RL Project: Autonomous Driving

Michele Sprocatti

Abstract

This project investigates the possibility of using a reinforcement learning agent to avoid crashes in the given highway environment (Leurent, 2018). This project compares the output of the best RL agent found with a specific baseline and with the human controlling the car. The baseline explained in Section 2 is defined in order to be an informed guess of the action that must be taken. To find the best RL agent, three different approaches are investigated: Double Deep Q-Network (DQN), Dueling Deep Q-Network (DQN), and Proximal Policy Optimization (PPO). These 3 approaches are presented in Section 3.

1. Introduction

The highway environment (Leurent, 2018) is a simple environment where we can have a variable number of lanes and a different number of vehicles. The objective is to give commands to the ego vehicle (the one that is controlled) to avoid crashes and maximize rewards. The rewards are completely controlled by the environment and are normalized between -1 (crash) and 1. Each episode has a maximum duration that is set when the environment is created, so each episode can be truncated if it reaches the maximum duration or ends with a crash.

2. Baseline

In order to define a good baseline, it was necessary to understand the information provided by the environment (Leurent, 2018) through the state. In the documentation, it is explained that the specific environment (highway) uses the kinematics observation format, so in the first row, it gives the absolute information about the ego vehicle while the other rows contain the relative information of the other cars with respect to the ego vehicle. The baseline provided uses two specific epsilons (one for the x-axis and one for the y-axis) to understand if there is a car in front of the ego vehicle. If this is true, then it moves up (if the action is available); otherwise, it moves down. If there is no vehicle in front of the ego one, the returned action will be 'FASTER'. This baseline is not good in all cases because it does not check if there are vehicles above or below the ego one when it changes lanes, and also, with the default settings, sometimes it does not have the most updated state (since the simulation is always running), so it may crash sometimes.

3. Approaches for RL agent

To define a RL agent, 3 different approaches are explored:

- Double Deep Q-Network (DQN) (van Hasselt et al., 2015)
- Dueling Deep Q-Network (DQN) (Wang et al., 2015)
- Proximal Policy Optimization (PPO) (Schulman et al., 2017)

All models were tried with different combinations of the hyper-parameters in order to find the best performing one. The three models are three deep RL approaches of increasing complexity; the first one is the simplest one of them, which is Double DQN, and then the Dueling DQN approach is explored. In the end, the state-of-the-art algorithm, the PPO, is run. The DQN base algorithm is not considered because of the problem of overestimating the function approximation.

3.1. Double Deep-Q-Network

In this approach, we have two networks (Q and Q' with the same topology) and they are used as follows: Q is used to make the prediction, and in order to update it, we minimize the mean squared error between the output and the target that is built as follows:

$$Q^*(s_t, a_t) = R_t + \gamma * Q'(s, argmax_a Q(s_{t+1}, a)) \quad (1)$$

The Q' network is updated after x training steps by copying the weights from the Q network. For an in-depth explanation of the algorithm, look at (van Hasselt et al., 2015).

3.2. Dueling Deep-Q-Network

In the Dueling Deep-Q-Network approach, we have two networks, but instead of predicting the Q value, each one of them has two different outputs $V(s)$ and $A(s, a)$ for every

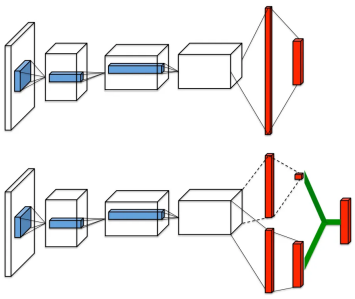

Figure 1. Extracted from original paper (Wang et al., 2015)
Top: Regular DQN architecture. Bottom: Dueling DQN architecture.

a (see figure 1), and they compute the value of $Q(s, a)$ as follows:

$$Q(s,a) = V(s) + [A(s,a) + \frac{1}{|A|} * \sum_{a'} A(s,a')] \quad (2)$$

Then the target is computed similarly to the Double Deep-Q-Network; see Equation 1. For an in-depth explanation of the algorithm, look at (Wang et al., 2015).

3.3. Proximal Policy Optimization

The Proximal Policy Optimization is an approximation of the Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) method that enforces a trust region with the computation of the Hessian matrix; the PPO, instead, avoids the case of going out of the trust region by using a min operator and a clipping operation.

$$J(\theta) = E_t[min(r_t(\theta) * A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon) * A_t] \quad (3)$$

Where $r_t(\theta)$ indicates the importance sampling ratio. For an in-depth explanation of the algorithm, look at (Schulman et al., 2017).

4. Training

The models were trained over 4000 episodes, and from that experience, the batches of (state, action, reward, next_state) are sampled to train the models. So, given the specific algorithm, the corresponding loss is computed, and then it is back-propagated in order to update the weights of the neural network. Since the Double DQN and the Dueling DQN use the experience replay, the batches are sampled over all possible data, while since the PPO is an on-policy method, the data are discarded after the finish of each episode. The outputs of the different trainings are presented in figures 2, 3, 4. These figures are created by plotting the average reward that the model gets for each of the 4000 episodes of the training. Since the plots are very dense, some other plots are presented to understand how the different models

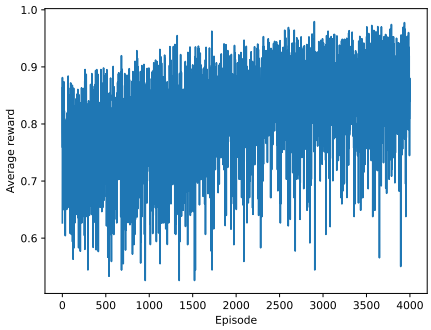

Figure 2. Double DQN training average reward

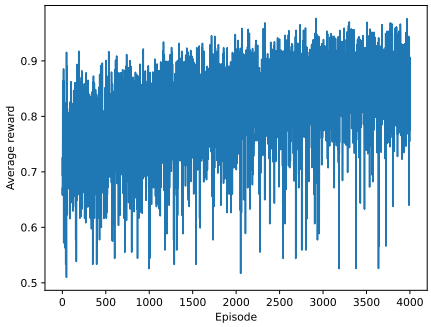

Figure 3. Dueling DQN training average reward

improve during training. An evaluation of the model every 250 episodes of the training is carried out. The evaluation involves running 20 episodes using the current model in a new environment. Then, the plot of the average return and also the average reward of these evaluation steps can be seen in figures 5, 8, 9, 6, 7, 10. From these figures, it seems that the Dueling DQN algorithm and the Double DQN are the ones that learn the most because of a continuous increase in the average return, while the PPO is more oscillating between low and high returns. From the plot of the average reward, it can be seen that the PPO does not change a lot during the training, while the Double DQN and the Dueling DQN seem to learn a lot, but the Dueling DQN reaches a better average reward in the end.

5. Results

The results are obtained by evaluating the different models on the environment and comparing them with the manual control and the specific baseline. The return and the average reward with manual control can be seen in figures 11 and 12. These are very high since the human can avoid crashes in a very good way. The baseline output can be seen in figures 13 and 14 . From these figures, we can see that the baseline is good when it comes to selecting the action because we have a very high average reward, and this comes from the fact that in the general case it returns 'FASTER', but it does not

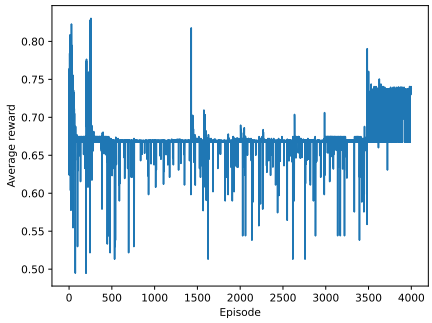

Figure 4. PPO training average reward

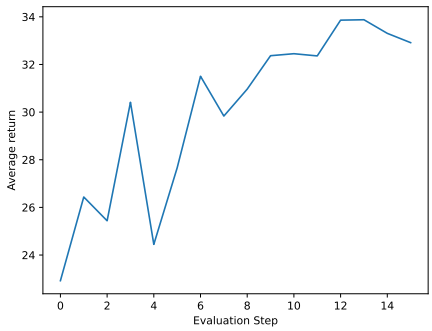

Figure 5. Double DQN training evaluation return

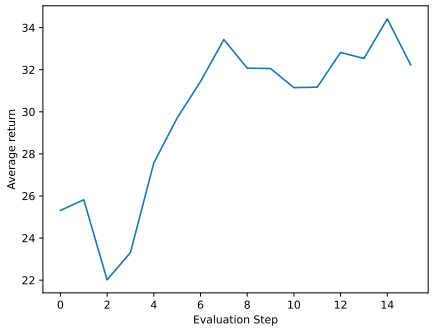

Figure 6. Dueling DQN training evaluation return

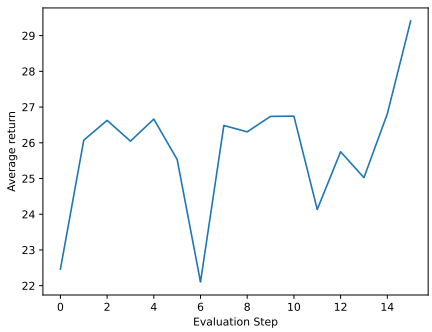

Figure 7. PPO training evaluation return

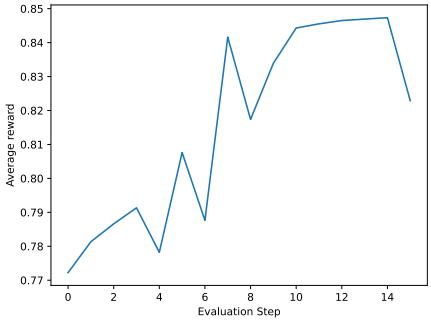

Figure 8. Double DQN training evaluation average reward

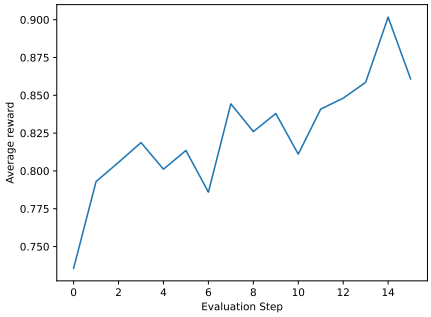

Figure 9. Dueling DQN training evaluation average reward

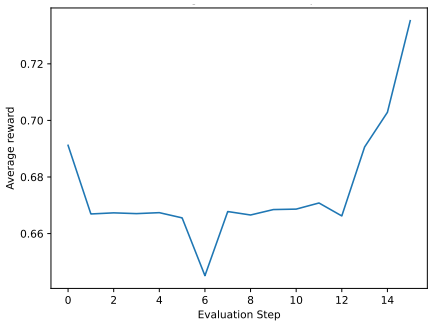

Figure 10. PPO training evaluation average reward

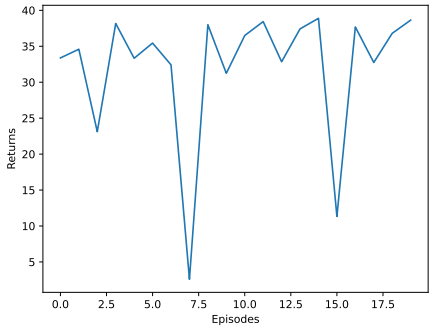

Figure 11. Return over the episode with manual control

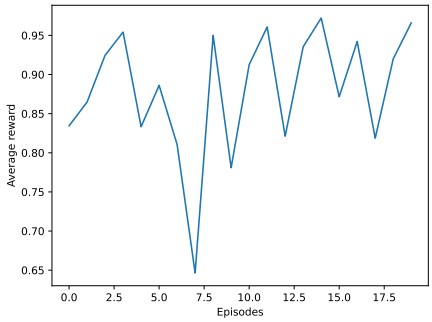

Figure 12. Average reward over the episode with manual control

avoid all crashes because when it changes lanes it does not check if the others are free, and also some crashes happen because it sees the vehicle in front too late, and this causes the returns to not be very high. For the three reinforcement learning agents, we can see the results in figures 15, 16, 17, 18, 19, 20. From these plots, we can see that the Dueling DQN algorithm provided the best model because it has the highest reward on average with also more stability and very high returns over the different episodes. The Double DQN has a very high return, but the average reward is less stable and oscillates a lot between low and high values, while the PPO has lower returns because it moves the ego vehicle in the rightmost lane and then stays there but avoids crashes by selecting 'SLOWER'. By comparing the Dueling DQN that seems to be the best reinforcement learning agent and the baseline, we can see that the agent performs better because it obtains a higher return thanks to the fact that the episode lasts longer, but also it obtains higher average reward. Obviously, the human-controlled one seems to be the best, but the Dueling DQN agent reaches very similar performance in terms of rewards and returns.

6. Conclusions

From the three approaches presented, we can see that the Double DQN and the Dueling DQN are the two that perform the best because the PPO has lower returns. Between the

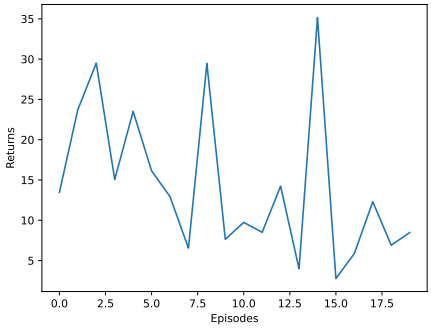

Figure 13. Return over episode baseline

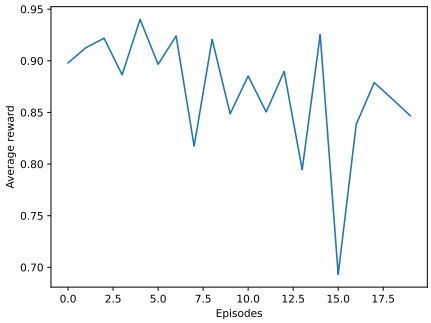

Figure 14. Average reward over episode baseline

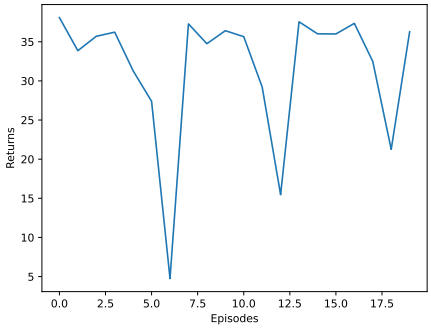

Figure 15. Return over episode Double DQN

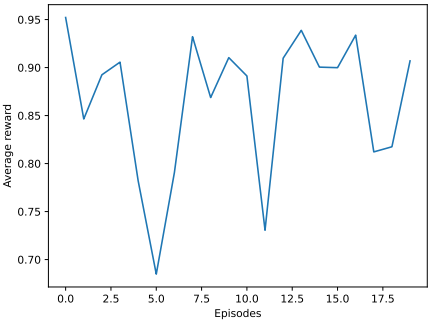

Figure 16. Average reward over episode Double DQN

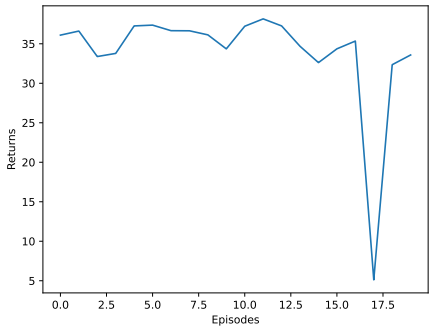

Figure 17. Return over episode Dueling DQN

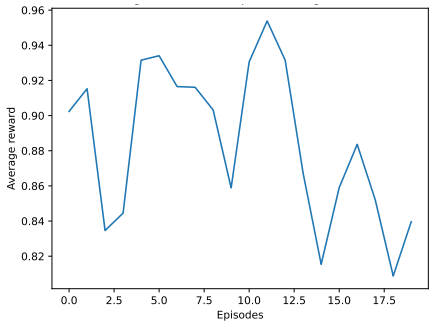

Figure 18. Average reward over episode Dueling DQN

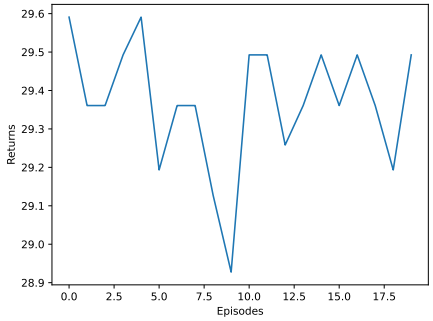

Figure 19. Return over episode PPO

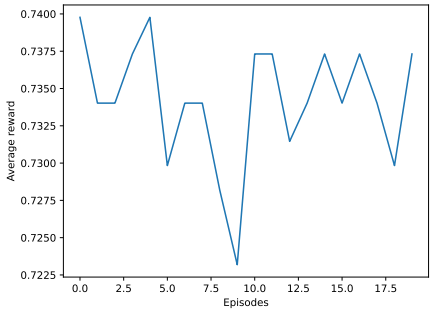

Figure 20. Average reward over episode PPO

Double DQN and the Dueling DQN, the one that seems to work best is the Dueling DQN because it achieves higher returns and the highest average rewards. By comparing the Dueling DQN with the baseline, we can see that the model outperformed the specific baseline in terms of returns, and it reaches similar performance compared to the human.

References

Leurent, E. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://arxiv.org/abs/1509.06461.

Wang, Z., de Freitas, N., and Lanctot, M. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015. URL http://arxiv.org/abs/1511.06581.