RL project report: Autonomous Driving

Francesco Boscolo Meneguolo 2119969

Abstract

This report presents different reinforcement learning approaches to autonomous driving in the Gymnasium highway environment, focusing on safety, efficiency, and adaptability. The RL agent is trained using different deep reinforcement learning algorithms to make real-time driving decisions, including lane changes, acceleration, and maintaining safe distances. Experimental results of the different trained models are compared among them, also with my baseline model and the manual control.

1. Introduction

In this task, the ego-vehicle is driving on a 3 lanes highway populated with other vehicles. The agent's objective is to reach a high speed while avoiding collisions with neighbouring vehicles. Driving in the right-hand lane is also rewarded. The environment configuration and the reward function are the default ones. The state is given by the Kinematic observation and there are five actions ('Lane Left', 'Idle', 'Lane Right', 'Faster', 'Slower'). No more discussion on this is needed since the information concerning the state, the actions, and the reward function are already present in the delivery of the project and in https://highway-env.farama.org/.
For the RL agent I implemented: my heuristic baseline, the manual control and three deep RL algorithms for discrete action space.

2. Heuristic Baseline

My heuristic baseline consists on the ego-vehicle immediately moving into the right-hand lane and then accelerating or slowing down depending on whether there is a vehicle within a fixed distance of 0.16 in the right-hand lane. The evaluation results are shown in Figure 1. The code is in **my_baseline.py**.

3. Manual Control

Evaluation results of manual control are displayed in Figure 2. Obviously, this method has the best results. The code is in **manual_control.py**.

4. PPO Clip

I implemented the PPO Clip version as described in (Schulman et al., 2017) with the hyperparameters in Table 1. The code is in the file **training_ppo_clip.py**. In Figure 3 performance during training is shown. In Figure 4 evaluation results are displayed. I also tried a different approach using Generalized Advantage Estimation (GAE) (Schulman et al., 2015), but it performed worse than the original implementation. I left commented in the code the part related to GAE.

5. Double DQN

I implemented the Double DQN (Van Hasselt et al., 2016). I perform soft updates on the target network, instead of copying the q-network weights every N updates of the q-network, since it proved to be better. I use a linear epsilon decay schedule concerning epsilon-greedy selection of the action for 2×10^4 timesteps.
The hyperparameters used are in Table 2. The code is in file **training_double_dqn.py**. In Figure 5 performance during training is shown. In Figure 6 evaluation results are displayed.

6. Dueling DQN with PER

I implemented the Dueling DQN (Wang et al., 2016) with Prioritized Experience Replay Buffer (Schaul et al., 2015) using the Sum Binary Tree data structure (which is a Full Binary Tree) to store the priorities. In the Sum Binary Tree each leaf stores the priority of an experience in the Experience Replay Buffer, while an inner node stores the sum of its two children's priorities; so the root contains the total priority. In this way the time complexity to retrieve an experience in the Experience Replay Buffer according to its priority is $O(log\ n)$, where n is the capacity of the Experience Replay Buffer. I perform soft updates on the target network, instead of copying the q-network weights every N updates of the q-network, since it proved to be better. I use a linear epsilon decay schedule concerning epsilon-greedy selection of the action and a linear increment of β parameter both for 2×10^4 timesteps. The hyperparameters used are in Table 3. The code is in file **training_dueling_dqn_per.py**. In Figure 5 performance during training is shown. In Figure

6 evaluation results are displayed. This is experimentally the best deep RL algorithm among the three, according to the evaluation performance. In order to choose the best model, I selected the four best models from Figure 8 and I evaluated them separately over 100 episodes. Then I noticed that the best model is the one obtained after 1500 training episodes, as can be seen in Figure 9. I saved the weights of the best model in the files `dueling_q_network.pth` and `dueling_target_network.pth`. A plot of the evaluation of the best model is in Figure 10.

7. Figures

7.1. Heuristic Baseline Figures

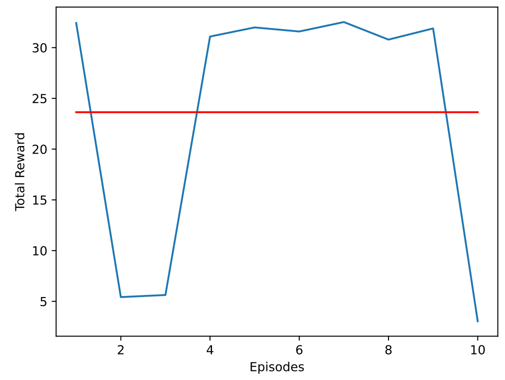

Figure 1. Heuristic Baseline Evaluation over 10 episodes. The red line is the average total reward.

7.2. Manual Control Figures

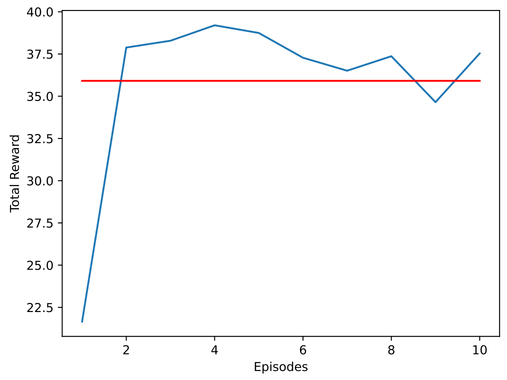

Figure 2. Manual Control Evaluation over 10 episodes. The red line is the average total reward.

7.3. PPO Clip Figures

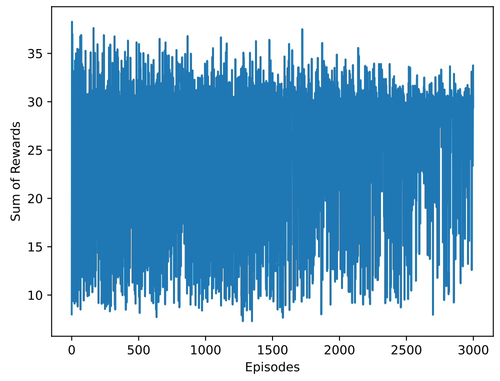

Figure 3. PPO Clip Total Rewards during 3000 episodes of training.

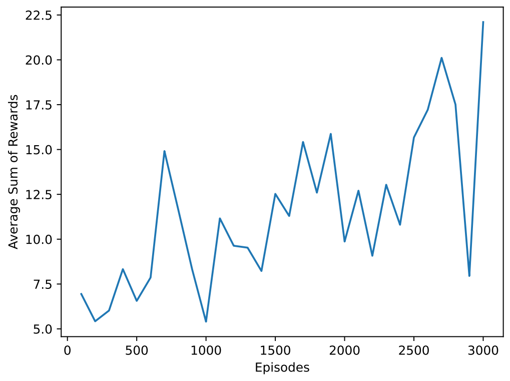

Figure 4. PPO Clip evaluation taking the average over 10 episodes every 100 episodes during training.

7.4. Double DQN Figures

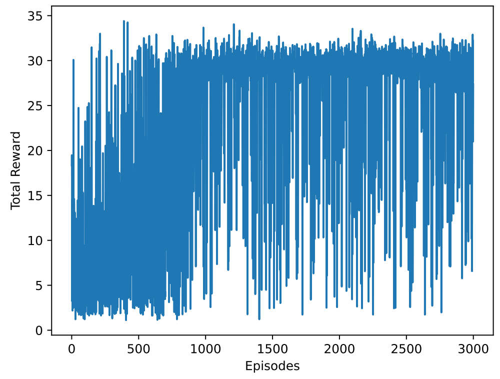

Figure 5. Double DQN Total Rewards during 3000 episodes of training

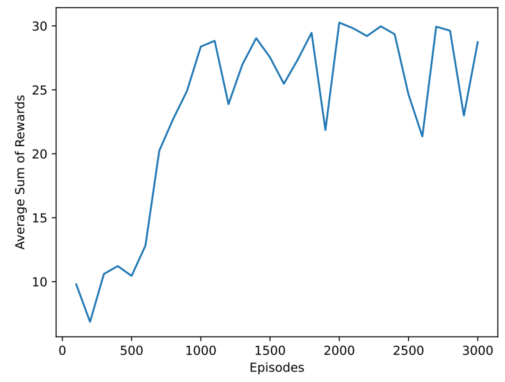

Figure 6. Double DQN evaluation taking the average over 10 episodes every 100 episodes during training.

7.5. Dueling DQN with PER Figures

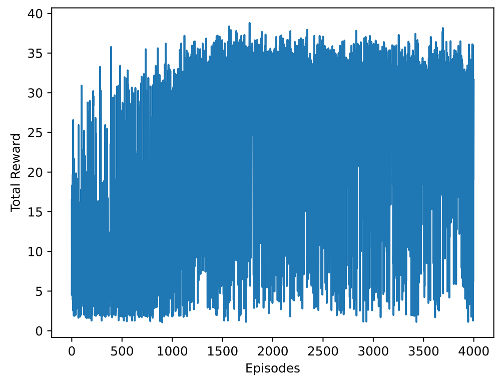

Figure 7. Dueling DQN with PER Total Rewards during 4000 episodes of training.

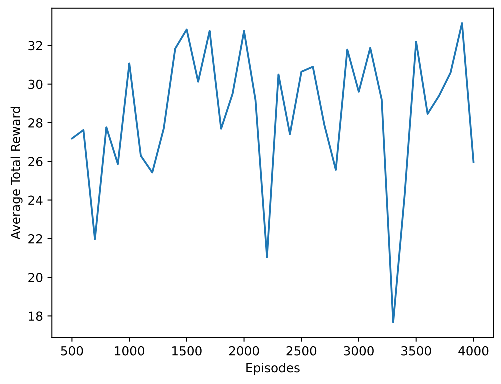

Figure 8. Dueling DQN with PER evaluation taking the average over 10 episodes every 100 episodes during training.

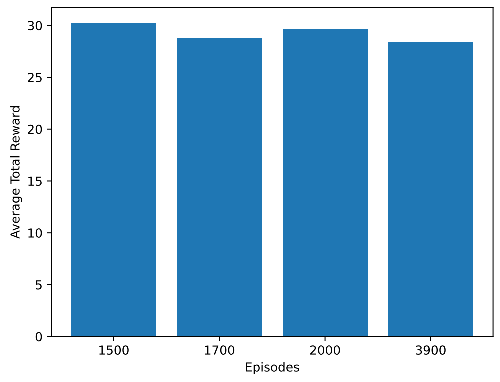

Figure 9. Best four Dueling DQN with PER models evaluation taking the average over 100 episodes.

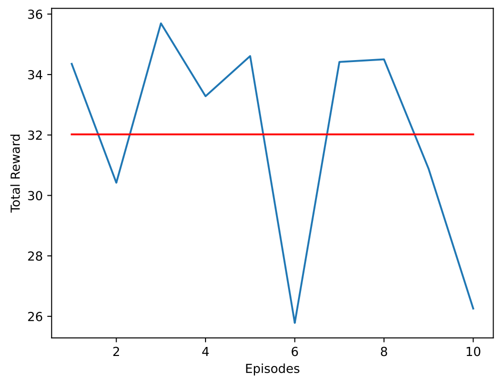

Figure 10. Best Model Evaluation over 10 episodes. The red line is the average total reward.

8. Tables

8.1. PPO Clip Table

Table 1. PPO Clip Hyperparameters

HYPERPARAMETERS	
TRAINING EPISODES	3×10^3
EPISODES PER BATCH	100
MAX TIMESTEPS PER EPISODE	40
UPDATES PER ITERATION	4
GAMMA FACTOR	0.99
ACTOR LEARNING RATE	3×10^{-4}
CRITIC LEARNING RATE	5×10^{-4}
CLIP FACTOR	0.2
ENTROPY COEFFICIENT	0.01
TARGET KL	0.02
MAX GRADIENT NORM	0.5

8.2. Double DQN Table

Table 2. Double DQN Hyperparameters

HYPERPARAMETERS	
TRAINING EPISODES	3×10^3
MAX TIMESTEPS PER EPISODE	40
BUFFER SIZE	1×10^4
BATCH SIZE	64
GAMMA FACTOR	0.99
LEARNING RATE	1×10^{-3}
ϵ MAX	1.0
ϵ MIN	0.05
τ FOR SOFT UPDATE	1×10^{-3}

8.3. Dueling DQN with PER Table

Table 3. Dueling DQN with PER Hyperparameters

HYPERPARAMETERS	
TRAINING EPISODES	4×10^3
MAX TIMESTEPS PER EPISODE	40
BUFFER SIZE	1×10^4
BATCH SIZE	64
GAMMA FACTOR	0.99
LEARNING RATE	1×10^{-3}
ϵ MAX	1.0
ϵ MIN	0.05
τ FOR SOFT UPDATE	1×10^{-3}
α	0.6
β MIN	0.4
β MAX	1.0

9. Comments

We can notice that PPO Clip performs worse with respect to the heuristic baseline, since it sticks in a suboptimal policy. Instead, both Double DQN and Dueling DQN with PER perform better, in particular Dueling DQN with PER. In fact, the best model has performance comparable to manual control, even if its performance is not exceeded. In the best model there are still sporadic collisions. I tried Rainbow DQN as well, but it did not show signs of convergence during training, possibly due to Distributional RL (Bellemare et al., 2017).

10. Citations and References

References

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. PMLR, 2017.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.