Evaluation of Deep Reinforcement Learning methods in Autonomous Driving

Nicolas Guarini
Department of Computer and Systems Sciences (DSV)
Stockholm University, Stockholm, Sweden
nigu2420@student.su.se – web@nicolasguarini.it

***Abstract*—This project evaluates the effectiveness of three Deep Reinforcement Learning (DRL) methods, Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C), in addressing autonomous driving challenges. Using a customizable simulation environment, the agents were trained and tested across four diverse driving scenarios: *highway*, *roundabout*, *merge*, and *intersection*. The analysis focused on both the training process (e.g., reward progression) and the post-training performance of the models, evaluating metrics such as total reward, collision rate, and driving behavior realism. Results showed that PPO generally achieved the best overall performance in terms of efficiency and realism. However, DQN delivered results that were often comparable or only slightly inferior to PPO, demonstrating robustness in various scenarios. A2C, while effective in some cases, struggled with consistency and adaptability. All code, metrics, and additional analyses are available on GitHub.**

***Index Terms*—reinforcement learning, deep reinforcement learning, self-driving cars, autonomous driving**

I. Introduction

Autonomous driving has emerged as one of the most promising applications of artificial intelligence, with the potential to revolutionize transportation by enhancing safety, efficiency, and convenience. Among the various approaches to tackle this challenge, reinforcement learning (RL) has gained significant attention due to its ability to learn complex policies in dynamic and uncertain environments [1]. RL allows agents to learn through trial and error, optimizing their behavior based on rewards received from interacting with the environment [1]. This characteristic makes RL particularly well-suited for autonomous driving, where decision-making in diverse scenarios is critical.

In recent years, deep reinforcement learning (DRL), which combines RL with deep neural networks [2], has shown remarkable success in tasks like continuous control, autonomous navigation, and lane-keeping [3]. Algorithms such as Deep Q-Networks (DQN) [4], Proximal Policy Optimization (PPO) [5], and Advanced Actor Critic (A2C) [12] have demonstrated strong performance across various simulated and real-world environments. For example, DQN has been widely applied in discrete action spaces [4], while PPO offers robustness and scalability for both discrete and continuous control tasks [5]. Similarly, A2C, with its synchronous update mechanism, strikes a balance between computational efficiency and learning stability, making it effective for policy optimization in diverse settings [12]. Furthermore, a more modern and complex evolution of A2C that shares the same core architecture based on Actor-Critic, SAC (Soft Actor-Critic), is used by Waymo (Google) [9] for its advanced autonomous vehicles.

This project aims to evaluate the effectiveness of these DRL methods in the context of autonomous driving. Using a highly customizable simulation environment, a range of scenarios such as multi-lane highways, roundabouts, intersections, and lane merging will be explored. The analysis will focus on both the training process (e.g., reward progression, exploration rate, and loss) and the performance of trained models across different driving scenarios. By comparing and contrasting the results, this study seeks to highlight the strengths and limitations of each algorithm and provide insights into their applicability to autonomous driving tasks.

Related work in this domain [6] [7] [8] has highlighted the potential of DRL to address complex decision-making in autonomous vehicles. For instance, Waymo has employed reinforcement learning in combination with imitation learning to enhance its autonomous driving systems [9]. Building on this foundation, this project aims to further contribute to the understanding of how different DRL methods perform in various driving contexts, thereby advancing the state of the art in autonomous driving research.

II. Purpose

The primary purpose of this project is to evaluate the effectiveness and applicability of different deep reinforcement learning (DRL) algorithms in solving decision-making tasks within the context of autonomous driving. Autonomous vehicles must operate in highly dynamic and diverse environments, where they encounter complex scenarios such as merging onto highways, navigating roundabouts, and dealing with intersections. These tasks require the ability to adapt and make decisions in real-time while ensuring safety and efficiency.

This project aims to systematically compare the performance of algorithms such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C) in a customizable simulation environment. By focusing on both the training process and the final performance of the trained models, the study seeks to identify the strengths and limitations of each algorithm when applied to autonomous driving tasks. The analysis will include metrics such as reward progression, episode length, exploration rate, and overall success in different scenarios.

The ultimate goal is to provide insights into the suitability of each algorithm for various driving challenges, contributing to the development of more robust and efficient DRL solutions for autonomous vehicles. This evaluation might also serve as a reference for future research, highlighting areas where improvements are needed and guiding the selection of algorithms for specific autonomous driving applications.

III. Method

All experiments were carried out on a 2022 Macbook Air with M2 Apple Silicon Chip and 16GB of RAM, running MacOS Sequoia 15.1.1.

The complete source code is available on GitHub at https://github.com/nicolasguarini/drl-autonomous-driving-evaluation.

A. Environment

The environment chosen for the simulation of urban driving scenarios is *highway-env* [10], which provides a collection of urban driving scenarios integrated with *gym* [11], a toolkit developed by OpenAI for the development and evaluation of reinforcement learning algorithms. It offers various environments such as games and simulators where agents can be trained and evaluated [11].

Specifically, the scenarios and configurations used in this project are as follows:

- ***Highway*** (*5-lane, one-way highway*): the agent drives on a highway populated with other vehicles. The agent's goal is to achieve a high speed while avoiding collisions with other vehicles. The agent also receives an additional reward for driving in the rightmost lanes;
  Max reward: 40;
- ***Roundabout*** (*2-lane roundabout with 4 one-way roads*): the agent approaches a roundabout with traffic inside and approaching. It must navigate its route, manage lane changes, and avoid collisions with other vehicles;
  Max reward: 10;
- ***Intersection*** (*unregulated, with 4 one-way roads*): the agent faces an unregulated intersection with heavy traffic. It must yield appropriately and avoid collisions;
  Max reward: 10;
- ***Merge*** (*an access ramp merging into a two-lane one-way highway*): the agent drives on a highway, and a vehicle will soon merge from the access ramp. The agent's objective is to maintain speed while making space for the merging vehicle to safely enter the highway.
  Max reward: 15;

1) Actions: A *discrete* action set was chosen instead of a *continuous* one to simplify the computational training process of the models.

The action space is defined as follows [10]:

```
ACTIONS_ALL = {
    0: 'LANE_LEFT',
    1: 'IDLE',
    2: 'LANE_RIGHT',
    3: 'FASTER',
    4: 'SLOWER'
}
```

Of course, not all actions are always feasible (e.g., `LANE_RIGHT` if the agent is already in the rightmost lane or accelerating beyond the maximum speed). If the agent attempts an infeasible action, it defaults to the `IDLE` action [10].

2) Reward Function: In general, across all scenarios (except the parking scenario, which is not covered in this project), the two main features of focus are maintaining high speed (within limits) and avoiding collisions with other vehicles [10].

The reward function is defined as follows [10]:

$$R(s,a) = a \cdot \frac{v - v_{\min}}{v_{\max} - v_{\min}} - b \cdot \text{collision}$$

where:

- v is the current speed;
- $v_{\min}$ is the minimum speed;
- $v_{\max}$ is maximum speed;
- a and b are two coefficients.

In highway scenarios, the reward is also influenced by the agent staying in the rightmost lanes [10].

An important detail is that negative rewards are forbidden, since they may incourage the agent to terminate the episodes early (e.g., causing a collision) instead of risking a negative return [10].

The behavior of other vehicles is designed to emulate realistic traffic dynamics. Vehicles follow a combination of predefined rules and stochastic behaviors. They adapt their speed and lane changes based on the traffic flow, maintaining safety distances and reacting to the agent's actions. This creates a dynamic environment where the agent must learn to navigate effectively while anticipating the behavior of surrounding vehicles [10].

B. RL Methods

Three Reinforcement Learning methods have been chosen:

- Deep-Q Networks (DQN) [4];
- Advantage Actor Critic (A2C) [12];
- Proximal Policy Optimization (PPO) [5];

The implementations of these algorithms from the *stable-baselines3* library were used, which is a set of implementations of RL algorithms based on OpenAI Baselines, integrated with the TensorFlow's TensorBoard tool [13].

Each model based on these methods was trained individually on each of the 4 selected scenarios, resulting in a total of $3 \times 4 = 12$ agents.

1) Deep-Q Networks (DQN): Q-learning is a reinforcement learning algorithm where an agent learns an optimal policy by estimating the value of state-action pairs, called Q-values [4]. Deep Q-Networks (DQNs) extend Q-learning, enabling agents to learn policies directly from high-dimensional sensory inputs, such as images, using Convolutional Neural Networks (CNNs). Unlike traditional CNNs that classify images, DQNs output Q-values for actions based on input states, allowing agents to learn from their behavior without requiring extensive labeled data [4].

In DQNs, the Q-value is updated as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\big),$$

where α is the learning rate, γ the discount factor, and $Q(s_t, a_t)$ the estimated value of taking action a_t in state s_t [4].

During training, the agent iteratively transitions between states, performing actions, receiving rewards, and adjusting its Q-values. Episodes may terminate early upon failures, such as collisions.

DQN is one of the most used, studied, and well-known algorithms in the RL field, but it is relatively simpler compared to PPO and A2C, which are based on policy gradients and require the optimization of probability distributions for actions. However, DQN remains a solid benchmark as it serves as a bridge between classic Q-learning and the high-dimensional continuous action spaces typical in urban driving scenarios.

The parameters selected for training with DQN are the following:

- `net_arch=[256, 256]`: specifies a neural network with two hidden layers of 256 neurons each, providing sufficient capacity for complex environments;
- `learning_rate = 5e-4`: a standard moderate value to ensure stable convergence without overshooting;
- `buffer_size = 10000`: size of the storage of past experiences for sampling during training;
- `learning_starts = 200`: starts training after 200 steps to ensure the buffer has enough data for meaningful updates;
- `batch_size = 64`: typical size for stable gradient estimation while balancing memory and computation;
- `gamma = 0.8`: balances the importance of immediate vs. future rewards, favoring near-term optimization.

2) Advantage Actor-Critic (A2C): A2C is a RL method that combines the strengths of the Actor-Critic approach with a streamlined, centralized model for parameter updates. A2C employs an actor to determine the policy and a critic to estimate the advantage, which is the difference between the expected value of an action and the average value of all actions in a given state [12].

This design reduces variance in gradient estimation, leading to more stable learning and faster convergence compared to other RL methods like DQN or pure policy gradient approaches [12].

Compared to more computationally demanding or complex methods, A2C offers an effective combination of simplicity, efficiency, and scalability, making it a robust choice for applications in dynamic environments.

The parameters selected for training with A2C are the following (`net_arch`, `learning_rate`, and `gamma` are defined the same as DQN):

- `n_steps=5`: accumulates gradients over 5 steps before updating the model, balancing responsiveness and computational cost;
- `gae_lambda=0.95`: controls the bias-variance tradeoff in Generalized Advantage Estimation, improving learning stability.

3) Proximal Policy Optimization (PPO): Proximal Policy Optimization (PPO) is a policy gradient method that improves the TRPO (Trust Region Policy Optimization) method by simplifying its implementation and enhancing sample efficiency while maintaining reliable performance [5]. Instead of relying on a hard constraint, PPO uses a "clipped" surrogate objective function to control policy updates:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\big[\min(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta),\ 1-\epsilon,\ 1+\epsilon)\hat{A}_t)\big]$$

where ϵ is a hyperparameter (e.g. $\epsilon = 0.2$), and $r_t(\theta)$ represents the probability ratio [5].

The clipping mechanism prevents drastic policy changes by bounding the probability ratio, ensuring stable and cautious updates. PPO can also incorporate an adaptive KL (Kullback-Leibler) penalty as an alternative to clipping, dynamically adjusting the penalty coefficient to regulate the KL divergence [5].

The advantages of PPO include its reliance on simpler first-order optimization methods, compatibility with parameter sharing and noisy architectures, and superior performance compared to methods such as A2C and TRPO in diverse tasks [5].

The parameters selected for training with PPO are the following (`net_arch`, `learning_rate`, `gamma` and `batch_size` are defined the same as DQN):

- `n_steps=2048`: collects 2048 steps of data per policy update, which helps improve gradient estimates while being computationally efficient;
- `n_epochs=10`: specifies the number of optimization epochs per update, allowing for thorough policy refinement;
- `clip_range=0.2`: restricts the probability ratio to the range $[1-\epsilon, 1+\epsilon]$, ensuring stable updates without large policy deviations.

C. Training Process

All models were trained for a total of 40,000 timesteps, which turned out to be a good compromise between the overall training duration (approximately 2 hours for the *highway* scenario) and the quality of the resulting models.

The collection of metrics during training was carried out through TensorBoard, a visualization tool for monitoring and analyzing metrics during the training of machine learning models, such as loss, accuracy, and other statistics [13].

The metrics chosen to analyze and compare the learning process of the models are:

- *Episode Length Mean*
- *Episode Reward Mean*
- *Exploration Rate*
- *Entropy Loss* (only for A2C and PPO)
- *Explained Variance* (only for A2C and PPO)

The code related to this part is inside the files in the `scripts/` folder, named using the format `train_{METHOD}.py`.

D. Performance Evaluation

The trained models were evaluated by running them in their respective scenarios for a total of 100 episodes, collecting the following metrics for each *episode*:

- *Total Reward*;
- *Episode Length*;
- *Crash* (boolean);
- *Average Speed*;
- *Collision Reward*;
- *Right Lane Reward* (only for *highway* scenarios);
- *High Speed Reward*;
- *On Road Reward*.

This approach allowed for the evaluation and comparison of the models by analyzing specific aspects of their behavior within the various scenarios.

These metrics were obtained using the *highway-env* API and collected into .csv files using *Pandas*.

The code related to this part is inside the files in the `scripts/` folder, named using the format `run_{METHOD}.py`.

IV. Results

We will begin by analyzing the training progress and subsequently the performance of the trained models. Practical behaviors of the agents will also be analyzed to assess the realism of their actions. Due to space constraints, the training analysis will focus on the trend of the *reward*, as it is the most comprehensive indicator of learning quality. Still, all graphs are available in the GitHub repository.

A. Highway Scenario

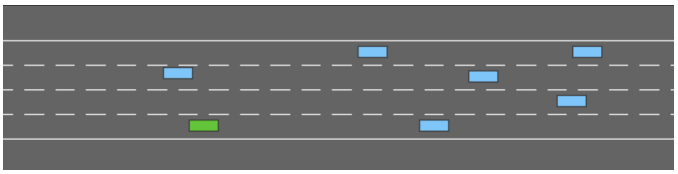

Fig. 1. Used Highway scenario configuration

1) Training: In Figure 2, we observe that DQN and PPO showed relatively consistent improvements, achieving average reward levels towards the end of training (around the 35k-th step) between approximately 20 and 23 (with a maximum of 30). However, PPO reached these excellent values around 15k steps, whereas DQN achieved them at around 35k. Additionally, PPO proved to be more computationally efficient, maintaining a stable 7 fps and completing training in just one and a half hours. A2C, on the other hand, was the most unstable algorithm in learning, consistently achieving inferior results compared to the other two at the same number of steps.

2) Trained Models Performances: Regarding driving style, the agent trained with PPO is the most balanced, maintaining a relatively steady speed consistent with other vehicles and most of the time changing lanes in advance when vehicles in the distance cause traffic. This allows it to "predict" uncomfortable

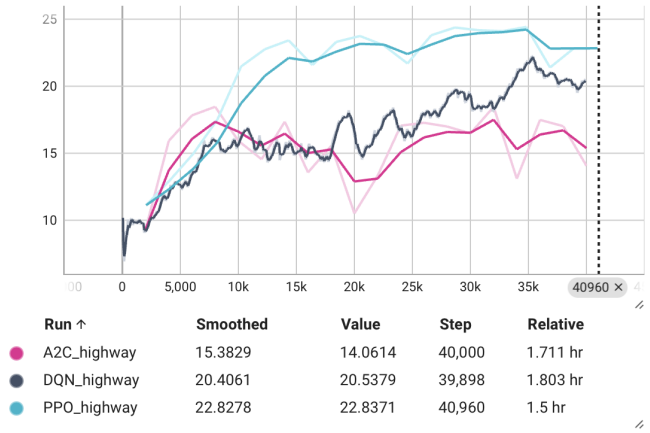

Fig. 2. Comparison of rewards during training in the *highway* scenario

Model	Ep. Len.	Rew.	Crashes	High Spd. R.	R. Lane R.
DQN	25.34	22.86	75%	20.29	15.98
PPO	35.57	27.33	18%	6.76	29.06
A2C	27.75	20.95	48%	6.54	15.49

TABLE I
TRAINED AGENTS' PERFORMANCES IN *highway* SCENARIO ON 100 EPISODES

situations and act accordingly. DQN is equally skilled at determining when to change lanes to avoid collisions but tends to significantly increase speed on clear roads without adequately reducing it when slower vehicles appear in the distance, sometimes failing to avoid collisions due to lack of time or space.

A2C developed a different and much more passive strategy, essentially moving at a particularly moderate speed. Consequently, its rewards are mediocre as it proceeds slowly, avoids overtaking, and rarely actively avoids collisions, and when it needs to, it fails to do so.

The results presented in Table I reflect these observations, highlighting how the PPO agent exhibits the most optimal behavior among the three.

B. Roundabout Scenario

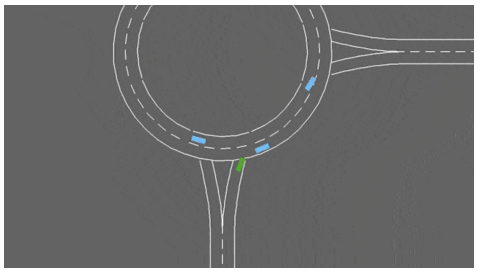

Fig. 3. Used Roundabout scenario configuration

1) Training: In Figure 4, we can observe that this time A2C achieves excellent reward values right from the start and in a very stable manner. This is unusual, as in the initial episodes, the agent has not yet developed a strategy and should therefore

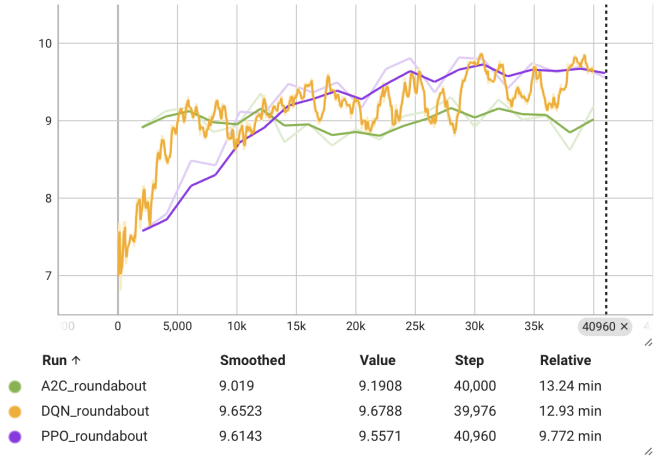

Fig. 4. Comparison of rewards during training in the *roundabout* scenario

achieve low rewards. The other two agents, on the other hand, exhibit a more natural learning curve, both stabilizing around an average reward of slightly less than 10.

Model	Avg. Ep. Len.	Avg. Rew.	% Crashes
DQN	10.71	9.68	7%
PPO	10.82	9.81	8%
A2C	9.79	8.7	28%

TABLE II
TRAINED AGENTS' PERFORMANCES IN *roundabout* SCENARIO ON 100 EPISODES

2) Trained Models Performances: Analyzing the behavior of the A2C agent, it was observed that it always enters the roundabout without slowing down or yielding to vehicles already inside and proceeds at a constant speed in the outermost lane. This strategy generally leads to moderately acceptable rewards but is not very natural and often results in collisions. DQN and PPO, on the other hand, exhibit much more realistic behavior, closely resembling typical human driving. They slow down near the roundabout, enter only after yielding to other vehicles if necessary, and move to the inner lane when the outer lane is occupied or will soon be occupied by other vehicles.

The results reported in Table II confirm these observations.

C. Merge Scenario

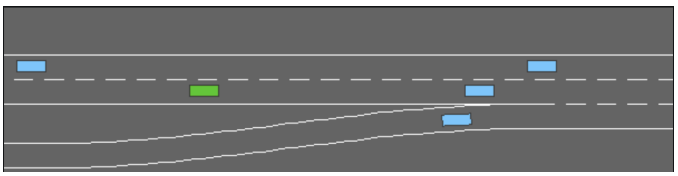

Fig. 5. Used Merge scenario configuration

1) Training: In Figure 6, we can observe that this time A2C appears to be the best-performing agent, with an average reward of approximately 14 by the end of training. However,

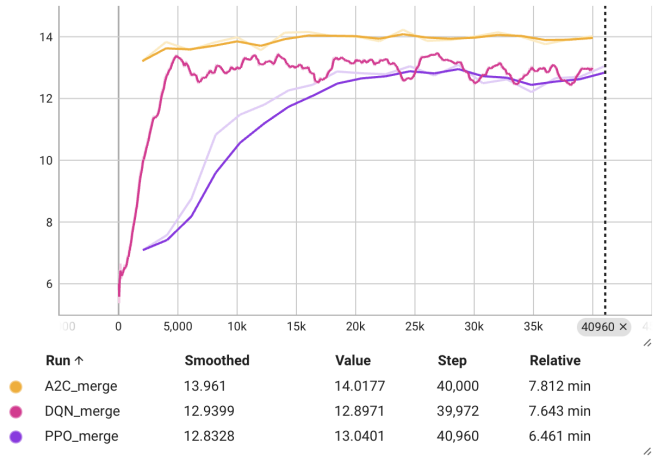

Fig. 6. Comparison of rewards during training in the *merge* scenario

we again notice that it achieves high reward levels right from the start, unlike the other two agents, which demonstrate a more natural and gradual learning process, reaching average rewards of around 13 by the end of training.

Model	Avg. Ep. Len	Avg. Rew.	Crashes	Avg. R. Lane Rew.
DQN	14.8	12.78	3%	11.16
PPO	15.72	13.48	5%	14.29
A2C	16.81	13.72	7%	16.81

TABLE III
TRAINED AGENTS' PERFORMANCES IN *merge* SCENARIO ON 100 EPISODES

2) Trained Models Performances: In Table III, we can see that the three agents achieved similar results, with A2C performing slightly better than the others, at the cost of a slightly higher number of collisions.

Analyzing the agents' behavior, it was observed that A2C achieves great results right from the beginning because, in this case as well, it simply follows the rightmost lane at the same speed as the other vehicles without taking any further actions. PPO and DQN, on the other hand, developed a more active strategy, changing lanes to avoid slow vehicles merging into the highway and adjusting their speed when necessary.

D. Intersection Scenario

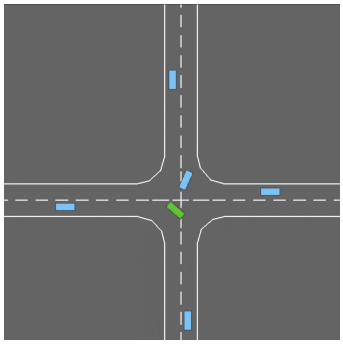

Fig. 7. Used Intersection scenario configuration

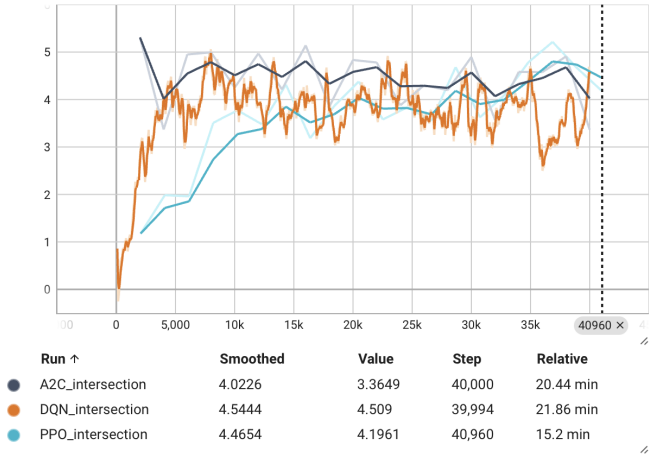

Fig. 8. Comparison of rewards during training in the *intersection* scenario

1) Training: In Figure 8, we can observe that the three agents achieve similar results, stabilizing at rewards around 4. However, in this scenario, the maximum reward, achieved by successfully completing the left turn while yielding and avoiding collisions, is 10. None of the agents, therefore, managed to develop an optimal strategy in this scenario.

Model	Avg. Ep. Len	Avg. Rew.	% Crashes
DQN	8.21	4.75	41%
PPO	7.9	5.24	41%
A2C	7.23	4.13	55%

TABLE IV
TRAINED AGENTS' PERFORMANCES IN *intersection* SCENARIO ON 100 EPISODES

2) Trained Models Performances: Observing the results reported in Table IV, we note that the collision percentage is extremely high for all agents, especially A2C.

The agents' behavior is similar in all cases, and it is worth noting that collisions are often caused by other vehicles crashing into each other. This leads to premature termination of the episodes, resulting in mediocre rewards for the agents, which consequently have limited opportunities to experience complete episodes and learn from them.

V. CONCLUSIONS

This study evaluated the performance of three deep reinforcement learning (DRL) algorithms—DQN, A2C, and PPO—in four diverse autonomous driving scenarios: highway, roundabout, merging, and intersection. The results highlighted notable differences in the strengths and weaknesses of these methods, emphasizing their applicability to various driving tasks.

PPO generally delivered the best overall performance, thanks to its robust policy update mechanism, which balances stability and adaptability. It demonstrated realistic and effective driving behaviors across all scenarios, excelling particularly in dynamic environments such as highway merging. DQN, while less sophisticated, achieved performance often comparable or only slightly inferior to PPO in most scenarios, showcasing its robustness and effectiveness despite its simplicity. A2C, on the other hand, showed moderate success but struggled with training stability and underperformed in complex traffic scenarios.

Theoretical insights might explain these observations: DQN's reliance on Q-value estimation [4] makes it well-suited for static or moderately dynamic environments but limits its performance in tasks requiring fine-grained action adjustments. A2C combines policy and value optimization through an actor-critic model [12], offering efficiency but facing challenges in convergence stability. PPO's policy clipping mechanism [5] enables steady and reliable improvements, making it the most adaptable and effective choice for diverse driving scenarios.

In conclusion, PPO emerged as the most reliable method for autonomous driving tasks, while DQN proved to be a competitive alternative with fewer computational demands. A2C demonstrated potential in simpler settings but struggled with complexity. These findings highlight the importance of aligning algorithm selection with task requirements and provide valuable insights for future research in DRL for autonomous systems.

REFERENCES

[1] Richard S. Sutton and Andrew G. Barto. "Reinforcement Learning: An Introduction". Second Edition. Cambridge, MA: The MIT Press, 2018.
[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017.
[3] Talpaert, Victor & Sobh, Ibrahim & Kiran, Bangalore & Mannion, Patrick & Yogamani, Senthil & Sallab, Ahmad & Perez, Patrick. (2019). Exploring Applications of Deep Reinforcement Learning for Real-world Autonomous Driving Systems. 564-572.
[4] T. Okuyama, T. Gonsalves and J. Upadhay, "Autonomous Driving System based on Deep Q Learnig," 2018 International Conference on Intelligent Autonomous Systems (ICoIAS), Singapore, 2018.
[5] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG].
[6] Talpaert, Victor & Sobh, Ibrahim & Kiran, Bangalore & Mannion, Patrick & Yogamani, Senthil & Sallab, Ahmad & Perez, Patrick. (2019). Exploring Applications of Deep Reinforcement Learning for Real-world Autonomous Driving Systems. 564-572.
[7] M. Dhinakaran, R. T. Rajasekaran, V. Balaji, V. Aarthi and S. Ambika, "Advanced Deep Reinforcement Learning Strategies for Enhanced Autonomous Vehicle Navigation Systems," 2024 2nd International Conference on Computer, Communication and Control (IC4), Indore, India, 2024.
[8] Duan, Jingliang, et al. "Hierarchical Reinforcement Learning for Self-driving Decision-making without Reliance on Labelled Driving Data." IET Intelligent Transport Systems, vol. 14, no. 5, Feb. 2020, pp. 297–305. Portico, Crossref.
[9] Lu, Yiren, et al. "Imitation Is Not Enough: Robustifying Imitation with Reinforcement Learning for Challenging Driving Scenarios." 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2023, pp. 7553–60.
[10] Leurent Edouard. "An Environment for Autonomous Driving Decision-Making". 2018. GitHub repository.
[11] Towers, Kwiatkowski, Terry, et.al. "Gymnasium: A Standard Interface for Reinforcement Learning Environments". 2024. arXiv.
[12] P.-H. Su, P. Budzianowski, S. Ultes, M. Gasic, and S. Young, "Sample-efficient Actor-Critic Reinforcement Learning with Supervised Data for Dialogue Management," pp. 147–157, 2018.
[13] Abadi et.al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". Google Research. 2015.