Chair of Computer Science 6 (Machine Learning and Reasoning)

RWTH AACHEN UNIVERSITY

The present work was submitted to the Chair of Machine Learning and Reasoning.
Diese Arbeit wurde vorgelegt am Lehrstuhl für Maschinelles Lernen und Inferenz.

Towards Generalizable Automated Driving: A Hierarchical Reinforcement Learning Approach with Engineered Low-Level Policies in a Scenario-Based Learning Environment

Master Thesis
Masterarbeit

Presented by / Vorgelegt von

Mahmoud Youssef Mahmoud Abdelhamid,
414308

1st Supervisor / 1. Betreuer Dr. Zlatan Ajanović

2nd Supervisor / 2. Betreuer Lennart Vater, M.Sc.

1st Examiner / 1. Prüfer Univ.-Prof. Hector Geffner, Ph.D.

2nd Examiner / 2. Prüfer Univ.-Prof. Dr.-Ing. Lutz Eckstein

Aachen, August 21, 2024

Abstract

Applying Reinforcement Learning (RL) algorithms in the context of Automated Driving Systems (ADS) face challenges of being prone to doing unsafe actions as well as being slow in learning and having inconsistent results. Current approaches utilize RL, Imitation Learning (IL), rule-based approaches, or a combination. Some of these attempts employ either temporal hierarchical policy or scenario-based environments. However, these attempts lack either efficiency or generalizability of learning. This thesis explores a potential solution for these challenges that combines the use of Hierarchical Reinforcement Learning (HRL) that helps reduce the complexity of credit assignment issues and thus helps with the efficiency and generalizability of learning. Moreover, this thesis uses a scenario-based environment which guarantees the assessment of the generalizability of learning. This also allows for incorporating challenging scenarios which enhances learning efficiency by exposing the ego vehicle to more complex interactions than those encountered on real roads during training. The method introduced in this thesis is the custom RL environment Scenario Automated Driving Reinforcement Learning (SAD-RL) that utilizes scenarios and uses the HRL low-level policies executor Scenario Automated Driving Pilot (SAD-Pilot). This method is evaluated on several custom scenario-based datasets. The results show that Synchronous Advantage Actor Critic (A2C), could demonstrate fast learning, transfer learning as well as consistent and promising results when trained on SAD-RL environment. Furthermore, comprehensive ablation studies revealed the significance of using HRL and using scenarios in achieving safety, efficiency, and generalizability of learning. The fully trained A2C agent achieved final results that were at least as good as, and in some cases better than, the agent that was trained exclusively on each specific type of scenario on their respective test.

Contents

1 Introduction

Highly automated driving is a promising technology for improved passenger comfort and road traffic safety [1]. However, it is a challenge to develop such a system that can participate in complex road traffic alongside other road users [2]. Even if the perception of the surroundings is already reliable and precise, highly automated driving systems must plan their actions, predict the behavior of surrounding road users, and make the right decisions. Furthermore, they must react even to unexpected behavior of surrounding road users. One example is a harshly braking lead vehicle. In these situations, it might be hard to avoid a collision. Thus, the system should make good tactical decisions to avoid such a bad situation with a limited possibility of avoiding a collision. Reinforcement learning (RL) is a promising approach for automatically handling complex decision-making processes [3]. The idea of RL is that an agent learns to master a task from trial-and-error interaction with the environment. This method is currently also receiving a lot of attention in the automated driving domain [4]. Usually, the agent drives in a simulation environment with random traffic, trying to learn how to drive just from experience. However, the learning process in RL is generally not sample efficient, i.e., it takes a long time to train an RL agent. One approach to increase the efficiency of the training process of an RL agent in the context of automated driving is to systematically present interesting traffic situations in the simulation. Furthermore, using Hierarchical Reinforcement Learning (HRL) can enhance the learning efficiency and safety of the system. HRL introduces a framework that separates decision-making into high-level options and low-level policies, allowing the agent to learn complex behaviors through a structured and abstract approach. This research aims to investigate the use of scenario-based techniques for RL in the context of automated driving, with a particular focus on HRL mechanisms. The first step is to conduct a detailed literature review on scenarios and RL in automated driving. The second step is to design and implement an approach that makes use of scenarios for efficiently training an RL agent, incorporating HRL to improve learning efficiency and decision-making safety. The scope of this thesis will be on highway driving. Furthermore, the developed approach shall be evaluated. This evaluation process will involve a rigorous examination and direct comparison with alternative approaches. The goal is to determine the algorithm's capacity to safely drive on a highway without colliding with other road users or going off-road.

2 Background

The integration of artificial intelligence technologies became essential to help make the driving experience both smoother and safer. This chapter provides an overview of the Levels of Driving Automation, driving scenarios, and Artificial Intelligence technologies, including RL and Imitation Learning.

2.1 Automated Driving Systems

This section provides an overview of the key concepts and technologies related to automated vehicles. It explores the levels of driving automation, ranging from driver assistance to full automation, as defined by the Society of Automotive Engineers (SAE). Additionally, it discusses various driving scenarios, including safety-threatening situations and the implementation of challenging driving scenarios. Lastly, it examines the layers of motion planning, detailing how automated vehicles perceive their environment, make decisions, and execute driving maneuvers.

2.1.1 Levels of Driving Automation

The Levels of driving automation rely on the activities and responsibilities associated with the vehicle operation defined in SAE J3016 and shown in Figure 2.1 [5] as Dynamic Driving Task (DDT). This includes tasks such as steering, acceleration, and deceleration all of which are typically performed by humans. The document defines the 6 Levels of driving automation ranging from Level 0 (No Driving Automation) where the human is responsible for the entire DDT, up to Level 5 (Full Driving Automation) where the vehicle is capable of performing all aspects of DDT. Levels 1 and 2 the driver has the overall responsibility of the DDT. For Levels 3, 4, and 5 the ADS is responsible for the entire DDT.

- **Level 0 - No Driving Automation** The entire DDT is handled by the driver, regardless of whether or not active safety systems are applied which take actions to prevent potential danger and not only detect it.
- **Level 1 - Driver Assistance** The driver performs either the lateral or the longitudinal motion control while the ADS handles the other. The ADS is not capable of fully detecting or responding to the driving events and objects. Therefore, the driver must supervise the ADS by handling the Object and Event Detection and Response (OEDR). Moreover, the driver can disengage the driving automation at request.

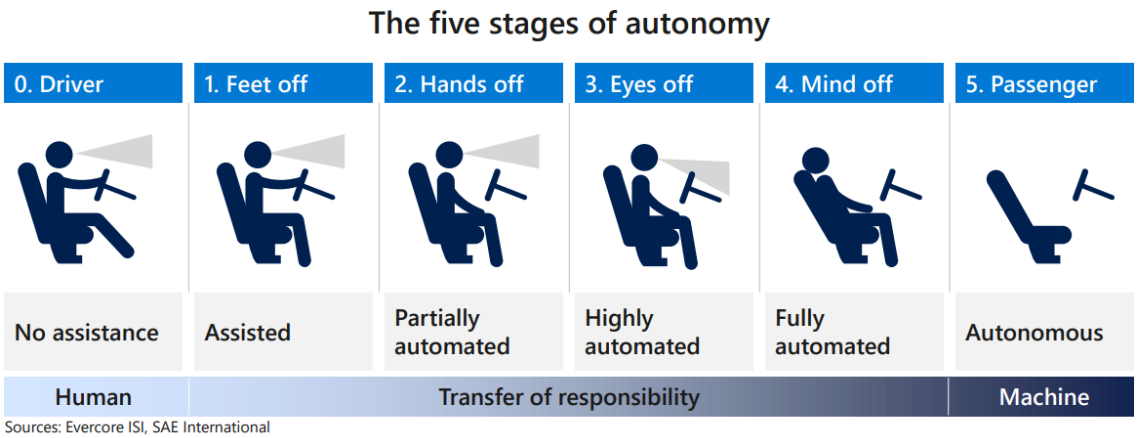


Figure 2.1: Levels of Driving Automation [5]

- **Level 2 - Partial Driving Automation** The ADS is capable of both lateral and longitudinal of the DDT. However, it can only handle limited OEDR, so some events and objects cannot be fully recognized or responded to. Therefore, the driver must always supervise the driving automation system.
- **Level 3 - Conditional Driving Automation** ADS performs the entire DDT within its domain. It also checks if the domain is about to be violated and executes DDT Fallback disengages appropriately after requesting an intervention from the user. It then disengages completely when the user requests for the disengagement of the ADS.
- **Level 4 - High Driving Automation** The user in this level does not need to supervise the ADS performance, but like Level 3, the ADS here performs the entire DDT within its domain while requesting the user's intervention when the domain is violated. On the other hand, it performs DDT Fallback when the DDT performance-relevant system failure occurs and it starts using the fallback system and minimal risk condition capability. At this level the ADS operates the vehicle if it's on a highway, however, after the vehicle exits the highway, the user will be requested to intervene.
- **Level 5 - Full Driving Automation** This level differs from the previous two levels by having an unconditional domain. This means that the ADS performs under all road conditions and any circumstances. In the event of DDT performance-relevant system failure, ADS in this level automatically activates DDT fallback and achieves minimal risk conditions. It also differs from all previous levels in the point that the user is not required to supervise the ADS.

2.1.2 Scenarios

Scenarios are generically associated with simulations used to test an interaction through a sequence of scenes [6]. They are mainly used in three applications: strategic management,

human-computer interaction, and software and systems engineering [7]. In the context of automated driving, a scenario can be defined as the storyline that includes the expected action of the driver [8].

2.1.2.1 Challenging Scenarios

The main challenge to ADS that stand as a barrier to progress is safety assurance [9]. The idea of a complex automated system operating in a dynamic real-world environment requires a minimum threshold of safety which should be very ambitious or at least as good as the human drivers [10]. Driving a car as a human driver through a regular road for hours does not result in frequent crashes. It was statistically shown that because the mean distance between car crashes is large, during a test it would be infeasible. An ADS according to [11] requires a testing distance of 6.6 billion testing kilometers to achieve an acceptable safety threshold. Recognizing this concern emerges the idea of using scenarios as a safety measure for the automated system. The authors of [12] identified the steps to generate a safety-challenging scenario. First, the challenger scenario concept is defined, in which it involves challenger objects that can be dynamic or static. These objects are not necessarily the cause of a car crash but they tend to pose a threat to the ego vehicle and it can either force it to move or cause an accident. Then these scenarios are generated from data with traffic events, based on a comprehensive Layer Model, that is primarily inspired by the PEGASUS project [13] that resulted in the 6 Layer Model presented in [14]. The Layer model consists of six layers, that contribute to the definition of challenging scenarios. They are designed to encapsulate different aspects of the driving environment.

- **Layer 1** - **Street layer:** Describes the road geometry and topology, which includes information about the road layout, like the number of lanes, lane marking, curvature of the road, and elevation.
- **Layer 2** - **Traffic infrastructure:** Defines traffic infrastructure, like barriers, signs, or traffic lights.
- **Layer 3** - **Temporal modifications:** Represents modifications over time to layers 1 and 2, to include scenarios with temporary construction sites or additional static objects.
- **Layer 4** - **Movable objects:** Represents the dynamic elements, like motor vehicles.
- **Layer 5** - **Environment conditions:** Describes factors that can externally affect the road condition (other layers), like the weather.
- **Layer 6** - **Data and communication:** This layer was extended to cover digital information like high-definition map data.

In the challenger scenario concept, Layers 1 and 4 hold the most interest as they contain the fundamental aspects of the driving environment and the dynamic interactions between objects. These are the core information needed to develop a challenging scenario that an ego vehicle can encounter otherwise in a long drive in real traffic. As mentioned by Riedmaier et al [9] the

scenario-based approach works on reducing the testing distance for ADS mainly by omitting the time steps at which no significant action or event occurs.

2.1.2.2 Applications of Challenging Scenarios

The driving scenarios have to be described uniformly to be used in testing simulation tools. For example, the representations used in the Pegasus project [13] were OpenDRIVE for static elements and OpenSCENARIO [15] for dynamic elements. However, some tools that are used in building simulations have specific format requirements that these representations do not yet support. To solve this issue, some researchers focused on building simulation tools that work with a common format. One of these tools is Lanelet2 [16] which is an open-source map representation and a library that can use this representation to develop an enclosed simulation framework that can be used for motion planning and highly automated driving applications. Another representation was developed by TUM university researchers, who developed a framework called CommonRoad [17] which is a framework for comparing motion planners on different types of roads. Along with the framework, they developed a representation of the simulation elements using XML files that consist of the information needed for representing a scenario. It includes details about lanelets, obstacles, and the scenario's planning problem.

2.1.3 Layers of Decision-Making in ADS

Motion Planning is an important component in the field of ADS, where the vehicles need to navigate through environments autonomously. This process is organized into a set of layers each of which represents a specific aspect of the planning and control process. These layers are different depending on the type of the system, They're usually considered when the motion planning solution is not end-to-end. Paden et al [18] called them Route planning, Behavioral layer, Motion planning, and Feedback control as shown in Figure 2.2

Route planning in automated driving involves algorithms that plan a high-level path from the vehicle's current position to a specified destination. Global path planning computes an initial path or waypoint to the destination, considering static obstacles such as roads, buildings, and traffic regulations.

The behavioral layer defines high-level behaviors and decision-making processes for the vehicle. Behavioral planning establishes strategies based on the current traffic conditions, environmental factors, and the vehicle's goals. Decision-making algorithms determine which behavior to activate in response to input from the environment.

Motion planning breaks down the global path from the behavioral layer into a set of waypoints and generates a smooth trajectory for the vehicle to follow. The selected behavior is then translated into a trajectory that can be achieved by low-level feedback controller.

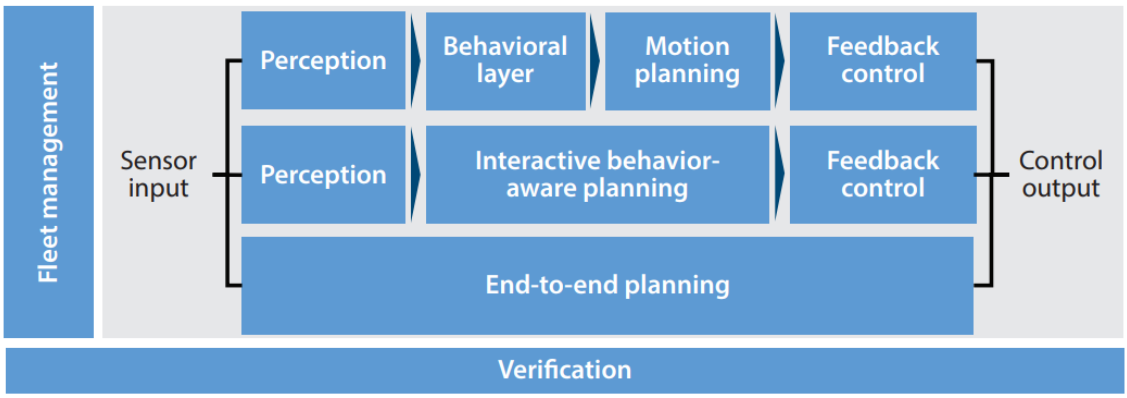


Figure 2.2: Planning and decision making for ADS [19]

The local feedback control layer focuses on real-time regulation of the vehicle's motion during automated driving. Proximity sensors, such as lidar and radar, provide immediate information about the vehicle's surroundings. Feedback control algorithms use this sensor data to adjust the vehicle's speed and direction, ensuring safe and smooth navigation through the environment. This layer plays a vital role in collision avoidance and handling unforeseen obstacles or sudden changes in traffic conditions.

In another review [19], the two layers behavioral layer and motion planning can be replaced by only one layer called interactive behavior aware planning.

2.2 Datasets and Scenarios

Developing a concrete ADS requires the use of data to test and evaluate the effectiveness of the system, as well as to train and validate in some cases like RL algorithms. There are mainly two types of datasets in this context, which are perception datasets and trajectory datasets. Perception datasets are mainly used to develop the ability of the vehicle to understand the surrounding environment. On the other hand, trajectory datasets are used by researchers who focus on developing automated driving systems from a broader perspective. As mentioned before in Section 2.1.2.1, testing these systems on regular roads requires driving through an infeasibly long road which is why some researchers focused on collecting traffic trajectory data which are used to generate scenarios based on them that can be used comprehensively for validation and testing of these systems.

2.2.1 Real-Roads Datasets

Several trajectory datasets have been recorded using different methods and focusing on different road structures. Some of these datasets were collected using infrastructure sensor technology like NGSIM [20], which means that static sensors are being installed on road infrastructure.

Some other datasets were collected using camera-equipped drones like highD [21] and exiD [22]. This section will present the approach of the highD dataset that is utilized in this thesis.

highD

Some critical papers have highlighted issues with this methodology of NGSIM [23] [24]. They showed erroneous trajectory behavior like false positive trajectory collisions and illogical vehicle speeds and accelerations which made it not very suitable for scenario-based safety validation. The highD approach used five requirements for their scenario-based suitable dataset:

- Naturalistic behavior: The behavior of the road users should be uninfluenced by the measurement, they should not be aware that the road is being recorded.
- Static scenario description: static road information should be defined based on the first three layers of the 6-layer model described earlier in Section 2.1.2.1. They include for example the number of lanes, lane width, speed limits, etc.
- Dynamic scenario description: Dynamic road information is based on the fourth layer of the 6-layer model, which describes road users' movements.
- Effort effectiveness: The ratio of measured scenarios over the combined permanent and initial effort.
- Flexibility: Measurements should cover every variance of traffic. The measurement should capture also data at every time of the day and every environmental condition.

The proposed method uses drone technology to record video data of road users, enabling the extraction of naturalistic driving behavior and detailed scenario descriptions. By utilizing computer vision algorithms and manual annotation, the highD dataset is created, encompassing 16.5 hours of measurements from six locations with 110,000 vehicles. This dataset offers valuable insights into real-world traffic dynamics, including complete lane changes and a total driven distance of 45,000 km. The highD approach compared well against NGSIM in the quantity of data, vehicle diversity, travel distance, and time. In addition to this, the authors employed a tracking algorithm in the post processing, which was used on each frame to match the objects in two consecutive frames by their distances and eliminate those infeasible to remove any false positive detections.

2.2.2 ADS Scenarios Sources

Some researchers have focused on generating scenarios to train and evaluate ADS on. Two ways are commonly used to generate scenarios:-

- **Synthetic Sets:** Scenarios are generated synthetically. These scenarios can either be generated at random or based on the challenger scenario concept mentioned earlier in Section 2.1.2.1
- **Naturalistic Sets:** These scenarios are based on real road datasets that are collected in different ways as explained in the previous section.

In this thesis, both approaches will be used to train, validate, and evaluate the designed implementation. To extract scenarios out of the mentioned earlier datasets (i.e. highD and exiD), using the mentioned framework in Section 2.1.2.2, The CommonRoad framework and scenario structure as they are very flexible and generic on all CommonRoad simulation tools. The same authors have also provided a dataset converter from different datasets which they keep updating as soon as they add more datasets to the CommonRoad XML file format as mentioned in this publication [25] in which they added a new converter from OpenScenario dataset. Their file format consists of:

- **Lanelet:** Those are the road segments that represent the road network and they are represented in the file by their left and right bound. Each Lanelet is modeled as a directed graph that has a successor, predecessor, adjacent left, and adjacent right. They also consist of traffic regulations
- **Obstacles:** They are defined by their role (static or dynamic), type (car/truck/bus/bicycle/pedestrian/constructionZone/parkedVehicle/priorityVehicle/unknown), shape (rectangular, circular or polygon) and movement over time if they belong to the dynamic object type.
- **Planning Problem:** Each ego vehicle is given an initial state as well as one or several goal regions that are considered the goal of the scenario that has to be reached.

Generating synthetic scenarios can be more challenging. As explained, scenario files include the attributes used by a simulation tool to show some aspect of the scenario. Testing the feasibility of these scenarios by manually examining the road and dynamic object attributes would be tedious. To streamline this process, the scenarios were described using the challenge description method [26]. This method employs graph search to generate human-readable parameters that describe the scenarios and help infer their difficulty. Therefore, be able to check their feasibility and compatibility with a given setup. These parameters include the minimum number of lane changes required to complete a scenario and the decision time needed for each required lane change.

2.3 Decision Making Methods

This section discusses some fundamental methodologies employed in the development of automated vehicles. The concepts and algorithms explored in this section are essential to understanding the subsequent sections of this thesis, where these techniques are applied and evaluated in the context of ADS.

2.3.1 Reinforcement Learning

RL is the third branch of Machine Learning along with supervised and unsupervised learning. RL algorithms define an agent that learns by interacting with an environment, by trial and

error, several algorithms are detailed in Sutton [27] which are summarized in this section. The decision-making process is based on the Markov decision process (MDP). Similar to unsupervised learning, RL algorithms do not depend on labeled data but rather focus on reaching a goal and building a policy to reach that goal. As shown in Figure 2.3, the agent tries to take the action at each time step that maximizes the cumulative reward, in a greedy way. Depending on the algorithm used, the agent balances the cumulative reward using a discount rate that shows how important are the upcoming rewards.

2.3.1.1 Markov Decision Process

At the core of many RL applications lies the form of MDP. MDP provide a mathematical framework to model decision-making problems in a sequential and stochastic setting. This setting allows for the development of the algorithms to find the optimal strategies for an agent [28].

Definition of MDP

An MDP is defined by a tuple (S, A, P, R), where:

- S is the state space representing all possible situations the agent can be in.
- A is the action space representing all possible actions the agent can take.
- P is the transition probability function, defining the probability of transitioning from one state to another given an action.
- R is the reward function, indicating the immediate reward the agent receives after taking action in a certain state.

The future state (next state) and reward are entailed by the current state and action. The current state and action only sufficiently capture all relevant information for predicting the next outcomes without depending on the sequence of events preceding it.

Bellman Equation

A fundamental equation in MDPs is the Bellman equation, it expresses the value of a state or a state-action pair in terms of the expected immediate reward and the future value:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \cdot \max_{a'} Q(s',a')$$

where $Q(s,a)$ is the action-value function, γ is the discount factor and the maximization is over the possible actions in the next state.

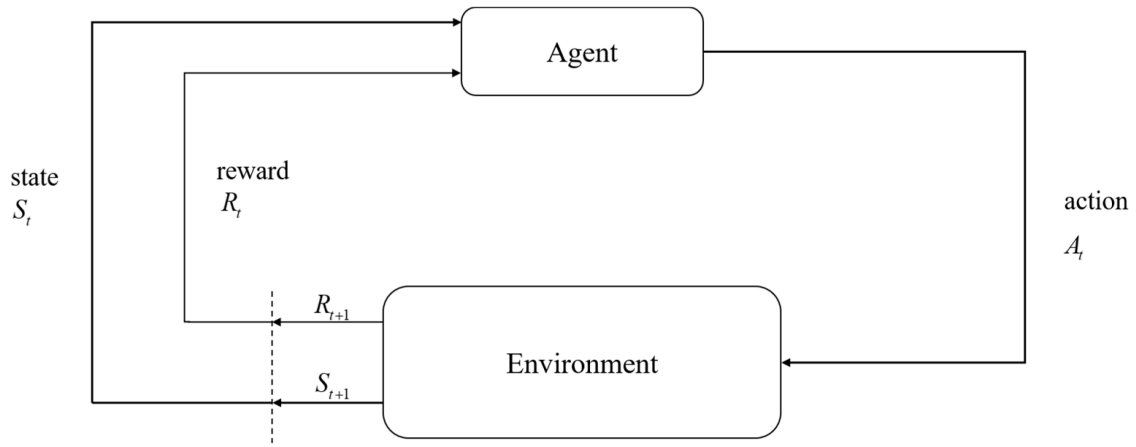


Figure 2.3: MDP Environment [27].

Exploration and Exploitation Dilemma

RL algorithms employ various strategies to balance exploration (trying new actions) and exploitation (choosing actions with known high rewards). A very common approach is Epsilon-greedy which will be explained in Section 2.3.2.1.

2.3.2 Types of RL Algorithms

RL has various algorithms that can be categorized into different approaches based on their methodologies. This section will provide an overview of the major types of RL algorithms Model-based RL algorithms build an internal model of the environment. These models are used to plan and simulate a prediction of the consequences of different actions and use them to make informed decisions. An example of these algorithms is Monte Carlo Tree Search (MCTS). On the other hand, model-free RL algorithms learn directly from interacting with the environment without having an explicit internal model. These algorithms try to approximate an optimal policy or a value function through trial-and-error while acting with the environment. Examples of these algorithms are policy-based algorithms and value-based algorithms which are explained in Sections 2.3.2.2 and 2.3.2.3 respectively. RL algorithms can also be classified into batch and online learning. Such that, batch learning trains the model on a fixed dataset, while online learning changes the model in real-time as new data becomes available. Furthermore, RL approaches can be categorized into off-policy and on-policy algorithms.

Off-Policy and On-Policy Methods

Off-policy algorithms learn using a different policy than the one being executed, this allows for more flexibility in usage of the data. On the other hand, On-policy algorithms, optimize the policy being executed during the learning process which might result in better convergence but affects data efficiency.

2.3.2.1 Epsilon-Greedy Approach

The Epsilon-Greedy approach is a fundamental exploration-exploitation algorithm commonly used in RL. This algorithm provides a way to balance the exploration of uncertain non-greedy actions and exploitation of already known greedy high-reward optimal actions in a certain state. In the Epsilon-Greedy algorithm, the agent selects the greedy action (exploitation) with probability $1 - \epsilon$ and explores a random non-greedy action with probability ϵ. The parameter ϵ (epsilon) is a small positive value between 0 and 1, representing the exploration rate. A common strategy is to start with a higher exploration rate and gradually reduce it over time, this allows the algorithm to learn first and then start using the learned actions when it's more informed about the environment.

The decision-making process can be formally described as follows:

$$\text{selected action} = \begin{cases} \text{Random policy with probability } \epsilon \\ \text{Optimal policy with probability } 1 - \epsilon \end{cases}$$

2.3.2.2 Value Based Algorithms

Value-based RL algorithms focus on estimating the value function, that represents the expected cumulative reward while being in a particular state and following a certain policy (a set of actions). These value functions are then used to derive an optimal policy that the agent can follow. the value function can be either a state-value function ($V(s)$), estimating the expected cumulative reward from a given state, or an action-value function ($Q(s, a)$), estimating the expected cumulative reward from taking action a in the state s. It uses the Bellman equation, which expresses the value of a state or state-action pair in terms of immediate rewards and the expected future value.

Temporal Difference (TD) Learning

TD learning estimates the value function ($V(s)$ or action-value function $Q(s, a)$). The key concept is the temporal difference error (δ_t), representing the difference between current and future estimated values. It uses different error functions depending on the function being estimated.

For state-value function:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

For action-value function:

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

It then updates the functions in the following way:

For state-value function:

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t$$

For action-value function:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\delta_t$$

Q-Learning

Q-learning algorithm that aims to learn the optimal action-value function ($Q(s, a)$), representing the expected cumulative reward of taking action a in state s. The core idea behind Q-learning is to iteratively update Q-values based on observed rewards and transitions. The Q-value update rule is given by:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big[R + \gamma \max_{a'} Q(s', a') - Q(s, a)\Big]$$

where $Q(s, a)$ is the Q-value for taking action a in state s, α is the learning rate, controlling the size of the updates, R is the immediate reward received after taking action a in state s, γ is the discount factor, determining the importance of future rewards, s' is the resulting state after taking action a, and $\max_{a'} Q(s', a')$ represents the maximum Q-value for the next state s'. Furthermore, it uses the exploration-exploitation trade-off through the use of an epsilon-greedy strategy which was explained earlier in this chapter.

Deep Q Network (DQN)

Deep Q Network (DQN) is an extension of Q-learning that uses deep neural networks to approximate the action-value function ($Q(s, a)$) [29]. DQN is particularly effective in handling high-dimensional state spaces, making it suitable for complex tasks and environments. The Q-network is trained to minimize the temporal difference error by comparing predicted Q-values with the target Q-values shown in the following equation:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big[R + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\Big]$$

where: θ are the parameters of the current Q-network, θ^- are the parameters of a separate target Q-network, $Q(s, a; \theta)$ represents the Q-value predicted by the current network, and $Q(s', a'; \theta^-)$ represents the target Q-value predicted by the target network.

Another important feature that DQN introduces is the concept of experience replay to address issues of correlated and non-stationary data. The agent stores experiences (s, a, r, s') in a replay buffer and then samples batches during training, breaking the correlations and improving data efficiency.

2.3.2.3 Policy Based Algorithms

Unlike value-based algorithms, Policy-based ones focus on directly learning the optimal policy, which is the mapping from states to actions. These algorithms aim to find the policy that maximizes the expected cumulative reward without estimating the value function beforehand. They create a policy and optimize its parameters to maximize the expected cumulative reward. These algorithms are better at handling high-dimensional action spaces as they do not require estimation of the value function which can be very tedious when dealing with these non-deterministic action spaces. A very important example of a policy-based algorithm is Proximal Policy Optimization (PPO). Proximal Policy Optimization (PPO) is a state-of-the-art policy optimization algorithm in RL. PPO aims to find the policy parameters that maximize the expected cumulative reward. The objective is to maximize the following surrogate objective function:

$$L(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

where: $L(\theta)$ is the surrogate objective, θ represents the policy parameters, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ is the ratio of new policy probability to old policy probability, A_t is the advantage function, ϵ is a hyperparameter controlling the size of policy updates, $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t in state s_t under the policy parameterized by θ, $\pi_{\text{old}}(a_t|s_t)$ is the probability of taking the same action under the old policy. One of the important properties of PPO is the clipped surrogate objective which prevents overshooting policy updates and thus ensures stability in the learning process. The clipped term $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ constrains the ratio within the bounds of $(1-\epsilon, 1+\epsilon)$ which is used instead of the ratio only in the surrogate objective function. The policy parameters are updated by maximizing the surrogate objective which is done by stochastic gradient descent:

$$\theta_{\text{new}} = \underset{\theta}{\text{argmax}}\, L(\theta)$$

2.3.2.4 Actor-Critic Algorithms

The actor-critic architecture is a hybrid RL algorithm that combines elements of both the worlds of policy-based (actor) and value-based (critic) methods. This architecture aims to use the strengths of both approaches to achieve more stable and efficient learning.

- **Actor** The actor is responsible for learning the policy, which is the mapping from states to actions. it focuses on finding a policy that maximizes the expected cumulative reward. In the context of policy gradient
- **Critic** The critic estimates the value function, to provide a sort of benchmark for the actor to measure the effectiveness of its selected actions. The value function represents

the expected cumulative reward for being in a state and following a certain policy. The critic's feedback helps guide the actor by evaluating the actions taken and giving a signal for reinforcement.

The actor and critic interact during training by suggesting actions (from the actor) on each state and the critic evaluates these actions by estimating their value. The difference between the estimated value and the actual reward acts as feedback to the actor so that it can change the policy accordingly. An example of an actor-critic algorithm is DDPG [30].

2.3.3 Hierarchical RL

A very common and important approach in the context of robotics and control is the use of HRL. In some complex problems, the agent has to decide on actions that take several time steps which is usually harder to learn when the agent has to do a huge amount of interaction with the environment before it can learn a reasonable policy. From this point emerged the idea of using HRL, which essentially decomposes the problem into a set of abstraction levels that limits the amount of interactions of RL agent with the environment by using a combination of previously learned smaller scope actions[31]. Hutsebaut-Buysse et al [32] has shown different mechanisms to handle HRL:

Temporal Abstractions

Temporal abstraction is one of the important mechanisms of HRL. As explained earlier, in some complex problems the decision can take several action steps that may require a learning process as well. In this mechanism, the temporally extended action (sub-behavior) is a sequence of primitive actions or other temporal extended actions that can be used by the learning agent to decide on a higher level of abstraction.

State Abstractions

Another important mechanism that has to be used alongside the temporal abstractions is the state abstraction. In order to facilitate the training process, the state space has to be simplified to allow the agent to reason only for high-level decisions. This is done by grouping similar states which can be defined by using Bisimulation for example. The bisimulation concept groups biosimilar states if they have the same probability distributions over future states and actions. Using those two mechanisms can help the agent avoid RL challenges such as credit assignment.

Credit assignment

It's the problem of defining which action had an impact on which reward. Using temporal and state abstractions can help make this task less complex by freeing the agent from having to reason about which action to take on every individual step. The agent can just use temporal abstractions which can run for several steps until the option it chose is applied.

3 Related Work On Automated Vehicles

ADS have made a swift shift in the field of transportation safety, efficiency, and convenience. This made many researchers focus on developing the ADS's minds or decision-making component, resulting in countless researches that went in several different directions trying to achieve the best possible system. These researches used different approaches that include but are not limited to imitation learning, reinforcement learning, and knowledge-based or rule-based ones. Along with these algorithms, some researchers focused on collecting datasets for testing of those systems. This chapter summarizes the most relevant approaches that align with this topic.

3.1 ADS Decision Making Approaches

The unique challenges posed by real-world scenarios demand sophisticated approaches to handle diverse situations. This section delves into decision-making methodologies for automated vehicles, focusing on strategies employed to navigate complex traffic scenarios.

3.1.1 Rule-based Approaches

Defined in [33], a rule is a description of the desired behavior for autonomous vehicles. It can be defined by different aspects such as traffic laws or common sense. One of the very simple and naive rules that is often used in the ADS context, is the speed limit, in which the automated driving vehicle is forced to follow the speed limit or recommended speed. Other rules might be more complex that focus on collision avoidance or adhering to other traffic laws like driving on the right side of the road or following the center of the lane when the vehicle is not switching lanes. A rule-based approach that focuses on both collision avoidance and adhering to traffic laws was done in previous work at the Institute for Automotive Engineering at the RWTH Aachen University that inspired several ideas and approaches done in this thesis is called Highway Pilot (HP). The HP represents a rule-based approach for simulating a simplified automated driving function of level 4. The system has both longitudinal and lateral control mechanisms. In terms of longitudinal control, the HP integrates an Adaptive Cruise Control (ACC) and an Automated Emergency Braking (AEB). ACC system's role is to maintain a constant reference velocity, initially set to the speed limit of the road. However, in the presence of a lead vehicle, the HP adjusts its speed to maintain a minimum safety distance from the leading vehicle. Lateral control comprises Lane Keep Assistance (LKA) and Lane Change Assistance (LCA). LKA ensures the vehicle remains centered in the lane, even during

curves. The LCA, however, makes decisions about lane changes, disallowing them if AEB mode is active. The Model Predictive Control (MPC) utilizes ACC, LKA and LCA parameters to optimize inputs. The AEB engages emergency braking in response to harsh braking by a preceding vehicle. Lane changes are executed based on conditions such as maintaining a safety distance or reaching the reference velocity. The rule-based approach involves setting reference trajectories and orientations for MPC, ensuring the HP follows a desired path while adapting to dynamic driving scenarios.

3.1.2 Planning-based Approaches

Planning-based approaches in the context of Automated Driving involve the algorithm focusing on creating a trajectory or a planned maneuver to navigate the vehicle from the current location to a desired destination based on a planning algorithm like search algorithms that look for the optimal way to minimize the cost depending on the problem's goal. One of the first planning-based approaches in the AD context is the attempt of [34] that used A* search to design a planning algorithm that looks for the shortest time trajectory while still handling dynamic objects. Another recent approach involved using Minimax planning algorithm to try to find the safest trajectory to navigate the vehicle to the desired destination [35]. It focused on maximizing the safety conditions while assuming that all the other vehicles in the environment are acting against it. The authors also assumed that if two trajectories are equally safe, the algorithm in this case would prefer the longer traveling distance trajectory to the shorter one. The attempt treats the problem as a two-player game for safety between the ego vehicle and surrounding vehicles.

3.1.3 Imitation Learning

Some attempts focused on using imitation learning models, in which the model aims to mimic human behavior for a given task including Automated driving [36]. Imitation learning has several approaches and designs, the most commonly used one is Behavioral Cloning (BC), in which the model focuses on minimizing the difference between the model's prediction and the expert action in the same states. The utilization of imitation learning methods in the AD context has been going on for a while dating back to the late 80s when the famous ALVINN [37]. This attempt was based on BC and a large amount of human driving experience to derive an end-to-end approach from the vehicle's sensor inputs to the controller directly. The authors used an Artificial Neural Network to with backpropagation learn the behavior, which is similar to solving a regular labeled problem. As most of the labeled machine learning problems, they faced the problem of scarcity of labeled data in different numerous conditions on the road. Symphony is another more recent approach introduced by this research [38]. Instead of just using BC, they used the approach of Generative Adversarial Imitation Learning (GAIL). This approach is inspired by a Generative Adversarial Network (GAN) which employs

a discriminator in the process of learning. This approach was used to solve the covariate shift issue, in which the model is unable to generalize well during the testing phase if the data encountered is different than that of the training. GAIL solves this by repeatedly trying out the learned policies on the environment and trying to minimize the divergence between the predicted trajectories and the expert demonstrations. BC usually has this issue since it uses off-line learning, meaning that the learned policy never encounters an actual environment until the testing phase. In addition to this, the authors used a hierarchical policy to avoid mode collapse, which typically happens when generative models converge to a limited set of modes and fail to capture the diversity of the data. A high-level goal-generating policy is specified depending on the initial state which is fixed during the rollouts and is provided to low-level goal-conditional policy and the discriminator. The goal-generating policy is trained to match the distribution of goals in the training data. They used the following three realism metrics that measure how well the learned policies match the distribution of real-world behaviors. It includes the collision rate, the average displacement error, and the off-road time. In addition to this, they used the diversity metrics that evaluate how well the learned policies cover the entire distribution of realistic behavior. They used minimum state-action distance error (MinSADE) and the curvature Jensen-Shannon divergence (JSD). Another type of metrics used is the novelty metrics which evaluate how well the learned policies can handle novel scenarios.

3.1.4 Reinforcement Learning

On the other hand, Other researchers focused on RL as it is commonly used in robotics and automated systems. Some chose to mix and match the methods to find an optimal solution that combines two or three of the mentioned methods in Sections 3.1.1 and 3.1.3. An example of an approach that used Reinforcement Learning combined with imitation learning would be the approach of [39]. In this research, the authors used an approach called BC-SAC (Behavioral Cloning - Soft Actor-Critic) which is a combination of imitation learning (IL) and reinforcement learning (RL) to enhance the safety and reliability of driving policies, particularly in challenging scenarios. The BC-SAC approach involves a dual actor-critic architecture similar to Twin Delayed Deep Deterministic Policy Gradients (TD3) and Soft Actor-Critic (SAC). This architecture consists of an actor-network $\pi(a|s)$, a double Q-critic network Q(s, a), and a target double Q-critic network $\overline{Q}(s, a)$. Each network is equipped with a separate Transformer observation encoder that encodes features such as vehicle states, road-graph points, traffic light signals, and route goals. The actor network outputs a tanh-squashed diagonal Gaussian distribution parameterized by a mean μ and variance σ. The training data used in the study is crucial, as the performance of learning-based methods heavily depends on the training data distribution, especially in safety-critical settings with long-tail distributions. The authors explore how the training distribution affects the method's performance and demonstrate that training on more difficult examples results in better performance than using unbiased training distributions. The study utilizes a dataset consisting of over 100k miles of expert driving trajectories, split into

training and testing segments. These trajectories contain features describing the autonomous vehicle (AV) state and the state of the environment as measured by the AV's perception system. The difficulty model described in the paper is used as a proxy for measuring the rarity of events, as challenging scenarios are generally less frequent. The authors demonstrate that while imitation learning can perform well in low-difficulty scenarios that are well-covered by the demonstration data, their proposed approach significantly improves robustness in the most challenging scenarios, leading to over a 38% reduction in failures. This highlights the effectiveness of the combined IL and RL approach in addressing the limitations of pure IL methods and improving the system's performance in challenging driving scenarios. Some researchers focused on a different problem that faces automated driving decision-making, they found out that terminating the problem very soon due to wrong actions like going out off the road or a car crash would lead to drastic safety issues, as well as limiting the exploration of the learner, due to the early termination of the simulation. Their approach was to use shielding [40], which they use to prevent the decision maker from taking wrong actions like these and at the same time give some kind of a penalty to allow the model to learn from this mistake. Using these safety precautions with RL makes it a safe reinforcement learning approach. Another attempt regarding safety is SafetyNet [41], where the authors employed an ML planner that is trained on 300 hours of expert driving demonstrations to deal with safety issues during driving and try to avoid collisions. However, using an ML planner alone has proved itself to be less safe than the rule-based approach but easier to implement and covers more situations. The authors introduced a method called fallback, where they used the ML planner to generate the first trajectory and in case it fails or has a potential risk of having a collision (overlapping the same location of another vehicle), the fallback trajectory which is a rule-based trajectory that is designed to avoid collisions takes place instead of the original plan. This attempt has proven itself to be better than the initial ML planner-only based approach and reduced the collision rate by 95%. This proved that using rule-based approaches together with automated ones like Imitation or Reinforcement Learning can in some ways help improve it to avoid safety-threat situations.

3.2 Overview of RL Environment

Since this thesis's scope is experimenting with RL in the context of ADS. This section will examine the current approaches done using Reinforcement Learning, starting with the setup of the environment and showing how different algorithms perform on different tasks. Several survey papers address this topic which are examined carefully. Aradi et al[4], compares different attempts done in the scope of ADS. The authors first define several scenarios that usually research focuses on. These scenarios are following a vehicle, lane keeping, merging, and driving in traffic. Depending on the task the research is interested in, the environment observation and action space may differ.

Car Following

The task of the vehicle in this case is to maintain a safe distance from the leading vehicle. The observation space of this task includes (v, dv, ds) tuple representing agent speed, speed difference to the lead, and headway distance. The action space in this case is only acceleration, while the reward is based on jerk, Time To Collision (TTC), or passenger comfort. Naturally, collisions are considered a failure.

Lane Keeping

This scenario focuses on lateral control, where it uses either the ”Ground truth” lateral position and angle or front-facing camera image. The reward in this case is given depending on how close the car is to the center line of the lane.

Merging

This scenario tries to find a gap from which to enter the highway. It uses both longitudinal and lateral control as it needs to control both steering and acceleration. Some other variations of this might include multiagent scenarios.

Driving in Traffic

Most complex scenario with scalable factors (network topology, surrounding vehicles, traffic rules). Two types: hierarchical approaches (behavioral layer decisions with classic control) and end-to-end solutions (direct control of steering and acceleration). Behavioral layer decisions include lane-keeping, changing lanes left or right, and overtaking.

Reviewing the attempts presented in Aradi et al[4]. As mentioned in Section 3.2 depending on the task or scenario being executed, the observation and action spaces vary as well as the RL algorithm used for the agent. Focusing on the following, lane keeping and lane change maneuvers are the most relevant to this thesis. In the case of following the observation space of the attempt [42] [43] focused on the velocity, distance to the leading vehicle, and relative distance, on the other hand, it used the acceleration only as the action space. The RL agent used was DDPG and DNN in this case. Other attempts that focused on Lane Keeping were [44] [45] where they focused on using the vehicle’s perspective as the observation space, with one of them using a Lidar-like tracking sensor and the other using the front camera. The action space in this case varied as one used both acceleration and steering speed/velocity, while the other used only the steering. The algorithms used varied between PPO, CNN, DNN, and DQN. Lastly, some other attempts [46] [47] [48] focused on Lane change maneuvering in which their observation spaces included the lateral and longitudinal positions as well as the velocity and the heading angle of the vehicle. One of the attempts included the acceleration, the id of the lane, the width, and the curvature as extra components of the observation. Furthermore, some included, steering, acceleration, yaw rate, and throttle as their action space while the algorithms used were TRPO, RARL, NFSP, DNN, DDPQ, and DQN. There are also other worth

mentioning attempts that were not mentioned in Aradi et al[4]. One of them is, the attempt of [49], in which the authors built an RL algorithm that made use of a rule-based approach to prevent the agent from making safety-threatening decisions. The attempt used the observation space of (p, x, y, v_x, v_y) such that, p is a binary identifier of the observability of the ego vehicle, (x, y) tuple is the global position and (v_x, v_y) tuple is the global velocity. The authors have used a discrete action space with 5 actions that are turn left, turn right, keep lane, faster, and slower. They have also used the DQN-attention RL algorithm with the safety measure of preventing the agent from being too close to the leading vehicle by using the inequality of safety:

$$d - t_{\min} * v > d_{\min}$$

such that d, v are the relative distance and velocity between two vehicles, t_{min} is the minimum TTC and d_{min} is the minimum distance before the action is executed. They used this inequality to decelerate in case the current TTC is less than the minimum TTC or maintain the acceleration otherwise. Another approach is the approach of ASAP-RL [50], the approach leverages expert knowledge to pre-train an actor and critic models to help increase the efficiency of learning the model. This approach used an observation space that consists of a 5-channel birds-eye-view (BEV) image with spatiotemporal information of the ego agent and surrounding agents, as well as information about the road geometry and navigation. ASAP-RL first parameterizes motion skills that are diverse enough to cover various complex driving scenarios and situations. The RL agent then learns from the set of skill parameters instead of the control space, which enables more informative exploration.

4 Problem Statement

Machine Learning systems such as RL have proven themselves to outperform many rule-based approaches in the context of ADS. However, there is a challenge that these systems could execute unsafe actions or have slow learning rates [51]. This thesis aims to introduce a decision-maker for the ADS that can demonstrate efficiency in learning and safety which can be proven with guaranteed generalizability of learning. This works as a step towards building an ADS that can be as safe as rule-based systems. In this chapter, the requirements of the decision-maker for the ADS will be defined in detail as well as the concrete questions that this thesis will try to answer.

4.1 Requirements

In this section, the main requirements that the decision-maker has to follow will be concretely defined as well as a comparison between the state-of-the-art algorithms regarding following those requirements. The requirements are:

R1 Safety: The developed system must ensure the safety of the vehicle by effectively taking measures to avoid traffic accidents and any situations that threaten vehicle safety.

R2 Sample Efficiency: The developed system must effectively take measures to address sample efficiency, enabling the system to learn to drive safely within a short training duration.

R3 Generalizability: The developed system must acquire general driving skills that are applicable across various conditions, avoiding overfitting to specific scenarios.

For the safety requirement (R1), the decision-maker of the ADS in this thesis' case, the RL algorithm has to have a comparable performance to other algorithms. The approach also has to take specific measures to address safety. The requirement (R1) can have numerous methods to measure the performance. Still, they have to be related to safety, such as collision rate, maintaining vehicle safe distance, or the rate of going off the road.

For the second requirement (R2), the rate of learning of the developed decision-maker of the ADS in this thesis has to be measured and compared to other algorithms and outperform their rate of learning.

The third requirement (R3) compares the generalizability of learning of the method. This is the ability of the model to learn driving skills and prohibited actions rather than overfitting the given scenarios. This is measured by masking or keeping away some of the scenarios from the

model during training and using them only during testing to make sure the model has learned the skill itself.

4.2 Research Gap

Other attempts that focus on the same context have already been introduced in Section 3.1.4. However, the main focus is to assess whether the approach is abiding by the requirements mentioned in Section 4.1. The attempts mentioned have different ways of following the requirements set for this thesis which is presented in Table 4.1.

Requirements	DQN-attention [49]	ASAP-RL [50]	BC-SAC [39]
R1 Safety	+	+	+
R2 Efficiency	+	+	-
R3 Generalizability	-	-	+

Table 4.1: Satisfaction of Requirements by Papers

the attempt of [49] used a combination of DQN algorithm with attention mechanism and rule-based approaches, it used safety measures like shielding as well as measuring the safety of the model by using collision rate metrics and in that way, it followed the requirement (R1). For the efficiency measures it used HRL and did ablation studies to compare the performance and in this way, it followed the requirement (R2). However, it did not use any guarantee for the generalizability of learning of the agent and in this way, it did not follow the requirement (R3)

The attempt of ASAP-RL [50] focused on comparing the model's efficiency of learning to other baseline models as well as using HRL to improve the efficiency of learning. The authors have used also safety metrics (like collision rate) but always compared them to the training iterations. In this way, ASAP-RL satisfies requirements (R1) and (R2) by outperforming the baseline models in both safety and efficiency of learning. However, it did not focus on the generalizability of learning of the model and in that way, it did not satisfy the requirement (R3).

The attempt of Behavioral Cloning - Soft Actor-Critic [39] used metrics of safety calling it failure rate that included collision, off-road, and overall against the collision weight which does not explicitly show the efficiency of learning, and in this way, it satisfies R1 but not R2. In this approach, the authors explicitly mentioned avoiding train-test leakage by splitting training and testing data as well as partitioning similar data. In this way, this approach satisfies the requirement (R3) as well as (R1). However, it did not take any measures to improve the efficiency of learning and therefore it did not follow the requirement (R2).

Going over these attempts has shown several ways to measure each of the requirements which can help in designing the evaluation part later in this thesis. In addition, it has shown a diverse set of observation spaces and action spaces that can be used in the implementation part later as

well as demonstrating the use of selected scenarios for training as in the BC-SAC [39] attempt. However, none of these attempts were able to satisfy all the requirements according to the information provided in the papers. This creates a research gap for this thesis to try and fill by attempting to satisfy all of the requirements or by answering potential questions that lead to the satisfaction of these requirements.

4.3 Addressed Questions

The questions being addressed in this thesis align with the requirements presented in Section 4.1. This thesis aims to develop steps towards building a reliable ADS that can be deployable as rule-based systems. In order to reach this goal, smaller sub-questions that represent steps toward the broader aim have to be answered. This thesis focuses on two of those questions that address the quality of an RL algorithm in safety, efficiency, and generalizability.

RQ1: How does HRL impact efficiency and generalizability of learning of the agent?

HRL focuses on the idea of using abstraction levels in terms of state representation and low-level policies to improve the learning efficiency of the agent. The point of the question is to test the ability of HRL to improve the efficiency and generalizability of learning as it can help with the credit assignment issue as mentioned in Section 2.3.3 as well as reducing the complexity of the environment which can facilitate the use of shielding that can help with the safety aspect. Therefore, this question aligns with the requirements (R1), (R2), and (R3).

RQ2: How can the use of challenging scenarios improve the efficiency of learning of the agent?

The second question handles the efficiency of learning as challenging scenarios are expected to help with the scarcity of safety-threatening situations on real-roads as mentioned in Section 2.1.2. In addition, using scenarios can help guarantee testing on unseen scenarios during the training of the agent which can help in the validation process of the generalizability of learning. The main point of this step is to question the effectiveness of using a set of scenarios, that are either actively selected or randomly selected, which are shown during the training of the model to help make the RL algorithm a faster learner and efficient in handling these challenging driving situations. This research question lies within the scope of the requirements (R2) and (R3).

5 Method

As explained in Chapter 4, the requirements and the questions focus on the methods that can help the agent achieve both safer and more efficient learning of driving skills. These ideas are summed up in points like utilizing challenging scenarios to help speed up the learning and validation process of the agent as well as using an engaging simulation that tries to mimic the process in a real-life driving environment. The use of a hierarchical policy to reduce the problem into manageable decisions that can help the agent focus on the decisions and not on how to implement them.

5.1 Simulation

The simulation of the environment that the model will learn is quite an essential point that many researchers have focused on building. Popular simulations like Carla and Sumo are very excessively used in research, however, they usually include many aspects and layers that are not required for the purpose of this thesis. For instance, CARLA provides a wide array of simulated sensors which is not required for this thesis scope. SUMO, on the other hand, provides traffic simulation which is also not relevant for this thesis. Therefore, the decision was to use a commonroad-based toolchain simulation that was developed at Institute for Automotive Engineering (IKA) previously. The simulation utilized the CommonRoad vehicle model to build an environment that presents the process of the scenario as well as visually rendering it. The simulation is able to represent a scenario using an XML scenario file as well as a JSON file as long as it includes all the necessary attributes needed to build a scenario. Visually as shown in Figure 5.1, the simulation shows the road lanes as well as the ego vehicle and dynamic objects in a rectangular shape with blue and green colors respectively. In addition, the label of the ego vehicle as well as the ID of the other vehicle are shown next to the current velocity of the vehicle in kilometers per hour. The goal position is also shown as a red line vertically across the street. The simulation creates the vehicles and positions them according to the time step position in the scenario. In addition, it allows the user to specify the actions of this vehicle by specifying a pilot that is responsible for making the decisions for each dynamic vehicle such as HP. Typically this scenario is being used with a no-action pilot for all the dynamic vehicles except the ego vehicle, making this simulation an open loop one as all the dynamic objects but the ego vehicle follow predefined trajectories. The observation space of this simulation includes the time step, the road map, and the dynamic objects that contain the parameters for each dynamic object which can be seen in Figure 5.2. The vehicle's s_x, s_y that, represents the position of the vehicle at the timestep. It also has the vehicle's velocity v, length

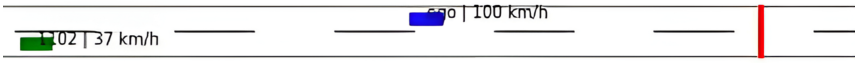


Figure 5.1: Simulation Snapshot

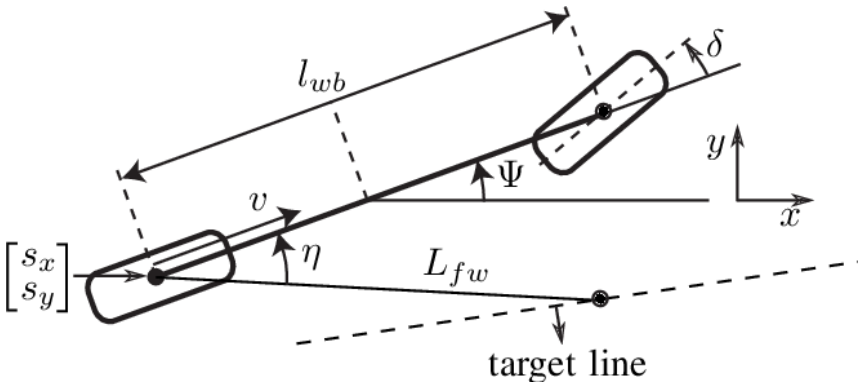


Figure 5.2: Kinematic Single Track model [52]

of the wheelbase l_{wb}, the steering angle δ and the heading angle ψ of the vehicle which is the angle between the vehicle's longitudinal axis and a reference axis (commonly the x-axis in an external coordinate system). The action space that was used in the simulation included the steering velocity and the acceleration. The simulation terminates for the 5 following reasons:

- **Goal Reached** The simulation terminates when the ego vehicle has reached the marked goal position.
- **Collision** The simulation terminates when the ego vehicle collides with another dynamic object.
- **Time Out** The simulation terminates if it exceeds the given time for the scenario (provided in the scenario file).
- **Standstill** The simulation terminates if the ego vehicle reaches a standstill state or a velocity of zero km/h.
- **Offroad** The simulation terminates if the ego vehicle goes outside the bounded road.

5.2 Scenarios

As explained in Section 2.2.2, scenarios used in this thesis are divided into two main types. Synthetically generated scenarios and naturalistic/real-roads extracted scenarios. In this section, the method used to generate synthetic scenarios as well as the one used to extract real-road scenarios will be explained.

5.2.1 Synthetically Generated Scenarios

Synthetically generated scenarios are the challenging scenarios that are hand-engineered (i.e. not based on real recorded sections of a road) as well as following a certain scheme that is based on challenger scenario concepts. The main point of these scenarios is to put the ego vehicle in a situation where a collision is imminent if the ego vehicle fails to get itself out of the situation or remains idle. In this thesis, three scenario types were generated based on the UN Regulations No. 157 for automated lane keeping systems critical scenarios [53]:

Type A: The concept of scenario type A is shown in Figure 5.3. The concept behind this type is spawning the ego vehicle in an initial lane and spawning another challenger vehicle in the same lane, preceding the ego vehicle which is decelerating to cause a collision with the ego vehicle, if it ceases to do any action before it reaches the challenger vehicle.

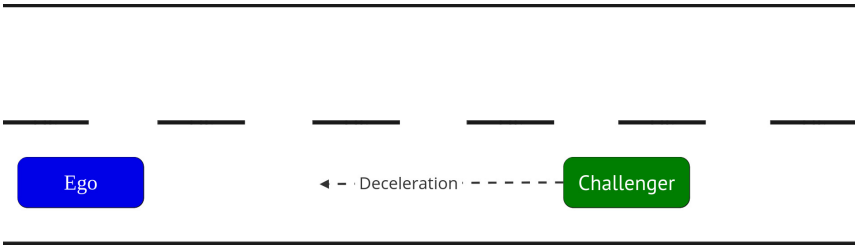


Figure 5.3: Scenario Type A Concept

Type B The concept of scenario type B is shown in Figure 5.4. The ego vehicle is spawned in one lane, and the challenger vehicle is spawned in the left adjacent lane, which then performs a cut-in maneuver to the right lane in front of the ego vehicle and starts decelerating to put the ego vehicle in a challenging situation. This type is considered more challenging for the ego vehicle than type A as the ego vehicle could collide with the challenger vehicle if it makes an early lane change. On the other hand, in type A, the ego vehicle can do a lane change to the left at any time before colliding with the challenger vehicle and still be able to solve the scenario.

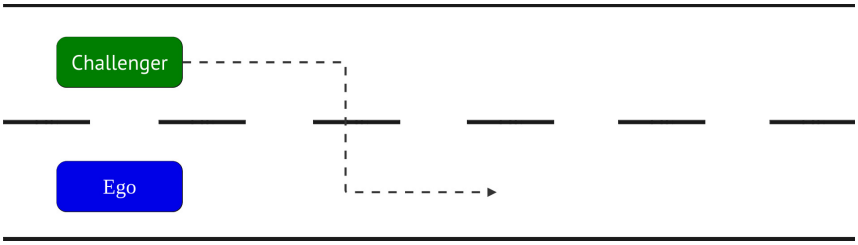


Figure 5.4: Scenario Type B Concept

Type Cutout The concept of scenario type Cutout is shown in Figure 5.5. In these scenarios, two vehicles are spawned in the same lane as the ego vehicle. The vehicle in the front decelerates while the vehicle in the middle between the ego vehicle and the front vehicle does a cut-out maneuver to the left adjacent lane and then decelerates too. This type is considered the most challenging among the three scenarios since it requires the ego vehicle to perform two lane

changes to the left. First, to avoid the challenger vehicle in the very front that decelerates and the second is to avoid the challenger vehicle that does a cut-out maneuver in the left lane. The ego vehicle can decide to do a brake but it then risks termination due to timeout as it might fail to reach the goal before the end of the simulation time. It might also risk termination due to being in a standstill position as it would have to decelerate to zero km/h to avoid colliding with the constantly decelerating challenger vehicle.

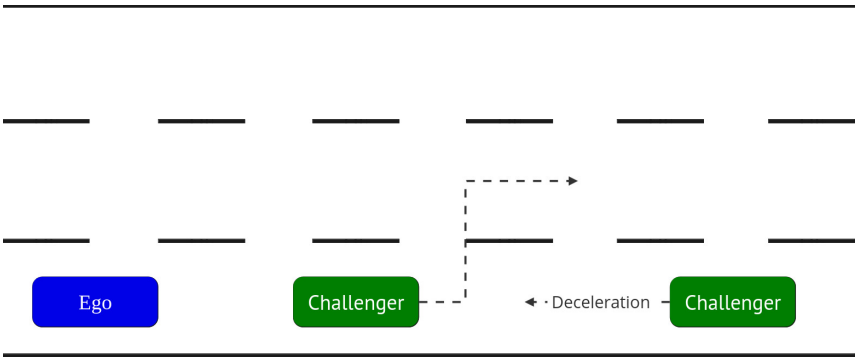


Figure 5.5: Scenario Type Cutout Concept

Before the generation of each scenario, it had to go through the feasibility checker function from the commonoad-drivability-checker library [54]. The first check was the trajectory feasibility which checked if the trajectory for each vehicle model is feasible for every consecutive state of the trajectory. The second check was the collision check between the dynamic obstacles of the scenario. This check was done to avoid having scenarios that have a collision between its dynamic obstacles, as this can not be the reason for terminating the scenario and at the same time would look very unrealistic. Therefore, the scenarios that failed to comply with one or both of the feasibility checks were not considered for generation from the beginning. During the generation of feasible scenarios, each scenario was named following a specific pattern to ensure average consistency and clarity. The naming convention adopted is as follows:

$$\text{scenario\_\{type\}\_v\_\{number\_of\_extra\_vehicles\}\_\{number\_of\_scenario\}} \tag{5.1}$$

where:

- **{type}**: Indicates the type of the scenario.
- **{number_of_extra_vehicles}**: Represents the number of extra vehicles present in the scenario. These vehicles have nothing to do with the concept of the scenario but are only present to make the scenario more complex.
- **{number_of_scenario}**: Denotes the index of the scenario.

For example, the name *scenario_a_v_3_1* refers to a scenario of type A involving three other vehicles. This systematic naming approach facilitates easy identification and categorization of different scenarios used for training.

5.2.2 Real Road Extracted Scenarios

Another source of scenarios that are used in this thesis, is the real-road extracted scenarios. These scenarios are essentially extracted from recorded roads with different methods as explained in Section 2.2. The dataset used here in this thesis is the highd dataset. The dataset converter implemented by CommonRoad [25] was used to extract XML scenarios that can then be used for training and validation on the simulator Section 5.1 used in this thesis. The idea behind using these scenarios is to provide a real factor to the validating scenarios and at the same time demonstrate the difference in the efficiency of learning between the challenging scenarios and the real road scenarios. In addition, these scenarios can be used to demonstrate the generalizability of learning as these scenarios were not seen during the training of the agent on challenging scenarios.

5.3 Hierarchical policy

As shown in Section 5.1, the simulation takes an input of steering velocity and acceleration which can directly be used as an action space in the RL environment. This aligns with the control layer in the layers of decision-making explained in Section 2.1.3. As explained in Section 2.3.3, in problems similar to the one addressed in this thesis one can use HRL to facilitate challenges such as credit assignment explained in Section 2.3.3. In the context of this thesis, the agent will hardly be able to attribute the reward to the set of primitive actions due to the continuous nature of the action space. This variability complicates the direct mapping of rewards to specific actions, making HRL an advantageous approach to handle such complexities. Therefore, applying abstraction mechanisms was the choice in this approach. The idea is to choose the high-level action (option) of changing lanes, and the temporally extended action (sub-behavior) is implemented by SAD-Pilot which is designed to choose the appropriate input at each time step to pass to the simulator to ensure a smooth lane change. This is following the temporal abstractions approach explained in Section 2.3.3. The actions are designed to cover both lateral and longitudinal decisions. Abiding by the state abstraction mechanism, the lateral action space is grouped to be only choosing an available lane to switch to or keep the current lane that is provided by the model accounting for a set of 3 actions:

- Switch to the right lane a_{right}
- Switch to the left lane a_{left}
- Keep the current lane a_{center}

During the lane switch process, the agent is prevented from making decisions and the SAD-Pilot takes over handling the input of the simulation until the change is done. Afterward, the decision capability is available again to the agent. In the case of longitudinal action space dimension, the agent is allowed to choose from 4 distinct actions:

- Accelerate $a_{accelerate}$
- Decelerate/brake a_{brake}
- Maintain velocity $a_{maintain}$
- Hard brake $a_{hard\ brake}$

such that accelerate increases the acceleration by a specific amount for the upcoming steps, and decelerate does the exact opposite by reducing the acceleration by a certain amount for the next steps. To provide more safety in risky cases, a hard brake is provided for the model to use to slow down the vehicle drastically to avoid a collision or any risky situation. The fourth action is the no action or maintain which keeps the velocity as is. In each step, the agent is allowed to choose a tuple of lateral and longitudinal actions. Therefore, the available actions for the agent in each step are n-distinct actions depending on the combination of action tuples that will be used in the implementation.

5.4 Scenario Selection

Filtering the scenarios according to feasibility and compatibility with our system is an important step of the method. As described in Section 2.2.2, the challenge description method provided parameters that can be used for filtering scenarios. The minimum required lane changes parameter *n_lc* and the decision times required to do each lane change *decision_times* were used in the filtering process. The method provided a *n_lc* of null when the scenario was not feasible. In addition, for the scenario to be compatible with the system, it had to take into consideration the inability of the agent to do consecutive lane changes in a very short decision time. This issue happens as the way SAD-Pilot handles the lane change on the simulator avoids doing 2 consecutive lane changes before a small waiting time. As shown in Algorithm 1, the scenarios were dropped if the decision time needed starting from the second lane change was greater than 6.5 seconds.

Algorithm 1 Filtering scenarios based on lane change criteria

```
function IS_FEASIBLE(n_lc, decision_times)
    if n_lc is None then
        return False
    if n_lc > 1 then
        for t in decision_times[1 :] do
            if t is None or t ≤ 6.5 then
                return False
        return True
    else if n_lc == 1 or n_lc == 0 then
        return True
    else
        return False
```

5.5 Shielding

As explained earlier in Section 3.1.4 preventing agents from making unnecessary mistakes like going off the road which can hinder the learning process by limiting the exploration of the agent as the simulation will terminate during the learning process. It can also be very dangerous if done in a real-life scenario. Preventing these mistakes can be easily done using manually engineered rules that can prevent the agent from choosing such actions and from making decisions that can lead to a safety-threatening situation. In this thesis, the shielding that will be used focuses on two aspects. The first is to prevent the agent from changing to an unavailable lane such as being on the very right lane and trying to go to the right lane or going to the left lane when the vehicle is on the very left lane. This can be achieved by using the information the simulation provides, as it shows on every step of the current lanelet map which contains the available lanes that can be utilized to make this decision. The other point at which the shielding will be applied is to prevent the agent from being too close to the leading vehicle as shown in Section 3.2. One of these ways is the inequality of safety which is mentioned in the same section. In this thesis, a similar approach will be applied to maintain a safe distance between the ego vehicle and the leading one. This will be done by calculating the TTC and the longitudinal action will be applied by the shield using this condition:

$$a_{\text{long}} = \begin{cases} -a_{\max} : & \text{Brake}, & \text{if } TTC \leq TTC_{\min}, \\ \pi(s) : & \text{Agent's choice}, & \text{otherwise.} \end{cases}$$

The action will be applied to the longitudinal dimension regardless of the choice of the model at this step to brake in the case that TTC is less than the minimum TTC and apply the agent's choice otherwise.

6 Implementation

In this chapter, the explicit implementation of the method explained in Chapter 5 will be elaborated in detail. This chapter will focus on how precisely were the synthetic scenarios used for training generated, how were they stored, and the specific implementation of the environment and how is it used as well as the hierarchical policy. Furthermore, it will explain the validation pipeline that is used to show the training progress throughout the training phases.

6.1 Scenarios Generation

One of the essential implementations done in this thesis is the generation of synthetic scenarios that are based on the challenger scenario concept Section 5.4. As explained in Chapter 5, there are 3 types of challenger scenario concepts used in the generation of the scenarios. Type A, B, and Cutout. The generation used a set of parameters required for each type that describe what happens in the scenario. The permutations between these parameters' range of values are computed to generate scenarios that cover all the possible combinations of parameter values.

6.1.1 Scenarios Parameters

As shown in Table 6.1 some of the parameters are common across all scenario types such as the number of lanes, curved road option, the goal position, s_0, t_0 and v_0. On the other hand, some parameters have to be explained for each type on its own:

Type A The challenger vehicle as explained earlier is spawned in front of the ego vehicle and then decelerates. The required parameters for the challenger vehicle are the time headway from the ego vehicle, the initial acceleration a_0 and the acceleration delay a_{delay}.

Type B In addition to the parameters in type A, there are 3 more parameters that are related to the lane change maneuver of the challenger vehicle, that are: the lane change direction $lc_{direction}$, the lane change delay lc_{delay} and the lane change duration $lc_{duration}$.

Type Cutout This type used the same parameters as type B but had 2 challenger vehicles, one of which used the lane change parameters to do a lane change to the left adjacent lane. In the meantime, the other challenger vehicle used the deceleration parameters to decelerate in the same lane.

In addition, each permutation of all parameters was given another parameter for extra vehicles that are initialized in the scenario to add variety and a sense of difficulty to the scenarios.

These vehicles use the scheme of the challenger vehicle of type A with random parameters for deceleration. Generally, the scenarios generated in this thesis have a duration for the scenario to keep running and cannot exceed. In this thesis, the scenarios are given a duration of 20 seconds.

Parameter	**Description**
Number of lanes	Choosing the number of lanes that will be used in this scenario.
Curved road option	Whether or not the road is curved.
Goal position	Choosing the goal position from the given options.
s_0	Initial longitudinal position.
t_0	Initial lateral offset to the center line.
v_0	Initial speed.
Time headway	Time gap from the ego vehicle for the challenger vehicle to spawn.
a_0	Initial acceleration of the challenger vehicle.
a_{delay}	Time to wait before applying acceleration to the challenger vehicle.
$lc_{direction}$	Direction of lane change (left or right).
lc_{delay}	Time to wait before executing the lane change.
$lc_{duration}$	Duration of the lane change maneuver.

Table 6.1: Parameters for Scenario Generation

6.1.2 Scenarios Storage

As explained in Section 5.1, the simulator is able to compile CommonRoad XML or JSON scenario files. In this thesis, The scenarios were generated as a JSON file representing each scenario by its ID and its attributes After the generation of these scenarios, they were stored in a NoSQL MongoDB database, such that each scenario was saved with its ID, and the JSON file that represents it. The total number of generated and saved scenarios is 577k scenario. The collection was indexed by scenario IDs to facilitate the importation during training by fast retrieval. The IDs of the scenarios needed for training are picked from the pool of scenarios and split into training and action space batches. In the implementation of this thesis, the training and testing splits were divided into 100 batches, in order to use 100 parallel environments to train the agent. During training, the environment is initialized with a training batch, which contains the names of the scenarios. The environment then preloads all of the needed scenarios for the batch in a dictionary, to have fewer I/O operations in the middle of training which increases the throughput of training iterations such that more batches can be processed per unit of time.

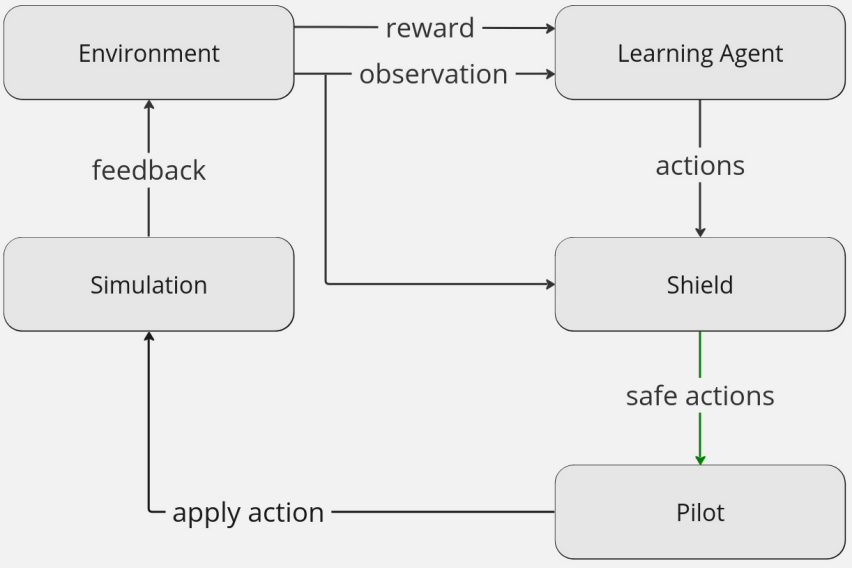


Figure 6.1: Environment Cycle of Sad-RL.

6.2 Environment Setup

In Section 5.1, the simulation design was explained in detail that it's based on CommonRoad scenario tools and that it had its own state and action setup. This simulation was used in this thesis as the environment built based on the generated scenarios. However, to use this environment with pre-implemented RL algorithms, it would be tedious to refactor them to interact with this environment and even more difficult and away from the scope of this thesis to build them from scratch. Therefore another environment was built on top of the simulation that utilizes its state and action setup to use the RL algorithms directly with it. The environment was designed to inherit the Farama Gymnasium [55] interface since several pre-implemented RL algorithms libraries are built to work directly with Gymnasium environments. An example of these libraries is stable-baseline3 [56]. The interface implements 4 main functions that are the observation space, the action space, the step, and the reset. The observation and action spaces specify the observation and action spaces that will be used in the environment while the reset function restarts the environment to the initial state and returns the initial observation. The step function applies the action selected by the agent to the environment and returns the new observation and reward. The environment designed for this thesis is called SAD-RL. It has a special consideration throughout its steps. The learning cycle can be seen in Figure 6.1. First, during the initialization of the environment, a new simulation will be created to apply the current scenario in the batch queue of the environment. It then uses this simulation throughout the whole process, it has to be reset and provided with the action selected by the agent at each step along with the other dynamic objects' actions. In this case, since the approach in this thesis is using open-loop simulation, the action for other dynamic objects follows the predefined trajectory. As for the action of the ego vehicle, it goes through the shield first, which in turn

assesses the action and decides whether or not it's doable and safe. It then passes this action if it's a lane change to the pilot and The simulation then returns to the environment the feedback that the agent then assesses to consider a new action, this action is filtered by the shield to make sure it follows the conditions of safety and then pass the action to the pilot that executes the action on the simulator. The simulator returns the feedback after the action is executed which is then interpreted by the environment to return a reward and an observation to the learning agent and the shield.

6.2.1 Observation Space

The environment is set up to use an observation space that allows the agent to assess the situation and map its actions in a certain situation to the reward given by the environment. The finalized observation space used in this thesis is shown in Table 6.2. The observation space is designed to give the agent a focused view of the headway vehicle with time headway, distance headway, and minimum time gap. The other part of the observation space includes the terminal states, to give a good indication to the agent about the state at the end of the episode which can help it connect the sparse reward with the termination reason. Furthermore, the last part of the observation gives the agent information about the 6 adjacent vehicles within a circular sensing range around the ego vehicle by providing the distances and relative velocities. Other cars in the non-adjacent lanes were not considered.

6.2.2 Reward

The environment evaluates this as well to return two reward equations that represent the sparse and dense rewards. The sparse one included the termination cases of the environment. In this thesis, the range of the sparse reward is [0-1]. If the agent reaches a goal it gets a reward of one, however, if it collides, goes off the road, reaches a timeout or a standstill, it gets only a zero

$$\mathrm{r}_{\mathrm{sparse}} = \begin{cases} 1 & \text{if the goal is reached} \\ 0 & \text{if there is a collision, offroad, timeout, or standstill} \end{cases}$$

On the other hand, the dense reward included the sparse reward, the safe distance keeping from leading vehicles reward, and driving on the right side of the road reward. The dense reward is given by this function:

$$\mathrm{r}_{\mathrm{dense}} = \mathrm{r}_{\mathrm{sparse}} + \mathrm{safe\_distance\_reward} + \mathrm{Coeff}_{\mathrm{rightlane}} \cdot \mathbf{1}_{\mathrm{is\_rightlane}}$$

$$\mathrm{safe\_distance\_reward} = -\exp\left(5 \cdot \left(\frac{cur\_dhw}{3}\right)\right) \cdot \mathbf{1}_{cur\_dhw \leq dhw_{min}}$$

Variable	Description
Lanelet ID	Identifier for the lanelet
Number of Lanes	Number of lanes in the current section
Velocity	Current velocity of the vehicle
Acceleration	Current acceleration of the vehicle
Current THW	Current Time Headway
Current DHW	Current Distance Headway
Current Minimum Timegap	Minimum timegap to the leading vehicle
Distance to Goal	Distance remaining to the goal
Is Goal	Boolean indicating if the goal is reached
Is Collision	Boolean indicating if a collision occurred
Is Time Out	Boolean indicating if time has run out
Is Offroad	Boolean indicating if the vehicle is offroad
Is Standstill	Boolean indicating if the vehicle is at a standstill
Distance to Lead	Distance to the leading vehicle
Distance to Rear	Distance to the rear vehicle
Distance to Left Lead	Distance to the leading vehicle in the left lane
Distance to Left Rear	Distance to the rear vehicle in the left lane
Distance to Right Lead	Distance to the leading vehicle in the right lane
Distance to Right Rear	Distance to the rear vehicle in the right lane
Relative Velocity to Lead	Relative velocity to the leading vehicle
Relative Velocity to Rear	Relative velocity to the rear vehicle
Relative Velocity to Left Lead	Relative velocity to the leading vehicle in the left lane
Relative Velocity to Left Rear	Relative velocity to the rear vehicle in the left lane
Relative Velocity to Right Lead	Relative velocity to the leading vehicle in the right lane
Relative Velocity to Right Rear	Relative velocity to the rear vehicle in the right lane

Table 6.2: Observation Space Parameters

These equations are inspired by the approach of CommonRoad-RL [57] which is a configurable RL environment for autonomous driving. Given that the *cur_dhw* is the current distance headway and dhw_{min} is the minimum distance headway. The dense reward here is only used as an extra setup to help the agent learn the problem faster and could be ablated if it does not show signs of faster learning or if it hinders the learning process.

6.2.3 Shield

The shield's role follows the mentioned idea in Chapter 5, by checking the two conditions that correspond to the action space in both dimensions (lateral and longitudinal). The lateral action is assessed from the observation of the environment which uses CommonRoad's laneletnetwork which is a class in CommonRoad library that represents the connections between the lanes of the road. The shielding condition checks if the current lane, has an available neighbor lane on the left or the right side depending on the action, and decides whether or not it will be executed, otherwise the maintain or keep lane action is used. In case there is an available lane, the action is passed to the pilot that executes it. On the other hand, the longitudinal action

is assessed by calculating the TTC between the ego vehicle and the leading vehicle using this formula that assumes constant acceleration:

$$TTC = \sqrt{\frac{1}{4}\left(\frac{2 \cdot dv}{da}\right)^2 - \frac{2 \cdot ds}{da}} - \frac{1}{2}\left(\frac{2 \cdot dv}{da}\right)$$

such that ds is the distance difference between the two vehicles (rear and leading), dv is the velocity difference or relative velocity, and da is the acceleration difference. This formula is then compared to the minimum TTC to decide on the safe action provided by the condition demonstrated in the approach.

6.2.4 Pilot

The pilot is considered the executor of actions selected by the agent. An overview of the whole system is demonstrated in Figure 6.2. The pilot executes the actions by providing the simulation with a tuple of steering velocity δ_v and acceleration a_{long} that represent both the lateral and longitudinal actions. The pilot executes the hierarchical policy approach explained in Section 2.3.3 to facilitate the problem from a continuous action space to a discrete action space. The agent decides on both lateral and longitudinal dimensions. On the lateral side, the pilot employs a Lane Change Assistant (LCA). The LCA first defines the target lane that it wants to move to, and then it sets a boolean that indicates that a lane change is currently being done. At this time the agent is not allowed to make decisions until the LC_{active} boolean is deactivated. During the lane switch, the function calculates the lane change reference trajectory using a helper method that calculates the longitudinal final position and then calculates the lateral trajectory during the lane change. Another function calculates the orientation at which the vehicle should be to make this change as a step by step This orientation is then passed to the Model Predictive Controller (MPC) which solves an optimization problem at each step that aims to minimize the cost of deviating from the calculated orientation, the speed and the time headway of the vehicle. It then produces the required steering velocity and acceleration that are passed to the simulation to execute the actual lane change. On each step, the pilot checks whether or not the lane change is done and if it is not yet done it re-calculates the remaining parts of the reference trajectory that are not yet followed and keeps on following them till the lane change is done. In case a lane change is not happening at the moment, the keep lane function is being called which creates a trajectory that follows the center line of the lane until the agent decides to switch lanes. During the keep lane lateral action, the agent is allowed to choose a longitudinal action out of the four mentioned in Chapter 5.

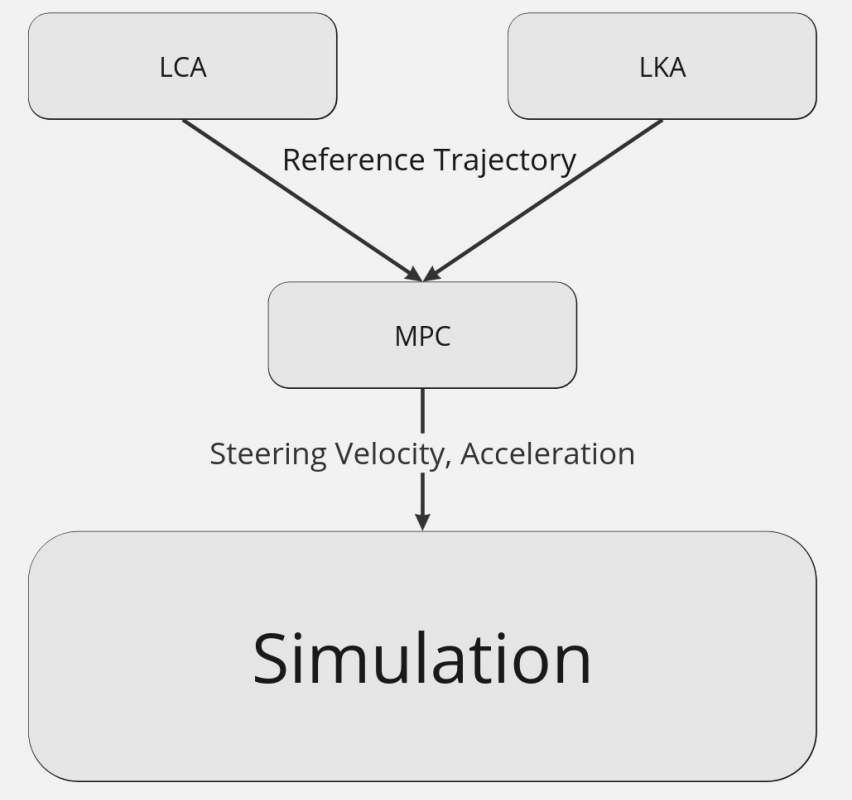


Figure 6.2: Pilot components interaction with simulation.

6.3 RL Agents

This thesis aims to use one or several RL agents to answer the main research questions asked in the research statement Chapter 4. The RL agent in this case doesn't have to be the best-performing agent or a very competitive one but has to show significant performance in the efficiency of learning when compared to the ablation results later. The purpose of this thesis does not include the hand engineering of the agents themselves, therefore off-the-shelf implementations of popular RL algorithms were used. Since the implementation of the environment is based on the Gymnasium interface, the use of the stable-baseline3 library on Python mentioned in Section 6.2 is the prime candidate in this thesis. The agents used in the experiments are 4 agents, DQN, QR-DQN, PPO, and A2C. Discrete flat actions were used in this case as a set of all given actions by the pilot (HRL). The action space was inspired by that of Farama Highway-env's [58] Discrete Meta-Actions, to also test the Highway-env trained models on the generated scenarios for comparison. The final action space of the SAD environment is shown in Table 6.3

The environment uses the scenarios from the MongoDB database to initialize the simulation and use it later during training or testing. The parameters of the algorithms used in the experiments were the default parameters of each algorithm as the fine-tuning of the model

Action ID	Description
0	Change to the left lane and maintain current velocity
1	Maintain current lane and velocity
2	Change to the right lane and maintain current velocity
3	Maintain current lane and accelerate
4	Maintain current lane and brake

Table 6.3: Definition of the action space of the SAD-RL environment

is not one of the focus points of this thesis. However, the seed was made constant during the training phase, to make the results more consistent. Since the observation space of the environment is a dictionary, the MultiInputPolicy was used in all 4 algorithms. Furthermore, the device used for training was consistently CPU as the use of GPU was not very useful in this case due to having several operations being done on the CPU for the simulation. This caused a bottleneck that prevented the use of GPU. The total timesteps used were counted by the number of steps taken by the agent and not the simulation timesteps. The steps of the agent and the simulation timesteps might differ if lane changes occur as explained in Section 2.3.3.

6.4 Validation Pipeline

During the training phase of the agent, there should be a designed consistent validation pipeline to ensure integrity for the training phase. The designed pipeline used in this case consists of 3 steps.

6.4.1 Pipeline Initialization

The choice of the agent, depending on the selected algorithm out of the mentioned 4 algorithms in Section 6.3, the pipeline gets initialized and refactored accordingly to handle the selected agent. The choice of the algorithm affects the initialization of the agent as well as the next steps to save and load the models in the correct directory as well as the charts. In addition, 100 SubprocVecEnv are initialized, each representing one of the 100 training batches that include a portion of the training scenarios as mentioned in Section 6.1.2. This is done to speed up the process of training as well as having a smaller amount of scenarios to preload before the beginning of training.

6.4.2 Intermediate Validation

The intermediate validation step is mainly the callback function that is passed to the model learn function. In this pipeline, a custom callback is used to generate consistently tailored information to the needs of the thesis and is independent from the used algorithm which is the case when using tracking and visualization tools for machine learning pipelines like Tensorboard [59] or Wandb [60]. The custom callback function saves the termination reason

for each episode in a CSV file as well as the timestep and the reward. Furthermore, it saves an intermediate model during training after a certain amount of environment steps (steps taken by the agent on all parallel environments if any). These models are then tested deterministically against the given scenario (in the case of training on one scenario only). In addition, it renders a video of the deterministic test. These videos are then used to evaluate the reasons for terminations in a qualitative manner as well as show the progress of the agent during training which helps in debugging the agent's performance.

6.4.3 Results Visualisation

The last part of the custom callback function generates a chart based on the saved CSV file. The chart used in this case is the moving average of each termination reason rate against the number of episodes. The callback function generates one chart per termination reason showing the progress over time. An example of these charts can be seen in Figure 6.3 which shows the moving average rate for goal reached termination against the episodes. Lastly, the final model is saved and the environments are closed.

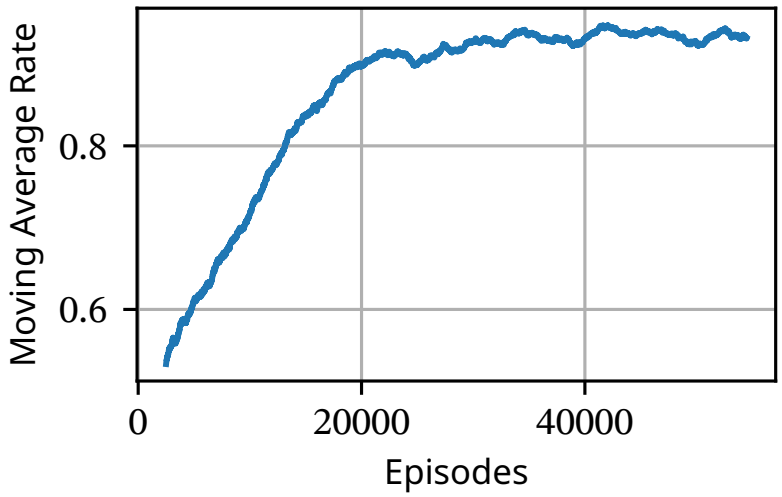


Figure 6.3: Goal Reaching Rate

7 Results

In the following chapter, the evaluation techniques as well as the metrics used in this thesis to evaluate the candidate agents are presented. The evaluation KPIs are also introduced to follow the requirements of the research statement that are related to the training/validation phase and the final testing phase. In addition, the results of testing the agents are presented in several phases to pick the most suitable model to answer the research questions of this thesis. The ablation study results are also presented to try to answer the research questions. Lastly, the results of extending the training data will be presented.

7.1 Evaluation Framework

To create reliable evaluation metrics, the requirements mentioned in Section 4.2 have to be followed. As shown in Chapter 3, there have been several approaches to evaluate each one of the mentioned requirements. In this section, the chosen evaluation metrics as well as the ablation study criteria shall be explained in detail as well as how they will align with the requirements of the research statement.

7.1.1 Evaluation metrics

The metrics will focus mainly here on the first two requirements (safety and efficiency of learning) that require, KPIs to measure their progress of learning and comparing them to each other. The last requirement (generalizability) will be considered throughout the experiments by always making sure that the data used for training are not seen in testing which will also be explained in detail.

7.1.1.1 Safety

The safety evaluation in this thesis focuses on the safety-threatening reasons that caused the termination of the episode. These are the termination reasons explained in Section 5.1 except for the goal reaching and timeout, so in total 3 termination reasons pose a safety-threat (Collision, off-road, and standstill). For training validation, these metrics will plotted as explained in the validation pipeline in Section 6.4, the moving average of the rate of each termination reason against the number of episodes. To evaluate the deterministic performance of the agent on the testing data, the percentage of each termination reason against the total number of testing episodes will be calculated. These metrics show how well the algorithm is avoiding safety-threatening situations. The evaluation metrics can be calculated by Equation (7.1).

$$P(T_i) = \frac{\sum_{k=1}^{M} T_{i,k}}{M} \times 100 \tag{7.1}$$

Where:

- $P(T_i)$ is the percentage of termination reason T_i.
- $T_{i,k}$ is the occurrence of termination reason T_i in evaluation episode k.
- M is the total number of evaluation episodes.

The percentage of every termination reason will be demonstrated in a bar chart for each agent for a single test set.

7.1.1.2 Efficiency of Learning

The efficiency of learning will be assessed through charts generated by the validation pipeline, as detailed in Section 6.4. These charts demonstrate the termination reason moving average rate over the number of episodes. Using the moving average helps to show a more independent representation of the agent's performance on different time windows, avoiding the potential bias caused by initial poor performance. The focus will be on the goal-reaching and collision moving average rates, which indicate the agent's learning progress and any visible learning patterns. It will show a decline in the collision rate and an incline in the goal-reaching rate if the model improves over time. The reward range will be also experimented with and the final reward range that was used throughout training will be documented.

7.1.1.3 Generalizability

Ensuring the generalizability of learning during the training and validation phases is crucial to demonstrate the model's ability to acquire driving skills, rather than merely overfitting or memorizing specific scenarios. This section formalizes the methodology for validating generalizability which will be defined for both synthetically generated scenarios and real roads.

Synthetically Generated Scenarios

The dataset $\mathcal{D}$ is divided into a training set $\mathcal{D}_{\text{train}}$ and a test set $\mathcal{D}_{\text{test}}$, with the condition that $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$.

As detailed in Section 6.1, scenarios are created using three distinct types: $\mathcal{T}_A$, $\mathcal{T}_B$, and $\mathcal{T}_{Cutout}$. Each type encompasses various scenarios defined by a set of parameters $\{p_1, p_2, \dots, p_m\}$. Thus, a synthetic scenario S_i belongs to a type $\mathcal{T}_i$ and is characterized by a unique parameter combination within the set $\mathcal{S}_i$, where $\mathcal{S}_i$ denotes all scenarios of type $\mathcal{T}_i$.

Testing on scenarios not seen during training ensures the model's generalizability. To further evaluate this, cross-test validation is employed by training on one scenario type and testing on another. For example, a model trained on type $\mathcal{T}_A$ (the easiest) can be tested on type $\mathcal{T}_B$ (harder)

or type $\mathcal{T}_{Cutout}$ (the hardest), and similarly for other type combinations. Cross-test validation is formulated as:

$$\text{Train}(\mathcal{D}_{\text{train}} = \mathcal{S}_i),$$
$$\text{Test}(\mathcal{D}_{\text{test}} = \mathcal{S}_j),$$
$$\text{such that } \mathcal{S}_i \cap \mathcal{S}_j = \emptyset.$$

This setup ensures that training and testing data come from different scenario types, allowing for a robust assessment of the model's ability to generalize across various scenario types.

Real-Roads Scenarios

Real-road data $\mathcal{R}$ consists of scenarios extracted from actual road recordings. Unlike synthetic scenarios, these are not based on predefined challenger concepts but are unique road snippets, enabling testing on previously unseen data. Consider the set of scenarios $\{r_1, \dots, r_m\} \subseteq \mathcal{R}$:

$$\text{Train}(\mathcal{D}_{\text{train}} = \{r_1, \dots, r_n\}),$$
$$\text{Test}(\mathcal{D}_{\text{test}} = \{r_{n+1}, \dots, r_m\}),$$
$$\text{such that } n < m.$$

Additionally, cross-test validation with synthetic and real-road scenarios can be performed as follows: let $\mathcal{R}_{test} \subseteq \mathcal{R}$

1. **Synthetic-to-Real:**
$$\text{Train}(\mathcal{D}_{\text{train}} = \mathcal{S}_i),$$
$$\text{Test}(\mathcal{D}_{\text{test}} = \mathcal{R}_{test}).$$
2. **Real-to-Synthetic:**
$$\text{Train}(\mathcal{D}_{\text{train}} = \mathcal{R}_{train}),$$
$$\text{Test}(\mathcal{D}_{\text{test}} = \mathcal{S}_i).$$

In the Synthetic-to-Real setup, the model is trained on synthetic scenarios from $\mathcal{S}_i$ and evaluated on real-road scenarios from $\mathcal{R}$. Conversely, in the Real-to-Synthetic setup, the model is trained on real-road scenarios from $\mathcal{R}$ and tested on synthetic scenarios from $\mathcal{S}_i$. These cross-test validations help assess the model's generalizability across different types of scenario sources.

7.1.2 Ablation Study

To address the research questions of this thesis, an ablation study will be conducted for each question. This approach involves systematically removing specific model components to evaluate their impact on performance.

Firstly, to assess the effectiveness of the HRL approach, the hand-engineered policies will be ablated as well as the lateral shielding as it won't be possible without the hierarchical policies. This will turn the problem into a continuous action space instead of the discrete one provided by HRL. This will help determine whether the absence of HRL affects the efficiency and generalizability of the learning algorithm. By comparing the performance with and without HRL, one can deduce if it was effective or not, and thus, answer the first research question.

Secondly, the role of scenario-based training in improving learning efficiency will be investigated by ablating the use of challenger scenario concepts. This involves removing these structured scenarios and observing the impact on the training process. The scenarios will include both randomly generated synthetic scenarios and those derived from real-road traffic data, as outlined in Section 2.2.

The results of these ablation studies will be compared against the same models that were learned using one of the components at each test. The models will be tested on the same set of testing scenarios and evaluated using the same consistent metrics to ensure a fair comparison. This analysis will provide a clear understanding of the contributions of each component on its own to the model's overall performance and help validate the research questions.

7.2 Experiments

After the defining of the Evaluation framework, the experiments themselves have to be defined and the results of the experiments shall be demonstrated. In this section, the generated dataset, and training-testing split of the experiments will be explained as well as the 4 phases of training and the results achieved by the 4 candidate agents in this thesis mentioned in Section 6.3.

7.2.1 Experiments Datasets

As explained in Section 6.1 there will be two sources of scenarios, and each of them will have its own training-test split required for the experiments. For the synthetic scenarios, the 3 generated scenario types A, B, and Cutout created a full dataset of 577k scenarios out of which around 500 scenarios were omitted due to not being feasible according to the scenario challenge description method. Then out of the remaining scenarios, 1000 scenarios were selected randomly as training data for each type. In addition, 200 scenarios were selected randomly as testing data for each type. Resulting in a total of 3000 scenarios for training and 600 scenarios for testing that are completely distinct. For the real-road scenarios, the dataset

converter from CommonRoad was used to extract 4000 scenarios that were randomly sampled from the given videos in the dataset of Highd. Out of these scenarios, only 350 scenarios were considered feasible enough as their goal position was far enough from the starting position. 300 scenarios in this case were used for training and 50 scenarios were used for testing.

7.2.2 Over-fitting Experiments

In this phase, the focus was to pick an algorithm that could be used to prove the hypotheses of the thesis. That is done by checking if there is a learning pattern generally found in the training of the agents. To simplify this, over-fitting only one scenario was the option. Each of the candidate agents was given a set of simple scenarios to try and show some pattern of learning during training on each of these scenarios learning them independently. Each one got a chance to train on these scenarios for 200k timesteps. This test helped define the model that is suitable for the custom environment.

The candidate agents used for this phase were DQN, QR-DQN, PPO, and A2C as mentioned in Section 6.3. After converting the continuous action space problem into a discrete action space one using hierarchical policy, the following justifications were made for the selected algorithms:

- DQN (Deep Q-Network): DQN is well-suited for discrete action spaces and can efficiently learn the optimal policy using Q-learning. Since the problem is now similar to problems solvable by DQN as well as it being one of the most commonly used algorithms in discrete autonomous driving problems. Therefore, it was the first on the list.
- QR-DQN (Quantile Regression DQN): QR-DQN provides a more robust understanding of the value function by estimating the distribution over returns, which is beneficial in environments with high variability. This made it the second candidate to provide a more stable value-based algorithm in addition to DQN.
- PPO (Proximal Policy Optimization): PPO was used in these experiments as it is known for its stability of learning due to the clipped objective function as well as its common usage in the context of ADS as well as performing well on high dimensional action spaces.
- A2C (Synchronous Advantage Actor-Critic): A2C leverages both value-based and policy-based methods, allowing it to learn the value function and policy simultaneously, resulting in stable learning. In addition, advantage Actor-Critic has been used frequently in similar HRL in the context of ADS in addition to performing well on high dimensional action spaces.

In this experiment phase, both rewarding systems (sparse and dense) were used to test the learning capabilities of the 4 algorithms. The main point of using both was to decide on which system the training should proceed depending on how well will the performance be.

Furthermore, to see if the dense reward helps in speeding up the process of learning of the algorithms. The KPIs of this phase are the charts generated by the validation pipeline. If the algorithm is learning, it will show a rise in the moving average rate of goal reaching against the number of episodes as the learning proceeds. It should also show a fall in the moving average rate of all the other termination reasons.

Results

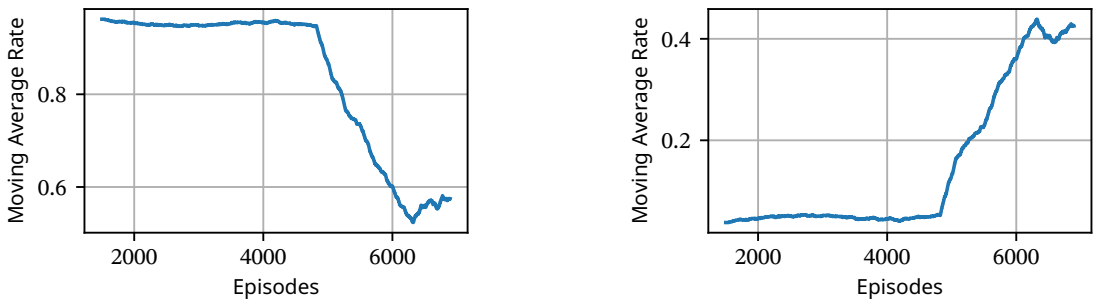
In this test, the 4 algorithms were given several scenarios to try to over-fit, while being validated using the validation pipeline. For simplicity, only one scenario results will be shown in this phase. *scenario_a_v_0_121* will be the example scenario for the comparison of the performances of the algorithms. This scenario's trajectory plot can be seen in Appendix A along with all the scenarios mentioned in the training phase. The results of the training are as follows:

DQN: Figure 7.1 shows that DQN was only able to achieve less than 45 percent of the moving average rate of goal-reaching episodes while the collision rate reached over 55 percent. Thus demonstrating low efficiency of learning.

QR-DQN: Figure 7.2 shows that QR-DQN was not able to show any indication of learning, as it could only reach a negligible rate of goal-reaching episodes at the end of training which was less than 1 percent. However, the collision rate was over 90 percent by the end of training.

PPO: Similar to QR-DQN results, Figure 7.3 shows that PPO was not able to have a goal-reaching rate above 1 percent, while the collision rate was above 95 percent.

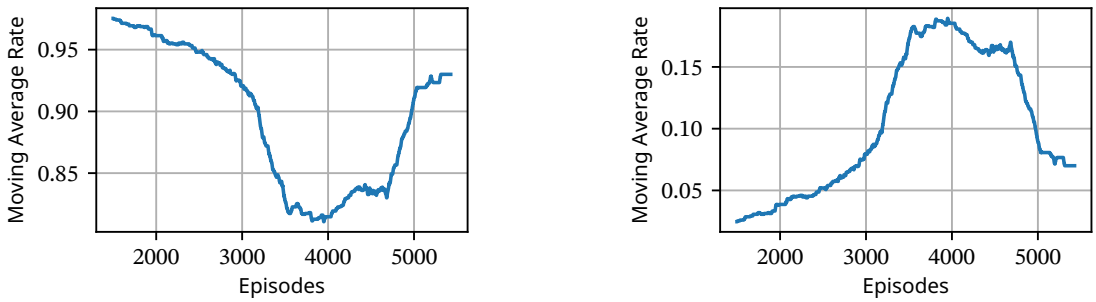
A2C: As seen in Figure 7.4 A2C was able to outperform all of the other algorithms reaching a goal-reaching rate of over 95 percent and a collision rate that is less than 5 percent at the end of the training phase.



(a) Collision Rate

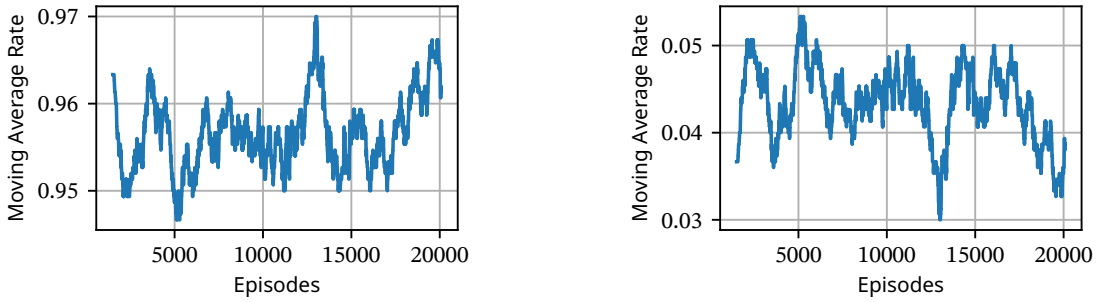
(b) Goal Reaching Rate

Figure 7.1: Termination reasons moving average collision and goal reaching rates of training on scenario *scenario_a_v_0_121* using DQN model



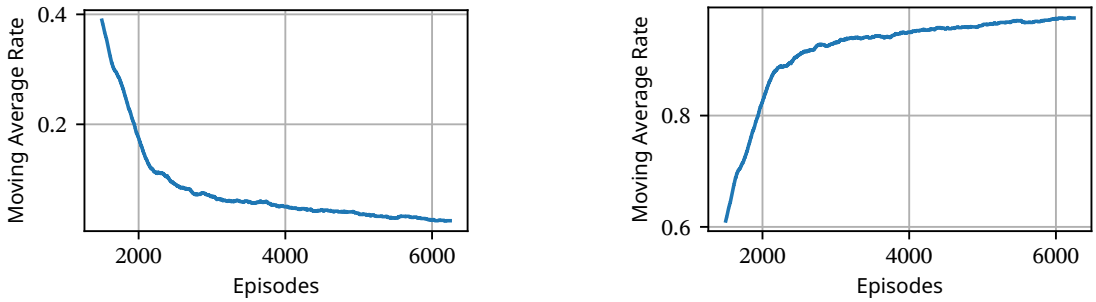
(a) Collision Rate (b) Goal Reaching Rate

Figure 7.2: Termination reasons moving average collision and goal reaching rates of training on scenario *scenario_a_v_0_121* using QR DQN model



(a) Collision Rate (b) Goal Reaching Rate

Figure 7.3: Termination reasons moving average collision and goal reaching rates of training on scenario *scenario_a_v_0_121* using PPO model



(a) Collision Rate (b) Goal Reaching Rate

Figure 7.4: Termination reasons moving average collision and goal reaching of training on scenario *scenario_a_v_0_121* using A2C model

Therefore, A2C was chosen as the algorithm that is qualified to continue with the test phases. In Appendix A A2C trained with different seeds demonstrated slight variations in the charts which does not affect the learning efficiency or the final rate achieved by the agent. Therefore,

a constant seed will be used to lower the variability for the rest of the phases. In addition, the use of dense reward hindered the learning process as it introduced more rules that the agent had to follow (e.g. driving on the right side of the road). On the other hand, sparse reward showed the required performance for this learning phase.

7.2.3 Cross-test Experiments

The focus of this test phase is to test the ability of the agent to excel in transfer learning. This can be tested by training on one scenario until over-fitting is reached, then using the saved model to test on another scenario deterministically and then asses the performance. The KPI of this test phase is the inference from the rendered video of the episode. The video will show the behavior of the agent in evading the challenger vehicle and reaching the goal on time. Since the results of the previous phase ended in ruling out all of the candidate algorithms except for A2C which demonstrated successful over-fitting, the only algorithm that will be tested in this case would be A2C. In addition, using the dense reward was hindering the process of learning, so it will also be omitted and only a sparse reward will be used from now on. The algorithm will be given 2 different scenarios of different types to over-fit on each one of them independently and then do the cross-test. If the algorithm is able to solve (reach the goal position) the test that it hasn't seen during training) in this, it proves that it can learn the skills of driving and not just memorize a certain scenario.

Results

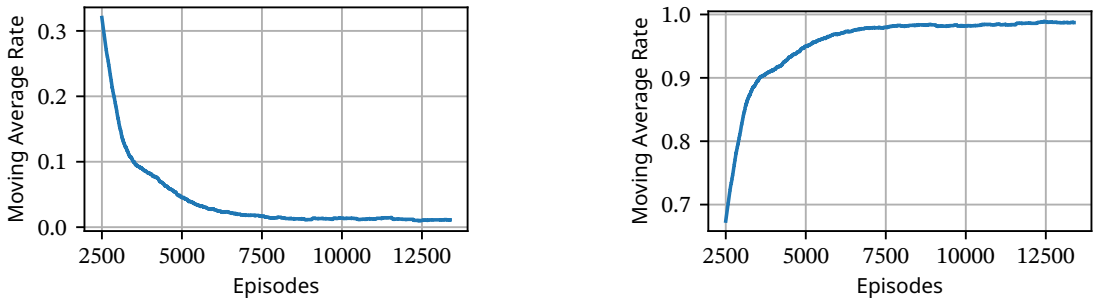
The scenarios used for this test phase were two of the already used scenarios in the previous phase of over-fitting. The first and the second models learned on scenarios *scenario_a_v_0_121* and *scenario_b_v_0_0* respectively. In this test, both models that learned about each of these scenarios independently were successful in solving the other one in a cross-test even though it hasn't been seen during training. Since those two scenarios had similar solutions, as all the agent had to do was switch to the left adjacent lane and keep going until it reached the goal. Therefore, it was expected that both models would be able to solve the other scenario successfully. To take this test a step further, another scenario was introduced to the test cases, *scenario_cutout_v_2_55156*. This scenario is a harder one to solve since it had 3 lanes while the other two had only 2 lanes. In addition, this scenario requires at least 2 lane changes to the left to solve. Testing both models in this scenario ended in a collision. The first model was able to do 2 lane changes to the left but before passing the vehicle on the middle lane it did another lane change to the right and ended up colliding with this vehicle. On the other hand, the second model was only able to do one lane change to the left and then collided with the vehicle in the middle lane. Keeping the extra test in mind, the first model was able to pass two more scenarios *scenario_a_v_0_103* and *scenario_b_v_3_38964* and it was able to solve them given that they were not seen during training. Therefore, it demonstrated that this test was successful enough to allow the algorithm A2C to proceed to the next phase.

7.2.4 Simultaneous Over-fitting Experiments

In this test phase, the goal is to demonstrate the ability of the agent to over-fit two scenarios at a time. The two chosen scenarios were *scenario_a_v_0_121* and *scenario_c_v_2_55156*. Choosing those two scenarios came from the extra test done in the previous phase. Since the model that could over-fit the former scenario could not solve the latter scenario. However, the agent that could over-fit the latter scenario could solve the former one. The training was designed to take 200k timesteps showing two KPIs in this case. The first KPI demonstrating the over-fitting is, the moving average rate against the number of episodes. This KPI would show that the agent is learning the two scenarios if the goal-reaching rate is increasing while the collision rate and other termination reasons are decreasing. On the other hand, the second one is the rendered videos of a deterministic test done during the training for each scenario every 7500 timesteps. These videos will help show the progress of learning simultaneously on the two scenarios by showing the behavior of the agent as the timesteps of training increase.

Results

The experiment results indicate that the agent successfully over-fitted both scenarios, *scenario_a_v_0_121* and *scenario_c_v_2_55156*, simultaneously.Figure 7.5 illustrates the moving average rate against the number of episodes. It shows a learning pattern as the rate of goal-reaching episodes increases throughout the training reaching almost 100 percent. In comparison, the collision rate graph shows a decline till it reaches almost 0 percent. The other termination reasons like standstill and offroad were constantly at 0 percent. Lastly, the timeout rates were negligible and also showed a decline. Therefore, these charts were omitted as they are redundant in this case.



(a) Collision Rate

(b) Goal Reaching Rate

Figure 7.5: Termination reasons moving average collision and goal reaching rates of training on scenarios *scenario_a_v_0_121* and *scenario_c_v_2_55156* simultaneously. The graphs combine both scenarios: goal-reaching and collision rates.

7.2.5 Dataset Learning Experiments

After the algorithm shows itself to be able to learn simultaneously two scenarios at a time, the last phase is to train it on larger sets of scenarios. Two training experiments will be conducted in this phase. Independent learning of each scenario type and full training on all scenario types.

Independent Learning of Types The training begins with a set of scenarios that belong to only one scenario type (A or B or Cutout), and then we train on a full dataset of the 3 types. In this training phase, A2C as the candidate algorithm was trained on 1000 scenarios from each of the 3 types independently. Then 200 scenarios from each type that were not taken from the training data were used for testing. Hence, having an independent dataset for training and also for testing. The scenarios used in this training were those that had inevitable collision if the agent remained idle. The models will learn on each scenario for 1 million timesteps and then will be cross-tested to assess whether transfer learning was achieved. In addition, the results are always compared against no policy agent (no action agent) and random policy agent (random action agent).

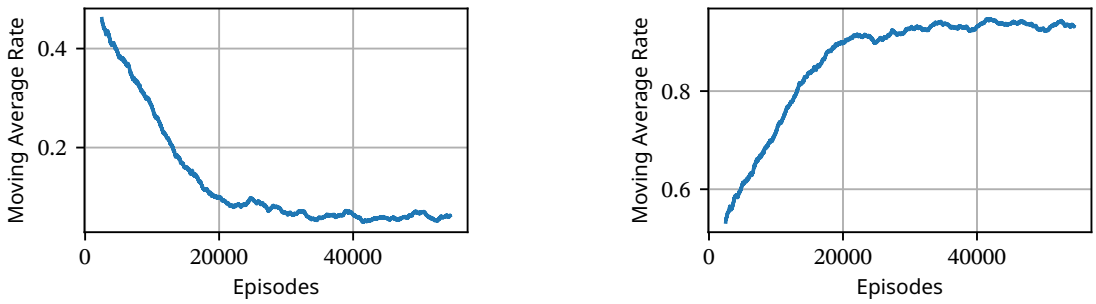
Full Learning on All Types

The final stage of training is on all of the scenarios together to demonstrate general skill learning across all scenarios. In this stage, a curriculum learning method was used. The agent first learns on scenarios of types A and B together, then the same model is used to train on the three types all at once. The idea to combine the learning process of types A and B came from the fact that they both have similar solutions in most of the scenarios (i.e. in both scenarios the agent has to do a left lane change maneuver to avoid the decelerating vehicle in front of it at some point). Then training on all three types altogether was used to include type Cutout to the dataset but also avoid catastrophic forgetting that the algorithm might suffer from in the cases of type A and B. In this phase also, the full learned model will be tested against all 3 types and will be compared to the results of the no policy and random-policy tests.

Results

The first training part results showed a clear learning pattern training on the 3 types independently. The method that was used to validate the training phase was the validation pipeline as in the previous phases. As shown in Figure 7.6, the agent was able to learn type A scenarios successfully as it reached almost 95 percent of goal-reaching rate by the end of training. The second model shown in Figure 7.7 demonstrated also learning on type B reaching a goal-reaching rate of about 66 percent by the end of training. The last model that learned on type Cutout as shown in Figure 7.8 as the goal-reaching rate has reached about 65 percent at the end of training. all the 3 types showed an increase in the goal-reaching rates. On the other hand, they showed also a clear decline in the collision rates, while the other reasons were

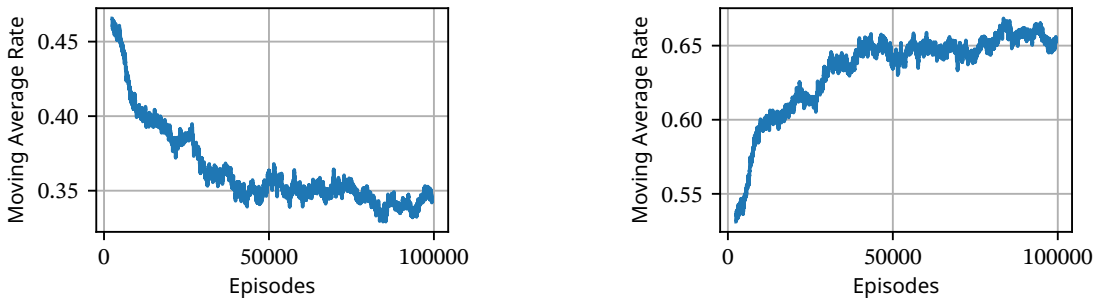
either constant zeros or negligible rates. The model that learned on type A showed the best learning pattern as it reached almost 95 percent of goal-reaching rates. The second best was the model trained on type B reaching about 66 percent goal reaching rate and the one trained on Cutout got the least goal reaching rate of about 65 percent. Furthermore, the test of the deterministic policies of each trained agent compared with no policy and random policy agents was conducted. Figure 7.9a shows the results of testing all the trained agents on test scenarios from type A. The trained agents on each type independently were able to outperform the No policy and random policy agents by having a better goal-reaching percentage of test scenarios and a lower collision percentage with the agent trained on type A outperforming all of the agents. In Figure 7.9b, which shows the trained agents tested on test scenarios from type B, the trained agents also outperformed the no policy and random policy agents. Similar to the test on type A, the agent trained on B was able to outperform all the agents. Lastly in Figure 7.9c, which showed the tests on type Cutout, the trained agents outperformed no policy and random policy agents except for the agent trained on type B which showed a similar performance to the random policy one but still had a better performance than no policy agent. In addition, the agent trained on type Cutout also outperformed all agents.



(a) Collision Rate

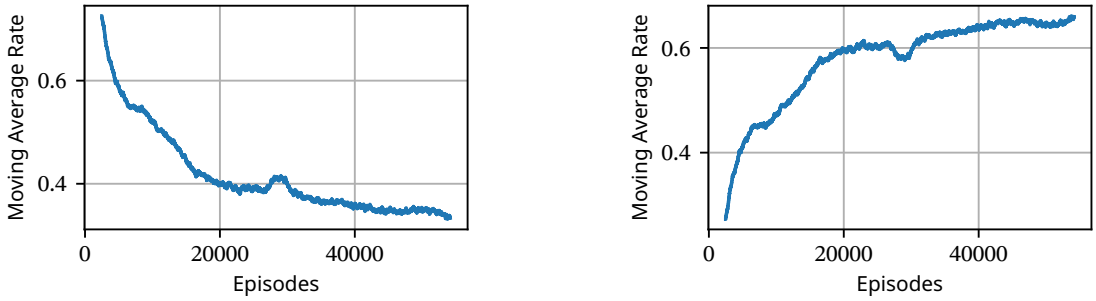
(b) Goal Reaching Rate

Figure 7.6: Termination reasons moving average collision and goal reaching rates for training on type A scenarios



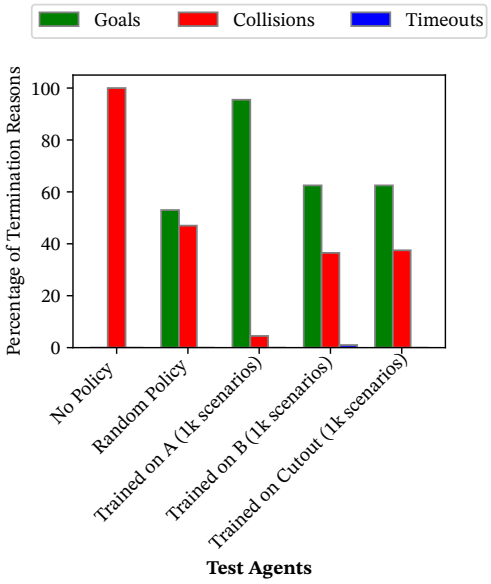
(a) Collision Rate (b) Goal Reaching Rate

Figure 7.7: Termination reasons moving average collision and goal reaching rates for training on type B scenarios

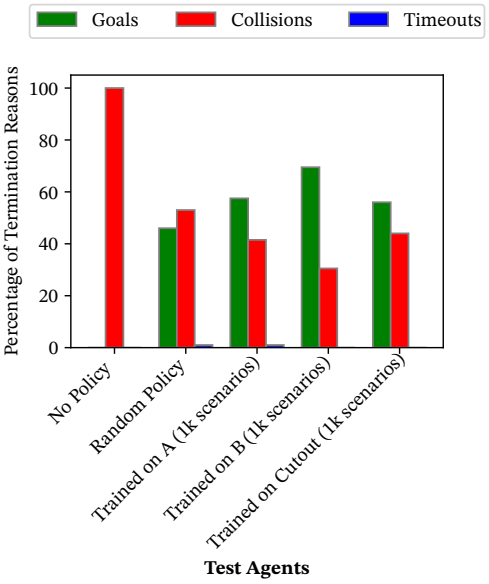


(a) Collision Rate (b) Goal Reaching Rate

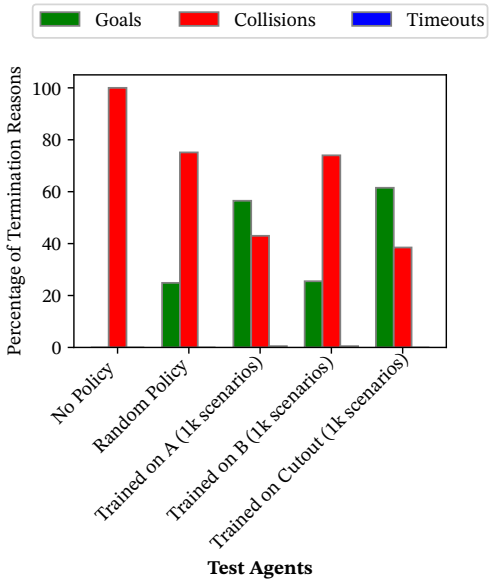
Figure 7.8: Termination reasons moving average collision and goal reaching rates for training on type Cutout scenarios



(a) Dataset Independent agents tested on type A



(b) Dataset Independent agents tested on type B

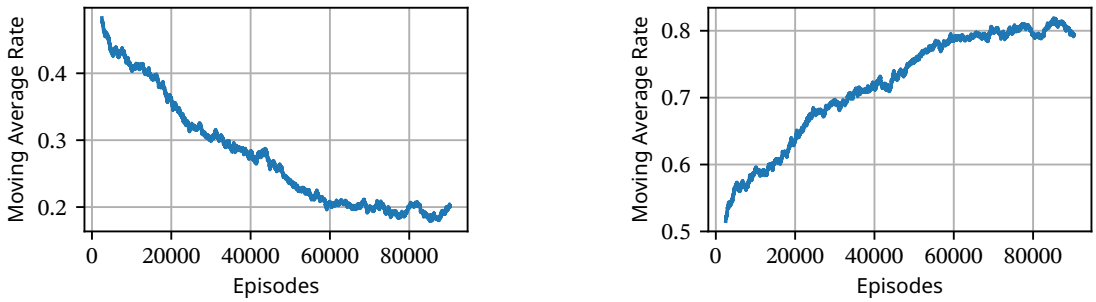


(c) Dataset Independent agents tested on type Cutout

Figure 7.9: Dataset Independent agents tested on types A, B, and Cutout

For the second dataset training phase, the trained model also showed a learning pattern in both stages of learning. As shown in Figure 7.10, the first stage which is learning types A and B together, showed a moving average rate of goal-reaching episodes of above 80 percent at the end of the training stage of 1 million timesteps. In the second stage, as seen in Figure 7.11,

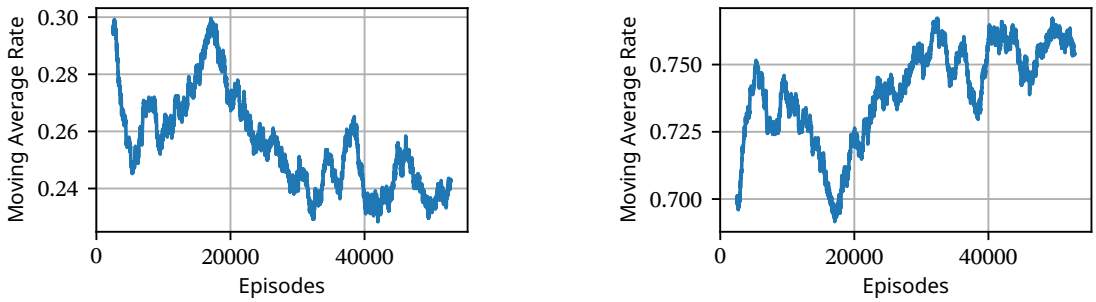
learning all 3 types together, the moving average rate of goal-reaching episodes was above 75 percent by the end of the training stage of 1 million timesteps as well. The fully trained agent was able to outperform all the other agents on test scenarios of type A in Figure 7.12a except for the agent trained on A only. In Figure 7.12b the fully trained agent also outperformed all the other agents in type B tests except for the one trained on type B. On the other hand, in Figure 7.12c, the fully trained agent could outperform all the agents in type Cutout tests except for the agents trained on type A and Cutout such that it got a similar performance to that of the agent trained on type A but less than that trained on type Cutout.



(a) Collision Rate

(b) Goal Reaching Rate

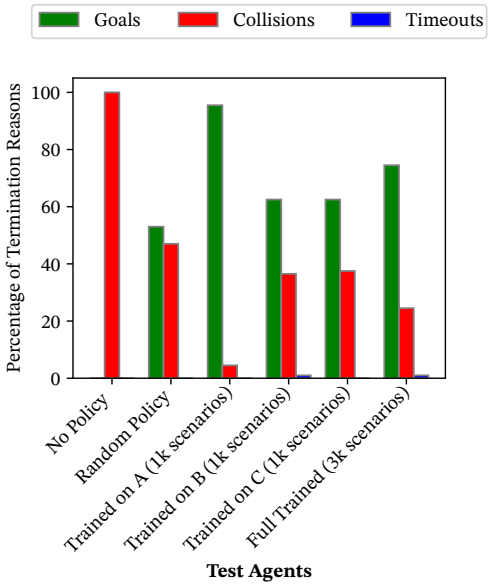
Figure 7.10: Termination reasons moving average rates for full training on types A and B



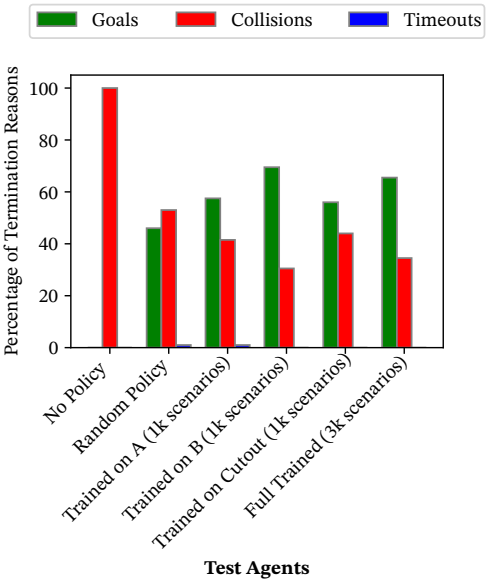
(a) Collision Rate

(b) Goal Reaching Rate

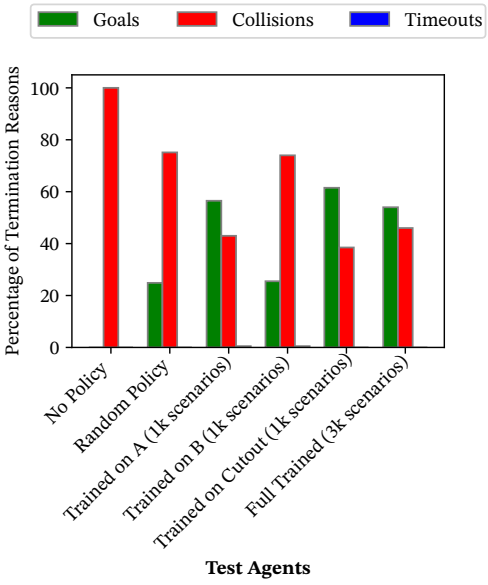
Figure 7.11: Termination reasons moving average rates for full training on types A, B, and Cutout



(a) Dataset Independent agents with Full Trained tested on type A



(b) Dataset Independent agents with Full Trained tested on type B



(c) Dataset Independent agents with Full Trained tested on type Cutout

Figure 7.12: Combined results of Dataset Independent agents with Full Trained tested on types A, B, and Cutout

7.3 Ablation Studies

As explained in Chapter 4, the research questions required an answer by showing the contrast between the use of the features designed in this thesis and without using them. The purpose of this section is to answer the research questions by ablating the use of HRL and challenging scenarios. The results of the ablation study will show whether or not these features are useful for solving the problem stated in this thesis.

7.3.1 Ablating HRL

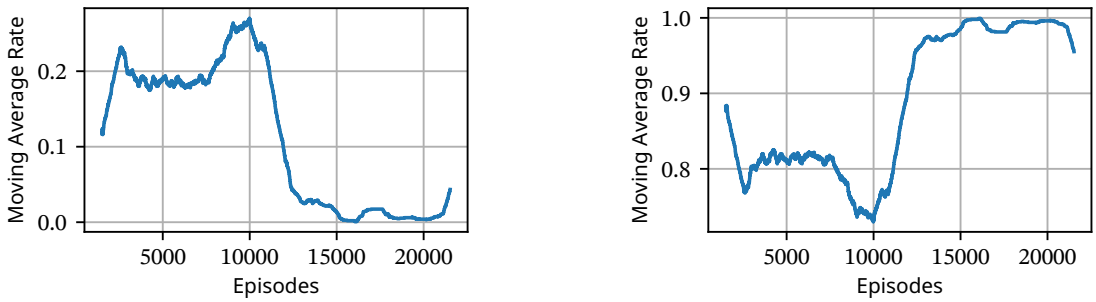
The first ablation study focuses on answering the first research question which questioned the feasibility of using hierarchical policy with reinforcement learning. For this study, the use of HRL was completely ablated returning the problem back to its original state that used a continuous action space as explained in Section 2.3.3. In addition, the use of the lateral shield will also be ablated as it's not possible without the use of HRL. In this state, the agent has to pass to the simulation two float numbers, the steering velocity and acceleration. The algorithms that will be tested in this case are PPO and A2C. The reason for using these 2 out of the 4 algorithms used earlier is that value-based algorithms like DQN and QR-DQN are generally not compatible with continuous action space problems. This ablation study will follow the phases of training used in Section 7.2.

Over-fitting Experiments

Both the given algorithms will try to over-fit one simple scenario to indicate if they show any learning pattern. In this test *scenario_a_v_0_121* was the scenario learned by the algorithms. The learning will be validated using the pipeline used earlier which shows the learning through the moving average rate of the termination reason against the number of episodes. Both algorithms will be given a chance of 1 million timesteps to over-fit the given scenario.

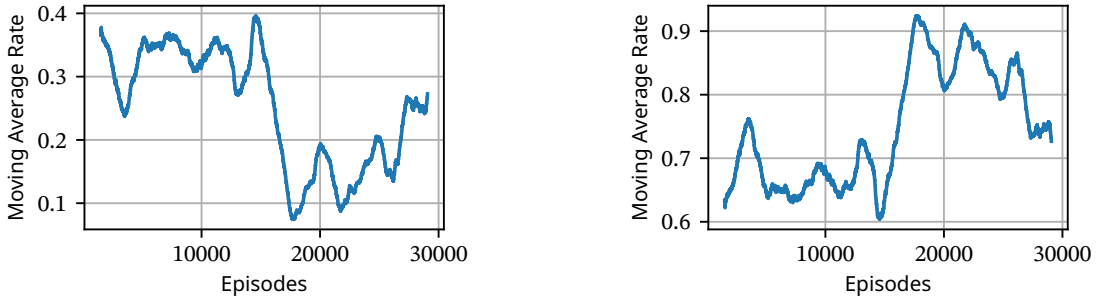
Results

According to the result chart in Figure 7.13, A2C algorithm showed a decline in the moving average rate of collisions that reached almost 0 percent at the end of the training. However, it failed to reach the goal even for one episode. On the other hand, the moving average rate of offroad episodes reached almost 100 percent at the end of the training. The goal-reaching rate chart was omitted due to its negligible results. The results achieved by PPO were not any better as it also showed in Figure 7.14 a trade-off between the moving average rate of collision and that of offroads with the former reaching over 20 percent and the latter reaching over 75 percent by the end of training. In the meantime, the results from goal-reaching and other termination reasons rates were either negligible or remained at 0 percent so their charts were omitted.



(a) Collision Rate (b) Offroad Rate

Figure 7.13: Termination reasons moving average rates for A2C training on *scenario_a_v_0_121* without HRL



(a) Collision Rate (b) Offroad Rate

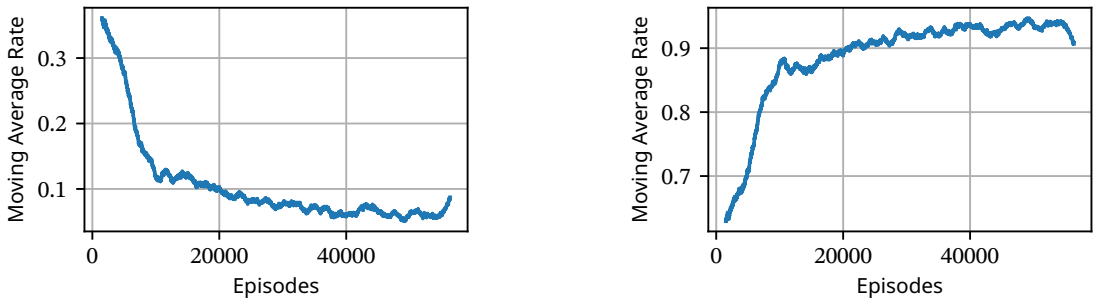
Figure 7.14: Termination reasons moving average rates for PPO training on *scenario_a_v_0_121* without HRL

7.3.2 Ablating Scenarios

The second ablation study that will answer the second question will be ablating the use of the challenger scenario concept in the training of the agents. The agents will be trained on real-roads extracted scenarios from the dataset highD that was mentioned in Section 2.2.1. The algorithm (in this case also A2C) is to be given a set of 300 real scenarios from highD for training and 50 scenarios for testing that were not seen during training. The model will learn on the 300 training scenarios for 1 million timesteps. Then, it will be validated using the same pipeline used before as well as having its model tested on the 3 scenarios types test cases independently and get its results compared to those of the other learned agents. In addition, the learned agents from the last phase of the training Section 7.2.5 will be tested against the highD real roads scenarios and compared to the result of highD trained model to show how well the trained agents can perform on real road scenarios.

Results

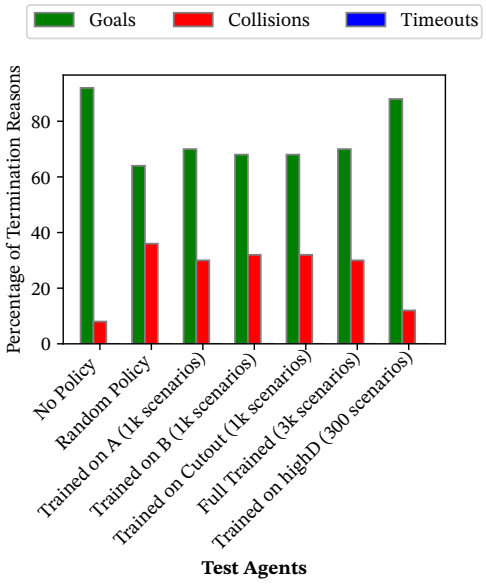
The validation pipeline showed in Figure 7.15 an incline in the moving average goal-reaching rate for the training of A2C on real highD roads that reached above 90 percent after the training. It showed a decline in the moving average collision rate that reached less than 10 percent by the end of training.Figure 7.16a, shows that the no policy agent outperformed all, while the second best agent was the one that learned on highD. The rest of the agents that learned independently on the 3 scenario types showed a similar performance with the full-trained model, and the model trained on type A slightly outperformed the other ones. These results show that in the case of real roads, no policy or staying idle is the best action that the model can take. In contrast, testing the highD trained agent on test scenarios of types A, B, and Cutout. As seen in Figure 7.16b the highD trained agent showed the lowest performance on tests from type A. In Figure 7.16c, the highD trained agent could outperform the trained agents on type A and Cutout as well as random policy and no policy agents but couldn't outperform the fully trained agent or the agent trained on type B. For the tests on type Cutout seen in Figure 7.16d, the highD trained agent had the lowest performance compared with other training agents.



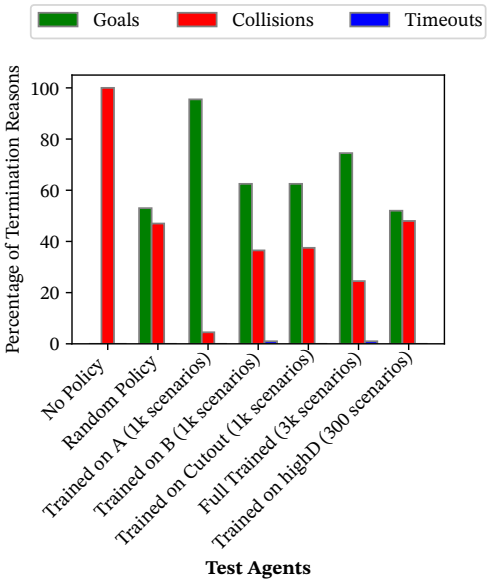
(a) Collision Rate

(b) Goal Reaching Rate

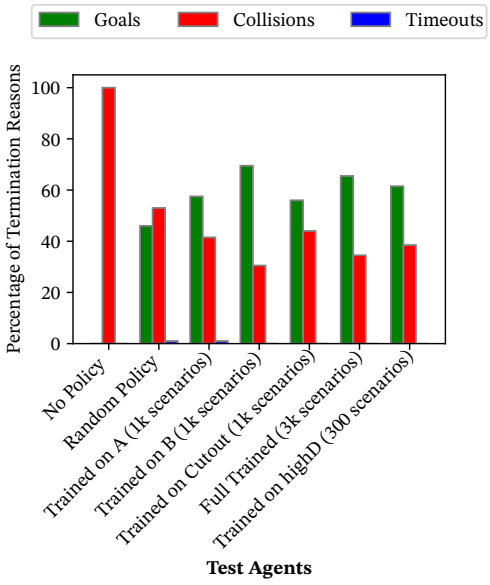
Figure 7.15: Termination reasons moving average rates for A2C training on 300 scenarios from highD



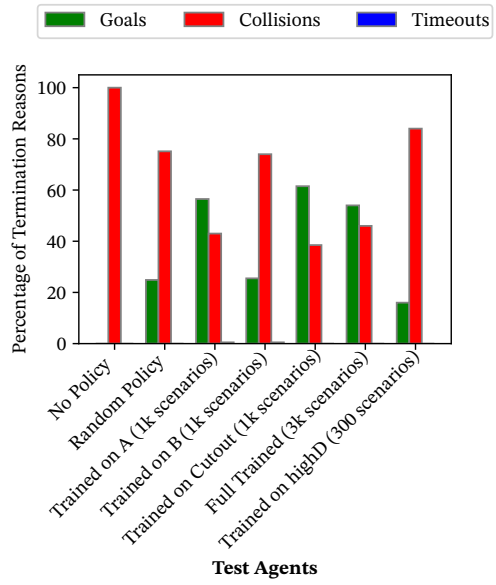
(a) Trained agents tested on highD 50 scenarios including highD trained agent



(b) Trained agents tested on type A scenarios including highD trained agent



(c) Trained agents tested on type B scenarios including highD trained agent



(d) Trained agents tested on type Cutout scenarios including highD trained agent

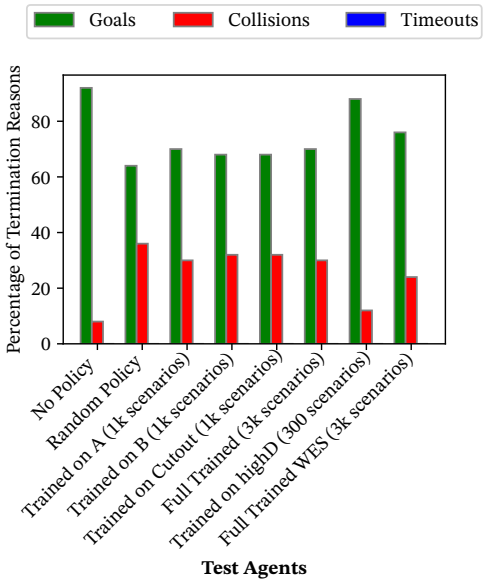
Figure 7.16: Performance of trained agents on highD 50 scenarios and types A, B and Cutout scenarios including highD trained agent

7.4 Easy Scenarios Training Extension

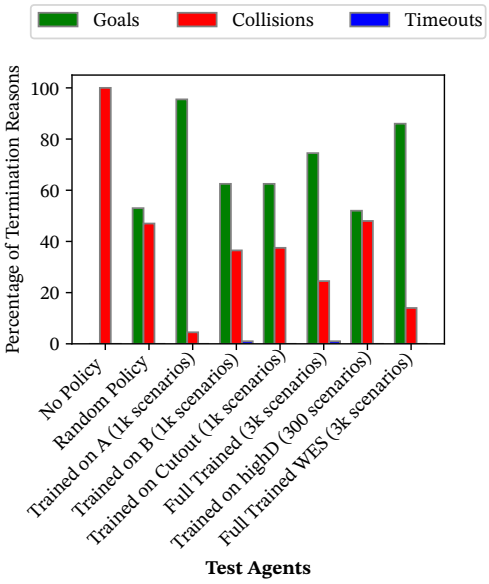
The previous learning process focused on teaching the agent that doing no action is always the worst decision as it always ends up in a collision. To resolve this, further experiments were conducted. This training phase trains a full model on the 3 scenario types with 1000 scenarios each. However, the dataset in this case will include easy scenarios. The easy scenarios are those in which the ego vehicle could reach its goal without having to take any actions. These scenarios existed because not all parameter combinations led to the generation of challenging scenarios. For instance, if the lane change delay (*lc_delay*) was set very high, the ego vehicle could safely pass the challenger vehicle in Scenario Type B before the challenger vehicle even started its lane change. Therefore, this experiment will show whether or not the agent can achieve better results in both challenging scenarios and also real-roads scenarios. The idea behind this is to teach the agent that it does not have to take an action if it is not necessary. The same curriculum learning method applied in Section 7.2.5 will be applied again in this phase.

Results

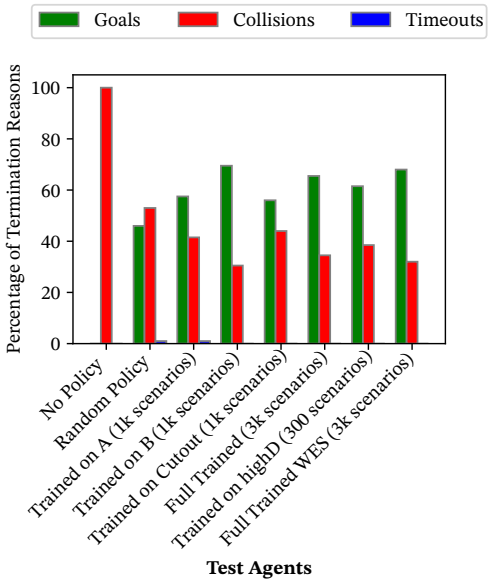
The results of this training on both types A and B together and then training on the types A, B, and Cutout for 1 million timesteps, showed also an incline in the moving average rate of goal reaching that reached over 80 percent while showing a decline in the collision rate that reached less than 20 percent by the end of training. Then the model that was fully trained agent With Easy Scenarios (WES) was tested against the test cases of all scenario types independently as well as those of highD. The results showed in Figure 7.17a the fully trained gained WES could outperform all the other agents on highD test scenarios except for the one trained on highD. For the tests on type A scenarios Figure 7.17b the fully trained agent WES could outperform all the agents except for the one trained on type A scenarios. In Figure 7.17c, the fully trained agent WES also could outperform all the other agents testing on type B except for the agent trained on type B. Lastly, the tests on type Cutout showed in Figure 7.17d that the fully trained agent WES could outperform all the agents except the one trained on type Cutout.



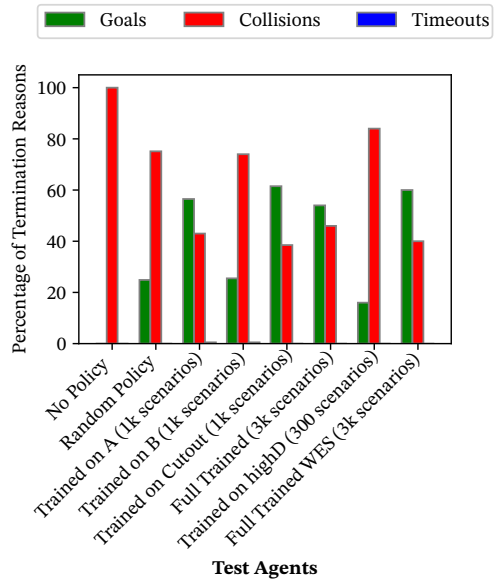
(a) Trained agents tested on highD 50 scenarios WES



(b) Trained agents tested on type A scenarios WES



(c) Trained agents tested on type B scenarios WES



(d) Trained agents tested on type Cutout scenarios WES

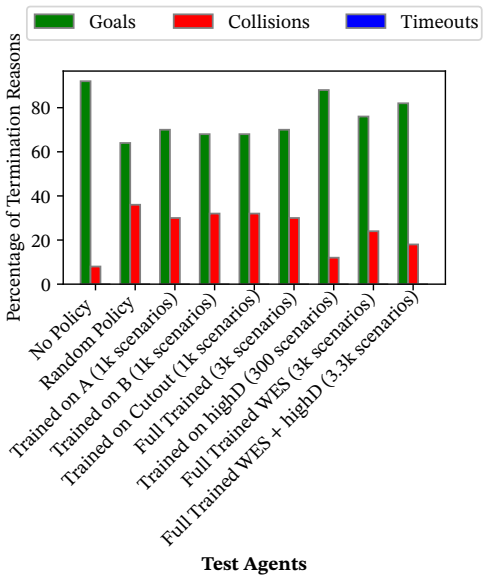
Figure 7.17: Performance of trained agents on highD 50 and types A, B and Cutout scenarios WES

7.5 Real-Roads Scenarios Training Extension

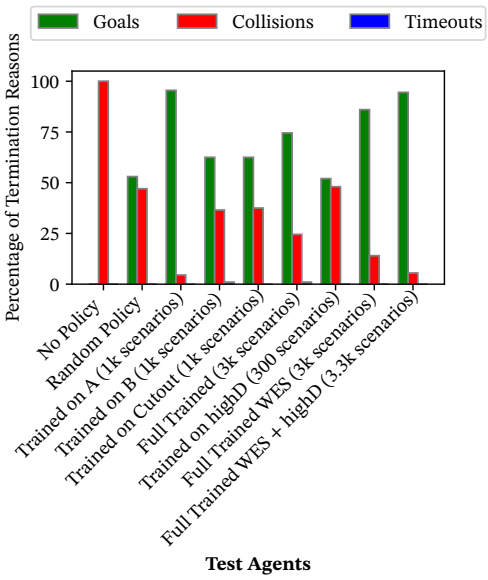
Another aspect that could be extended in the dataset used for training the full model, is the use of real-roads scenarios also in the dataset. The scenarios used in this experiment will be those extracted from highD data. The Full trained agent WES will be given extended training on the new dataset that consists of type A, B, Cutout WES (3000 scenarios) + highD 300 scenarios used for training the highD trained agent. The idea behind this experiment is to include real-roads experience for the learned agent. The training will be extended for 1 million timesteps and then will be compared against all scenario types independently as well as on highD test set.

Results

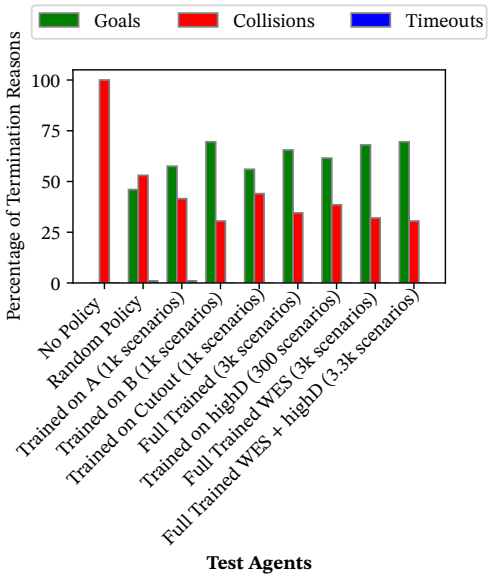
The fully trained against WES + highD that got trained on the extended dataset with easy scenarios as well as highD data could outperform all the other agents testing on highD scenarios as seen in Figure 7.18a. It could even outperform the fully trained WES agent but it could not outperform the no policy agent and the agent trained on highD even though it got very close results to highD trained agent. In Figure 7.18b the fully trained agent WES + highD outperformed all the agents on type A test scenarios and even got a very close performance to that of the agent trained on type A. On type B tests the fully trained agent WES + highD could also outperform all the agents and got very close results to the agent trained on type B as seen in Figure 7.18c. Finally, in the type Cutout tests shown in Figure 7.18d, the fully trained agent WES + highD could outperform all the agents even the one trained on type Cutout scenarios. This demonstrated that training on challenging scenarios mixed with easy scenarios and real-roads improves the model's skills in learning generally across all the test types.



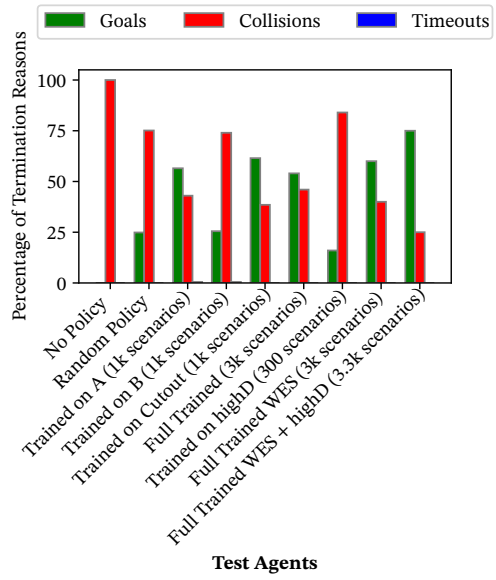
(a) Trained agents tested on highD 50 scenarios WES + highD



(b) Trained agents tested on type A scenarios WES + highD



(c) Trained agents tested on type B scenarios WES + highD



(d) Trained agents tested on type Cutout scenarios WES + highD

Figure 7.18: Performance of trained agents on highD 50 and types A, B and Cutout scenarios WES + highD

8 Conclusion

RL approaches hold great promise in the context of ADS. However, they currently lack safety guarantees as well as efficiency of learning. This thesis aims to tackle these issues by the utilization of HRL mechanism in a scenario-based environment. In this thesis, the architecture of the custom environment SAD-RL was introduced which separates the actions performed by the agent into high-level options and low-level policies according to the temporal abstraction mechanism of HRL. The types of challenging scenarios used in this thesis were demonstrated in detail as well as the method designed in this thesis to generate synthetic scenarios based on these types. The way used in this thesis to extract real-road scenarios was also presented in detail. Furthermore, An existing simulation was used in this thesis to load the scenarios and handle the actions supplied by the low-level policies. The algorithms used in the experiments to assess the requirements of the thesis were DQN, QR-DQN, PPO, and A2C. Multiple algorithms were selected to evaluate the effectiveness of the proposed architecture across different RL algorithms. DQN, QR-DQN, and PPO failed to show a learning pattern in overfitting one scenario on the used custom environment. On the other hand, A2C was able to achieve a learning pattern in independent training on two or more scenarios from each type as well as generalizability of learning in a cross-test. The results of training on a full dataset that included all scenario types also showed a learning pattern and could outperform independent learned agents on each scenario type test except the agent that learned on the same scenario type as the test. To answer the research questions of the thesis, ablation studies were conducted on HRL and challenging scenarios. Ablating HRL showed a weaker learning pattern as well as an inability to compete with the results achieved by HRL learned agents. The ablation of challenging scenarios by learning on real-roads extracted scenarios instead showed a weak performance compared to agents that learned on challenging scenarios when tested on challenging scenarios. On the other hand, real-roads learned agent could outperform all the learned agents when tested on real-roads extracted scenarios. However, when complemented with easy scenarios and real-roads extracted scenarios, the fully trained agent could reach better results in both real-roads tests and scenario types tests that either were as good as the result of the corresponding trained agent or even outperformed it in some cases. Moreover, challenging scenarios have provided a guaranteed generalizability test for the trained agents and thus helped in the generalizability of learning. Therefore, using HRL and challenging scenarios have shown potential in overcoming the safety and efficiency challenges of using RL as well as generalizability in the context of ADS.

Limitations and Future Work

The capacity of this thesis did not include extensive experimenting with the rewarding system in the context of HRL as the dense reward was omitted immediately after hindering the learning performance of the agents which did not allow for experimenting on compliance with traffic rules of the learned agents. In addition, ablation studies on longitudinal shielding were not performed as they did not lie within the scope of the research questions. Exploring these parts can help design a safer ADS that can avoid safety-threatening situations as well as improve compliance with traffic rules. The implementation of this thesis depended on pre-implemented algorithms from stable-baseline3 while using these algorithms in their default form without experimenting with different parameters (fine-tuning). The scope of the thesis was limited to finding an algorithm that shows the potential of the architecture and answers the research question but did not focus on finding the best-performing agent. Therefore, more research can be done on comparing more algorithms that can perform well on the system as well as fine-tuning them to reach the best possible results which can yield a safer and more efficient agent.

A Appendix

Trajectory Plots

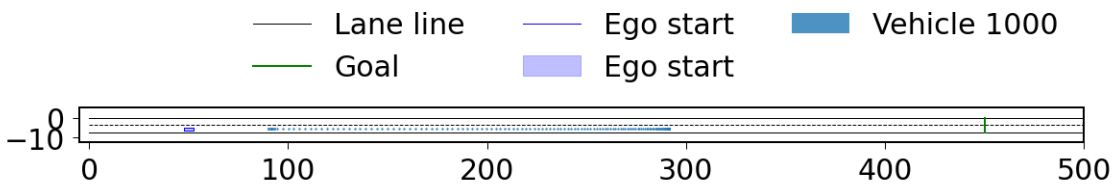


Figure A.1: *scenario_a_v_0_103* trajectory

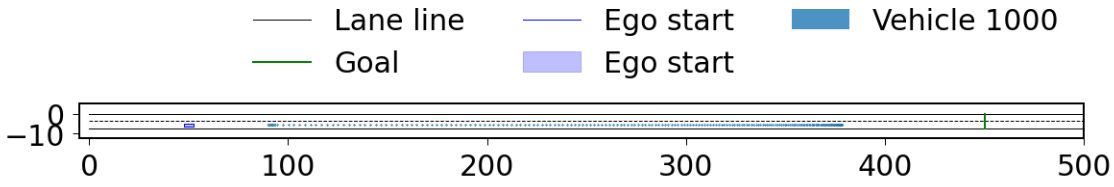


Figure A.2: *scenario_a_v_0_121* trajectory

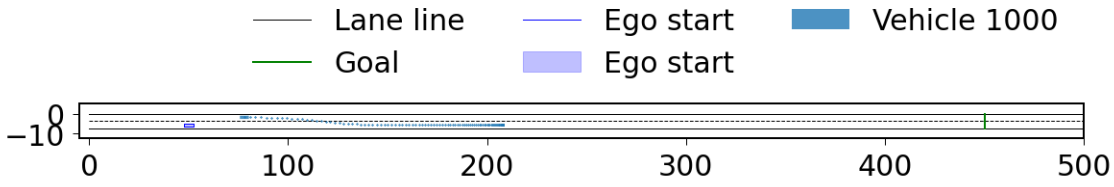


Figure A.3: *scenario_b_v_0_0* trajectory

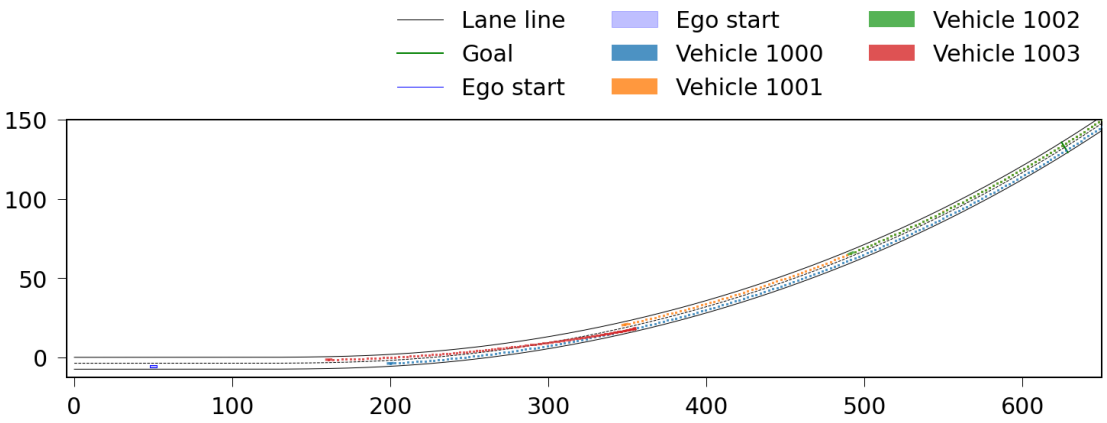


Figure A.4: *scenario_b_v_3_38964* trajectory

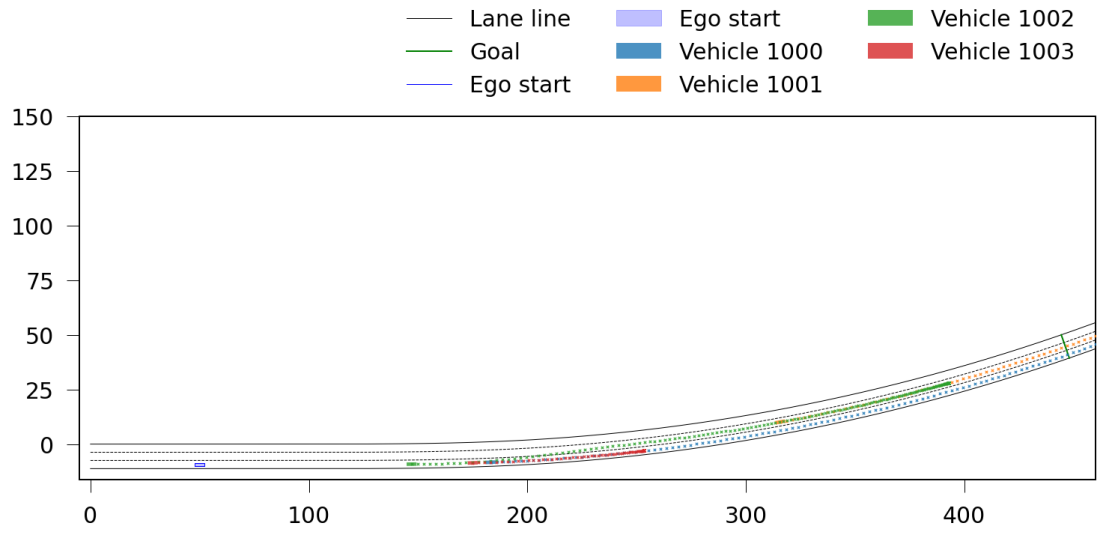
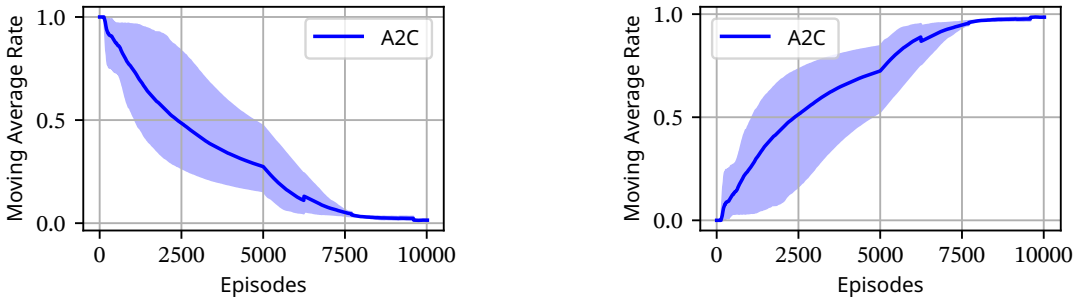


Figure A.5: *scenario_cutout_v_2_55156* trajectory

Training Mean Confidence Interval



(a) Collision Rate

(b) Goal Reaching Rate

Figure A.6: Termination reasons moving average rates for training A2C on *scenario_a_v_0_121* mean and confidence interval on 4 seeds

List of Acronyms

ADS	Automated Driving Systems
DDT	Dynamic Driving Task
HP	Highway Pilot
HRL	Hierarchical Reinforcement Learning
IKA	Institute for Automotive Engineering
IL	Imitation Learning
LCA	Lane Change Assistant
MDP	Markov Decision Process
MPC	Model Predictive Controller
OEDR	Object and Event Detection and Response
RL	Reinforcement Learning
SAD-Pilot	Scenario Automated Driving Pilot
SAD-RL	Scenario Automated Driving Reinforcement Learning
TTC	Time To Collision
WES	With Easy Scenarios

List of Figures

List of Tables

List of References

[1] F. Hartwich, C. Hollander, D. Johannmeyer, and J. F. Krems, "Improving passenger experience and trust in automated vehicles through user-adaptive hmis: "the more the better" does not apply to everyone," *Frontiers in Human Dynamics*, vol. 3, 2021, ISSN: 2673-2726. DOI: 10.3389/fhumd.2021.669030. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fhumd.2021.669030.

[2] M. Kremer, S. Christiaens, C. Granrath, *et al.*, "Scenario- and model-based systems engineering for highly automated driving," *ATZ Worldw*, vol. 122, pp. 16–21, 2020. DOI: 10.1007/s38311-020-0330-x. [Online]. Available: https://doi.org/10.1007/s38311-020-0330-x.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2022. DOI: 10.1109/TITS.2020.3024655.

[5] Society of Automotive Engineers, "SAE J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE International, Standard J3016_202104, 2021. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/.

[6] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 982–988. DOI: 10.1109/ITSC.2015.164.

[7] M. Jarke, X. T. Bui, and J. M. Carroll, "Scenario management: An interdisciplinary approach," *Requirements Engineering*, vol. 3, pp. 155–173, 1998.

[8] S. Geyer, M. Baltzer, B. Franz, S. Hakuli, M. Kauer, M. Kienle, S. Meier, T. Weißgerber, K. Bengler, R. Bruder, *et al.*, "Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance," *IET Intelligent Transport Systems*, vol. 8, no. 3, pp. 183–189, 2014.

[9] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE access*, vol. 8, pp. 87 456–87 477, 2020.

[10] P. Junietz, U. Steininger, and H. Winner, "Macroscopic safety requirements for highly automated driving," *Transportation research record*, vol. 2673, no. 3, pp. 1–10, 2019.

[11] W. Wachenfeld and H. Winner, "The release of autonomous vehicles," *Autonomous Driving: Technical, Legal and Social Aspects*, pp. 425–449, 2016.

[12] H. Weber, J. Bock, J. Klimke, C. Roesener, J. Hiller, R. Krajewski, A. Zlocki, and L. Eckstein, "A framework for definition of logical scenarios for safety assurance of automated driving," *Traffic injury prevention*, vol. 20, no. sup1, S65–S70, 2019.

[13] PEGASUS. "Anforderungen & rahmenbedingungen–stand 4.szenarienbeschreibung." Accessed: 2023-12-15. (2018), [Online]. Available: https://www.pegasu-sprojekt.de/files/tmpl/PDF-HZE/04_Szenarienbeschreibung.pdf.

[14] M. Scholtes, L. Westhofen, L. R. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. H. Bollmann, F. Körtke, J. Hiller, M. Hoss, J. Bock, and L. Eckstein, "6-layer model for a structured description and categorization of urban traffic and environment," *IEEE Access*, vol. 9, pp. 59 131–59 147, 2021. DOI: 10.1109/ACCESS.2021.3072739.

[15] J.-M. Jullien, C. Martel, L. Vignollet, and M. Wentland, "Openscenario: A flexible integrated environment to develop educational activities based on pedagogical scenarios," in *Proceedings of the 2009 Ninth IEEE International Conference on Advanced Learning Technologies*, ser. ICALT '09, USA: IEEE Computer Society, 2009, pp. 509–513, ISBN: 9780769537115. DOI: 10.1109/ICALT.2009.24. [Online]. Available: https://doi.org/10.1109/ICALT.2009.24.

[16] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *Proc. IEEE Intell. Trans. Syst. Conf.*, Hawaii, USA, 2018. [Online]. Available: http://www.mrt.kit.edu/z/publ/download/2018/Poggenhans2018Lanelet2.pdf.

[17] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 719–726. DOI: 10.1109/IVS.2017.7995802.

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

[19] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.

[20] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale, "The next generation simulation program," *Institute of Transportation Engineers. ITE Journal*, vol. 74, no. 8, p. 22, 2004.

[21] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2118–2125. DOI: 10.1109/ITSC.2018.8569552.

[22] T. Moers, L. Vater, R. Krajewski, J. Bock, A. Zlocki, and L. Eckstein, "The exid dataset: A real-world trajectory dataset of highly interactive highway scenarios in germany," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 958–964. DOI: 10.1109/IV51971.2022.9827305.

[23] B. Coifman and L. Li, "A critical evaluation of the next generation simulation (ngsim) vehicle trajectory dataset," *Transportation Research Part B: Methodological*, vol. 105, pp. 362–377, 2017.

[24] M. Montanino and V. Punzo, "Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns," *Transportation Research Part B: Methodological*, vol. 80, pp. 82–106, 2015.

[25] Y. Lin, M. Ratzel, and M. Althoff, "Automatic traffic scenario conversion from openscenario to commonroad," *arXiv preprint arXiv:2305.10080*, 2023.

[26] L. Vater, S. Tarlowski, M. Schuldes, and L. Eckstein, *Determining the tactical challenge of scenarios to efficiently test automated driving systems*, 2024. arXiv: 2404.02599 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2404.02599.

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

[28] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] 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.

[31] A. Barreto, D. Borsa, S. Hou, G. Comanici, E. Aygün, P. Hamel, D. Toyama, S. Mourad, D. Silver, D. Precup, *et al.*, "The option keyboard: Combining skills in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] M. Hutsebaut-Buysse, K. Mets, and S. Latré, "Hierarchical reinforcement learning: A survey and open research challenges," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 172–221, 2022.

[33] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8536–8542.

[34] T. Fraichard, "Dynamic trajectory planning with dynamic constraints: A'state-time space'approach," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, IEEE, vol. 2, 1993, pp. 1393–1400.

[35] M. Saraoglu, H. Jiang, M. Schirmer, İ. Mutlu, and K. Janschek, "A minimax-based decision-making approach for safe maneuver planning in automated driving," in *2023 American Control Conference (ACC)*, IEEE, 2023, pp. 4683–4690.

[36] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

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

[38] M. Igl, D. Kim, A. Kuefler, P. Mougin, P. Shah, K. Shiarlis, D. Anguelov, M. Palatucci, B. White, and S. Whiteson, "Symphony: Learning realistic and diverse agents for autonomous driving simulation," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 2445–2451.

[39] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, *et al.*, "Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 7553–7560.

[40] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[41] M. Vitelli, Y. Chang, Y. Ye, A. Ferreira, M. Wołczyk, B. Osiński, M. Niendorf, H. Grimmett, Q. Huang, A. Jain, *et al.*, "Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 897–904.

[42] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102 662, 2020.

[43] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transportation research part C: emerging technologies*, vol. 97, pp. 348–368, 2018.

[44] J. Lee, T. Kim, and H. J. Kim, "Autonomous lane keeping based on approximate q-learning," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, IEEE, 2017, pp. 402–405.

[45] W. Xia, H. Li, and B. Li, "A control strategy of autonomous vehicles based on deep reinforcement learning," in *2016 9th International Symposium on Computational Intelligence and Design (ISCID)*, IEEE, vol. 2, 2016, pp. 198–201.

[46] X. Ma, K. Driggs-Campbell, and M. J. Kochenderfer, "Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1665–1671.

[47] H. An and J.-i. Jung, "Decision-making system for lane change using deep reinforcement learning in connected and automated driving," *Electronics*, vol. 8, no. 5, p. 543, 2019.

[48] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1379–1384.

[49] T. Liu, Q. Liu, H. Liu, and X. Ren, "Combining deep reinforcement learning with rule-based constraints for safe highway driving," in *2022 China Automation Congress (CAC)*, IEEE, 2022, pp. 2785–2790.

[50] L. Wang, J. Liu, H. Shao, W. Wang, R. Chen, Y. Liu, and S. L. Waslander, "Efficient reinforcement learning for autonomous driving with parameterized skills and priors," *arXiv preprint arXiv:2305.04412*, 2023.

[51] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 2156–2162.

[52] Y. Lin, S. Maierhofer, and M. Althoff, "Sampling-based trajectory repairing for autonomous vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 572–579.

[53] United Nations Economic Commission for Europe, *Un regulation no. 157 - automated lane keeping systems (alks)*, `https://unece.org/sites/default/files/2023-12/R157e.pdf`, Accessed: 2024-07-09, 2023.

[54] C. Pek, V. Rusinov, S. Manzinger, M. C. Üste, and M. Althoff, "Commonroad drivability checker: Simplifying the development and validation of motion planning algorithms," in *2020 IEEE intelligent vehicles symposium (IV)*, IEEE, 2020, pp. 1013–1020.

[55] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, *Gymnasium*, Mar. 2023. DOI: `10.5281/zenodo.8127026`. [Online]. Available: `https://zenodo.org/record/8127025` (visited on 07/08/2023).

[56] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html.

[57] X. Wang, H. Krasowski, and M. Althoff, "Commonroad-rl: A configurable reinforcement learning environment for motion planning of autonomous vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 466–472.

[58] E. Leurent, *An environment for autonomous driving decision-making*, https://github.com/eleurent/highway-env, Accessed: 2024-07-10, 2018.

[59] T. Authors, *Tensorboard: Visualizing learning*, https://github.com/tensorflow/tensorboard, Accessed: 2024-03-25, 2016.

[60] A. Lavin, L. Etter, C. Choy, S. Montgomerie, S. Kumar, and L. Biewald, *Weights & biases*, https://www.wandb.com/, Accessed: 2024-03-25, 2020.