Investigation of Different Social Agents in Reinforcement Learning for Autonomous Driving Training in Simulation

by

Yuan Tian

Bachelor of Applied Science, University of British Columbia, 2018

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

December 2023

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Investigation of Different Social Agents in Reinforcement Learning for Autonomous Driving Training in Simulation

submitted by **Yuan Tian** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

Ian M. Mitchell, Professor, Computer Science, UBC
Supervisor

Frank Wood, Professor, Computer Science, UBC
Supervisory Committee Member

Abstract

Automated driving has been pursued for more than fifty years, but with the widespread adoption of machine learning to train autonomous driving agents, simulators have begun to play an increasing role in the development and deployment of autonomous vehicles because they are not only faster, cheaper, and safer than physical experiments, but also more controllable, observable, and repeatable. In order to create realistic simulations, it is critical to have a good "social agent" to control the action of the other vehicles in the simulation. To date, however, no systematic comparison between different social agents has been performed. Here we describe a series of experiments which trained reinforcement learning (RL) agents in the SMARTS (Scalable Multi-Agent RL Training School) traffic simulation environment with three different social agents, and performed a comparison of the performance of the resulting RL agent. Along with SMARTS supported social agents SUMO (Simulation of Urban Mobility) and ZOO (provided by SMARTS), we integrated the DRIVE social agent (provided by Inverted AI) into the simulator for this study. The RL agents were trained and tested in six different task scenarios on five different maps. Overall, the experimental results show that RL agents trained in the environment with the DRIVE social agent performed more consistently on criteria including completion rate, collision rate and off-road rate, indicating DRIVE's heightened behavioral diversity and meaningful interactivity with the ego vehicle. We further conducted an analysis of some of the characteristics of the traffic generated by the different social agents: Traffic density, traffic speed and acceleration distribution, and average neighbor distance.

Lay Summary

There has been remarkable progress in the development of autonomous vehicles in recent years thanks to machine learning. When training an autonomous driving agent, simulation offers many advantages over physical experiments. This study investigated a critical component of simulators, the so-called "social agents" which control the behaviour of the other vehicles in a training simulation. We systematically compared three different social agents within a common simulator and evaluated their impact on the quality of an autonomous agent trained through reinforcement learning on the simulator. Additionally, we analyzed some characteristics of the traffic generated by these three social agents. The study offered insights into which traits in social agents contribute to the improved performance of autonomous driving agents, contributing valuable knowledge to the ongoing advancements in autonomous vehicle technology.

Preface

All work presented henceforth was conducted in the VCR lab, in the Department of Computer Science at the University of British Columbia, Vancouver Campus. This thesis is an original, unpublished work by Yuan Tian, written under the supervision of Ian Mitchell. A short, preliminary version of the work placed second in the NeurIPS 2022 Competition Driving SMARTS for both Track 1 and Track 2 in December 2022.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisor, Dr. Ian Mitchell for his invaluable mentorship and advice, and Dr. Frank Wood for being a constant source of advice and encouragement throughout my M.Sc. degree.

I extend my sincere thanks to InvertedAI for kindly providing their API service to run the DRIVE social agent, and to SMARTS for their support in code integration and debugging of both the simulator and the ZOO social agent. I'm grateful for their expertise and commitment.

I would also like to thank my colleagues at UBC, specifically Wilder, Justin and Yuni. Finally, I thank my friends and family, my partner Xuxin, my parents, and my grandparents for supporting me emotionally through this wonderful journey.

Chapter 1

Introduction

In recent years, there has been remarkable progress in the development of autonomous vehicles. However, research indicates that demonstrating the desired level of reliability for these vehicles would require driving billions of miles on the road [20]. Therefore simulation emerges as a crucial element in the development and deployment of autonomous vehicles. Simulation provides a controllable and cost-effective environment, enabling developers to rapidly test new algorithms and features without the need for real-world vehicle deployment. Simulation not only prevents potential hazards to test participants, such as in pedestrian jaywalking scenarios, but also ensures observable and reproducible test cases for debugging purposes.

Despite the numerous advantages of simulators, the quality and fidelity of a simulator significantly impacts the outcomes of autonomous driving system development. Several critical aspects define a high-quality simulator, including a realistic physics engine, the capability for diverse map and scenario generation, and high quality social agents to control the other vehicles in the simulation environment. Creating social agents that are interactive, behaviorally-diverse, and realistic has been one of the biggest bottlenecks due to the complicated nature of the human driving behaviors they seek to mimic. Even though several widely used open-source microscopic simulators—such as CARLA [12], SUMO [29], and SMARTS (Scalable Multi-Agent RL Training School) [47]—claim to provide simulation for autonomous vehicle development, there are no publicly available comparisons of

the social agents used in the different simulators.

Much recent research attention has been directed at reinforcement learning (RL) for so-called "end-to-end" autonomy, where the agent is trained to take sensor data as input and produce control signals as output, without any assumed models of sensing, dynamics, or control and without any specified intermediate features (such as a system state). We adopt this end-to-end RL approach in this study, and use it to train autonomous ego agents to drive a vehicle within the SMARTS simulation environment. Using SMARTS allowed us to train our ego agents against three different social agents: the agent provided by SUMO, the new agent ZOO developed in SMARTS, and the DRIVE agent developed by InvertedAI. The first two were already integrated into the SMARTS simulator, and we worked with InvertedAI to integrate the DRIVE agent. The RL ego agents were trained and assessed in SMARTS simulation environments featuring these different social agents, and the resulting data were analyzed to characterize the behavior of the traffic generated by these social agents and the qualities of the RL ego agents trained against them.

The rest of this thesis is organized as follows: Chapter 2 reviews prior related work. A detailed experimental setup is provided in Chapter 3. Performance evaluation and analysis are performed in Chapter 4, and Chapter 5 concludes the study with directions for future work.

Chapter 2

Background and Related Work

Autonomous vehicles (AVs) have been studied for decades, but the recent rapid growth in the power of machine learning has dramatically increased the level of interest in the academic and commercial R&D communities. We can identify two broad approaches to autonomous driving. The "modular" approach consists of multiple perception algorithms for cameras and other sensors that produce a simplified and often pre-specified set of features describing the state of the environment and ego vehicle, and then uses model-based planning and control algorithms [15] to decide what actions to take. In contrast, the "end-to-end" approach learns to map sensory input directly into control commands [8]. While the modular approach can take advantage of existing or independently developed models of sensing, dynamics, and control, and the internal feature representations can simplify auditing and interpreting behaviours, the design simplicity of the end-to-end approach makes it increasingly popular.

For an end-to-end learning algorithm, the supervised learning method was first introduced in the late 1980s when Pomerleau built the first end-to-end decision-making system for NAVLAB [34]. This system mapped the current image and laser range finder data to steering angle actions. Later in 2004, inspired by this work, Lecun et al. [24] trained a convolutional neural network (CNN) which can remotely drive a truck on a open terrain and avoid obstacles such as rocks and ponds. In 2016, Bojarski et al. [9] applied an end-to-end deep neural network in practice and firstly tested the lane keeping task in traffic on local roads. However, the supervised

learning method requires a large collection of labelled driving data. Furthermore, different human drivers may make completely different decisions even in the same situation, which results in an ill-posed problem for which it is difficult to train an effective regressor. Consequently, most researchers have focused on reinforcement learning (RL) approaches for self-driving cars, as RL does not require human-labeled training data.

2.1 End-to-end Reinforcement Learning for Autonomous Driving in Simulation

In RL, agents learn behaviours by optimizing a reward function, without relying on manually designed rules or human driving data [42]. While RL encompasses various subdomains, we will limit our scope to end-to-end RL for autonomous driving in simulation. When it comes to autonomous driving development in simulation, TORCS (The Open Racing Car Simulator) [14] and CARLA [12] are the two most popular simulators. While TORCS is designed for racing simulation, CARLA focuses more on providing urban driving simulation environments which are rich in sensory and physical complexity.

In 2010, Daniele et al. [28] used a tabular Q-learning model[1] to learn overtaking strategies in TORCS, with the resulting trained agent outperforming programmed NPCs provided in TORCS. The first modern attempt to apply a vision based end-to-end RL technique for developing autonomous driving policy in the TORCS simulator was presented by Koutnik et al. [22] in 2013. Their work used images from the driver's perspective, and processed the data with both a CNN and an RNN. Lillicrap et al. [27] introduced a deep deterministic policy gradient model that effectively learned deterministic control policies in TORCS. They used an RGB image of the current frame as the observation space, and mapped the current observation space to acceleration, braking and steering control actions. Also in the TORCS environment, an asynchronous advantage actor-critic algorithm (A3C) was proposed by Mnih et al. [31].

In recent years, Peng et al. [33] introduced a deep reinforcement learning al-

[1] Although we use the term "model" here, this neural network design is still an "end-to-end" approach to vehicle control, and is not related to the "model-based planning and control" used in "modular" approaches mentioned above.

gorithm Dueling Double Deep Q-Network (DQN) to achieve autonomous driving, with the trained agent able to outperform a human driver in TORCS. Later, Basile et al. [7] used DQN and Deep Deterministic Policy Gradient-based (DDPG) to achieve a safe driving policy that is robust to sensor faults. The proposed method was validated in TORCS.

When it comes to the CARLA platform, one of the major problems in applying RL is the relatively realistic but high-dimensional sensor inputs, such as complex RGB images. As a result, most of the RL examples on this platform have been focused on simple driving tasks, such as lane following [21]. However, in recent years, there have been some efforts made to make RL work with high-dimensional inputs. For example, Agarwal et al. proposed a framework that first created a low-dimensional representation that comprises a stack of bird's view images, desired trajectory, and traffic-light states, and then fed the representation to the RL model [5]. In 2018, Liang et al. [26] first pretrained a supervised network based on human labeled data, and then initialized a Deep RL model (DDPG) with the pretrained weights of the supervised network. This pipeline alleviated the low exploration efficiency of large continuous action spaces in RL. As a result, their work demonstrated the first effective RL approach for an end-to-end vision-based driving pipeline that outperformed the modular approach at the time on the CARLA simulator.

In recent years, there has been a lot of literature that ensembles various RL or other neural network techniques, such as with supervised neural network. Ahmed et al. [6] proposed an end-to-end AD system that combined a supervised network and a Deep RL agent (DDPG). The input RGB images were first encoded into a set of affordances, and then the DDPG mapped the affordances along with vehicle measurements and a navigation command into control signals for the vehicle.

2.2 Simulation in Autonomous Driving Training

In order to build performance in AVs and confidence that they will not malfunction, we need to collect a *lot* of data. Simulation systems are indispensable not only for reducing cost but also for obtaining massive data sets. Simulation may also be the best way to explore the potential influence of AVs on broader concerns around the

safety and social impact of vehicle traffic.

2.2.1 Simulator Models: Agent-based or Not?

While there are a huge variety of different approaches to vehicle and traffic simulation, Agent-Based Models (ABMs)[17], Discrete-Event Models (DEMs) [23] and System Dynamics Models (SDMs) [40] stand out as the three major categories [32].

An ABM model is a collection of autonomous decision-making agents (such as vehicle drivers or pedestrians) within a specific environment and time-scale. During the simulation, each agent independently makes decisions based on its own sensing, intelligence, memory, and predefined rules. ABM represents a bottom-up approach to modeling traffic, and offers several advantages for AV applications. First, by modeling individual interactions between different agents, ABMs can explore how individual behaviours contribute to collective traffic patterns. Second, ABMs provide a more realistic simulation of urban traffic due to the bottom-up perspective. Third, each agent in the simulation can quickly adapt to new characteristics or complexities [25]. Although computationally more expensive than the alternatives, ABMs have become the predominant tool for urban AV impact assessment and validation due to the growing availability of computational power [25], and the simulation approach of choice when training RL driving agents.

The DEM simulation approach assumes that a system's state variables change only at discrete and separate points in time. In a DEM simulation, a complex system is modeled as an ordered sequence of events, which may involve complicated sequences and hierarchical structures. However, in order to generate the sequence of events, a DEM generally requires agents to define their paths in advance. Since human drivers and pedestrians do not behave or react to one another in a deterministic fashion, predicting their paths in advance is not terribly realistic [13].

SDM traffic simulations dispense entirely with the notion of individual agents, and instead model traffic as if it were a fluid flowing through a network of interconnected pipes. The dynamics of this fluid does take into account the effect of vehicle interactions and hence it does not flow like water or air, but SDMs are typically built to explore network-wide effects, and hence do not typically try to track

the movement of an individual vehicle or its interplay with the traffic around it. SDMs are well-suited for addressing long-term and dynamic management problems; for instance, Wen et al. [44] applied SDM to model urban traffic in Beijing and analyze factors such as energy consumption and carbon emissions. The lack of information about individual vehicles makes SDMs inappropriate for training and testing RL agents for individual vehicles, although they could be used to analyze the high level impacts of increasing penetration of AVs into traffic streams.

2.2.2 Simulator Validation

A driving simulator need not be realistic if the goal is simply to train an agent that can drive in that simulator; for example, driving simulators and the agents that control the competing vehicles in racing games are often more fun with cartoonish physics. But if simulation is used for training agents that may eventually control real-world AVs, it is critical that the simulator be validated against real-world data. Sewall et al. [38] introduced an interactive hybrid simulation of large-scale traffic, and they validated their simulation by comparing the numbers of cars, average velocity, and flux with respect to real-world data. A study by Shaaban et al. [39] compared the performance of modeling dual lane and triple lane roundabouts between two agent-based traffic simulation tools, imTraffic and VISSIM. Their comparison focused on operational measurements such as average delay under various scenarios and traffic flow rates.

In terms of simulation platforms designed for AV development, Yang et al. [45] conducted numeric simulations to compare vehicle based and movement based traffic control for AVs under different intersection configurations. However, the comparison primarily focused on operational measurements, such as delay versus traffic demand and maximum throughput. For evaluating traffic behavior performance, Huber et al. [19] presented a proof of concept for evaluating traffic conflicts with respect to the behavior of a autonomous driving policy in simulation. In the study, they introduced a subset of metrics such as time to collision and deceleration rate to avoid collision.

For multi-scenarios autonomous driving, Sun et al. [41] proposed a reinforcement learning framework that includes an auxiliary task for scenario recognition,

and trained agents in the SMARTS environment. They tested their trained agents in six scenarios and showed the success rate as one of the performance metrics. Yang et al. [46] proposed a reinforcement learning algorithm with a action judgment network which predicts the safety state given current observation and action for developing safer AVs, and collision rate was selected as the evaluation metric. Overall, the majority of performance evaluations focus on the effect of different architectures and training algorithms for the ego vehicle rather than the effect of the social agents that control the other vehicles in the simulation.

2.2.3 AV Competitions in Simulation

The wide availability of ABM simulators and RL training packages have led to a growing crop of simulator based competitions for AV agents. Most focus on agents designed to operate in one of two environments: high-speed racing or urban driving.

The Generalized Racing Intelligence Competition (GRAIC) organized multiple events between 2019 and 2023, focusing on AVs navigating diverse race scenarios while avoiding static and dynamic obstacles along the track [30]. Sensing was simplified: A perception oracle provided known-to-be correct data including a local view of obstacles, lanes, and gates on the track. For the evaluation stage, the submitted controllers were tested not only against dynamic obstacles provided in the simulator, but also against other submitted controllers. Later in 2022, the Learn to Ride Challenge [2] was hosted by Carnegie Mellon University, which tasked participants with training an RL agent to achieve maximum speed while adhering to safety constraints. The challenge featured two tracks: one for training and another for testing. Notably, the testing track did not include dynamic obstacles or social agents. In high-speed racing competitions, it seems that there was little to no involvement of intelligent social agents.[2]

While race competitions typically prioritize maximizing speed, urban driving challenges tend to emphasize the robustness and safety of the controllers given various traffic conditions. In 2019, CARLA hosted the CARLA Autonomous Driving Challenge [1, 12], in which participants were tasked with training a self-driving

[2]GRAIC did feature competitions between submitted agents, but that represents rather constrained social interaction in which all agents are pursuing exactly the same goal.

agent to reach a target destination along a pre-defined route without committing traffic infractions; for example, driving on the wrong side of or off the road. Challenging traffic situations, such as uncontrolled intersections, were included in the testing scenarios, and tests were run under various simulated weather conditions. The evaluation metrics included of route completion percentage before termination (e.g. collisions with other vehicles) and the number of traffic infractions.

In 2022, SMARTS [47] organized the NeurIPS 2022 Driving SMARTS Competition [3]. Participants aimed to develop controllers that can drive as quickly and safely as possible from a start location to a destination amid background traffic. The competition featured various maps and scenarios for both training and testing, with evaluation metrics encompassing safety, comfort (smoothness and safe driving), task completion (percentage of completed scenarios), traffic rule violations, and completion time. The competition included both offline and online learning conditions, and a preliminary version of our work (which used a supervised learning algorithm rather than RL) secured second place in both the offline and online condition.

While both the SMARTS and CARLA urban driving competitions used their own interactive social agents during training and testing, no cross-validation between different social agents was performed by either competition. A study by Gutiérrez-Moreno et al. [18] trained an RL agent for intersection scenarios in both the SMARTS and CARLA simulators, but their goal was to improve training effectiveness and efficiency by first training the ego agent in the faster but less physically accurate SMARTS environment before further training in the slower but more physically accurate CARLA environment. In both environments the same social agent (from CARLA) was used for the other vehicles. Our study seeks to address a gap in the literature by exploring what is essentially the complementary experiment to that performed in [18]: comprehensively comparing ego agents trained in the same simulator environment but with different social agents controlling the other vehicles.

Chapter 3

Setup

Having covered the related work in reinforcement learning and simulation for autonomous driving, in this chapter we will discuss the following aspects that are related to preparing the experiments. The simulation environment section will cover the simulation platform and the simulation scenarios used for training the ego agents. Then the configuration used for all agents' training will be shown in Section 3.2. Last but not least, the details about the three different social agents we trained ego agents with will be discussed in Section 3.3.

3.1 Simulation Environment

The training environment configuration including simulation platform, training scenarios, and social agents is discussed in the following subsections.

3.1.1 Simulation Platform

Vehicle traffic can be simulated at many different levels of fidelity, but here we focus on the microscopic level in which the continuous motion of each vehicle is simulated with the time between simulation steps smaller than one second. There are several widely used open-source microscopic simulators, including CARLA [12] and SUMO [29]; however, for these experiments we chose to use the relatively recently developed SMARTS simulator [47] for a number of reasons:

- SMARTS uses a reduced fidelity vehicle and road model, which permits

faster simulation of individual episodes. That allowed these experiments to proceed on a modest computational budget.

- SMARTS was designed from the beginning to support multiple different vehicle agents, which allowed us to easily train the ego agents using several different social agents for the other vehicles. Out of the box SMARTS supports SUMO agents and its own agents from their so-called Social Agent Zoo (ZOO). We integrated support for the DRIVE agent from InvertedAI (IAI) [37].
- SMARTS follows the standard OpenAI Gym APIs [10], which makes implementation of new features smoother. The metadata collection, visualization as well as reward functions adjustment were easily achieved due to this interface.
- SMARTS is a new simulation environment, so it would be useful to know if it shows training capabilities comparable to established simulators.
- These experiments were a natural extension of code that we developed for the "Driving SMARTS@NeurIPS2022" competition (in which we were awarded second place).
- The development team for SMARTS provided active support during the competition and throughout the implementation of these experiments.

For the scope of this study, the SMARTS version of 0.6.1 and the ZOO social agents available in this version was used. Note that while SUMO is a open-source simulator, SUMO is also integrated in the SMARTS simulator as background traffic provider which is able to control social agents. In all following sections, "SUMO" refers to the social agent controller, not the simulator.

3.1.2 Simulation Scenarios and Episodes

The six scenarios that we used for training and evaluation are described below, roughly in order of degree of difficulty:

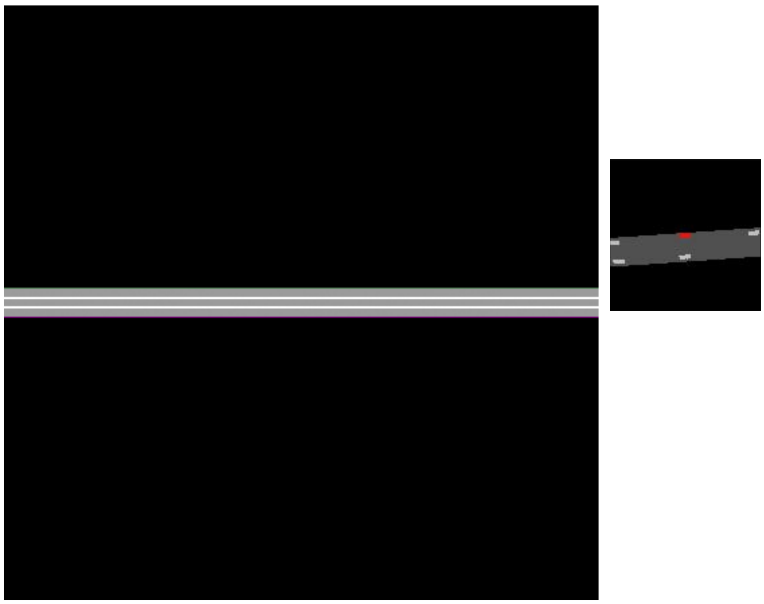


Figure 3.1: Left: Highway cruising map. Right: The 50m x 50m relative ego view to the same scale as the map.

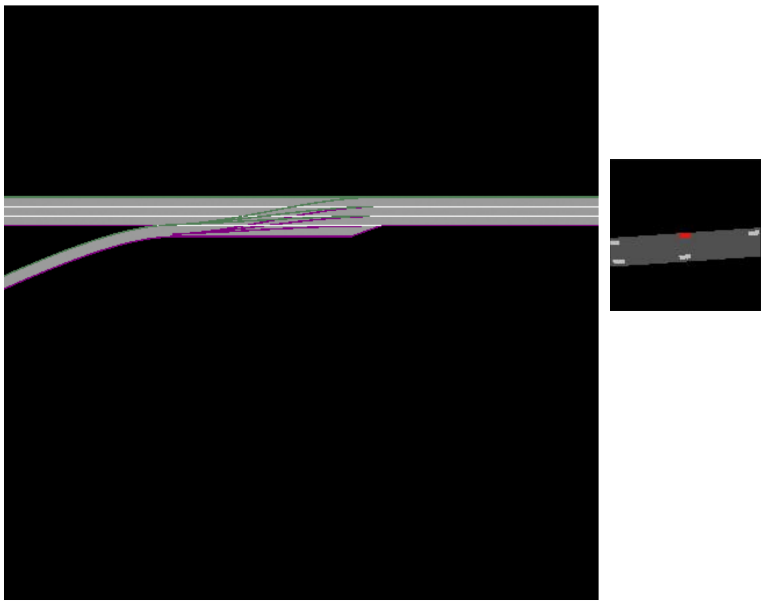


Figure 3.2: Left: Highway merging map. Right: The 50m x 50m relative ego view to the same scale as the map.

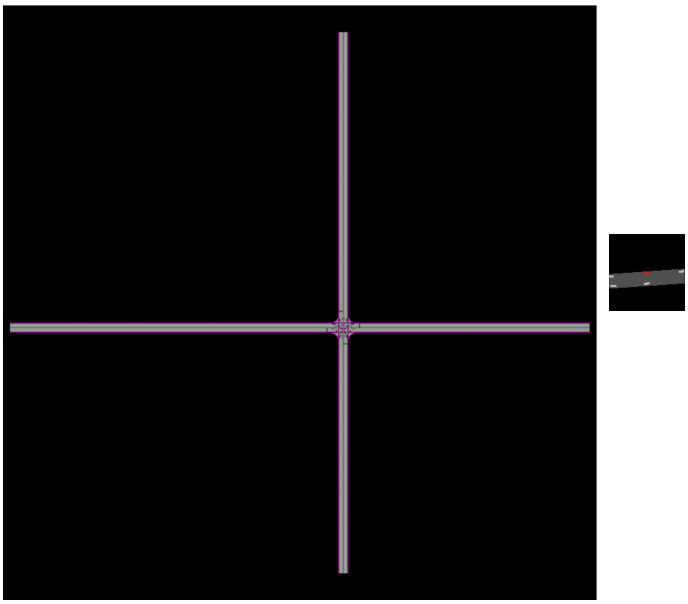


Figure 3.3: Left: Map for one lane to one lane cross intersection scenario with all way stop signs. Right: The 50m x 50m relative ego view to the same scale as the map. Note that this map is larger scale than all of the others, so the relative ego view region looks smaller.

1. Highway cruising: A ego agent starts from one end of a three-lane highway and drives straight to the other end. This scenario requires the ego agent to perform the highway cruising task without colliding into other social agent vehicles which may cut in, overtake, or change speed. Fig 3.1 shows the map definition.

2. Highway merging from an on-ramp: The ego agent starts from a single ramp lane which curves gently into a separated merge lane. Because the merge lane has finite length, the ego agent must merge into the flow of traffic on the three-lane highway to complete the scenario. Fig 3.2 shows the map definition.

3. One lane to one lane cross intersection with all way stop signs: Ego agent starts from the west road linked to a junction with all way stop signs and turns left at the junction onto the to north road to reach the final goal location at

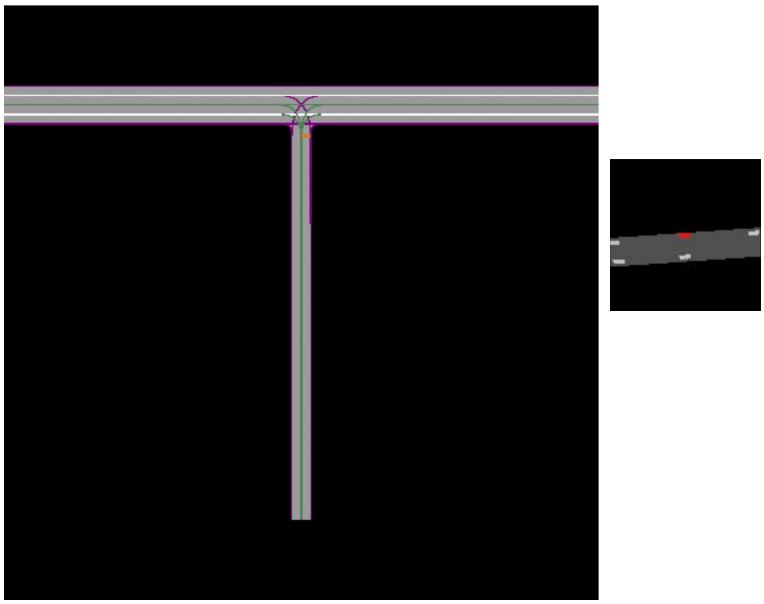


Figure 3.4: Left: Map for T intersection scenarios. The primary (east-west) road has two lanes and traffic priority. The secondary (north-south) road has one lane. Right: The 50m x 50m relative ego view to the same scale as the map.

the end of the road. The social agents controlling the other vehicles know about the stop signs, but the ego vehicle agent must learn these features. Fig 3.3 shows the map definition.

4. One lane to two lane T intersection left turn: Ego car starts on the secondary (south) road with one lane and turns left onto the primary road heading westward. The social agents controlling the other vehicles know that the primary (east-west) road has priority, but the ego vehicle agent must learn this feature. Fig 3.4 shows the map definition.

5. One lane to two lane cross intersection left turn: Same as the T intersection above except that the secondary (north-south) road continues north of the intersection, so other vehicles may start or stop on that additional segment. The primary (east-west) road maintains priority. Fig 3.5 shows the map definition.

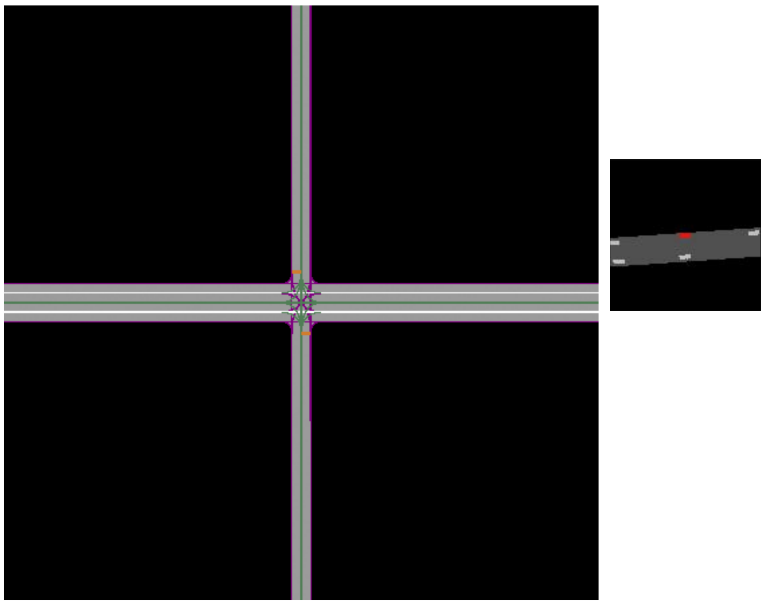


Figure 3.5: Left: Map for the cross intersection scenarios. The primary (east-west) road has two lanes and traffic priority. The secondary (north-south) road has one lane. This map is used for two scenarios (left turn and right turn). Right: The 50m x 50m relative ego view to the same scale as the map.

6. One lane to two lane cross intersection right turn: Same as the cross intersection above but the ego car will turn right from the south secondary road onto the primary road heading eastward. The primary (east-west) road maintains priority. Fig 3.5 shows the map definition.

The physical dimensions that each map represents can be found in Table 3.1. To provide some intuition, the relative size of the ego vehicle's view field is also shown beside each map in Figures 3.1–3.5.

In an episode of a given scenario, the ego agent always has the same start and goal location. Other vehicles are randomly spawned at the start of or during the episode. The same social agent is used for all of the other vehicles in a given episode, depending on against which of the social agents this episode is training or evaluating.

In intersection scenarios, the traffic flow rate ranged from 5 to 15 vehicles per

Table 3.1: Physical Map Size for the Scenarios

Scenario Map Name	Length (m)	Width (m)
Highway Cruising	200	200
Highway Merging	200	200
One Lane to One Lane Cross Intersection	400	400
One Lane to Two Lane T Intersection	200	200
One Lane to Two Lane Cross Intersection	200	200

minute, whereas in highway scenarios, the flow rate varied from 10 to 25 vehicles per minute. The higher spawn rate in highway scenarios was designed to introduce additional complexity to the conditions, as these scenarios lacked intersections, thereby lacking interactions between agents.

All other aspects of the driving task are held fixed across scenarios; for example, the observation, action, and reward functions are the same for all scenarios (see section 3.2).

3.2 Reinforcement Learning Configuration

Because SMARTS is designed to support RL algorithms, the integration of an RL training environment was intuitive and smooth. Since this study is an extension from the "Driving SMARTS@NeurIPS2022" competition, we reused several design parameter choices from it, such as the observation space and choice of RL learning algorithm. However, there were still several hyper parameters we chose to tune, such as the action space and reward function. All hyper parameters were tuned with respect to training against the SUMO social agent, and then they were fixed for the experimental training and evaluation with all social agents (see section 3.3 for more details about the social agents and justification of our tuning process). We discuss the following RL elements below: Observation space, action space, reward function and choice of RL algorithm.

3.2.1 Observation Space

At each time step the ego agent is given one frame showing the state of the simulation at each of the current and previous two time steps (a total of three frames).

Each frame includes:

- The position of the ego vehicle's goal location relative to the ego vehicle's current location. This observation ignores the road network.
- An eight-bit, 112×112 pixel color image showing a bird's-eye view centered at the ego agent's location and showing a 50×50 meter square observation area around it. For a sense of scale, this observation area would cover roughly 6 % of a 200m x 200m scenario map at any given time, and each pixel would represent a patch of ground 0.2 meters square. These images show roadways as grey, off-road (out of bounds) as black, the space occupied by the ego vehicle as red, and other vehicles as white.

Note that the observation image that SMARTS provides does not include any stop sign or roadway priority information. This limitation of the current SMARTS observation space will need to be removed before the simulator can be used for more complex road networks; however, for this study we treated that information as another feature which the RL agent would need to learn. The RL agent faces this same challenge no matter which social agent it is trained against, and it seems quite feasible to learn these features because the stop sign and/or roadway priority is fixed for any given map in our set of scenarios.

3.2.2 Action Space

We initially tried a small space of just four discrete actions: stop, drive straight at 40 km/h, turn left 15° at 40 km/h, or turn right 15° at 40 km/h; however, this level of discretization resulted in very jerky motion by the ego vehicle as it tried to approximate intermediate angles or speeds by switching rapidly between actions. After some experimentation, we settled on ten discrete actions as shown in Table 3.2. The asymmetry between the sharp left and sharp right turn actions arises because left turns have a larger turn radius (and hence a smaller turning angle) than right turns on road networks which have a "drive on the right" rule.

It is worth noting that although the ego vehicle's agent specifies a desired speed in its choice of action, the simulator also enforces maximums on acceleration (2.6 m/s^2) and deceleration (4.5 m/s^2) of vehicles; consequently, the vehicle's actual speed may take several time steps to achieve an indicated desired speed.

Table 3.2: Discrete action spaces.

Action Index	Speed (km/h)	Turning Angle (degree)
0	0	0
1	50	0
2	30	+2
3	50	+2
4	30	-2
5	50	-2
6	30	+10
7	50	+10
8	30	-20
9	50	-20

3.2.3 Reward Function

It is critical to design suitable reward signals to accelerate the learning process and provide a valid baseline. The reward function consists of a weighted sum of functions to encourage the agent to reach the final goal, while satisfying road rule constraints and avoiding collisions. Following [16], it has the form

$$r = w_h r_h + w_r r_r + w_p r_p + r_g \tag{3.1}$$

where r_h, r_r, r_p, and r_g denote reward components for rewards of humanness, rules, making progress to final goal, and reaching the final goal respectively, and w_h, w_r, w_p denote their respective weights chosen to sum to 1. Each reward term r_x is bounded between 0 to 1 as well (except r_g for reasons discussed below).

The reward for humanness is defined as

$$r_h = 0.3\tanh(\frac{d_t}{\sigma_d}) + 0.3\exp(-\frac{d_l^2}{2\sigma_l}) + 0.4 r_{steer} \tag{3.2}$$

where d_t is the distance to the closest obstacle in ego agent's visual field (span of 80 degrees centered at ego car's heading direction) and driving in the same direction as ego car (filtering out cars in the opposite lanes). As we want to maximize the d_t, we use a *tanh* function and choose σ_d to be the distance the ego car can travel

given current speed for 1.5 seconds. Secondly, d_l is the absolute offset of the ego agent from the center of the lane. We want to encourage the offset to be close to 0, so we chose the Gaussian function and set σ_l to be 10 percent of the lane width. Last, we reward action that keeps the same steering angle as the previous action, where r_{steer} is defined as

$$r_{steer} = \begin{cases} 1, & \text{if the current action has the same steering angle as the previous action} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

We admit that this definition of "humanness" is overly simplistic and ignores or overly simplifies factors which are known to influence human driving, such as acceleration and jerk. However, any attempt to quantify humanness in a universal manner using only local simulation state is doomed to failure anyway. We choose this form because it captures some desirable aspects of driver behaviour, and we note that our evaluation criteria stick to other, easily quantifiable properties (such as whether the ego vehicle reaches the goal location) rather than qualitative properties like humanness. It is also important to remember that this reward function only impacts the training of the ego vehicle, and it is the job of the (already trained) social agents to generate human-like trajectories for the other vehicles.

The reward for rule is defined as

$$r_r = 0.7r_w + 0.3r_o \tag{3.4}$$

where r_w is 0 when the ego car is on the wrong way and 1 otherwise, and r_o is equal to 1 when the ego car is on the shoulder and 0 otherwise. On shoulder is a flag that SMARTS provides, and this flag is on when any corner point of the vehicle's bounding box is off the road while the center of the vehicle is not located off the road. This term would also be a natural place to provide a reward for respecting roadway priority, but we have chosen to force the ego vehicle to learn that priority without a direct reward signal.

Table 3.3: Hyper parameter values for reward functions

Hyper Parameters	Description	Value
w_h	Weights for reward of humanness	0.2
w_r	Weights for reward of rules	0.3
w_p	Weights for reward of making progress	0.5
σ_d	Parameters for reward of distance to obstacles	$1.5*v_{ego}$
σ_l	Standard deviation for reward of lane center offset	0.32

The reward for making progress is defined as

$$r_p = \begin{cases} 1, & \text{if the ego car is closer to the final goal than last time step} \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

Finally, the reward for actually reaching the final goal is defined as

$$r_g = \begin{cases} 50, & \text{if the ego car reaches the final goal} \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

The constant "jackpot" reward of 50 for reaching the final goal is chosen to be larger than the maximum reward that the ego car can achieve if it just keeps driving until the episode time limit is reached; consequently, it will prefer to achieve the goal (thus ending the episode) rather than drive in circles until time expires.

The table 3.3 shows the hyper parameters and the corresponding values we chose in the reward function design. These values were chosen by hand-tuning. All training included in this study was using this fixed set of hyper parameters.

3.2.4 Neural Network Architecture and Training

All training was done via Python library SB3 [35] version 1.4.0 using the default network architecture. Figure 3.6 shows the overall architecture that SB3 provides. This network architecture can be separated into two main parts:

1. Feature Extractor: A neural network that extracts features from high dimensional observations. SB3 uses a CNN for image data, and a flatten layer for

vector observation data. The further details of the shape of the network will be discussed in below.

2. Last Network: A two layer fully-connected network that maps the features to actions or values. For PPO, each layer has 64 nodes.

All observations were first pre-processed before being fed to the feature extractor. The image data was normalized by dividing the values by 255 to have values between 0 to 1. The non-image observation data—the relative goal position—is converted to one-hot vectors.

For the image observations, SB3 used the "Nature CNN" for feature extraction. As PPO algorithm was selected, only a linear layer was presented after the CNN. This CNN architecture was shared between actor network and critic network. The details of the architecture used in the "Nature CNN" is shown in Figure 3.7. With a three frame stack of colored images at each timestep, a total of nine channels of image data with 112 by 112 pixel resolution is fed into the "Nature CNN". After three convolutional layers and one linear layer, a feature vector of length 256 is generated.

The output of the CNN is combined with the relative distance and heading with respect to the goal and they are fed to the final two-layer fully-connected network to map to the action space or value space. This last network used 64 units per layer for the PPO algorithm.

More details of the architecture of the "Nature CNN" or the fully-connected network architecture can be found at [35]. The pytorch code definition of the model used in this study can be found in the Appendix A.

The Python library Stable Baseline3 is used for training the RL agents. The learning algorithm is chosen to be Proximal Policy Optimization (PPO) for its relative robustness towards hyperparameter choices [36]. The default hyper parameters from Stable Baseline3 are used (specific values can be found in Table 3.4). More detail can be found at [35].

3.3 Social Agent

We divide the vehicles in our simulations into two categories: A single ego vehicle and all the other vehicles. The ego vehicle's behaviour is controlled by a novel RL

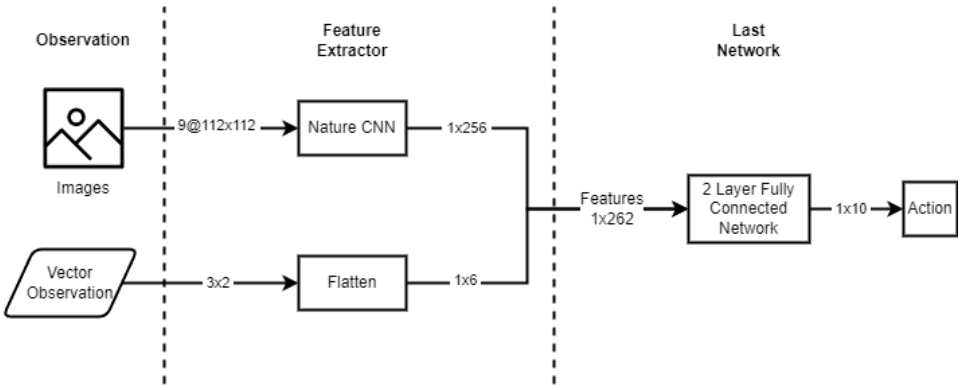


Figure 3.6: The default network architecture that SB3 provided. After feature extraction for image data, the Nature CNN generates output of dimension of 256 while the vector observation gets flattened to a 1 by 6 vector. This combined vector of features of dimension of 262 is fed into the last network. The two layer fully connect network has 64 nodes for each internal layer, and generates output of dimension 10 (the discretized action space dimension).

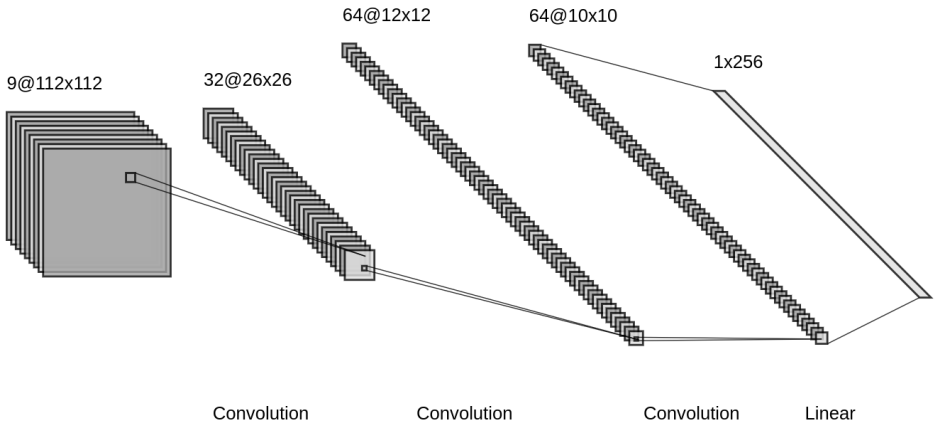


Figure 3.7: Nature CNN architecture used for image input data.

Table 3.4: The default hyperparameter values from Stable Baseline3

Hyper Parameters	Description	Value
Learning Rate	Learning Rate	$3e^{-5}$
N Step	The number of steps to run for each environment per update	2048
Batch Size	Minibatch size	64
N Epoch	Number of epoch when optimizing the surrogate loss	10
Total Timestep	The total number of samples (env steps) to train on	$4e^{6}$

agent which we are training or evaluating in the experiments below. The other vehicles' behaviours are controlled by a fixed or pre-trained social agent. Besides the two social agents that SMARTS already supported—SUMO and ZOO—we also integrated DRIVE social agents from a third party to compare the performance. DRIVE was selected due to its easy to integrate API and responsive customer service. To conclude, three different social agents were selected for this study:

- SUMO [29] has been an open-source traffic simulation suite since 2001, and it has been widely used as a baseline social agent ever since. It is also by default the background social agent that SMARTS provides. SUMO version of 1.7.0 was used in the SMARTS simulator for this study.

- ZOO [47] is the more intelligent social agent that SMARTS claims to be, and it is developed and deployed along with SMARTS simulator. It could come from self play or population-based training [43]. We were curious about the performance of this ZOO agent compared to SUMO agent.

- Last of all, DRIVE [37] developed a deep generative model for multi-agent trajectory prediction which claims to provide a more realistic, behaviorally diverse and interactive behavior model for social agents. They provide API support for integrating their behavior model into the SMARTS simulator.

The social agents for the other vehicles are not given any path or goal information; they are simply supposed to mimic how human agents might behave in

similar circumstances. Each other vehicle is also an independent agent, and is given the bird-view image centered at itself, all neighbor vehicles' state including position, heading, speed and bounding box, as well as the stop sign information as their observation space. The social agents are **not** aware of which vehicle is the ego vehicle. The SMARTS simulator provides the same observation space to all social agents, and it is the social agent's own decision on which information to use to generate their next action. The social agent's policy is packed in their own code base, so the details will not be discussed here but can be found in each citation.

We consider the SUMO agent as the baseline social agent in our experiments in the sense that training algorithm hyper parameters were (lightly) tuned to improve results against SUMO agents, and then those hyper parameters were reused without tuning when training against the other two social agents. We adopted this approach because SUMO is the most widely used of the three social agents and hyper parameter tuning is computationally expensive; therefore, we believe the most common use case for the other two agents would be swapping them into a training regime already tuned for SUMO agents.

In addition to the three specified social agents, we established a training environment that included all social agents, referred to as "ALL" in subsequent sections. This training environment with ALL social agents consisted of six scenarios for each social agent, totaling 18 scenarios. To ensure a fair comparison, the total training steps for all ego agents remained consistent, as illustrated in Table 3.4. Consequently, in the training environment with ALL social agents, the time spent learning against each individual social agent was roughly a third of what was encountered for training sets involving a single social agent.

Chapter 4

Results

With respect to each social agent (SUMO, ZOO, DRIVE, and ALL), we repeated the training process for an ego agent three times from different random initial neural network parameters in the six training scenarios described in Section 3.1.2. This protocol resulted in a total of twelve independently trained ego agents. The purpose of training three independent ego agents against each social agent is to get some intuition for the variability in results arising from the randomization in the initial network parameters and the training algorithm itself; if there had been time, we would have trained five or perhaps seven to improve that intuition. Then we evaluate each trained agent in the test scenarios and analyze the results.

4.1 Evaluation Configuration

The test scenarios were identical to the training scenarios in Section 3.1.2, but initialized with different random seeds. Every trained ego agent was tested against each of the three social agents, including the one with which it was originally trained, in the 6 scenarios. For each scenario, 20 episodes were evaluated. This resulted in a total of 120 episodes evaluated for each trained agent.

All three instances of ego agents trained against each social agent were evaluated, and the median of those three evaluation scores is presented in the following sections. The detailed data for all twelve individual ego agents can be found in Appendix A.

Table 4.1: Median Completion Rate, Collision Rate, and Off-road Rate for evaluation in SUMO environment.

	Test in SUMO		
Trained with	Completion Rate	Collision Rate	Off-Road Rate
SUMO	0.692	0.233	0.075
DRIVE	0.575	0.283	0.142
ZOO	0.533	0.225	0.242
ALL	0.517	0.100	0.383

Table 4.2: Median Completion Rate, Collision Rate, and Off-road Rate for evaluation in ZOO environment.

	Test in ZOO		
Trained with	Completion Rate	Collision Rate	Off-Road Rate
ZOO	0.600	0.208	0.075
DRIVE	0.475	0.442	0.083
ALL	0.458	0.267	0.275
SUMO	0.358	0.600	0.058

4.2 Performance Analysis

Table 4.1, Table 4.2, and Table 4.3 show the results of trained agents evaluated in the SUMO, ZOO, and DRIVE test environments respectively. We did not *test* against the ALL social agent because the results can be inferred by averaging the results for the other three. The tables report average completion rate, collision

Table 4.3: Median Completion Rate, Collision Rate, and Off-road Rate for evaluation in DRIVE environment.

	Test in DRIVE		
Trained with	Completion Rate	Collision Rate	Off-Road Rate
DRIVE	0.608	0.217	0.175
SUMO	0.529	0.379	0.093
ZOO	0.525	0.275	0.100
ALL	0.509	0.275	0.217

rate and off-road rate on 120 episodes among six scenarios ranking from highest completion rate to the lowest. All values shown in the tables are the median of the three evaluation results from the three instances of ego agents trained in each social agent environment. There are a couple of details worth noting about the rates reported in these tables:

- In addition to collision, off-road, and completion, there is another (rare) termination condition: exceeding the time step limit. Consequently, the collision rate, off-road rate, and completion rate may not sum to one.
- If the ego car leaves the map—for example, overshoots the goal location—it is counted as "off-road."

We visualize the data in Table 4.1, Table 4.2, and Table 4.3 in Figure 4.1. The figure shows the percentage chart for how episodes terminated in each of the twelve test cases. Here it is easy to see that for the majority of the test cases the off-road rate is lower than the collision rate, which means that collision with other social agents is the main reason that an ego car could not complete the task within the time constraint.

To get a feel for the variability between different training runs, Figure 4.2 shows a bar chart for the completion rates of all three instances of each trained ego agent evaluated in all three testing environments. Unsurprisingly, agents trained in the environment with the same social agents as the test environment always scored the highest completion rate, which is 69.2% for SUMO, 60.0% for ZOO, and 60.8% for DRIVE. Averaging across the other three ego agents in each of the three test environments (so ZOO, DRIVE and ALL tested against SUMO; SUMO, DRIVE and ALL tested against ZOO, ...), we see that the average completion rate is highest when tested against SUMO and lowest against ZOO. Based on these comparative results, we hypothesize that SUMO provides the least challenging social agent environment, while ZOO provides the most challenging. We will discuss this hypothesis further in Section 4.3 and Section 4.4.

Figure 4.2 also shows that agents trained in the DRIVE environment consistently scored the second best in the other two social agents' test environments. Quantitatively the ego agent trained against DRIVE scored 57.5% tested against

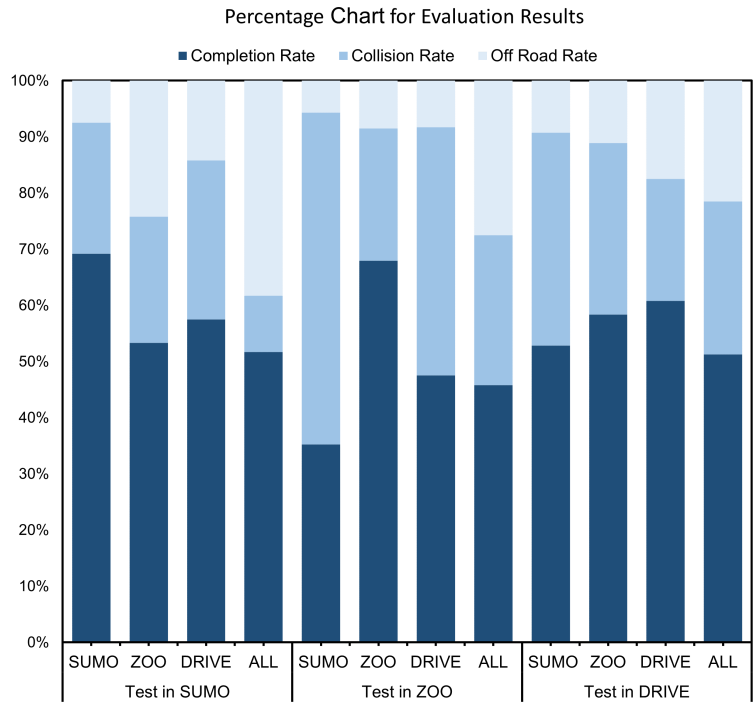


Figure 4.1: Percentage chart for completion rate, collision rate and off-road rate for all twelve evaluation results. These values are for the median of the three ego agent instances trained against a particular social agent, and correspond to the data reported in Table 4.1, Table 4.2, and Table 4.3.

SUMO and 47.5% tested against ZOO (see Table 4.1 and Table 4.2). These results potentially indicate that the DRIVE social agents may provide the most behaviorally diverse training environment, thereby resulting in a more robustly trained ego agent.

In contrast, we now draw attention to the large disparity in completion rates for ZOO trained ego agents against SUMO social agents, and vice versa. Furthermore, while the ZOO trained ego agent achieves roughly the same collision rate in the SUMO test as DRIVE and even SUMO itself, the SUMO trained ego agents has by far the worst collision rate of all test cases for the ZOO test. These results provide further support for the hypothesis that the ZOO and SUMO social agent

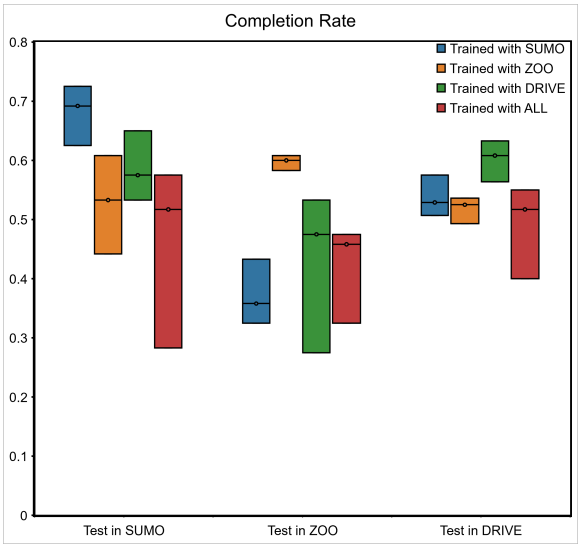


Figure 4.2: Completion rate of trained ego agents in different testing environments. A given bar shows the completion rate for all three independently trained instances of the ego agent: the top and bottom of the bar correspond to the rates of the best and worst instances, and the line through the middle is the median instance. It is this median instance which is reported in the first columns of Table 4.1, Table 4.2, and Table 4.3.

behaviours deviate significantly from one another, with the ZOO social agents the more challenging of the two. To further investigate this, we perform a more detailed analysis of the reasons for collisions and the characteristics of the driving environments presented by the different social agents.

Turning now to the bottom rows of Table 4.1, Table 4.2, and Table 4.3 (or the red bars in Figure 4.2), we see that the ego agent trained against ALL social agents appears to have the most consistent performance among all test environments; for example, its median completion rate is 51.7% tested against SUMO and DRIVE social agents, and slightly lower at 45.8% against ZOO social agents. It does have the lowest median completion rate among the ego agents in both the SUMO and DRIVE tests, but in both cases it is only a tiny bit worse that the ZOO trained ego agent. Looking at the reasons for failure, its high off-road rate may indicate a need for longer training time or a deeper neural network architecture, because it

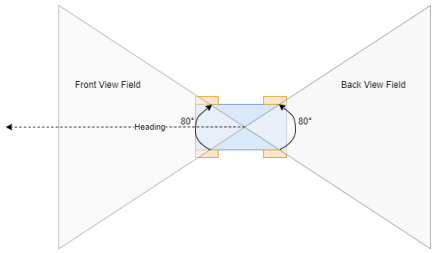


Figure 4.3: Front and rear view field of an ego car. They are both defined as a span of 80° centered along the heading axis.

is trying to learn a more complex environment (more different behaviours for the other vehicles) with the same number of training episodes.

We did test for the statistical significance of the completion rate results with both paired and unpaired t-tests; however, given that we had only three instances of each agent it should come as no surprise that the data is insufficient to claim any strong result of this nature.

4.3 Collision Rate Analysis

To further analyze the collision incidents detected during test evaluations, we define three collision types. For the purpose of these definitions, Figure 4.3 shows the front and rear view fields of the ego car.

1. Front Collision: The other vehicle is within the front view field of the ego car and the difference between the other car's heading and the ego car's heading is smaller than 40 degrees. In this type of accident it is highly likely that the ego car is at fault, as the collided vehicle was struck from behind by the ego vehicle.

2. Rear Collision: The other vehicle is within the rear view field of the ego car. In this type of accident, it is highly likely that the social agent controlling the other vehicle is at fault, as the ego car was the one being struck from behind.

3. Other Collision: All collisions that cannot be categorized into the former two types, such as head-on or T-bone collisions, are binned into this category.

Table 4.4: Median collision rate for different collision types for evaluation in SUMO environment.

Trained with	Test in SUMO Front Collision Rate (Proportion)	 Rear Collision Rate (Proportion)	 Other Collision Rate (Proportion)
SUMO	0.050 (21.4%)	0.033 (14.3%)	0.150 (64.3%)
ZOO	0.192 (85.2%)	0.008 (3.7%)	0.025 (11.1%)
DRIVE	0.067 (23.5%)	0.050 (17.6%)	0.167 (58.8%)
ALL	0.058 (58.3%)	0.000 (0.0%)	0.042 (41.7%)

Determining fault in this type of collision can be complicated, so we do not attempt to do so.

Through this collision type categorization we are better able to understand the collision rates during testing against different social agents, and hence potentially some qualitative characteristics of the three social agents.

With that in mind, Table 4.4, Table 4.5, and Table 4.6 show the rates of the different categories of collisions during testing against SUMO, ZOO and DRIVE social agents respectively. The percentage values inside the parentheses are the fraction of all collisions attributable to that category. In other words, the rates (outside the parentheses) should sum across a row to the rate reported in the second columns of Table 4.1, Table 4.2, and Table 4.3 respectively, while the percentages (inside the parentheses) should sum across a row to 100%.

Collisions against the SUMO social agents: The rear collision rates for all trained ego agents in Table 4.4 stand out for being very near zero, and the lowest rear collision rates in any of the tables. From this we may hypothesize that the SUMO behaviour model may contain a certain hard-coded driving mechanism that prevents them from colliding into the vehicle in front. While it may represent defensive driving at its best, it is unrealistic and may cause the ego vehicle to learn to ignore the vehicles behind it and therefore stop as quickly as possible whenever it wants. Evidence of this negative effect on ego agent training can be found in Table 4.5 and Table 4.6, where the category responsible for the most collisions for ego agents trained against SUMO social agents is a rear collision.

Table 4.5: Median collision rate for different collision types for evaluation in ZOO environment.

Trained with	Test in ZOO Front Collision Rate (Proportion)	Rear Collision Rate (Proportion)	Other Collision Rate (Proportion)
SUMO	0.075 (12.5%)	0.367 (61.1%)	0.158 (26.4%)
ZOO	0.025 (12.0%)	0.125 (60.0%)	0.058 (28.0%)
DRIVE	0.033 (7.5%)	0.308 (69.8%)	0.100 (22.6%)
ALL	0.042 (15.6%)	0.150 (56.2%)	0.075 (28.1%)

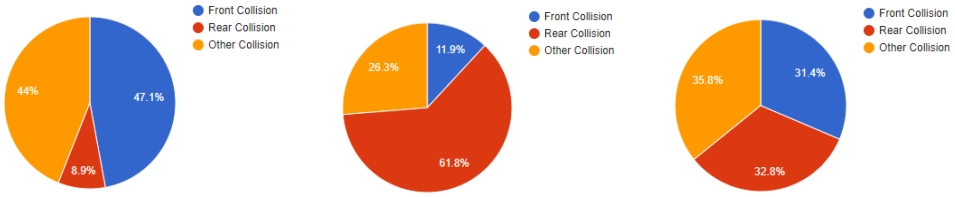
Table 4.6: Median collision rate for different collision types for evaluation in DRIVE environment.

Trained with	Test in DRIVE Front Collision Rate (Proportion)	Rear Collision Rate (Proportion)	Other Collision Rate (Proportion)
SUMO	0.093 (24.5%)	0.150 (39.6%)	0.136 (35.8%)
ZOO	0.083 (30.3%)	0.058 (21.2%)	0.133 (48.5%)
DRIVE	0.042 (19.2%)	0.100 (46.2%)	0.075 (34.6%)
ALL	0.142 (51.5%)	0.067 (24.2%)	0.067 (24.2%)

Collisions against the ZOO social agents: The opposite effect can be seen in Table 4.5, where rear collisions make up the overwhelming majority of collisions. From this pattern we hypothesize that the ZOO social agents are more aggressive and less interactive than SUMO or DRIVE social agents. As a consequence, ego agents trained against ZOO social agents may learn to be very cautious about the vehicles behind them, as demonstrated in Table 4.4 and Table 4.6 by the fact that the ego agent trained against the ZOO social agents has a significantly lower rate of being rear-ended than ego agents trained against SUMO or ZOO.

Collisions against the DRIVE social agents: It is much harder to discern a pattern from testing against DRIVE social agents as reported in Table 4.6, as the category of the most common collision type varies depending on which social agent the ego vehicle was trained against. To further explore this lack of an obvious pattern, Figure 4.4 shows for each test environment the fraction of collision types

averaged across the ego agents which were trained against all four social agent types. From this we see that when testing against DRIVE social agents, the three categories of collision are much more balanced than when testing against either SUMO or ZOO social agents. In fact, comparison of the data in Table 4.6 against the other two tables confirms that the ego agent trained against each of the four social agents demonstrates greater balance between the categories of collision in the DRIVE tests than in tests against SUMO or ZOO. This observation of the collision type distribution suggests that DRIVE may have the most behaviorally diverse social agent model.



(a) Test in SUMO **(b)** Test in ZOO **(c)** Test in DRIVE

Figure 4.4: Average proportion of collision types under different testing environments.

Comparing against collision categories in real life is difficult, since each real life collision does not involve an "ego" and an "other" vehicle, and is categorized by a trained human assessor rather than with a simplistic mathematical definition. For example, the US National Highway Traffic Safety Administration (NHTSA) Crash Report Sampling System (CRSS) during the period 2016–2020 reported that front-to-rear collision accounts for 43.9% of traffic accidents and angle accounts for 33.8% [4]. These CRSS categories are not equivalent to our definitions—for example, a CRSS front-to-rear collision means that one of the vehicles suffered a front and one of the vehicles suffered a rear collision according to our categories—but it does indicate that the direction of the vehicles before collision is reasonably well balanced. Figure 4.4 makes it clear that the DRIVE social agent comes closest to achieving a balanced portfolio of failure modes.

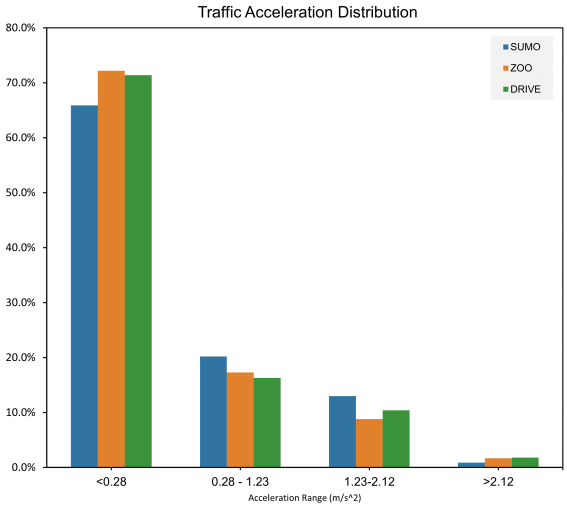


Figure 4.5: Traffic acceleration distribution for social agents.

4.4 Characteristics of Social Agents

To further investigate the characteristics of the three social agents, we collected some metadata about the environment which those social agents created. Note that the ego agent's behaviour has only indirect and limited effect on this metadata as it is mediated through its interaction with the social agents in its immediate vicinity.

4.4.1 Traffic Acceleration and Speed

Figure 4.5 shows the distribution of traffic acceleration for the three social agents. The data is categorized into four ranges: less than 0.28 m/s^2, between 0.28 to 1.23 m/s^2, between 1.23 to 2.12 m/s^2, and more than 2.12 m/s^2. These thresholds are derived from a study by [11], where longitudinal accelerations up to 0.28 m/s^2 were deemed 'Excellent,' and accelerations surpassing 2.12 m/s^2 were regarded as 'Terrible.' The inflection point for acceptable acceleration, described as 'So-so,' was found to be at 1.23 m/s^2.

As shown in the figure, for all social agents, over 65% of accelerations fall within the "Excellent" category, while less than 2% of accelerations are categorized as "Terrible". Notably, there is no substantial difference in the distribution of

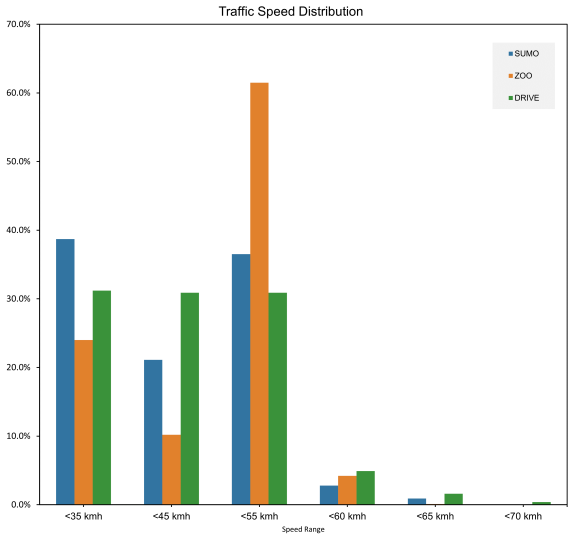


Figure 4.6: Traffic speed distribution for social agents.

accelerations among the different social agents.

Figure 4.6 shows the distribution of traffic speeds. The most conspicuous outlier is within the speed range of 45 km/h to 55 km/h, where over 60% of ZOO agents fall, while only around 30% of both DRIVE and SUMO agents fall within this range. As a result, the average speed of ZOO agents is notably higher than that of SUMO or DRIVE agents, which likely contributes to the high rear-end collision rate for ego vehicles in the ZOO tests. In contrast, DRIVE agents display an even speed distribution across bins below 55 km/h and a greater fraction of speeds above 55 km/h, again indicating a more diverse behavioral model.

4.4.2 Traffic Density

We define traffic density as the count of social agent vehicles within a 50×50 meter view zone centered around the ego car at each time step. The average traffic density has been calculated for each scenario and each social agents, and the outcomes are presented in Figure 4.7.

For scenarios like Highway Cruise, Highway Merge, 1 to 2 lane left turn c (cross intersection), and 1 to 2 lane left turn t (T-junction), the traffic densities

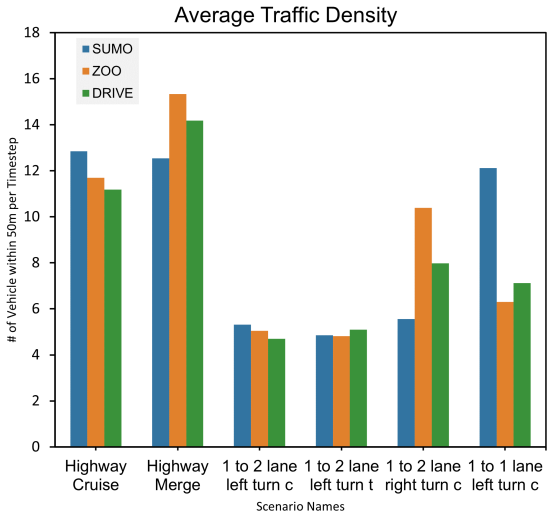
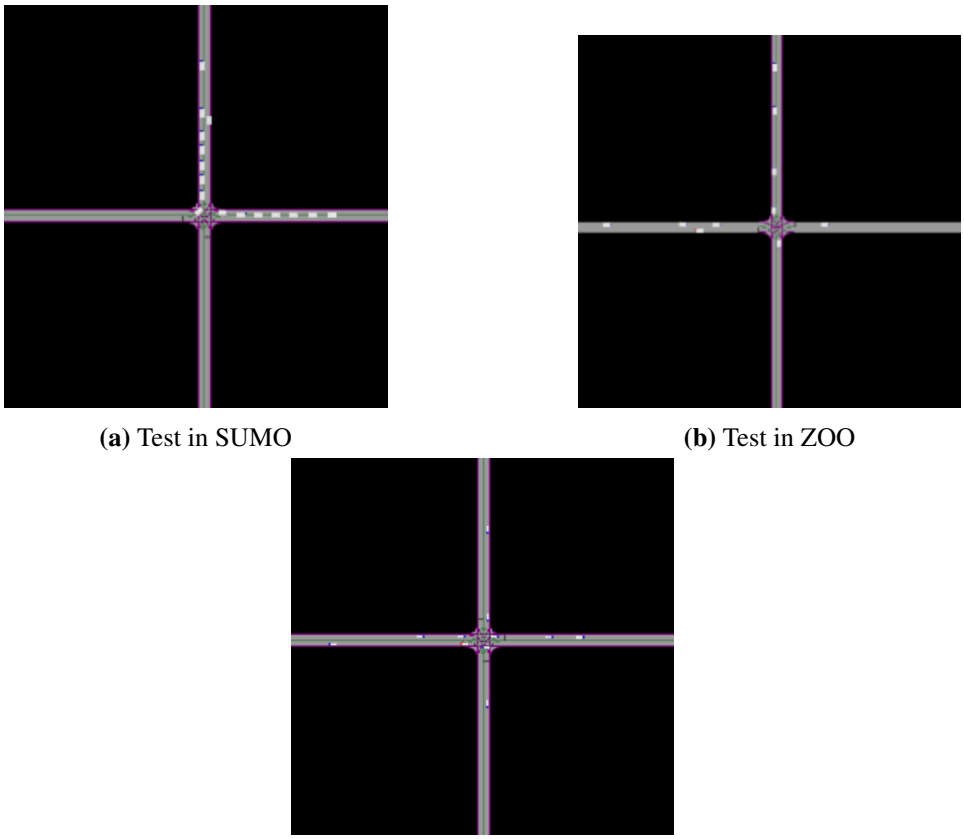


Figure 4.7: Traffic density for the environments generated by each social agent for each scenario.

are within a similar range across different social agents. Unsurprisingly, highway scenarios generally have higher traffic density compared to intersection scenarios, as intersections introduce more complicated conditions that may slow down traffic flow.

Interestingly, in the case of 1 to 1 lane left turn c scenarios, the SUMO environment exhibits notably higher traffic density than both the ZOO and DRIVE environments. To investigate this observation, animations of several episodes in the SUMO environment were observed and compared with those in the ZOO and DRIVE environments. The comparison between representative snapshots of these scenarios running under different social agent environments are presented in Figure 4.8. It is evident from these snapshots that in the SUMO environment, traffic congestion prior to entering the intersection is considerably worse than in the ZOO and DRIVE environments.

The cause of this severe traffic jam before the intersection in the SUMO environment might be attributed to the behavioral model of SUMO agents, which strictly follows hard-coded rules. As a result, when an ego car's behavior deviates from SUMO agents' expectations, SUMO agents can easily end up in deadlock sit-



(a) Test in SUMO

(b) Test in ZOO

(c) Test in DRIVE

Figure 4.8: Snapshots of running scenarios 1 lane to 1 lane left turn cross intersection under different social agents environment.

uations. On the other hand, ZOO and DRIVE agents appear to be less likely to wait for a long time (if not forever) when presented with out-of-distribution behaviors.

A second outlier is 1 to 2 lane right turn c, where the traffic density is higher for ZOO and DRIVE environment than SUMO environment. The cause of this outcome remains unclear and is worth further investigation.

4.4.3 Traffic Neighbor Distance

A "neighbor" of a given agent is defined as the closest vehicle within its field of view and heading in roughly the same direction. This definition is similar to the

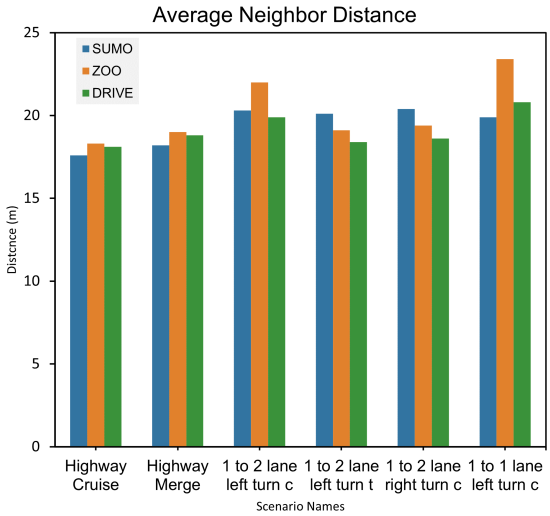


Figure 4.9: Average neighbor distance between vehicles for the environments generated by each social agent.

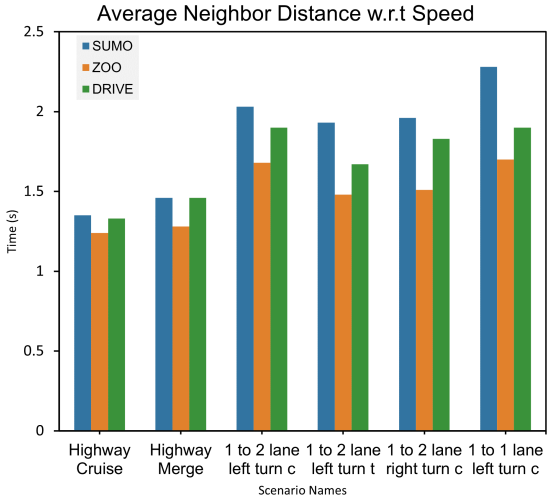


Figure 4.10: Average neighbor distance between vehicles with respect to vehicles' speed for the environments generated by each social agent.

definition stated for the front collision type as outlined in Section 4.3, and ignores vehicles behind, to the side, or traveling in a different direction. It is important to note that a vehicle may have no neighbor when no other vehicle on the road meets the specified criteria. As a consequence, no neighbor distance data is collected for that vehicle during that particular time step.

The results for neighbor distances are presented in Figure 4.9 and Figure 4.10, with the former showing results in meters and the latter in units of time. In the latter figure, the distance is divided by the speed of the subject vehicle to approximate the reaction time available to the driver if the neighboring vehicle were to come to an abrupt halt.

As shown in Figure 4.9, the neighbor distances seem to be relatively consistent across all social agents for each scenario, with the ZOO agent displaying slightly longer distances in four out of the six scenarios. However, when considering the neighbor distances with respect to the vehicle's speed as shown in Figure 4.10, it becomes clear that ZOO agents have the shortest driver reaction times across all scenarios, while SUMO agents have the longest reaction times. This outcome aligns with the findings from Section 4.4.1, where ZOO environments demonstrate higher traffic speeds and SUMO environments have the slower speeds. The short reaction time of ZOO agents may also contribute to their high rate of rear collisions, as mentioned in Section 4.3.

4.5 Limitations

This study has a number of limitations. All trained agents were based on a simple default neural network architecture provided by Python library stable baseline3 and detailed in Section 3.2.4. This architecture is likely insufficiently deep to enable an ego car to learn to respond consistently to the complex traffic scenarios presented by the various interacting social agents, as evidenced by the completion rates of each trained agent. The highest completion rate, 0.692, is attained by an agent trained and tested in the SUMO environment; however, this rate would be considered dismal for a real autonomous vehicle. This study was intended to be a lightweight exploration of the relative merits of training against different social agents, but an ego agent intended for actual use would certainly require a deeper

and perhaps more complex network architecture, and a corresponding greater training time.

The second limitation arises from the discrete action space adopted in this study. We chose a discrete action space in alignment with the default settings of the "Driving SMARTS@NeurIPS2022" competition and to avoid the complexity of a continuous action space. However, our study adopted a relatively coarse action space with only ten options. This choice reduced training time but led to non-human driving behaviors, such as abrupt braking and zig-zag motions. If time allows, exploring a higher-dimensional discrete action space or a continuous action space would be worth considering.

Last but not least, our testing environment was not as diverse as it could be. The testing scenarios were identical to those used in training. While different random seeds were applied to the evaluation environments, no novel maps or ego car tasks were introduced. Although additional novel scenarios were available from the competition, integrating DRIVE or ZOO agents into each scenario required effort from the agents' authors and we did not want to strain their already generous support for this preliminary study. If this study is extended, adding more scenarios would be a straightforward if somewhat time-consuming task.

Chapter 5

Conclusion

To supplement the existing studies on social agents for autonomous driving development provided by different simulators, this study compared the performance of the ego agent policies learned through RL from training in a common simulator environment but with three different social agents: SUMO, ZOO, and DRIVE. Specifically, we implemented six training scenarios with the three social agents: highway cruising, highway merging, one lane to one lane cross intersection left turn, one lane to two lanes cross intersection left turn, one lane to two lanes T intersection left turn, and one lane to two lanes cross intersection right turn. For a fair comparison among the ego agents trained with different social agents, all of the training environment factors—including observation and action space, reward function, neural network architectures and related learning hyper parameters—were set the same. The trained ego agents were then evaluated with each social agent under six test scenarios, which were identical to the ones in training but with different random seeds. Performance was compared under the criteria completion rate, collision rate, off-road rate, and collision type distribution. To further explore observation the effect of social agent, we investigated the general traffic characteristics of each social agent: traffic density, traffic speed distribution, traffic acceleration distribution, and average neighbor distance.

We data shows some interesting patterns. Unsurprisingly, all trained ego vehicles scored the highest completion rate under the test environment with the same social agents as their training environment. However, autonomous ego vehicles

trained in environments with the DRIVE social agents scored the second best in all test environments with other social agents consistently. As for collision rate, the test environment with the ZOO social agents showed the highest average collision rate of 37.9%, while it is 28.7% for DRIVE and 21.0% for SUMO. We characterized the type of collision into three easily distinguished categories: front (ego agent likely at fault), rear (social agent likely at fault), and other (unknown fault). More than 60% of collisions when testing in ZOO environments were rear collisions, while only 9% of collisions when testing SUMO environments were rear collisions. These statistics lead us to believe that the ZOO agents are quite aggressive and the SUMO agents quite defensive. DRIVE social agents appeared to have the most balanced collision type distribution where each type accounted for roughly one third of the total collisions.

We further investigated the characteristics of the traffic generated by each social agent. Many of the measures were essentially the same for all social agents, but a few stood out. While more than 60% of the ZOO social agents fell into the speed bins above 45 km/h, only around 40% of either SUMO or DRIVE social agents was driving more than 45 km/h. In addition, ZOO social agents had the shortest average neighbor distance with respect to their current speed for all six scenarios among all social agents. These characteristics further explained the high collision rate and the high rear-end collision proportion for ZOO social agents.

In general, the findings of our study reveal that each social agent has its own characteristics, and those different characteristics impact the policies of ego agents trained against those social agents. ZOO social agents appeared to be too aggressive on the road and not interactive enough with the ego vehicles, which resulted in overly cautious ego vehicles. While SUMO social agents showed a fair amount of interactivity by yielding to ego vehicles most of the time, it also resulted in aggressive ego vehicles with the highest collision rate in the test environments with other social agents. Lastly, DRIVE appeared to be the most behaviorally-diverse social agent among them all, and it also appeared to offer a happier medium on the scale of aggressive to defensive interactivity with ego vehicles.

During the experiment's setup stage, we integrated DRIVE into the SMARTS simulator, and implemented code that supported five different map settings and six scenarios. With the option to modify the ego car's task, additional scenarios can be

easily created. The integration code is open source.

This study discussed several representative social agents for training autonomous ego agents in simulation, but our comparisons cannot cover all possible social agents or simulators. From the results of this study, we believe it a worthwhile endeavor to explore other social agents in traffic simulation. If future study is undertaken, it would be worthwhile to explore more complicated neural network architectures, hyper parameter tuning, and action spaces to see if those have an effect on ego agent performance. Regarding the comparison between different social agents, a notable limitation of the current study was the use of the same scenarios for testing as for training; it would be good to introduce unseen maps and/or goals at the testing phase.

Bibliography

[1] Carla autonomous driving challenge. https://carlachallenge.org/, 2019. → page 8

[2] Learn to ride, safe learning for autonomous vehicle. https://www.aicrowd.com/challenges/learn-to-race-autonomous-racing-virtual-challenge, 2021. → page 8

[3] 2022 neurips driving smarts competitionpermalink. https://smarts-project.github.io/competition/2023_driving_smarts/, 2022. → page 9

[4] V. Adewopo, N. Elsayed, Z. Elsayed, M. Ozer, V. Wangia-Anderson, and A. Abdelgawad. Ai on the road: A comprehensive analysis of traffic accidents and accident detection system in smart cities, 2023. → page 33

[5] T. Agarwal, H. Arora, and J. Schneider. Learning urban driving policies using deep reinforcement learning. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 607–614, 2021. doi:10.1109/ITSC48978.2021.9564412. → page 5

[6] 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*, 23(8):12562–12571, 2022. doi:10.1109/TITS.2021.3115235. → page 5

[7] G. Basile, A. Petrillo, and S. Santini. Ddpg based end-to-end driving enhanced with safe anomaly detection functionality for autonomous vehicles. In *2022 IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, pages 248–253, 2022. doi:10.1109/MetroXRAINE54828.2022.9967647. → page 5

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. → page 3

[9] 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. *arXiv preprint arXiv:1704.07911*, 2017. → page 3

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016. → page 11

[11] K. N. de Winkel, T. Irmak, R. Happee, and B. Shyrokau. Standards for passenger comfort in automated vehicles: Acceleration and jerk. *Applied Ergonomics*, 106:103881, 2023. ISSN 0003-6870. doi:https://doi.org/10.1016/j.apergo.2022.103881. URL https://www.sciencedirect.com/science/article/pii/S0003687022002046. → page 34

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator, 2017. → pages 1, 4, 8, 10

[13] B. Dubiel and O. Tsimhoni. Integrating agent based modeling into a discrete event simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 9–pp. IEEE, 2005. → page 6

[14] E. Espié, C. Guionneau, B. Wymann, C. Dimitrakakis, R. Coulom, and A. Sumner. Torcs, the open racing car simulator. 2005. URL https://api.semanticscholar.org/CorpusID:16920486. → page 4

[15] U. Franke. Autonomous driving. *Computer Vision in Vehicle Technology: Land, Sea & Air*, pages 24–54, 2017. → page 3

[16] Y. Fuchioka, Z. Xie, and M. van de Panne. Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors, 2023. → page 18

[17] N. Gilbert. *Agent-based models*. Sage Publications, 2019. → page 6

[18] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, J. Araluce, and L. M. Bergasa. Reinforcement learning-based autonomous driving at intersections in carla simulator. *Sensors*, 22(21), 2022. ISSN 1424-8220. doi:10.3390/s22218373. URL https://www.mdpi.com/1424-8220/22/21/8373. → page 9

[19] B. Huber, S. Herzog, C. Sippl, R. German, and A. Djanatliev. Evaluation of virtual traffic situations for testing automated driving functions based on multidimensional criticality analysis. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, 2020. doi:10.1109/ITSC45102.2020.9294169. → page 7

[20] N. Kalra and S. M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016. ISSN 0965-8564. doi:https://doi.org/10.1016/j.tra.2016.09.010. URL https://www.sciencedirect.com/science/article/pii/S0965856416302129. → page 1

[21] 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, 2018. → page 5

[22] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, page 1061–1068, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319638. doi:10.1145/2463372.2463509. URL https://doi.org/10.1145/2463372.2463509. → page 4

[23] A. M. Law, W. D. Kelton, and W. D. Kelton. *Simulation modeling and analysis*, volume 3. Mcgraw-hill New York, 2007. → page 6

[24] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp. Off-road obstacle avoidance through end-to-end learning. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, page 739–746, Cambridge, MA, USA, 2005. MIT Press. → page 3

[25] J. Li, E. Rombaut, and L. Vanhaverbeke. A systematic review of agent-based models for autonomous vehicles in urban mobility and logistics: Possibilities for integrated simulation models. *Computers, Environment and Urban Systems*, 89:101686, 2021. ISSN 0198-9715. doi:https://doi.org/10.1016/j.compenvurbsys.2021.101686. URL https://www.sciencedirect.com/science/article/pii/S0198971521000934. → page 6

[26] X. Liang, T. Wang, L. Yang, and E. Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving, 2018. → page 5

[27] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. → page 4

[28] D. Loiacono, A. Prete, P. L. Lanzi, and L. Cardamone. Learning to overtake in torcs using simple reinforcement learning. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010. → page 4

[29] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL https://elib.dlr.de/124092/. → pages 1, 10, 23

[30] Minghao Jiang and Zexiang Liu and Kristina Miller and Dawei Sun and Arnab Datta and Yixuan Jia and Sayan Mitra and Necmiye Ozay. Graic: A simulator framework for autonomous racing. https://popgri.github.io/Race/, 2021. → page 8

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016. → page 4

[32] Y. B. Moon. Simulation modelling for sustainability: a review of the literature. *International Journal of Sustainable Engineering*, 10(1):2–19, 2017. doi:10.1080/19397038.2016.1220990. → page 6

[33] 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*, 4:328 – 337, 2021. URL https://api.semanticscholar.org/CorpusID:237717051. → page 4

[34] D. A. Pomerleau and D. S. Touretzky. Advances in neural information processing systems 1. *ch. ALVINN: An Autonomous Land Vehicle in a Neural Network*, pages 305–313, 1989. → page 3

[35] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html. → pages 20, 21

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. → page 21

[37] A. Scibior, V. Lioutas, D. Reda, P. Bateni, and F. Wood. Imagining the road ahead: Multi-agent trajectory prediction via differentiable simulation, 2021. → pages 11, 23

[38] J. Sewall, D. Wilkie, and M. C. Lin. Interactive hybrid simulation of large-scale traffic. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 30(6), December 2011. → page 7

[39] K. Shaaban and I. Kim. Comparison of simtraffic and vissim microscopic traffic simulation tools in modeling roundabouts. *Procedia Computer Science*, 52:43–50, 2015. ISSN 1877-0509. doi:https://doi.org/10.1016/j.procs.2015.05.016. URL https://www.sciencedirect.com/science/article/pii/S1877050915008169. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015). → page 7

[40] J. Sterman. *Business dynamics*. Irwin/McGraw-Hill c2000.., 2010. → page 6

[41] J. Sun, X. Fang, and Q. Zhang. Reinforcement learning driving strategy based on auxiliary task for multi-scenarios autonomous driving. In *2023 IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 1337–1342, 2023. doi:10.1109/DDCLS58216.2023.10166271. → page 7

[42] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. → page 4

[43] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1913–1922, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi:10.1145/3357384.3357902. URL https://doi.org/10.1145/3357384.3357902. → page 23

[44] L. Wen and L. Bai. System dynamics modeling and policy simulation for urban traffic: a case study in beijing. *Environmental Modeling & Assessment*, 22:363–378, 2017. → page 7

[45] C. Yang, X. Lin, M. Li, and F. He. Simulation comparisons of vehicle-based and movement-based traffic control for autonomous vehicles at isolated intersections. *IEEE Transactions on Intelligent Transportation Systems*, 23 (12):22954–22970, 2022. doi:10.1109/TITS.2022.3210772. → page 7

[46] F. Yang, X. Li, Q. Liu, C. Liu, Z. Li, and Y. Liu. Filling action selection reinforcement learning algorithm for safer autonomous driving in multi-traffic scenes. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7, 2023. doi:10.1109/IV55152.2023.10186804. → page 8

[47] M. Zhou, J. Luo, J. Villella, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen, A. C. Huang, Y. Wen, K. Hassanzadeh, D. Graves, D. Chen, Z. Zhu, N. Nguyen, M. Elsayed, K. Shao, S. Ahilan, B. Zhang, J. Wu, Z. Fu, K. Rezaee, P. Yadmellat, M. Rohani, N. P. Nieves, Y. Ni, S. Banijamali, A. C. Rivers, Z. Tian, D. Palenicek, H. bou Ammar, H. Zhang, W. Liu, J. Hao, and J. Wang. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving, 2020. → pages 1, 9, 10, 23

Appendix A

Supporting Materials

Inverted AI provided some characteristic data about real traffic, which we compare in the figures below against the data collected in section 4.4. In each graph the red bars show the real data.

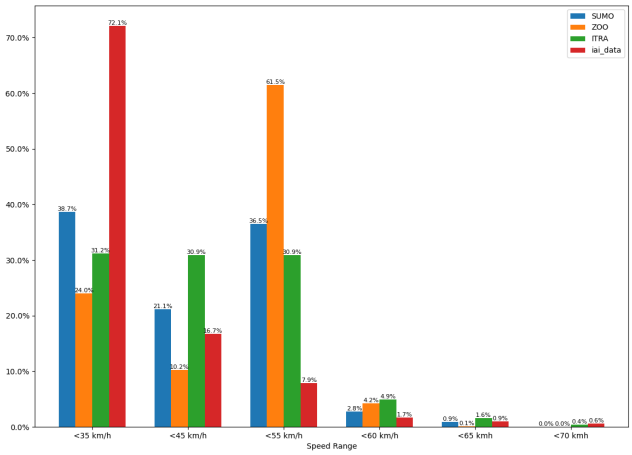


Figure A.1: Traffic speed distribution for social agents and real traffic data provided by InvertedAI.

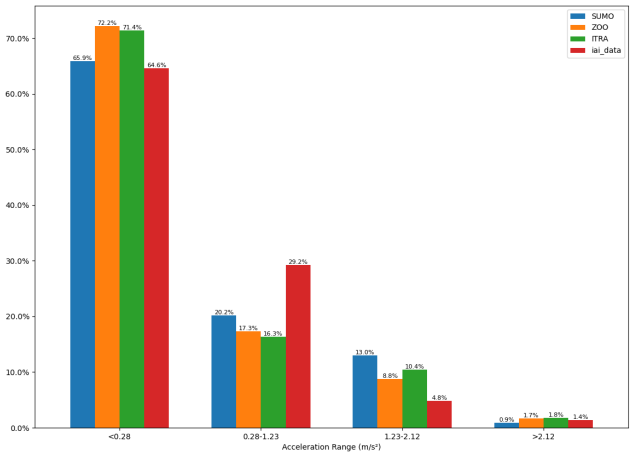


Figure A.2: Traffic acceleration distribution for social agents and real traffic data provided by InvertedAI.

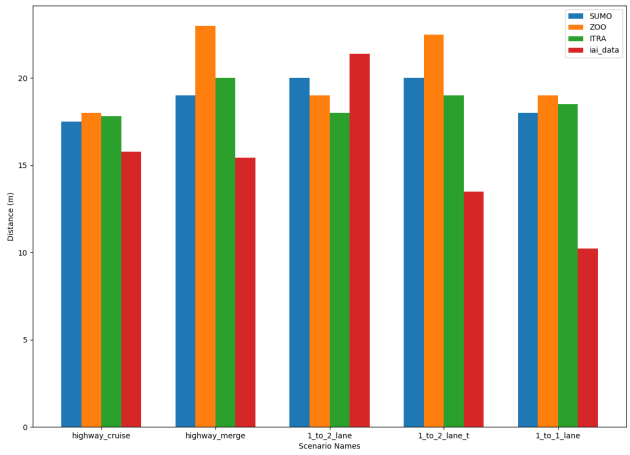


Figure A.3: Traffic average neighbor distance for social agents and real traffic data provided by InvertedAI.

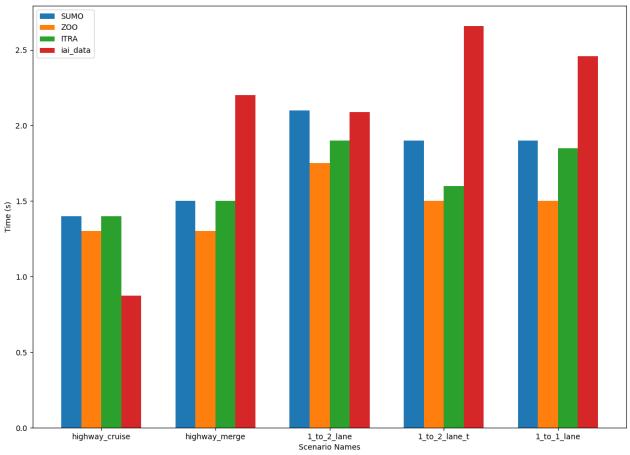


Figure A.4: Traffic average neighbor distance with respect to vehicles' speed for social agents and real traffic data provided by InvertedAI.