NOVA IMS
Information Management School

Reinforcement Learning Group Project

Master Degree Program in Data Science and Advanced Analytics

Developing a Reinforcement Learning Agent for Decision-making in Autonomous Driving

Group Members

Arif Maharramov 20230770

Arda Kaykusuz 20230766

June 2024

Contents

Abstract

The objective of this study is to develop and evaluate reinforcement learning (RL) agents for autonomous driving using the highway environment simulation. We focused on three RL algorithms: DQN, SAC, and A2C, each tested with various observation types and action spaces. Through a systematic grid search and rigorous hyperparameter tuning, we identified the optimal configurations for each model. Custom reward functions were also explored to enhance performance. Our findings indicate that while DQN learned much faster and demonstrated good performance, A2C achieved similar results with the same observations and actions. Meanwhile, SAC, although slower in learning compared to A2C and DQN, performed competitively in terms of rewards. In summary, this work underscores the potential of RL in advancing autonomous driving technology by showcasing the strengths and limitations of different RL algorithms in this context.

Key words: Reinforcement Learning, Autonomous Driving, Deep Q-Network, Soft Actor-Critic, Deep Deterministic Policy Gradient, Highway Environment, Hyperparameter Tuning, Custom Rewards, Simulation, Machine Learning.

1. Introduction

Autonomous driving stands as one of the most exciting frontiers in artificial intelligence, with the potential to transform the transportation industry fundamentally. When we consider the development of this technology, it will be one of the trend research topics in the following years.

Autonomous vehicles need to navigate complex environments, make split-second decisions, and ensure passenger safety, all while dealing with unpredictable driving conditions and behaviors from other vehicles. [1] These cars must make real-time decisions, process information, and act quickly. At the same time, safety must be a priority. When all combinations should be done, of course, requiring robust algorithms to handle unexpected situations and rare events. This is where Reinforcement Learning (RL) plays a key role, as it can learn the best actions through continuous interactions with the driving environment.

In our project, we focused on utilizing the "highway-env" project, which provides a simulated environment closely resembling real-world driving scenarios. [2][3][4][5] This simulation presents numerous challenges such as high-speed driving, lane changing, and collision avoidance, making it an ideal testbed for developing and evaluating RL algorithms. Our primary goal was to develop an RL agent capable of effectively navigating the Highway environment, balancing speed optimization with collision avoidance through strategic real-time decision-making. To achieve this, we tested and compared various RL algorithms to determine their effectiveness. In addition, we employed hyperparameter tuning to optimize the learning performance of our RL agents. Furthermore, we assessed their performance through evaluation metrics, providing insights into their decision-making processes and overall effectiveness.

2. Methodology

2.1. Grid search of different models and Model Selection

Our first step was a grid search to denote the most suitable configurations for the algorithms. It was beneficial to define the parameters and main directions of development in this initial stage, which allowed for the identification of key priorities for further improvement. As seen in Table 1. according to the grid search results, we decided to use 3 different models in our project.

Table 1. Grid Search Results of Models Using Default Parameters

Model	Number of Runs	Avg. Reward	Avg Timestamp
DQN	186	54.45888106	137.6855733
A2C	180	52.82753497	176.5560979
DDPG	65	31.5332644	331.8211815
TD3	72	48.32053474	344.9137309
PPO	34	8.546315235	883.8255642
SAC	72	34.937055653	281.1905229

When we observed the results, we proceeded in accordance with average reward scores. As seen in Table 1 DQN, A2C, and TD3 showed better performances compared to the other models. However, we also considered training time. Even if we saw a high average reward in TD3, the computational time was not great, and we decided to use SAC instead of TD3. After selecting models, finally we had 3 models to continue our analysis.

3. Experiments, Hyperparameter Tuning, and Results

3.1. DQN (Deep Q- Network) Models

As previously mentioned in Section 2.1. we applied DQN because it showed high performance in our initial grid search DQN is a reinforcement learning algorithm where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. It uses a neural network to approximate the Q-value function, which estimates the expected return of taking specific actions in given states. In our project, we configured our environment with parameters like lanes, vehicles, and TimeToCollision observations to help the agent avoid collisions. We then trained the DQN agent with different hyperparameters, optimizing them using Optuna to maximize average rewards. The best-performing hyperparameters were recorded for further analysis to robust our algorithm.

Additionally, we utilized the DiscreteMetaAction to simplify the agent's action space into a set of predefined actions. We applied the same procedure to it, and we got the results after hyperparameter tunings. Its parameters can also be seen in **Table 2.**

Table 2. Hyperparameters of DQN agents

Hyperparameters	**DQN with TimetoCollision**	**DQN with DiscreteMetaAction**
learning_rate	0.001241975	0.0005860512472
buffer_size	5000	25000
learning_starts	378	239
batch_size	128	64
gamma	0.895557481	0.8304643095
train_freq	9	2
gradient_steps	6	5
target_update_interval	32	27
verbose	1	1

After creating our agent with the best parameters, we conducted an extensive evaluation of our DQN agent in the highway environment by running it through 30 episodes and collecting detailed performance metrics. These metrics included total reward, collision count, average speed, cumulative rewards, collision positions, and speed profiles for each episode. We then visualized these metrics through various plots as seen from **Figure 1** to **Figure 4**.

When we examined Figure 1 to Figure 4, the total reward per episode remained relatively stable, typically ranging between 70 and 80, with a notable drop to nearly zero in episode 26, we observed a strong drop. As for the average speed, it generally fluctuated between 20 and 22, with a significant spike in episode 26, correlating with the drop in total reward, implying the agent may have accelerated uncontrollably, leading to poor performance. Lastly, the reward trajectories showed a consistent upward trend. After completing our analysis, we captured a video with this model as seen in Image 1. (GIF) The car followed the other cars appropriately and it approached vehicles at an appropriate distance.

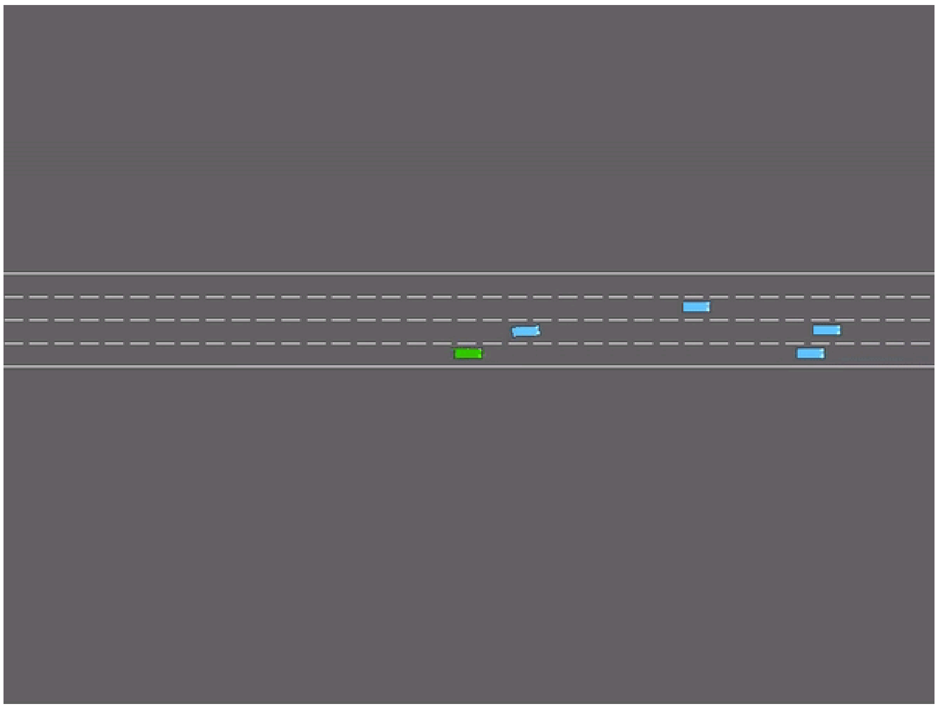

Image 1. The Simulation with DQN-TimetoCollision Agent

3.2. A2C Model

As the second agent, we decided to use A2C because it effectively combines policy-based and value-based methods. A2C synchronizes updates, allowing all agents to collect experiences and update the model together, ensuring stable and efficient training.[6] It brought some advantages to our project as mentioned in Section 2.1. We trained the model faster than the other agents.

We used TimetoCollision observation as we used for DQN while training. After training the model with 15000 timesteps, we saved and later loaded it for evaluation, with the environment set up to record video footage of the agent's behavior. The reason we used 15000 timesteps for it, we have not observed any improvement after this timestep.

In analyzing the agent's performance, we utilized data from Figures 5 to 8, which indicated a performance very similar to DQN. We saw that the total reward per episode remained consistently high, typically around 80, except for a significant drop in episode 25, where it fell to nearly zero, indicating a major failure. The average speed was stable between 20 and 21.5 units across most episodes. The reward trajectories showed a consistent upward trend. Lastly, the speed profiles revealed a rapid decrease from the initial speed, stabilizing around 20 units for most episodes, but episode 25 showed a sharp drop. As a result, the agent mostly showed a good performance but the anomaly in episode 25 highlighted a need to investigate.

And in Image 2, the car’s movements can be seen.

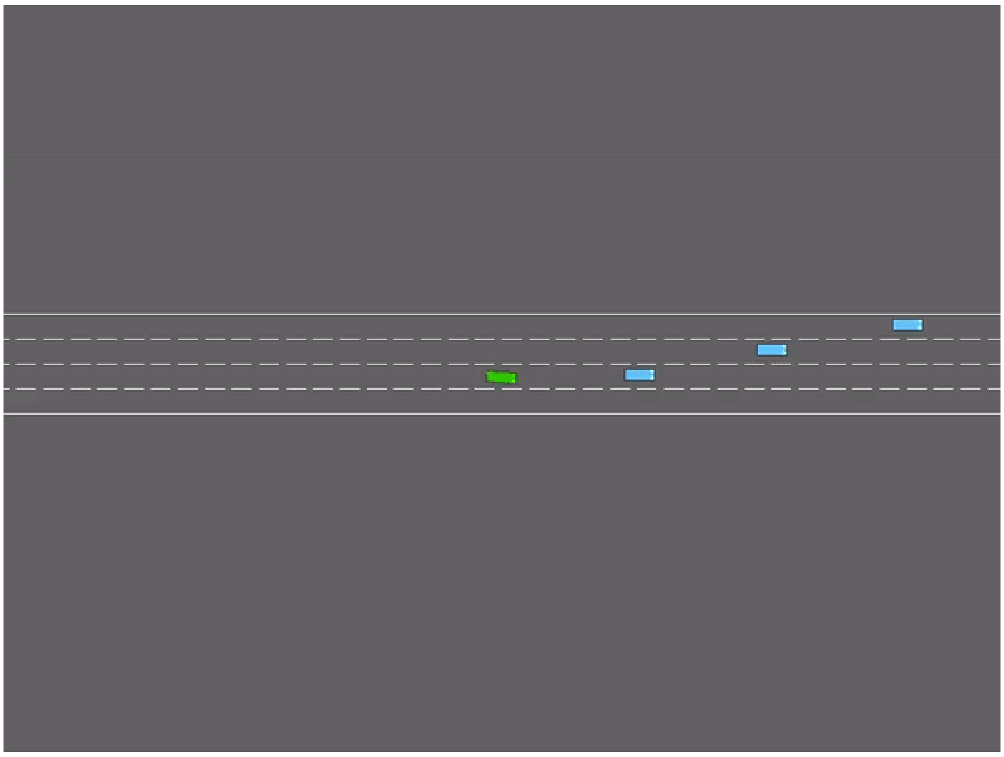
Image 2. The Simulation with A2C agent

3.3. SAC Model (Soft Actor-Critic)

As the final agent, we implemented Soft Actor-Critic (SAC). SAC is a sophisticated reinforcement learning algorithm that merges the advantages of value-based and policy-based approaches, while also integrating entropy maximization.[7] This integration makes SAC highly effective in environments that need a balance between exploration and exploitation.

When we examined the outputs of the SAC model it was seen that it showed good performance. The total reward per episode was mostly stable. The average speed per episode was generally between 20 and 22, with a notable spike in episode 21 correlating with the drop in reward, the car may have accelerated uncontrollably. As for the reward trajectories consistently showed an upward trend. The speed profiles revealed a rapid decrease from the initial speed, stabilizing around 20 units for most episodes, but certain changes in some episodes indicate variability in the agent's speed control.

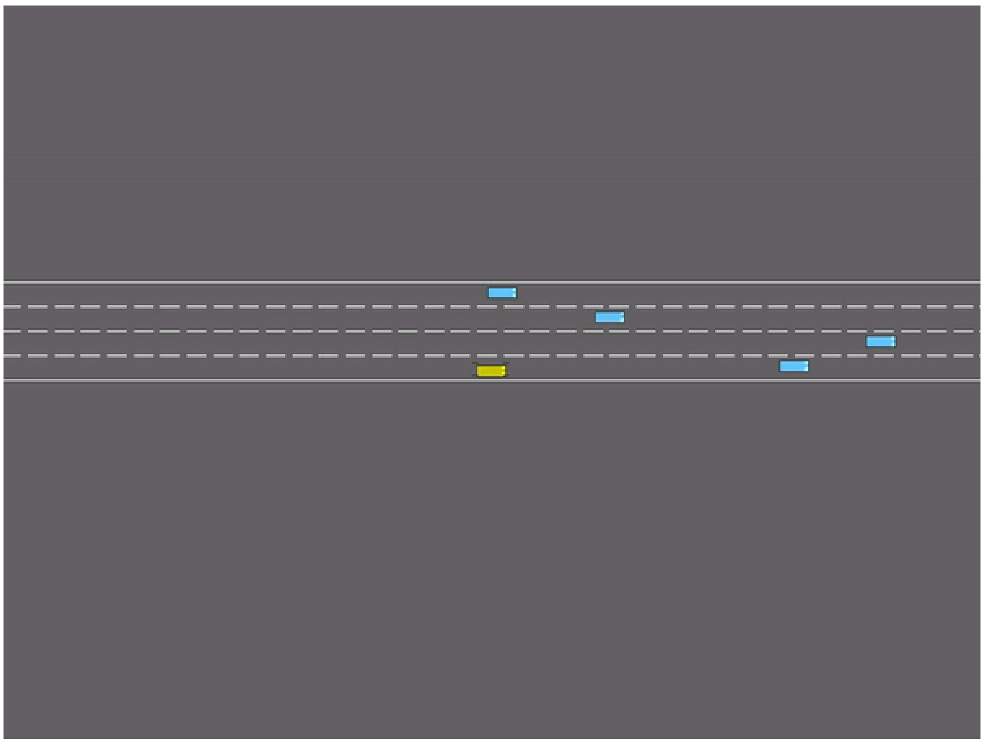

Image 3. The Simulation with SAC agent

4. Discussion

When we compared the performances of DQN, SAC, and A2C models, we found some interesting facts: DQN and A2C both learned faster than SAC and performed similarly with the same observation and action type. Both of these models demonstrated finer control over the vehicle's behavior, which led to smoother driving and better collision avoidance. At the same time, SAC's computational requirements were higher and made it less feasible for real-time applications without solid resources.

Custom reward structures have a high impact on learning outcomes, which proves the importance of aligning rewards with desired behaviors. SAC's stability and lower collision rate made it more suitable for environments that prioritize safety. For A2C, its flexibility in handling complex driving scenarios showed its potential in more dynamic settings. The challenges in tuning and balancing custom rewards showed the iterative nature of developing effective RL agents for the task treated.

5. Limitations and future work

5.1. Limitations:

During our coding, we faced many limitations:

- Insufficient Hyperparameter Tuning for TD3 and DDPG: Due to time constraints, hyperparameter tuning for SAC was not as extensive, potentially affecting their performance.
- Ineffective Models and Time Constraint: Attempts to train TD3 and PPO for 30,000 timesteps with default parameters were unsuccessful. The models failed to keep the cars within road boundaries and were computationally too slow for our devices. It is possible that these models could learn with higher timesteps such as 100K or 200K but because of computational resources and time limits, we did not try them. Meanwhile, our chosen algorithms were faster: SAC results stabilized after 30,000 timesteps when we attempted 50,000, and DQN results stabilized around 5,000 timesteps out of the 10,000 attempted.

5.2. Custom Rewards

We experimented with custom reward functions that would have saved many lives if it was applied in real life:

- Speed Reward: give a reward to the agent for maintaining high speed when no other vehicles are within a safe distance.
- Lateral Movement Penalty: penalize quick movements in the y-axis to encourage smooth lane changes.
- Distance Reward: reward the agent for maintaining an appropriate distance from the front vehicle.

5.3. Ideas for Future Research

If we had more time, we would have tried additional different methods on our project to overcome the limitations mentioned and to improve performance:

- More Extensive Hyperparameter Tuning: A more thorough one for SAC and A2C models to better optimize their performance.
- Longer Training Durations: Extending training time to allow the models to learn more effectively.
- Further Exploration of Custom Rewards: try and experiment with more sophisticated custom reward functions to better align agent behavior with the driving objectives wanted.

6. Conclusion

In conclusion, this study shows the potential of reinforcement learning (RL) in improving autonomous driving technology. By using a simulated highway environment, we carefully tested three RL algorithms: DQN, A2C, and SAC. Through detailed tuning of settings and creating custom reward functions, we improved each model's performance in various driving scenarios. Our results showed that DQN learned quickly and performed well in terms of average reward, showing more stability and safety, making it better for avoiding collisions. A2C also performed similarly to DQN with the same observation and action. Meanwhile, SAC learned slower than A2C and DQN but also performed well in terms of reward.

The insights gained from this study provide a strong base for future research in the field of autonomous driving. The challenges we faced, such as the need for more detailed tuning and limited training times, highlight the ongoing nature of developing effective RL agents. Future work could focus on better tuning of SAC and A2C models, longer training times, and exploring more advanced custom reward functions to better match agent behavior with desired driving goals. Additionally, addressing the high computational needs of these algorithms will be important for their real-time use in autonomous vehicles. Ultimately, this research contributes to the ongoing efforts to create safer and more efficient autonomous driving systems using the powerful tools of reinforcement learning.

7. References

[1] Liao, J., Liu, T., Tang, X., Mu, X., Huang, B., & Cao, D. (2020). Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning. *IEEE Access*, *8*, 177804–177814. https://doi.org/10.1109/ACCESS.2020.3022755

[2] https://highway-env.farama.org/environments/highway/

[3] https://highway-env.farama.org/observations/

[4] https://highway-env.farama.org/actions/

[5] https://highway-env.farama.org/rewards/

[6] Gupta, M. (2023, April 14). Advantage Actor Critic (A2C) Algorithm in Reinforcement Learning with Codes and Examples. *Medium*.

[7] Liao, J., Liu, T., Tang, X., Mu, X., Huang, B., & Cao, D. (2020). Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning. *IEEE Access*, *8*, 177804–177814. https://doi.org/10.1109/ACCESS.2020.3022755

8. Annexes

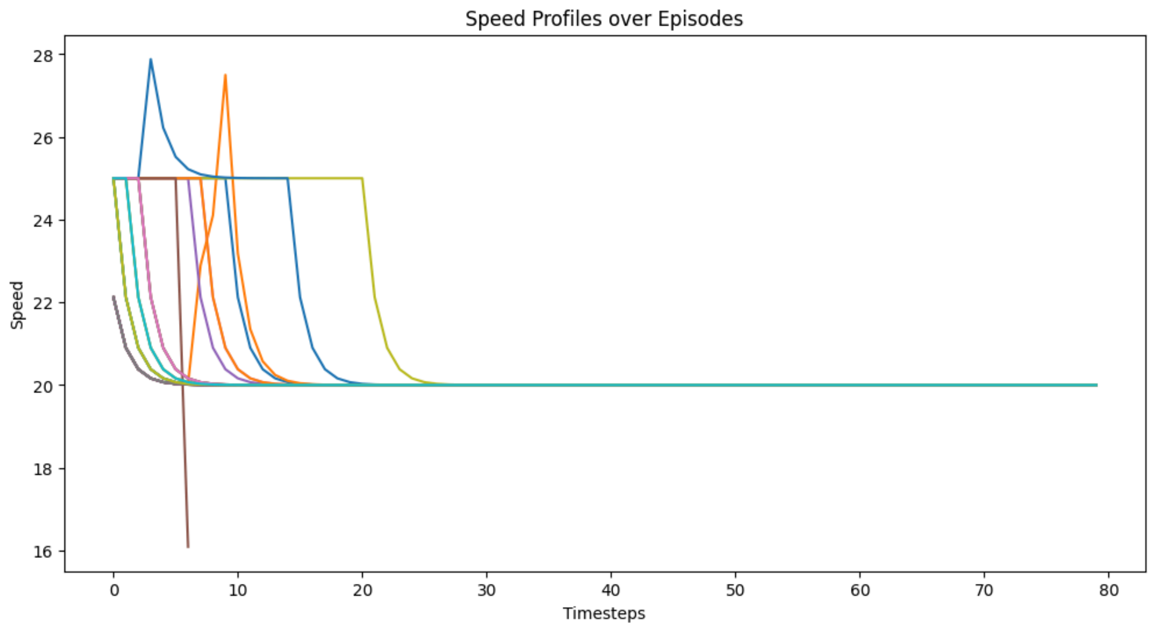

Figure 1. DQN - Speed Profiles Over Episodes

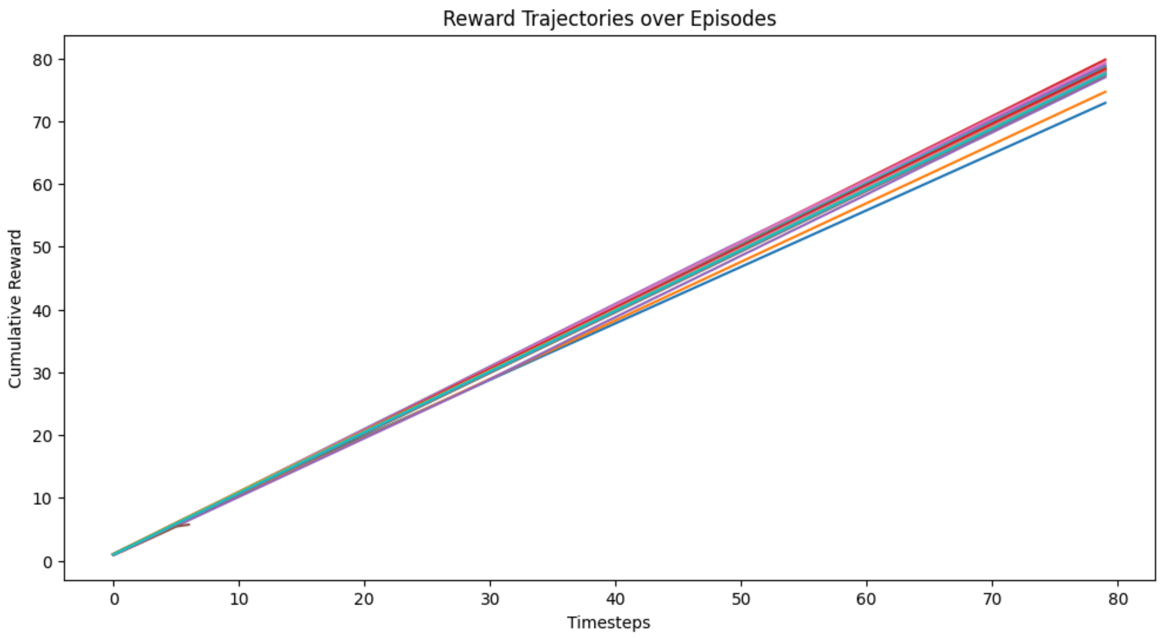

Figure 2. DQN - Reward Trajectories over Episodes

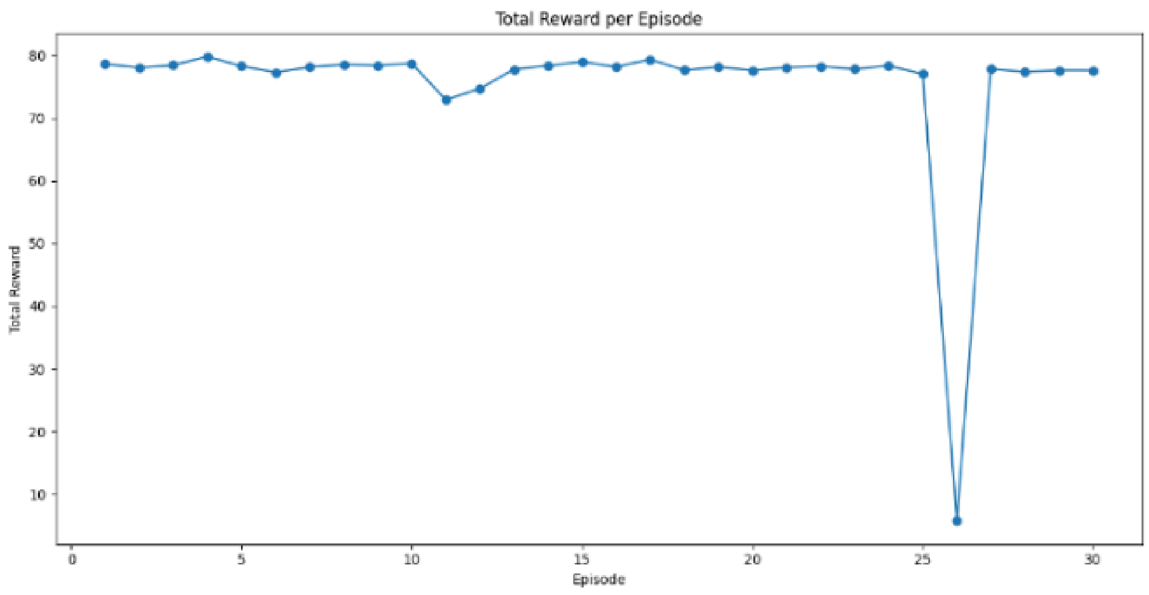

Figure 3. DQN - Total Reward per Episode

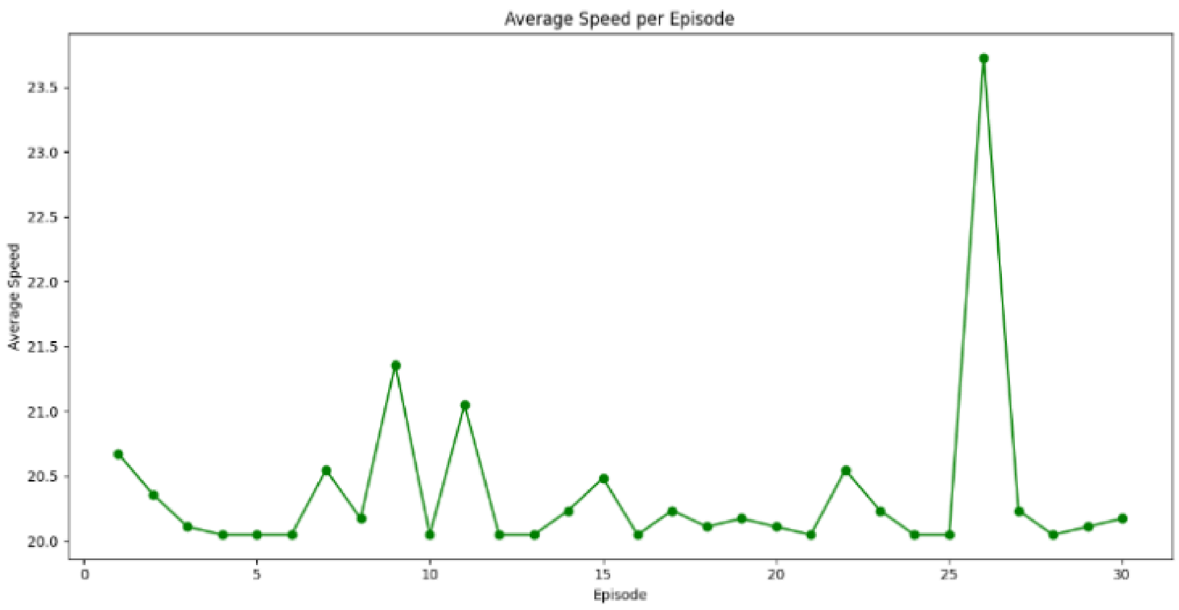

Figure 4. DQN - Average Speed per Episode

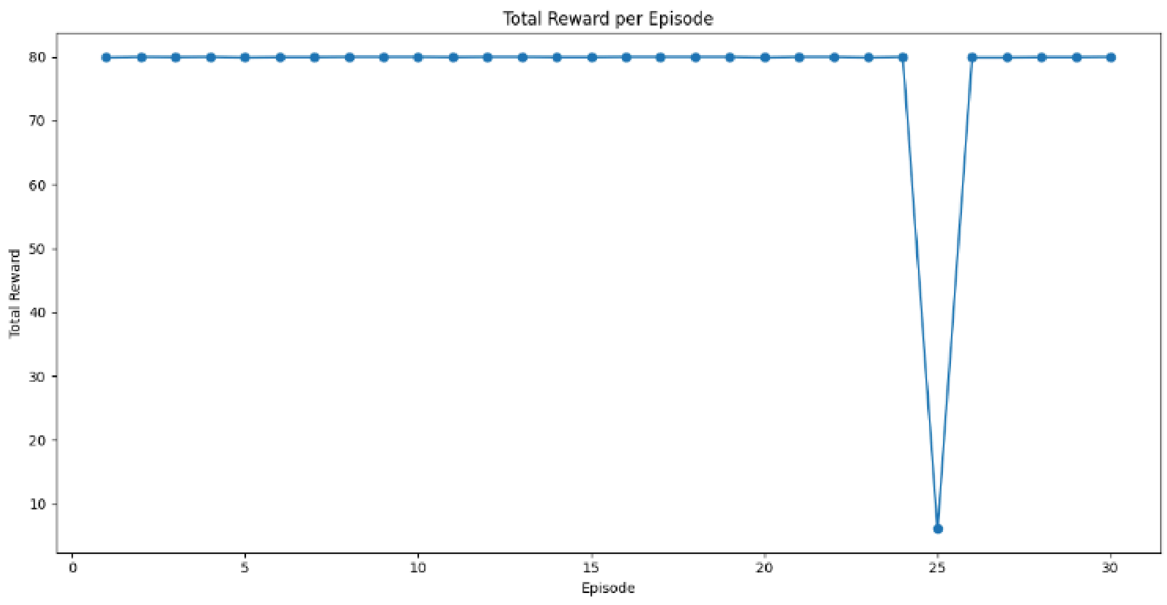

Figure 5. A2C - Total Reward per Episode

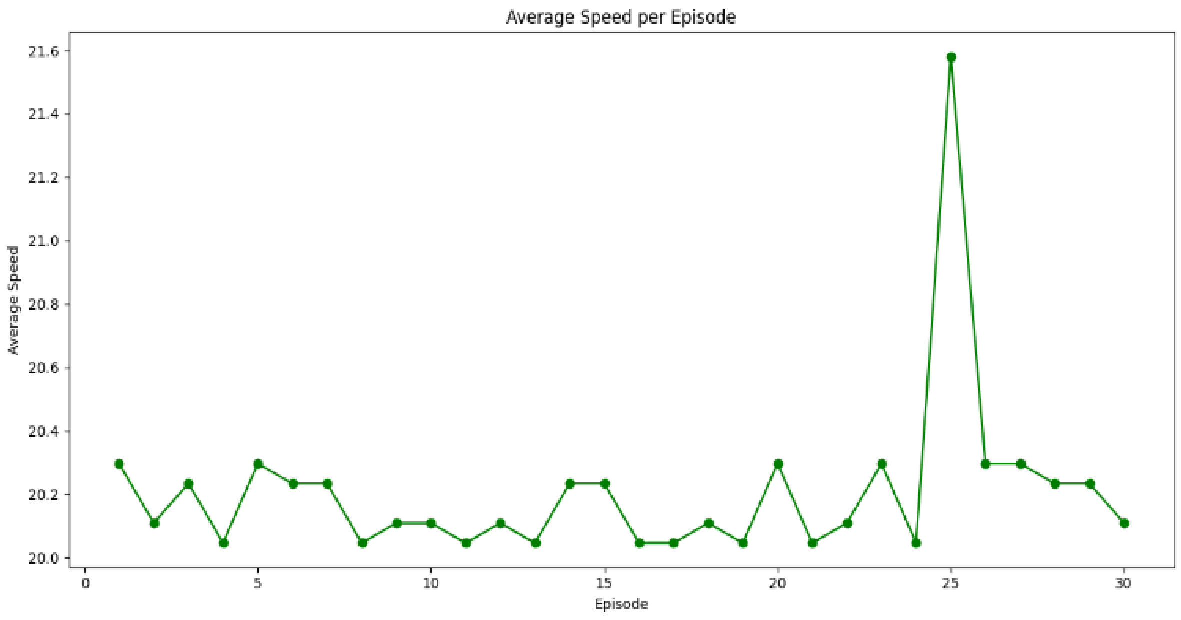

Figure 6. A2C - Average Speed per Episode

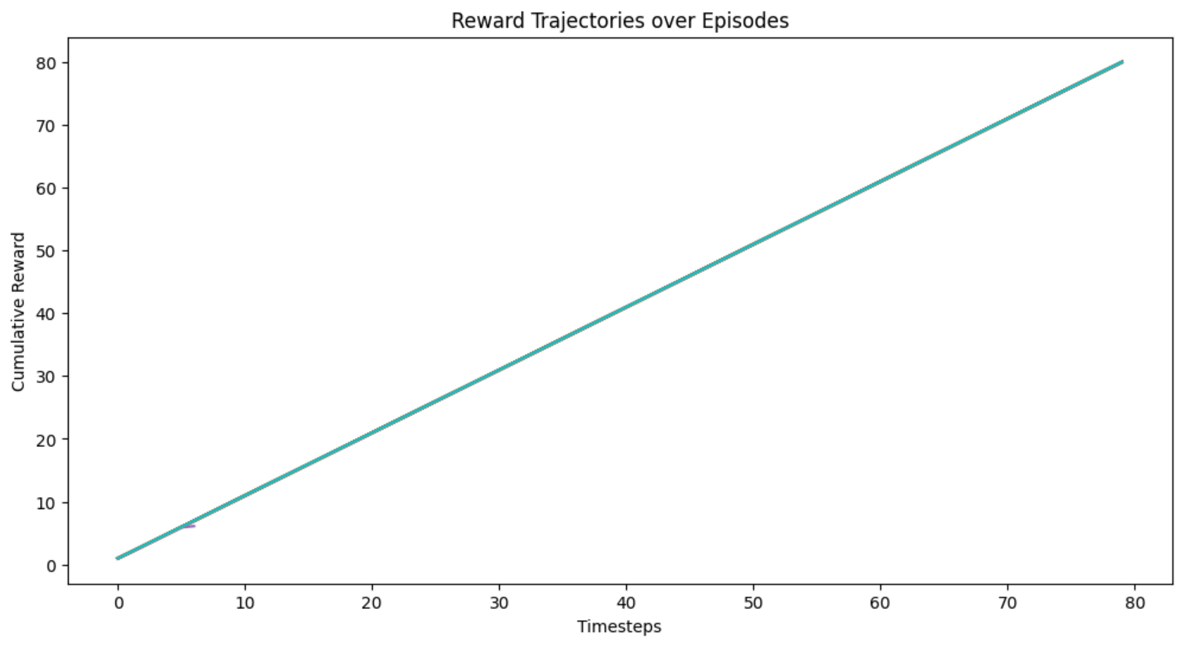

Figure 7. A2C Agent - Reward Trajectories over Episodes

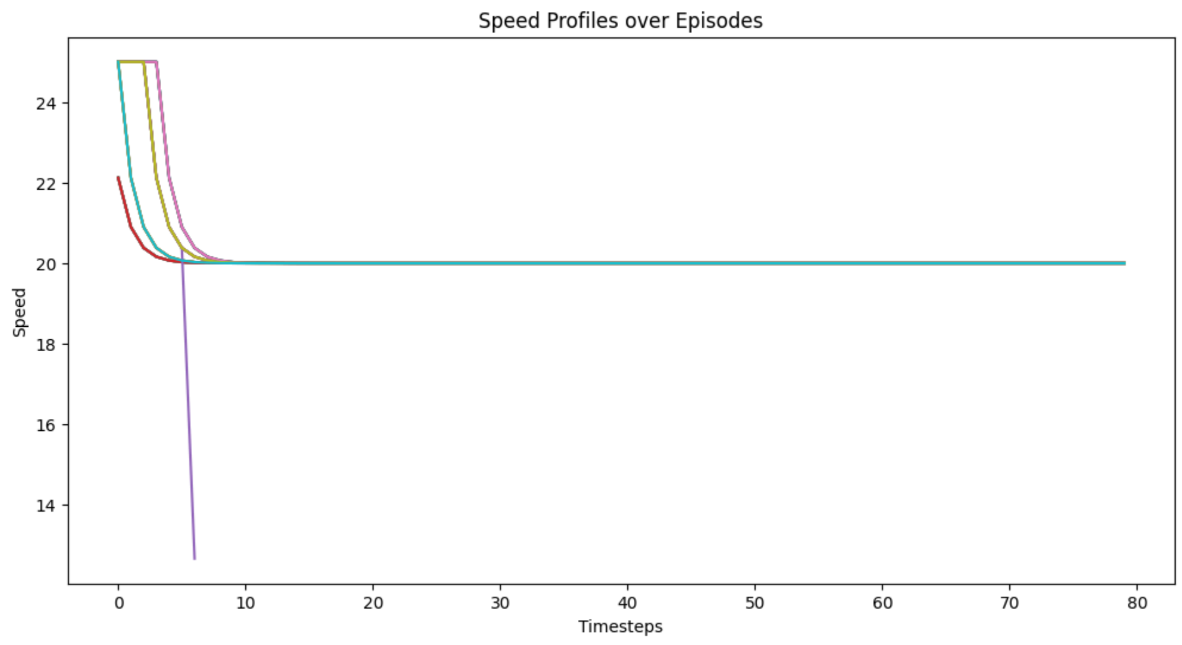

Figure 8. A2C Agent - Speed Profiles Over Episodes

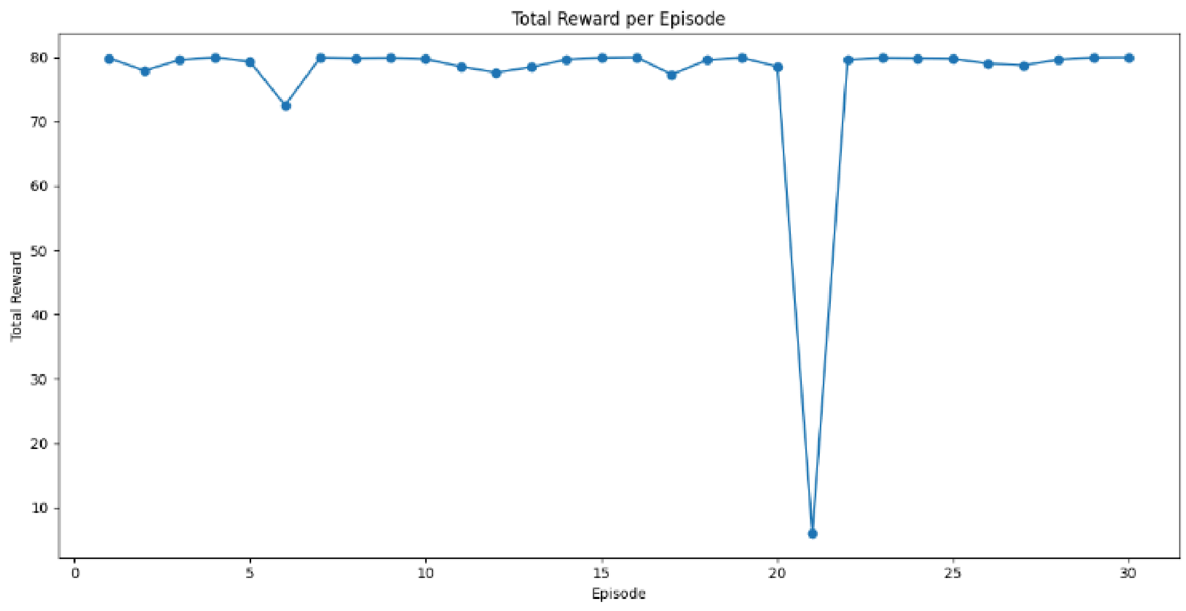

Figure 9. SAC - Total Reward per Episode

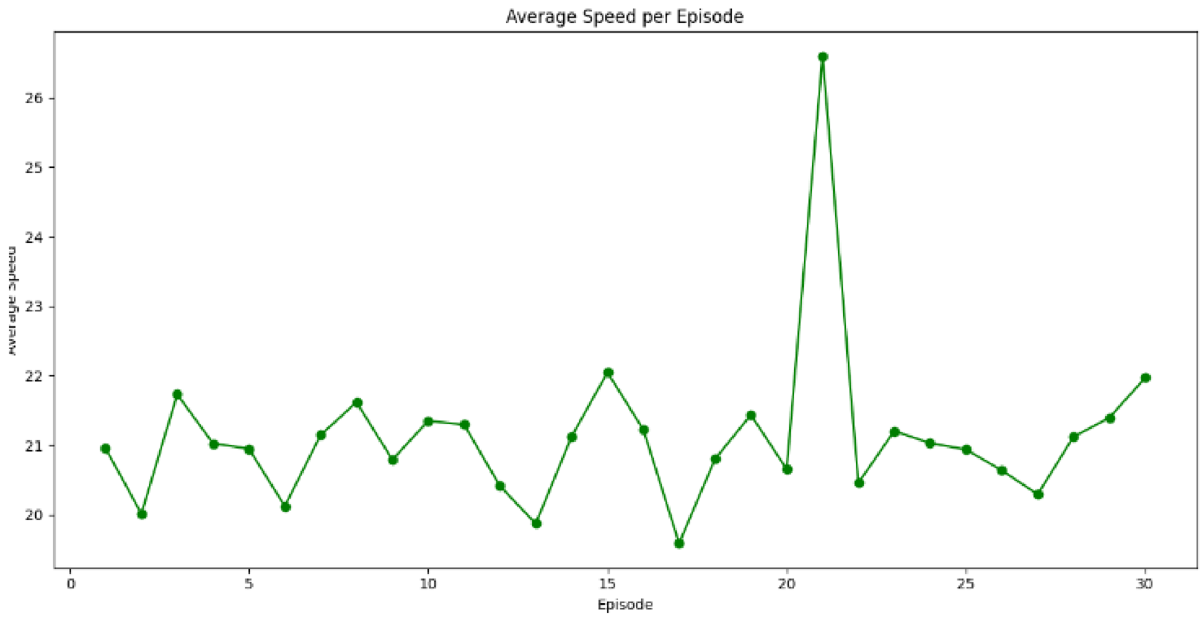

Figure 10. SAC - Average Speed per Episode

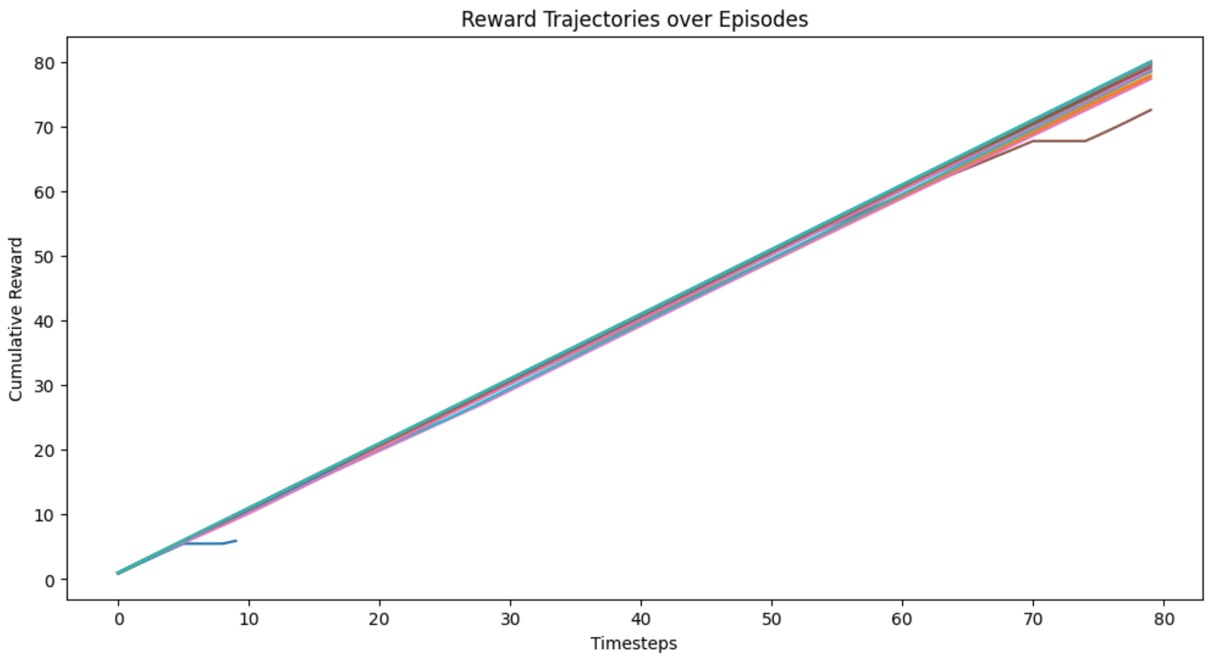

Figure 11. SAC - Reward Trajectories over Episodes

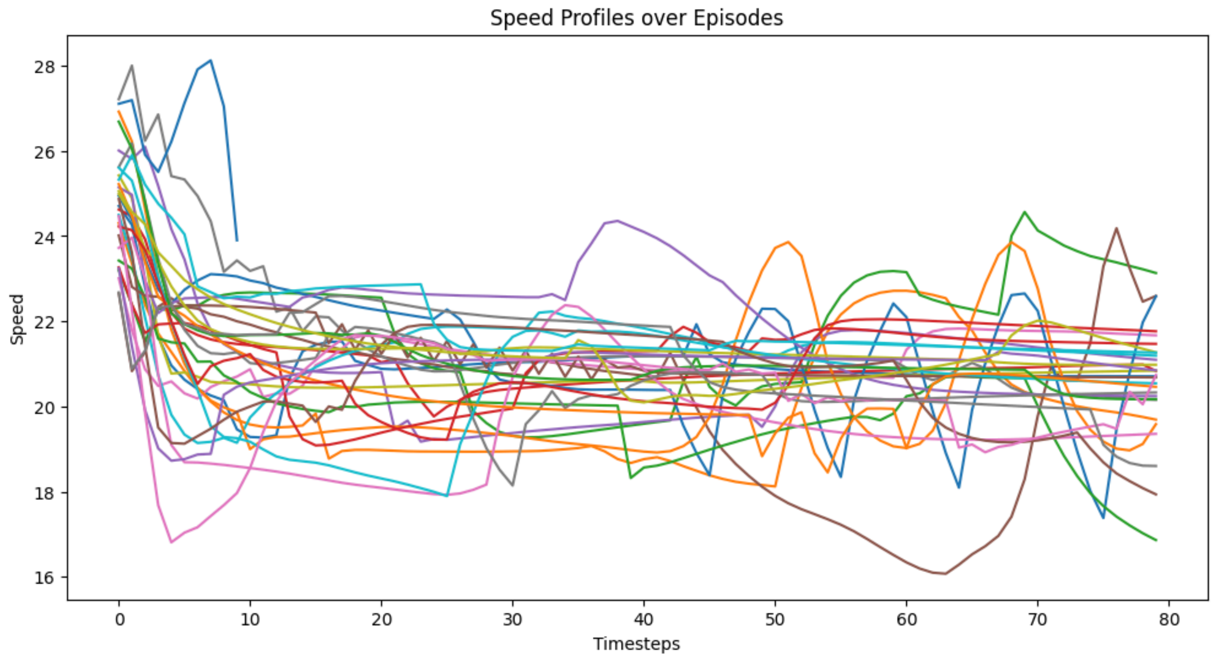

Figure 12 SAC - Speed Profiles over Episodes