Autonomous Highway-Driving in highway-env

Francesco Carlesso

Abstract

In this project I train a virtual agent to efficiently drive through a crowded highway, experimenting with three kinds of Deep Q-Networks: Vanilla DQN, Double DQN, and Dueling DQN. Test performance is compared against a heuristic baseline and a manually controlled situation.

1. Introduction

Autonomous driving represents one of the most compelling and challenging applications in the realm of artificial intelligence. In this project, I focus on training a Reinforcement Learning (RL) agent that can navigate a densely populated highway. The objective is to maximize the agent's progress along the highway, and therefore also to avoid collisions with other vehicles. RL is a crucial application to real-world problems, where safety is as crucial as efficiency. In designing my approach, I considered of particular importance the sparseness of crashing events, since even if collisions occur rarely, their outcomes can yield catastrophic consequences.

In the following sections, I provide a comprehensive description of the environment, the benchmarks, the algorithms chosen, and discuss some experimental adjustments, to carry out an overall performance review. The code is available at the following GitHub repository.

2. Environment Setup

The simulation environment used is highway-env (Leurent, 2018), which is designed to mimic realistic highway dynamics (Figure 1). In this case, the ego-vehicle (green) is proceeding forward in a highway composed by three lanes.

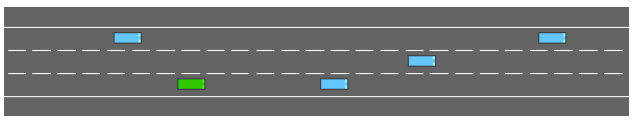

Figure 1. highway-env rendering example

The state is represented by a matrix S, where the first row corresponds to the agent, and the remaining rows describe the surrounding vehicles. Each row is characterized by five features, which are: *Presence* (boolean), *Normalized position along x and y axis*, *Normalized velocity along x and y axis*; where these normalized values are given with respect to the ego-vehicle, except of course for the first row which yields information about the absolute reference frame.

Discrete action space with five possible actions:

- *Change lane to the left* (Return 0)
- *Idle* (Return 1)
- *Change lane to the right* (Return 2)
- *Accelerate* (Return 3)
- *Brake* (Return 4)

The default reward function is made up by various terms, such as a bonus term for progressing quickly, a small positive reward for staying on the rightmost lane, and a penalty term for collisions, where these collisions also terminate the episode, thus forgoing potential future rewards.

Furthermore, performance evaluations are conducted using identical random seeds for having fair comparisons, so designed in a way that ensures the agent is seeing the exact same traffic conditions throughout the episodes when it is leveraging the different policies. For the same reason, the results are also consistent between different evaluation runs, keeping stochasticity in check.

A complete episode without crashes lasts 40 steps, where each step outputs a state matrix S. Given that such complete episodes last around 23 seconds, S is updated with intervals of approximately 0.6 seconds, which is twice the reaction time of an average human. I mention this because I think one should take into consideration the limitations this technical detail of the simulation imposes on the algorithms. The issue can be easily fixed in real-life scenarios by having more frequent information, but in my case it would increase the computational costs too much.

3. Baselines

To evaluate the performance of my agent, I defined two baselines: Manual Control and a Heuristic Baseline. These

reference points allowed me to have human and rule-based (non-learning) scores, to compare against the RL algorithms.

3.1. Manual Control

The manually controlled simulation is the one that yields the best results after all. This is of course given by the simplicity of the task, which is very intuitive. However, I have to admit that such high rewards are obtained through a very dangerous and not advisable driving style, like surpassing on the right and completely forgetting about safety distances.

3.2. Heuristic Baseline

The heuristic baseline is a rule-based approach built on basic driving strategies. The algorithm repeatedly evaluates the state matrix S to determine appropriate driving actions based on the relative positioning of surrounding vehicles. The pseudocode in the next column (Algorithm 1) details this heuristic. This baseline obtains lower cumulative rewards, but it fosters a much safer approach. Although there are a few crashes, the ego-vehicle prioritizes breaking when other vehicles are too close, as it can be seen from the second `if` condition. This leads to a suboptimal strategy, mainly going very slow with just some exceptions, therefore foregoing the high-speed reward, which makes up the majority of the episode. The heuristic was designed to be robust and easily interpretable. However, its lack of adaptability means that it can be overly cautious in situations where a more dynamic response would be beneficial, and this is one of the cons when apprlying non-learning algorithms.

The results of the comparison are shown below in Figure 2 where 40 is the highest reward, obtainable running at full speed on the right lane for the entire episode.

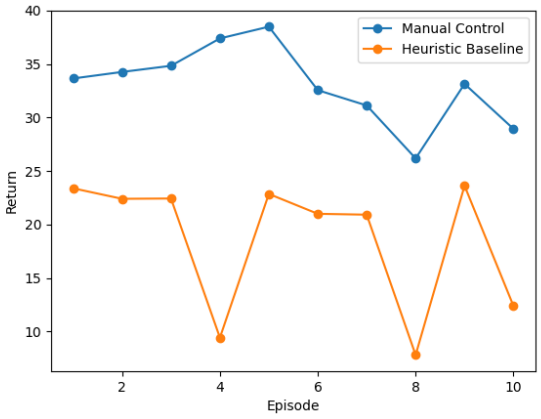

Figure 2. Baselines comparison

Algorithm 1 Heuristic Baseline

Input: State matrix S with rows $(s_0, s_1, ..., s_n)$ where (s_0) refers to the ego vehicle, and $(s_1, ..., s_n)$ to the other vehicles present on the road.

Initialize $action \leftarrow 1$ {Default: idle}

for each vehicle v in $(s_1, ..., s_n)$ **do**
  $pos \leftarrow v[1]$ {Relative distance}
  $lane \leftarrow v[2]$ {Relative lane}

  if $|lane| < 0.05$ **and** $pos > 0$ **then**

    if $pos < 0.15$ **then**
      $action \leftarrow 4$ {Emergency brake}
      break

    else if $pos < 0.4$ **then**
      $left\_clear \leftarrow$ **true**
      $right\_clear \leftarrow$ **true**
      for each vehicle v' in $(s_1, ..., s_n)$ **do**
        if $0 < v'[1] < 0.2$ **and** $-0.4 < v'[2] < -0.05$ **then**
          $left\_clear \leftarrow$ **false**
        end if
        if $0 < v'[1] < 0.2$ **and** $0.05 < v'[2] < 0.4$ **then**
          $right\_clear \leftarrow$ **false**
        end if
      end for

    end if

    if $right\_clear$ **then**
      $action \leftarrow 2$ {Change to right lane}
    else if $left\_clear$ **then**
      $action \leftarrow 0$ {Change to left lane}
    else
      $action \leftarrow 4$ {Brake}
    end if

    break

  end if

  if not (any vehicle v with $|v[2]| < 0.05$ **and** $0 < v[1] < 0.4$) **then**
    $action \leftarrow 3$ {Accelerate}
  end if

end for

Return $action$

4. Deep Q-Networks

To allow the agent to learn an optimal driving strategy, I experimented with three variants of Deep Q-Networks (DQN).

4.1. Vanilla DQN

While known for overestimating Q-values and sometimes exhibiting instability, in my experiments the Vanilla DQN outperformed the other methods. Empirically, this DQN showed the highest average return and the lowest variability compared to the other DQNs, suggesting that it converged to a more stable and effective policy. This is somewhat surprising, as the literature often cites instability and overestimation issues in Vanilla DQNs (Mnih et al., 2015), yet my results indicate that, with appropriate tuning, it can be both robust and performant in the driving environment tested.

This strong performance could stem from the relatively low stochasticity of the driving environment, where deterministic rewards and structured transitions reduce the negative impact of overestimation bias.

4.2. Double DQN

Double DQNs were developed to correct the overestimation problem in standard DQN by using separate networks for action selection and value estimation in the target calculation (Van Hasselt et al., 2016). This generally results in more conservative and accurate value updates.

In my case, Double DQN showed very bad performance, being enormously surpassed by both Vanilla and Dueling DQNs in terms of average return and training stability. This suggests that not only reducing overestimation has not helped stabilize learning, but it may have also led to under-exploration or overly cautious policies, especially in a task like driving where bold maneuvers (e.g., overtaking) can yield higher rewards.

4.3. Dueling DQN

Dueling DQN introduces a separation between the estimation of state value V(s) and advantage A(s,a), which are then combined into the Q-value estimate (Wang et al., 2016). This design helps the agent learn the importance of states, even when the choice of action is not critical.

However, my empirical results show that while Dueling DQN is better than Double DQN in terms of average return, it is underperforming relative to the Vanilla DQN. Performance fluctuates heavily between episodes, failing to stabilize, even though the progress can be seen when averaging across different episode windows (as shown in Figures 3, 4, and 5). Moreover, watching the simulation, one can observe the agent exhibiting the most proactive driving style, after manual control.

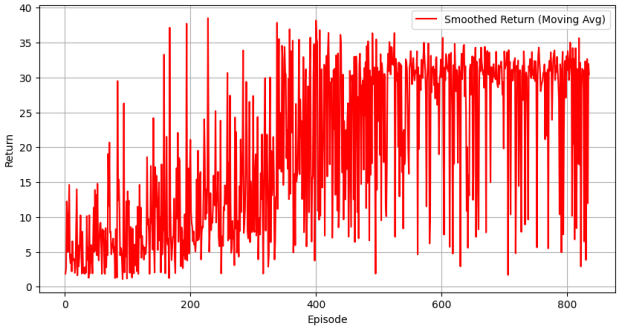

Figure 3. Return across episodes

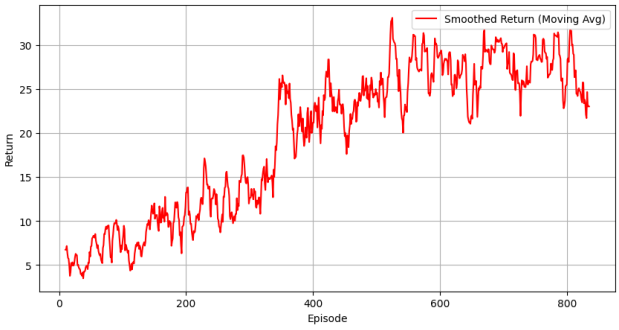

Figure 4. Smoothed return across 10 episodes

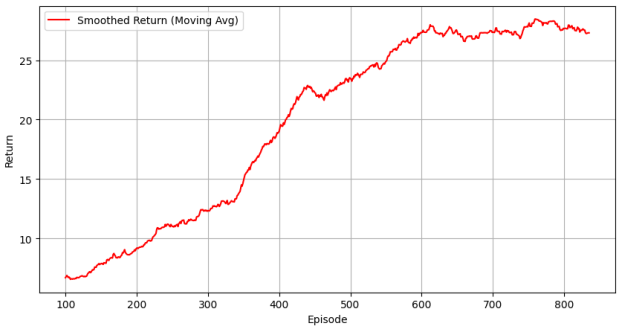

Figure 5. Smoothed return across 100 episodes

5. Evaluation

Results in Table 1 clearly demonstrate that, while the heuristic baseline offers a somewhat safe behavior, the Vanilla DQN is able to achieve superior performance by dynamically balancing the trade-offs between speed and safety. However, the other two RL algorithms (Figure 6) are still very unstable in my case, and even the Vanilla DQN is not able to achieve results comparable to human performance.

To do so, much more experimentation and training should have been carried out, and given the potential of RL algorithms, I believe that a thoroughly crafted method could even surpass manual control.

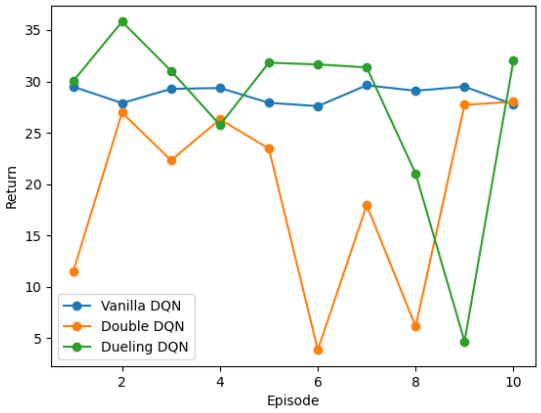

Figure 6. DQNs comparison

Table 1. Mean cumulative rewards with std, over 10 episodes.

METHOD	RETURN
MANUAL CONTROL	33.05 ± 3.67
HEURISTIC BASELINE	18.62 ± 6.20
VANILLA DQN	28.75 ± 0.84
DOUBLE DQN	19.42 ± 9.16
DUELING DQN	27.53 ± 8.97

6. Key Considerations

Throughout the project, I encountered a few challenges that required thoughtful adjustments:

- **Hyperparameter Sensitivity:** I extensively tuned the learning rate, discount factor, and replay buffer size. Small changes in these parameters had a notable impact on the convergence speed and final performance.
- **Reward Function:** Most of the cumulative reward composition depends on the *high speed reward*, which strongly influences the comparisons. This was set to 1 during training, and therefore also for the testing results discussed, but one needs to account that higher numbers could have promoted much different behaviors for the RL agents. It is anyways interesting to change this value in the evaluation scripts, to be able to grasp which are the methods that most exploit this reward term, that I consider to be the most informative.

In addition, several critical issues became apparent:

- **Robustness to rare events:** The sparsity of collision events remains a challenge, and further work is needed to ensure that the agent robustly handles rare but dangerous situations.
- **Sim-to-Real gap:** The policies learned in simulation might not directly transfer to real-world conditions. Addressing this gap would require additional research in domain adaptation and more complex environment modeling.
- **Scalability:** As the number of vehicles increases, the state space becomes more complex. Future work should consider more scalable representations or hierarchical models to handle increased traffic density.

7. Conclusion

In this project, I have demonstrated that reinforcement learning can effectively learn to drive on a crowded highway. Despite the challenges of sparse collisions, the RL approaches outperformed the heuristic baseline in terms of cumulative rewards.

The insights gained from this work underscore the potential of RL for complex, safety-critical applications and highlight several avenues for future improvements. I am confident that with continued refinement and access to a wider range of resources, RL-based methods will become a viable complement to traditional control systems in autonomous driving.

References

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

Mnih, V., Kavukcuoglu, K., Silver, D., et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and de Freitas, N. Dueling network architectures for deep reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1995–2003, 2016.