Machine Learning for Cooperative Driving in a Multi-Lane Highway Environment

Aashik Chandramohan, Mannes Poel, Bernd Meijerink, Geert Heijenk
Department of Computer Science, University of Twente, The Netherlands
{a.chandramohan, m.poel, bernd.meijerink, geert.heijenk}@utwente.nl

***Abstract*—Most of the research in automated driving currently involves using the on-board sensors on the vehicle to collect information regarding surrounding vehicles to maneuver around them. In this paper we discuss how information communicated through vehicular networking can be used for controlling an autonomous vehicle in a multi-lane highway environment. A driving algorithm is designed using deep Q learning, a type of reinforcement learning. In order to train and test driving algorithms, we deploy a simulated traffic system, using SUMO (Simulation of Urban Mobility). The performance of the driving algorithm is tested for perfect knowledge regarding surrounding vehicles. Furthermore, the impact of limited communication range and random packet loss is investigated. Currently the performance of the driving algorithm is far from ideal with the collision ratios being quite high. We propose directions for additional research to improve the performance of the algorithm.**

***Index Terms*—Autonomous driving, Cooperative driving, Reinforcement Learning, Q learning, SUMO, Highway environment**

I. Introduction

With the current developments in technology, autonomous vehicles have become a reality. Autonomous vehicles are the type of vehicles that are capable of sensing their environment and navigating without human input [1]. One of the approaches for realizing autonomous driving, is to use cooperative driving based on vehicular networking for obtaining information regarding surrounding vehicles. In cooperative driving vehicles can communicate information such as their current position and velocity with each other. A driving algorithm can then learn based on the information received and the actions performed by the autonomous vehicle such that over time the autonomous vehicle starts taking the correct actions to move through the traffic.

There is already some research done that involves controlling an autonomous vehicle in a highway environment. Shota Ishikawa et al. discuss how cooperative learning among autonomous vehicles can reduce traffic jams [2]. In [3], Xin Li et al. discuss an approach of using reinforcement learning for controlling an autonomous vehicle in a 2-lane highway environment using actions of lane change and constant acceleration and deceleration by the autonomous vehicle.

This paper investigates how and to what extent reinforcement learning can be used for cooperative driving to control an autonomous vehicle in a multi lane highway environment. It also investigates how the performance of the driving algorithm is affected by the communication range and by random packet loss in the underlying vehicular networking. In order to do so, a simulated traffic environment is set up.

This paper is organized as follows. In Section II, we explain how we designed, trained, and tested the machine learning algorithms for cooperative driving. The design of the algorithm itself is discussed in Section III. Section IV discusses test results for our algorithms, after which we draw some conclusions, and sketch how to continue this research in Section V.

II. Designing, training, and testing machine learning for cooperative driving

Machine learning algorithms require extensive training and testing. Because the cost and risks of training algorithms in real cars in real traffic scenario's is prohibitive, we have set up a simulated traffic environment. We use SUMO (Simulation of Urban Mobility) [4] to create traffic scenario's, where non-autonomous vehicles are behaving according to certain driver models. Using its TraCI (Traffic Control Interface) interface, one or more autonomous vehicles can be controlled using the machine learning algorithm under development. Our machine learning algorithms are developed in Python, using Keras [5] as a high level neural network API, on top of the Tensorflow library for dataflow graphs [6].

In the remainder of this section we will first explain why and how we use reinforcement learning as the machine learning algorithm of choice. We will then give some background information on Q learning, the used reinforcement learning type, in Section II-B. Finally, we will explain how we trained and tested the designed algorithms in Section II-C.

A. Reinforcement Learning

Our driving algorithms use reinforcement learning because it is a type of learning algorithm that adapts based on changes in the environment [3]. Fig. 1 shows the interaction between the driving algorithm and the simulation environment. The block diagram of reinforcement learning taken from [7] is shown inside the driving algorithm. Here the environment is the unit that interacts with the simulation environment and defines the input state, S_t at time t for the the agent. Based on S_t, the agent takes an action A_t, which is passed on to the simulation environment by the environment in the driving algorithm. Due to the taken action, there would be some change in the simulation environment which leads to a new state S_{t+1}. Based on whether the outcome of the action was

978-1-7281-0117-0/19/$31.00

favourable or not, rewards are given (positive for favourable actions and negative rewards for unfavourable actions). The accumulated rewards are used to train the agent with the aim of the agent being able to take the actions that lead to the maximum overall rewards.

B. Q Learning

Q learning is a type of model-free reinforcement learning. It provides agents with the capability of learning to act optimally by experiencing the consequences of actions, without requiring them to build maps of the environment domain [8]. Here the actions are taken based on the Q value for the state action pair. The Q value is the expected discounted reward for executing action A from state S and following the best policy to select the next action [8]. The action contributing to the maximum Q value from that state is chosen as the action of the driving agent for the corresponding input state. As at the start of the training the driving agent is unaware of the correct actions to be taken, initially a large portion of the actions are taken at random, according to an Exploration rate (ϵ), and the Q table is updated based on the calculated Q values. With progress in training the exploration rate is decreased so that actions are taken more based on the Q values and not at random. For a driving environment the number of possible input states are quite high. Hence it is difficult to maintain a Q table. Therefore a Deep Q Network (DQN) is used in the driving agent. In DQN the input state is passed on to a neural network which consists of different nodes in different layers and the nodes in the output layer correspond to the possible actions of the driving agent. During the training period the weights of these nodes are modified such that the nodes of the output layer correspond to the Q values for the actions.

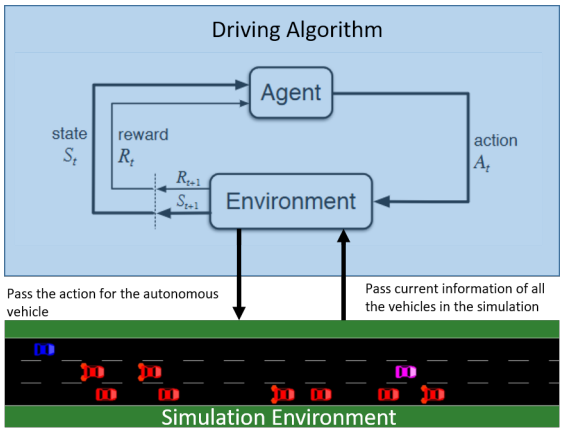

Fig. 1: Interaction between driving algorithm and simulation environment.

C. Training and Testing

a) Scenario: A 3-lane and a 2-lane highway environment is set up using SUMO. Only the autonomous vehicle is controlled by reinforcement learning. All other vehicles in the simulation are controlled by SUMO itself. All vehicles are considered to be of the same length of 3 meters. Other vehicles are inserted into the simulation at random times following a Poisson distribution, a slower moving type of vehicle which the autonomous vehicle can overtake, and a faster moving type of vehicle with maximum velocity similar to the autonomous vehicle. The probability of inserting the slower moving vehicle (max speed = 11.1m/s) into the simulation is 0.1 per second and for the other type of vehicle (max speed = 55.55m/s) it is set to 0.01 per second. All these vehicles are configured to follow all traffic rules (a.o. driving on the right, no right overtaking) and maintain a minimum distance with the vehicle in front. This means that none of these vehicles would initiate a collision and hence collisions could only be caused by the autonomous vehicle. In the autonomous vehicle, a collision avoidance system external to the learning algorithm is used which detects if the vehicle in front is very close and automatically brakes to avoid rear end collisions by the autonomous vehicle. This is used as trying to avoid collisions using just the learning algorithm was ineffective with the number of collisions being very high. Hence now the collisions are only possible during lane changes. The speed limit on the highway is set to 22.22m/s.

Each episode of a simulation consists of 160 time steps with each time step representing 1 second. The autonomous vehicle is entered into the simulation at the 60^{th} time step, so that there will already be vehicles on the road before the autonomous vehicle. As the learning algorithm is for controlling the autonomous vehicle, it is active only while the autonomous vehicle is in the simulation. In case of a collision the current episode is ended.

b) Training the Driving Agent: Experience Replay is used for training the neural network in the driving agent. The input state, new state, action, reward and the episode end status during each time step is saved in memory and a number of these past experiences are chosen at random to train the neural network [9] as consecutive time steps are highly correlated.

The exploration rate is set to a high value of 0.9 at the start of training and it is decreased exponentially during each time step by a factor of 0.9992 such that by the end of the training most of the actions by the agent are taken based on the maximum Q value for the actions.

c) Testing the Driving algorithm : The driving algorithm is also tested in random traffic scenarios similar to the training. The difference between training and testing is that, in training actions are also taken at random whereas in testing actions are only taken based on the Q values.

III. COOPERATIVE DRIVING ALGORITHM DESIGN

In this section, we introduce the design of the cooperative driving algorithm, based on Q learning with a DQN. In the following subsections, we define the basic elements of our Q learning algorithm: (1) actions and (2) input features of the driving agent, and (3) the reward mechanism for its training.

A. Actions of the driving agent

Possible actions for a vehicle in a highway environment are:
1) Change to the left lane.

2) Change to the right lane.
3) Accelerate.
4) Decelerate.
5) Idle Action.

The idle action states that the autonomous vehicle can remain in the same lane and with the same velocity that it had during the previous time step.

B. Input features of the driving agent

As the main actions of the vehicle in a highway environment is either changing lanes or changing its velocity, in this design the autonomous vehicle is only using the vehicle information of the vehicles immediately surrounding it as shown in Fig. 2. The input consists of the current distance, d_{i}, to and the velocity, v_{i}, of each of the surrounding vehicles ($i = 1, 2, 3, 4, 5, 6$), where $i = 1$ and $i = 2$ denote the vehicle in front and behind in the same lane, $i = 3$ and $i = 4$ denote the vehicle in front and behind one lane to the left, and $i = 5$ and $i = 6$ denote the vehicle in front and behind one lane to the right. If such a lane to the left or to the right does not exist because the autonomous vehicle is already driving in the leftmost or rightmost lane, the relevant distance is set to a maximum value, and the speed is set to 0. Finally, the current state of the autonomous vehicle, i.e., its velocity, v_a, acceleration rate, a_{a}, and lane index, L_{a}, are used as input.

C. Reward mechanism for training the driving agent

Rewards are the most important part of reinforcement learning. The effectiveness of the learning algorithm depends on the reward mechanism used. Based on the reward system, the driving agent can be very cautious, trying to avoid collisions by moving really slow, or it can be very aggressive by trying to reach maximum speed and reduce travel time at all cost. Hence it is important to design the reward mechanism properly. Rewards are given such that the autonomous vehicle tries to follow the basic traffic rules and initiates an overtake whenever possible. A high negative reward is given in case of a collision, as it is the most unfavorable result. Also a negative reward is given if the velocity of the vehicle is zero to discourage the driving agent from stopping the vehicle in the middle of the highway. Negative rewards are also given if the vehicle is close to the vehicle in front or if it is in the overtaking lanes unnecessarily, i.e. if there are no vehicles nearby in the lower lane. To encourage the vehicle to drive at the speed limit, a positive reward is given if it maintains the speed limit and a negative reward is given when its velocity goes over the speed limit. Also positive rewards are given if the vehicle tries to overtake slower moving vehicles. The full

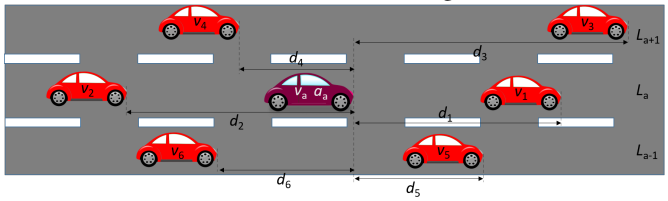

Fig. 2: Input features of the driving algorithm.

TABLE I: Reward Calculation

S.No	Reward	Reward Condition
1	-101	Collision
2	-50	Else & $v_{\text{a}} = 0$
3	-5	Else & $L_a \neq L_{\text{max}}$ & $d_1 < d_{\text{proximity}}$
4	$50 - d_5$	Else & $L_{\text{a}} \neq L_0$ & $d_5 < d_{\text{proximity}}$ & $a_{\text{a}} > 0$
5	$-1.5 \times d_5$	Else & $L_{\text{a}} \neq L_0$ & $d_5 > d_{\text{proximity}}$
6	0.5	Else & $L_{\text{a}} = L_{\text{max}}$ & $d_1 < d_{\text{proximity}}$ & $a_{\text{a}} < 0$
7	-0.5	Else & $L_{\text{a}} = L_{\text{max}}$ & $d_1 < d_{\text{proximity}}$ & $a_{\text{a}} > 0$
8	-1	Else & $v_{\text{a}} > SpeedLimit$
9	1	Else & $a_{\text{a}} > 0$
10	2	Else & $v_{\text{a}} = SpeedLimit$
11	0	Else

list of rewards and the conditions used is given in Table I. Here, L_0 is the rightmost lane of the road, and L_{max} is the leftmost lane. $d_{\text{proximity}}$ is the distance between the vehicles that is considered to be too close. Detailed motivation for the values of the rewards is explained in [10].

IV. Evaluation

In this section, we will evaluate the performance of the driving algorithm. We have trained and tested our agent in both a 2-lane and a 3-lane highway. After extensive experimentation, we choose a training period of 7000 episodes, and a testing period of 500 episodes. We also experimented with different number of hidden layers and different number of nodes per layer for the DQN. The results presented here are for 3 hidden layers with 1500 nodes each, which seems to give most consistent performance. In the Q learning, we used a learning rate $\alpha = 0.001$, and a discount factor $\gamma = 0.9$. We measure the percentage of training or test episodes resulting in a collision and the average speed during the episodes. More extensive results can be found in this report [10].

The cumulative collision percentage during the training period of the driving agent in a 3-lane scenario is shown in Fig. 3. It can be seen that initially almost all episodes result in a collision. The collision percentage starts decreasing as the exploration rate (ϵ) starts to decrease from 0.5, as then the actions are taken mainly based on the Q values. Towards the end of the training period, the cumulative collision percentage decreases to around 30%. At this point, only a rather small percentage of the new episodes result in a collision.

The results after training has been completed, during the test episodes, are summarized in Table II. From these results, it can be observed that our current results are far from acceptable for a real autonomous vehicle. Furthermore, it can be observed that the collision percentage in a 2-lane environment

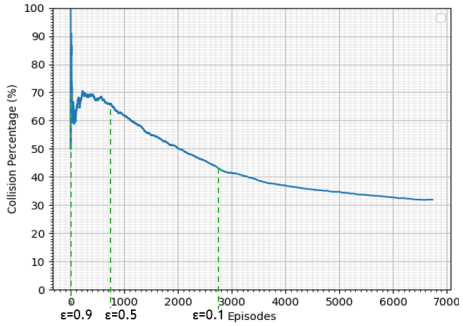

Fig. 3: Cumulative collision percentage during the training period.

TABLE II: Test Results

	95% conf. interval of collision %	average speed
3-lane scenario	$[8\% - 14\%]$	18.4 m/s
2-lane scenario	$[1.5\% - 4.8\%]$	13.4 m/s

is significantly better than in a 3-lane environment. This is due to the fact that collisions result from lane changes, and in a 2-lane highway, there are fewer opportunities for lane change. Because of this the autonomous vehicle also was slowed down by a slow vehicle more often, resulting in lower average speed.

We also experimented with limiting the communication range of the autonomous vehicle as that could be one of the major issues in cooperative driving. To this end, the driving agent, which was trained in a regular situation would only have accurate information regarding surrounding vehicles when they are within the communication range. Below a communication range of 70 meters, collision percentages became very high. Above 70 meters, results were in the range given in Table II. Next, we also experimented with packet loss. In this case, the agent, trained in a situation without packet loss, would at random occasions not have accurate information regarding surrounding vehicles. We used an error concealment technique, where the agent would be fed the last known values of d_i and v_i in that case. The results of this experiment for a 3-lane highway are shown in Figure 4. Here, it can be observed that the collision percentage of the agent is decreasing with increasing packet loss probability. The reason for the collision percentage to be 0% when no communication is received at all is because, in such a case the driving agent believes that there are no vehicles in the vicinity and hence drives the vehicle only in the driving lane. When a vehicle is encountered in the front, the external collision avoidance system in the vehicle becomes active and prevents a rear-end collision. The corresponding average speed measured for the autonomous vehicle at such a scenario was found to be very low at 9m/s. The exact reason for fewer collisions at intermediate values for the packet loss probability is still unclear. At these values, the autonomous vehicle does achieve a reasonable average speed.

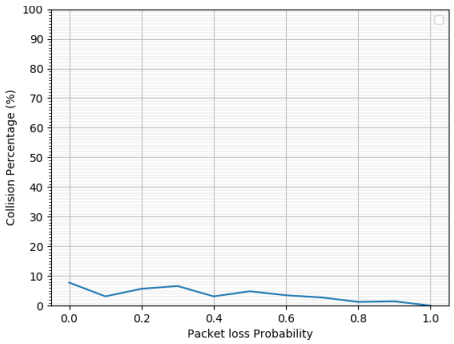

Fig. 4: Collision percentage for varying packet loss probability.

V. CONCLUSIONS AND FUTURE WORK

The main goal of this research was to investigate to what extent reinforcement learning can be used with cooperative driving in a highway environment. The collision percentage achieved with this design is currently prohibitively high and hence further design improvements need to be carried out to decrease the collision ratio. Another goal was to find how the driving algorithm performed with change in communication range and with packet loss during communication. It was found that communication range plays a vital role in the performance of the driving algorithm and that the communication range should be large enough so that the driving algorithm has enough time to react to the vehicles around it. We did not find a strong effect of packet losses on the performance of the driving algorithm.

We see a number of research directions to improve the performance of a reinforcement learning approach for cooperative driving. (1) We would like to experiment more with variation in the neural network configuration like the number of hidden layers, number of nodes in each layer, and the activation function of each node in the neural network. (2) Different agents can be used for different driving aspects, such as speed control and lane changing. (3) We want to explore longer training periods and different learning strategies. (4) The collision ratio can be further reduced by also including information of the vehicles ahead of the vehicle in front in the input features of the driving agent, as then the vehicle will have more time to react based on the change in state of those vehicles. (5) We might be able to further improve the reward structure to achieve better performance. (6) We can have the vehicles negotiate among each other before changing lanes, so that the other vehicle which might be on a path to collision can either decrease its speed to allow the vehicle to change lanes or send a negative response so that the vehicle does not undertake lane changing.

Summarizing, we conclude that cooperative driving using reinforcement learning is a prospective approach for autonomous driving, but much more research needs to be conducted before it can be put into practice.

REFERENCES

[1] S. K. Gehrig and F. J. Stein, "Dead reckoning and cartography using stereo vision for an autonomous car," in *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1999, pp. 1507–1512.

[2] S. Ishikawa and S. Arai, "Cooperative learning of a driving strategy to suppress phantom traffic jams," in *Agents (ICA), IEEE International Conference on*. IEEE, 2016, pp. 90–93.

[3] X. Li, X. Xu, and L. Zuo, "Reinforcement learning based overtaking decision-making for highway autonomous driving," in *Intelligent Control and Information Processing (ICICIP), 2015 Sixth International Conference on*. IEEE, 2015, pp. 336–342.

[4] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility," in *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, vol. 42, 2011.

[5] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io

[6] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[7] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[8] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[9] R. Liu and J. Zou, "The effects of memory replay in reinforcement learning," *arXiv preprint arXiv:1710.06574*, 2017.

[10] A. Chandramohan, "Machine learning for cooperative automated driving," Master's thesis, 2018. [Online]. Available: https://essay.utwente.nl/76407/