 2022] Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., and Geiger, A. (2022). Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Codevilla et al., 2018] Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE.

[Codevilla et al., 2019] Codevilla, F., Santana, E., López, A. M., and Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338.

[Dosovitskiy et al., 2017] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

[Fu et al., 2023] Fu, Z., Kreidieh, A. R., Wang, H., Lee, J. W., Delle Monache, M. L., and Bayen, A. M. (2023). Cooperative driving for speed harmonization in mixed-traffic environments. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–8. IEEE.

[Fujimoto et al., 2018] Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR.

[Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S.,

Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

[Harley et al., 2023] Harley, A. W., Fang, Z., Li, J., Ambrus, R., and Fragkiadaki, K. (2023). Simple-bev: What really matters for multi-sensor bev perception? In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2759–2765. IEEE.

[Hart, 2023] Hart, R. (2023). Elon musk predicts tesla self-driving cars will arrive 'this year. Last accessed 28-08-2023.

[Hessel et al., 2018] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Korosec, 2015] Korosec, K. (2015). Elon musk says tesla vehicles will drive themselves in two years. Last accessed 03-08-2023.

[Kumamoto et al., 1980] Kumamoto, H., Tanaka, K., Inoue, K., and Henley, E. J. (1980). Dagger-sampling monte carlo for system unavailability evaluation. *IEEE Transactions on Reliability*, 29(2):122–125.

[Li et al., 2022] Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., and Dai, J. (2022). Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference on computer vision*, pages 1–18. Springer.

[Liang et al., 2018] Liang, X., Wang, T., Yang, L., and Xing, E. (2018). Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599.

[Lillicrap et al., 2016] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *ICLR (Poster)*.

[Lin et al., 2021] Lin, I.-C., Yağan, O., and Carlee, J.-W. (2021). Mixed-autonomy era of transportation: Resilience & autonomous fleet management. Technical report.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[Muller et al., 2005] Muller, U., Ben, J., Cosatto, E., Flepp, B., and Cun, Y. (2005). Off-road obstacle avoidance through end-to-end learning. *Advances in neural information processing systems*, 18.

[Ohn-Bar et al., 2020] Ohn-Bar, E., Prakash, A., Behl, A., Chitta, K., and Geiger, A. (2020). Learning situational driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Organization, 2018] Organization, W. H. (2018). *Global status report on road safety 2018*. World Health Organization.

[Philion and Fidler, 2020] Philion, J. and Fidler, S. (2020). Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European Conference on Computer Vision*, pages 194–210. Springer.

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

[Prakash et al., 2021] Prakash, A., Chitta, K., and Geiger, A. (2021). Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7077–7087.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Singh, 2015] Singh, S. (2015). Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report.

[Smith and Topin, 2019] Smith, L. N. and Topin, N. (2019). Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE.

[Tan and Le, 2019] Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

[Toromanoff et al., 2020] Toromanoff, M., Wirbel, E., and Moutarde, F. (2020). End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7153–7162.

[Wu et al., 2022] Wu, P., Jia, X., Chen, L., Yan, J., Li, H., and Qiao, Y. (2022). Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132.

[Zhang et al., 2023] Zhang, J., Huang, Z., and Ohn-Bar, E. (2023). Coaching a teachable student. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7805–7815.

[Zhang et al., 2021] Zhang, Z., Liniger, A., Dai, D., Yu, F., and Van Gool, L. (2021). End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15222–15232.

Appendix A

ADDITIONAL ANALYSIS OF ROACH

In our attempts to generate a successful sensorimotor RL agent, we investigated several other approaches to improving upon the state-of-the-art. Although none of these avenues of research ended up being the focus of this thesis, we briefly share our findings here in hopes that they help future researchers make more informed choices.

A.1 Action Space Alternatives

An alternative action representation is used in recent state-of-the-art imitation learning methods [Chen et al., 2020, Chitta et al., 2021, Chen and Krähenbühl, 2022, Prakash et al., 2021] where instead of raw actions, relative positions of the ego-vehicle in future time steps are predicted as waypoints. These waypoints are converted into raw actions by using two PID controllers, one for lateral and one for longitudinal control. Although imitation learning methods report higher performance when predicting future waypoints, state-of-the RL methods tend to directly predict the suitable raw actions [Chekroun et al., 2021, Toromanoff et al., 2020, Zhang et al., 2021, Chen et al., 2021]. Since our initial experiments showed no significant benefit to changing the ROACH[Zhang et al., 2021] output representation, we kept the original model's raw action predictions $\mathbf{a} \in [-1, 1]^2$.

A.2 Limitations of the Reward Function

The reward function proposed by Toromanoff et al. [Toromanoff et al., 2020], incentivizes the agent to navigate safely, follow the desired trajectory, maintain an appropriate speed, and make context-aware decisions at intersections and traffic lights. The disadvantage is that the rewards heavily discourage moving away from

the lane designated by the waypoint information. This can be problematic, as the ideal route is calculated without any information regarding the position of other vehicles, or possible hazards on the road like potholes or blockages. In fact, the optimal agent according to this reward function would never do a maneuver to overtake other vehicles, always being content to wait behind a blocking object. However, our attempts to relax the desired position component of the reward function and give the RL agent more flexibility in lane selection have resulted in significant drops in performance. We reason that the learning problem becomes much harder when the agent must also determine which lane it should follow, but a better reward function that considers these issues might be necessary to solve the more complex scenarios of the CARLA[Dosovitskiy et al., 2017] simulator.

A.3 Reinforcement Learning Algorithm

The choice of which learning algorithm to use for an RL problem can be critical. In addition to the PPO[Schulman et al., 2017] algorithm used by Zhang et al.[Zhang et al., 2021], we experiment with two other state-of-the-art RL algorithms that have been successfully applied to a variety of problems, twin delayed deep deterministic policy gradients (TD3)[Fujimoto et al., 2018], and soft actor-critic (SAC) [Haarnoja et al., 2018]. The agents trained with SAC fail to learn the majority of the time, getting stuck in strange policies. We observe the performance of agents trained with TD3 to be better than SAC, yet the performance of agents appeared much more random than PPO, with repeated experiments giving wildly different results just based on random seeds. It is possible we simply did not find the correct hyperparameter configuration required to stabilize the off-policy learning algorithms, as a thorough hyperparameter search in a problem like this requires considerable hardware resources.