Learning Realistic High Level Decisions for Autonomous Driving at Complex Intersections

Danial Kamran, Martin Lauer and Christoph Stiller*

Abstract: In this work we use deep Q network (DQN) approach to learn high level actions for automated driving at un-signaled intersections. Using lanelet map, history of positions and velocities for vehicles close to the intersection are calculated that provide a generic state representation for the Reinforcement Learning (RL) agent to handle different types of intersections. Moreover, we define safety and utility reward functions and use weighted average of them as total reward in order to evaluate the situation more precisely. The goal is to learn optimal policy which is safe but also not overcautious. According to our experiments, using such reward function, the agent can successfully learn an optimal policy which only stops for the vehicles close to the intersection and will drive fast when vehicles are far from the intersection.

Keywords: Decision Making for Automated Driving, Reinforcement Learning, Un-signaled Intersections, Deep Q Networks

1 Introduction

One of the most important challenges for automated driving in urban areas is navigating through un-signaled intersections. In such situations, there is no traffic light to give way and take way to vehicles and therefore, the vehicles should themselves realize when to pass the intersection without having collisions. Besides safety, utility and comfort are also two important factors during crossing intersection which means the vehicles should prevent being too much cautious or having jerky maneuvers.

Several rule-based solutions have been presented for handling un-signaled intersections. In [1], hierarchical state machines were designed in order to handle different situations at intersections. In most of cases, time-to-collision [2] between ego vehicle and other vehicles is used as the main reasoning feature for handling intersection crossing. Although TTC can provide a safe and reliable solution, it assumes constant velocity for other vehicles and therefore it has similar reaction to vehicles with different intentions. In other words, TTC as the only reasoning factor can not provide information about future intention of vehicles which will result overcautious maneuvers.

Recently reinforcement learning (RL) techniques have been utilized for high level decision making for automated driving [3]–[6]. Generally RL based decision making approaches try to increase utility as much as possible without forgetting about safety of maneuvers. By utilizing plenty of interaction experiences between agent and environ-

*Institute of Measurement and Control Systems, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany (e-mail: {danial.kamran, martin.lauer, stiller}@kit.edu).

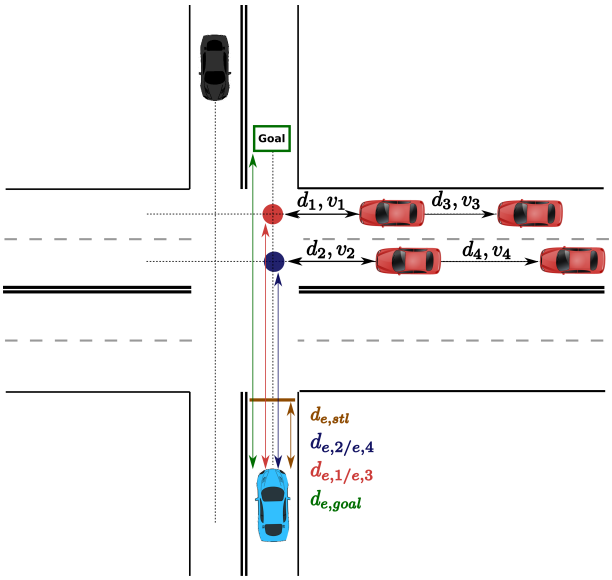

Figure 1: Overview of the yielding scenario and the features we used for situation representation. Blue car is ego vehicle and red cars are vehicles that have collision zone with ego lane. Gray car which has no collision zone with the ego lane is discarded.

ment (simulator), these approaches try to find long term optimal decisions which increase cumulative future reward.

One of the main challenges that still needs to be addressed for RL based automated driving solutions is realistic and generalized problem formulation for better portability in different simulation scenarios and also in real world. Therefore, the features that are being used for learning should be generalized and less dependent on the scenario. Moreover, complexity of learning structure and the amount of required training time for converging to an optimal policy has to be reduced as much as possible in order to prevent damages that the agent is likely to suffer specially due to the so called catastrophic forgetting [7], [8] and also overfitting [9] issues.

The main contribution of this paper is presenting a generic Reinforcement Learning (RL) algorithm for learning optimal behavior to drive through different types of unsignaled intersections safe and also without being conservative. Utilizing Lanelet2 maps [10] with enriched information about geometry of intersections, we propose a parameterized state representation based on the distance and velocities of the vehicles along the curvature (figure 1) which makes it less sensitive to the geometry and structure of intersection.

Another contribution of the proposed approach is learning high level actions as the behavior instead of low level control of the ego vehicle, which makes the RL agent suitable to be used as a conventional decision making module for generating velocity constraints for trajectory planning and control module in an automated driving pipeline (figure 2).

1.1 Related Works

There are some approaches that use reinforcement learning for the yielding scenario similar to this paper [3], [4]. In [3], authors try to learn optimal time or sequence of accelerations for passing occluded intersections using deep Q Networks (DQN). They use top view image

of the intersection marked with location and velocity of vehicles as state representation of their RL algorithm. In the output, they compared three different actions: time to start driving, sequential accelerations and creep-and-go for passing the intersection. The results show small amount of collisions and faster drives comparing to the TTC approach. However, the main disadvantage for this work is using image based state representation which makes the algorithm not generalizable and more sensitive to intersection and road geometries as part of state representation. To overcome this problem, parameterized state representation similar to [4], [5] can be more efficient and make the algorithm more generic. In [4] parameterized state has been used for the same yielding scenario but generating high level decisions in the output instead of time-to-go or acceleration values. Although they provide different techniques to improve network convergence, there is no evaluation on complex intersections where we can see how good the algorithm is generalized. Since there is only safety evaluations of the algorithm by measuring percentage of accidents, we also do not know if the presented algorithm is overcautious or not and how fast can drive through intersections.

2 Problem Statement

As explained in 1, we assume that the automated vehicle is supposed to cross an unsignaled intersection without any traffic light. We assume that there is one stop line behind intersection and the ego vehicle can stop and wait behind that line until the situation become safe for crossing. The goal is to cross the intersection without blocking oncoming vehicles on the other lanes. For that purpose, the RL agent decides about the action given to trajectory planning and control module during crossing the intersection (figure 2). The action can be stop, drive fast or drive slow (crawl) and is selected using DQN network (2.2). The input for the DQN is other vehicles velocities and distances to the collision point and also the ego vehicle velocity and distance to collision points and the stop line. The main challenge is to provide a generalized approach that can handle a variety of intersections with different number of lanes and structures.

2.1 Decision Making and Planning Pipeline

Figure 2 depicts the structure of decision making and planning pipeline that we present in this paper. Decision making module gets the information about upcoming intersections from the Lanelet map [10]. This information includes distance to the stop line and crossing lanes at the intersection. A Lanelet matching module compares the position of detected vehicles from perception with the crossing lanes and keeps those that have collision zone with our vehicle. Distance and velocity along curve for these vehicles are then sent to the RL module in order to choose the best optimal action as velocity constrain for the ego vehicle. Since the path in our scenario is fixed, trajectory planning only performs speed control in order to follow velocity constraints generated by the DQN module.

2.2 Reinforcement Learning with Deep Q Networks

Reinforcement Learning (RL) is used to solve different problems having interaction with the environment specifically for robotics. It models the problem as Markov Decision

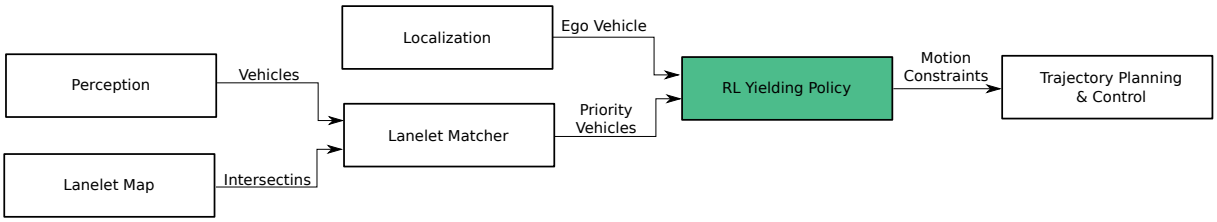

Figure 2: Overview of decision making and planning pipeline used in this paper for driving through intersections in urban area.

Process (MDP). RL agent selects the best action (a_t) for the current state (s_t) which provides the highest expected cumulative reward. The action is executed and the environment changes to a new state (s_{t+1}). According to the reward function defined for the RL agent, the action that was selected is evaluated and its reward value will be provided as $r(s_t, a_t)$ according to the new situation in the whole environment. After recording several experiences through interactions with the environment, the agent should learn the best actions for each state which will have highest discounted future reward:

$$R_t = \Sigma_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i) \tag{1}$$

In order to find the best policy, action-value function helps to find out the cumulative discounted reward of each action for each state:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r_t + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))] \tag{2}$$

If we assume that we have optimal Q function, a greedy policy selects actions with maximum Q value as the best action for each state [11]:

$$\mu(s_i) = arg\, max_a\, Q(s_i, a|\theta^Q) \tag{3}$$

The Q function can be approximated using deep neural networks known as Deep Q Networks (DQN)[12]. Assuming θ^Q as the parameters of the Q function estimator, the Q function is learned through minimizing the loss function using B random samples from reply buffer:

$$L(\theta^Q) = \sum_{i=1}^{B} (y_i - Q(s_i, \mu(s_i)|\theta^Q)) \tag{4}$$

where y_i is the network training target:

$$y_i = r(s_i, a_i) + \gamma Q(s_{i+1}, \mu(s_{i+1})|\theta^Q) \tag{5}$$

3 Learning Yielding Policy at Intersections

3.1 Parameterized State Representation

In this section, the main structure of reinforcement learning algorithm for finding best optimal actions during driving at the intersection will be presented. As explained in section 2.1, using lanelet map, the distance and velocity for ego vehicle and all other vehicles at intersection will be calculated along their path as one dimensional values.

Also the ego vehicle distance to the stop line will be required for full representation of the situation for the RL agent. All of these values are normalized assuming maximum distance and maximum velocity of objects in our experiments. Finally, the situation for the RL agent at time step $t$is represented as below:

$$situation_t = \begin{bmatrix} d_{e,stl} & d_{1,e} & d_{2,e} & ... & d_{n,e} \\ v_e & v_1 & v_2 & ... & v_n \\ d_{e,goal} & d_{e,1} & d_{e,2} & ... & d_{e,n} \end{bmatrix}^T$$

The first row in the situation represents distance of ego vehicle to the stop line, velocity of ego vehicle and its distance to the goal as $d_{e,stl}$, v_{el} and $d_{e,goal}$ respectively. In the remaining rows, the distance of each vehicle to the collision zone with ego vehicle, its velocity and the distance of ego vehicle to this vehicle are represented as $d_{i,e}$, v_i and $d_{e,i}$ respectively where i is the id of vehicle. It should be noted that using this representation, only N vehicles can be feed into DQN network. Therefore, when there are more than n vehicles at intersection which have collision zone with ego lane, the ones with highest criticality c_i will be selected where c_i for each vehicle is calculated as below:

$$c_i = 1 - \frac{\sqrt{d_{i,e}^2 + d_{e,i}^2}}{\sqrt{2}} \tag{6}$$

Using this equation 3.1, the vehicles which have smaller distance to the collision zones and closer to ego vehicle will be more critical and are selected among the others with lower criticality. When the number of vehicles is lower than N, we fill the rows of representation with $1, 0, 1$ values meaning virtual cars at maximum distance with zero velocity that can help the DQN network to neglect them.

In order to give more information about previous distances and velocities for ego and other vehicles for better reasoning, we provide history of situations for the RL algorithm as our final state representation:

$$s_t = \begin{bmatrix} situation_t & situation_{t-1} & situation_{t-2} & situation_{t-3} & situation_{t-4} \end{bmatrix}$$

3.2 Action Representation

In the output, the reinforcement learning policy will choose about the vehicle behavior at intersection. Every decision can be interpreted as a high level action which will be sent for the trajectory planner as a velocity constraint:

- Stop: Full stop with maximum deceleration (a_{min})
- Drive-fast: Reach v_{fast}
- Drive-slow: Reach v_{slow}

3.3 Reward Function

The reward function that is used for the proposed DQN algorithm will qualify the status of ego vehicle and other vehicles at the intersection in terms of safety and utility. For

that, two factors $R_{safety}(v_i)$ and $R_{utility}$ are defined, where $R_{safety}(v_i)$ qualifies safety of ego vehicle with respect to vehicle v_i and $R_{utility}$ calculates the reward according to ego vehicle velocity. The final reward given to the RL algorithm is calculated as weighted average of these two factors:

$$R_{total} = \lambda_s[\min_{0<i\leq n} R_{safety}(v_i)] + \lambda_u R_{utility} \tag{7}$$

We will explain each of these two reward factors in the remaining parts of this section.

3.3.1 Utility Reward

The utility reward will only qualify the velocity of the ego vehicle:

$$R_{utility} = \frac{v_{ego}}{v_{des}} \tag{8}$$

3.3.2 Safety Reward

Utility factor of the reward function will motivate the RL agent to drive as fast as possible (i.e. $v_{ego} = v_{des}$) at the intersection. However, in order to yield for other vehicles which are close to the intersection, safety reward will penalize critical situations and force the RL agent to reduce the speed. If the ego vehicle is leaving the intersection, safety reward will penalize situations where the ego vehicle is not able to leave before other vehicles enter the intersection. Therefore, two safety conditions are defined to penalize each of these situations. It should be noted that only one of these two conditions need to be valid for the ego vehicle in order to be safe:

- **Safe Stop (SS):** If the ego vehicle has the possibility to stop before entering collision zone.
- **Safe Leave (SL):** If the ego vehicle has the possibility to leave the collision zone before other vehicles can enter there.

A safety gap for each condition is calculated which should be bigger than a minimum value to be completely safe and the safety reward for that condition become 0 (maximum safety reward). If the safety gap is smaller than critical value, it means the ego vehicle is completely unsafe and the safety reward would be -1 (minimum safety reward). For other cases where the ego vehicle is not completely safe or unsafe, the reward will be calculated as below:

$$R_{SS(v_i)} = \begin{cases} -1 & \text{if } d_{SG} < d_{critical} \text{ (unsafe situation)} \\ 0 & \text{if } d_{SG} > d_{min} \text{ (fully safe situation)} \\ -(\frac{d_{SG}-d_{e,stl}}{d_{e,stl}-d_{critical}})^2 & \text{else} \end{cases} \tag{9}$$

$$R_{SL(v_i)} = \begin{cases} -1 & \text{if } t_{SG} < t_{critical} \text{ (unsafe situation)} \\ 0 & \text{if } t_{SG} > t_{min} \text{ (fully safe situation)} \\ -(\frac{t_{SG}-t_{min}}{t_{min}-t_{critical}})^2 & \text{else} \end{cases} \tag{10}$$

Figure 3: Top view images from the simulation during one episode. Ego vehicle (red vehicle) stops behind stop line in order to yield to other vehicles (image left), starts driving through the intersection (middle image) and reaches the goal point (right image).

Where d_{SG} and t_{SG} are the safety gaps for SS and SL conditions respectively. Also d_{min} and t_{min} are minimum required safety gap to be completely safe for each condition.

For each vehicle (v_i), one of the safety conditions should be valid to make it a complete safe situation, i.e., ego vehicle should either be able to stop behind collision zone with v_i or leave that collision zone before v_i can enter there. Therefore, maximum of $R_{SS}(v_i)$ and $R_{SL}(v_i)$ is selected as final safety reward according to that vehicle:

$$R_{safety}(v_i) = \max(R_{SS}(v_i), R_{SL}(v_i)) \tag{11}$$

Finally, we take minimum safety reward for all vehicles as the total safety reward:

$$R_{safety} = \min_{0<i\leq n} R_{safety}(v_i) \tag{12}$$

In this way we make sure that for each vehicle at least one of the safety conditions is valid and otherwise the ego vehicle is not safe.

4 Training and Evaluation

4.1 Simulation Environment

In order to learn the proposed DQN approach and also evaluate it, we use Carla simulator [13] to simulate automated driving through an un-signaled intersection with random vehicles driving at the other sides of intersection (figure 3). At the beginning of each training episode, ego vehicle and random number of other vehicles are positions at random distances from the intersection. Each vehicle has random desired speed and is randomly assigned to drive on one of intersection lanes. The position of stop line and also geometry of all intersection lanes are mapped to be used for situation representation as explained in section 2.1.

4.2 Training Setup

Figure 4 shows overall structure of DQN network. In the input, current state (s_t) is processed using h_ego and h_{vi} hidden layers for motion features of ego vehicle and other vehicles accordingly. Similar to [4], we used shared weights for processing all sub layers of other vehicles (w_{veh}) in order to be independent from the order of vehicle features in

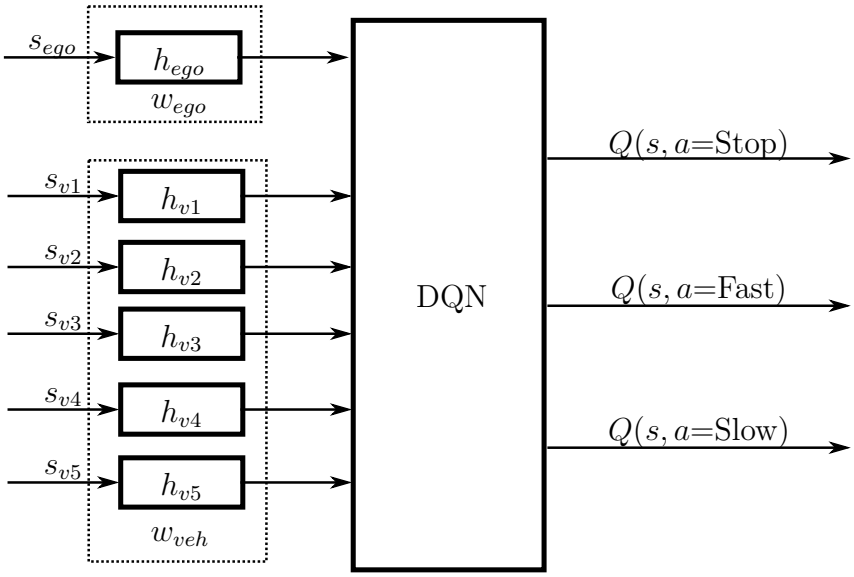

Figure 4: Structure of proposed feature extraction and DQN network. All hidden layers for processing intersecting vehicles (h_{v_i}) share same weights (w_{veh}) and ego vehicle data is processed by a separate hidden layer (h_{ego}). All extracted features are then concatenated as the input of DQN network.

the state. After first hidden layer, all extracted features for ego vehicle and other vehicles are concatenated and fed into the DQN network in order to estimate expected cumulative reward for each action at the current state ($Q(s_t, a_i)$).

4.3 Evaluation

After training more than 1000 episodes with 600,000 experiences, the learned policy reached collision free results for 10 test scenarios. As an example, figure 6 shows which actions have highest Q values using the trained policy at different distances and velocities of ego vehicle ($d_{e,goal}$ and v_e) at an intersection without any other vehicle. According to this figure, the policy drives slow when the ego vehicle is far from the intersection (to be careful if any vehicle enters the intersection area) and when it gets closer starts to drive fast. For better evaluating the trained policy, we compared it with a rule-based policy for 10 evaluation scenarios. The rule-based policy selects the highest velocity at each decision time which is safe according to the safety conditions explained in section 3.3.2. As it is visible in figure 5, performance of learned agent is close to the rule-based agent and in some cases it outperforms the rule based decisions.

5 Conclusions

In this paper, a DQN network was proposed as a decision making module in order to learn optimal high level actions for automated driving through un-signaled intersections. Defining safety and utility terms in the reward function, we tried to learn actions which are safe and also not too much conservative as the output of DQN network. Results show that the agent can learn optimal behaviors in order to drive as fast as possible. However,

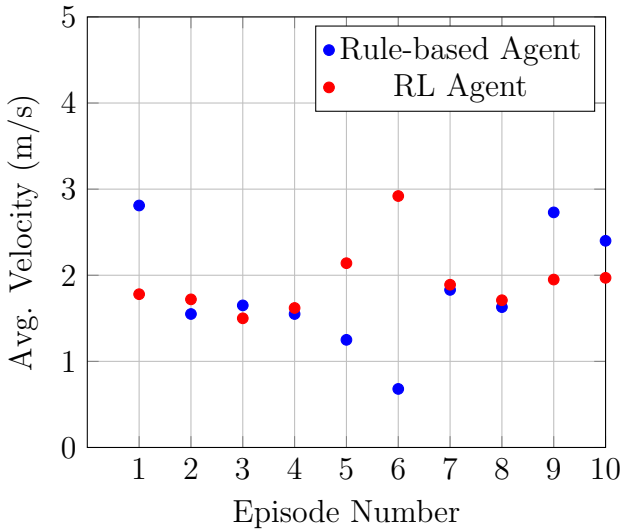

Figure 5: Comparing trained policy and rule based policy for 10 test scenarios.

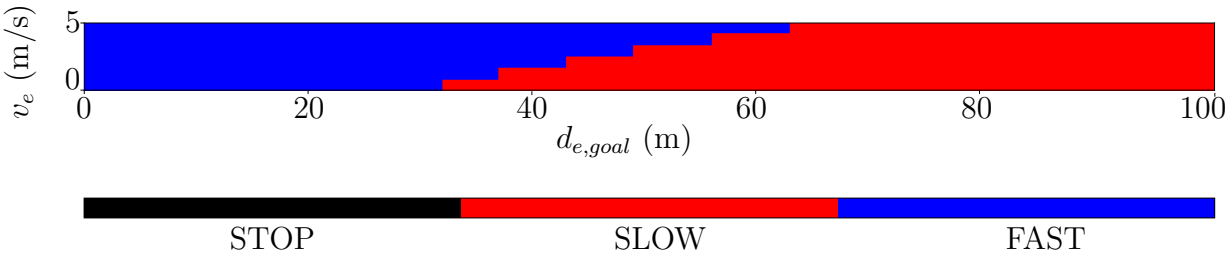

Figure 6: Evaluating the best action ($arg\,max_a\, Q(s,a)$) for different distances to the goal and velocities ($d_{e,goal}$ and v_e) of ego vehicle assuming no other vehicle at the intersection.

there are still some challenges that should be addressed in the future works specifically about safety verification of the DQN decisions in order to provide a completely safe and also not overcautious policy.

6 Acknowledgment

This research is accomplished within the project "UNICARagil" (FKZ 6EMO0287). We acknowledge the financial support for the project by the Federal Ministry of Education and Research of Germany (BMBF).

References

[1] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, *et al.*, "Making Bertha Drive–An Autonomous Journey on a Historic Route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[2] R. van der horst and J. Hogema, "Time-to-collision and collision avoidance systems," Jan. 1994.

[3] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[4] T. Tram, A. Jansson, R. Grönberg, M. Ali, and J. Sjöberg, "Learning negotiating behavior between cars in intersections using deep q-learning," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[5] 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)*, 2018.

[6] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving," *arXiv e-prints*, arXiv:1610.03295, 2016.

[7] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in, ser. Psychology of Learning and Motivation, G. H. Bower, Ed., vol. 24, Academic Press, 1989, pp. 109 –165.

[8] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[10] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, *et al.*, "Lanelet2: A high-definition map framework for the future of automated driving," in *Proc. IEEE Intell. Trans. Syst. Conf.*, Hawaii, USA, 2018.

[11] C. J.C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[12] V. Mnih and D. Silver, "Playing Atari with Deep Reinforcement Learning," 2013. arXiv: `1312.5602`.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.