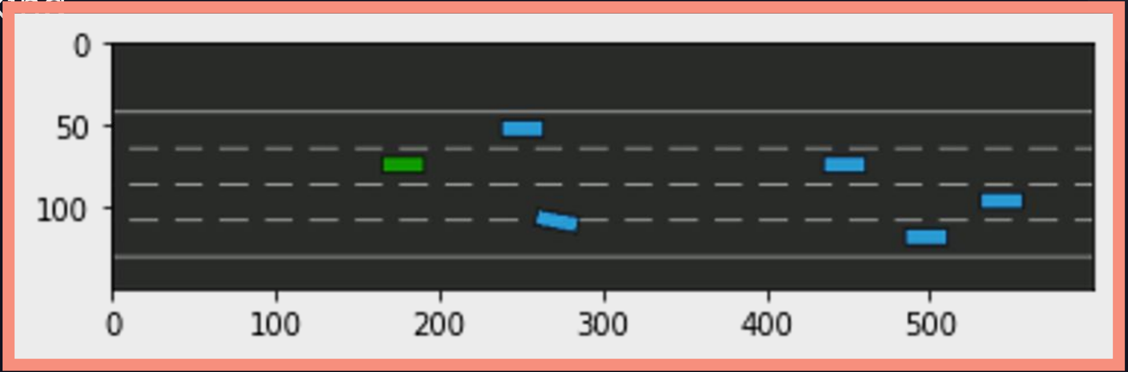
Autonomous Driving Optimization using Reinforcement Learning

Group - 1

Shyam Akhil Nekkanti - 8982123
Layanika Vinay Saravanan - 8934459
Samyukth Lalith Lella Gopal - 9005574

Project Overview

- **Goal:** Develop autonomous driving agents using Reinforcement Learning (RL)
- **Focus:** Optimize routing and lane management while ensuring safety
- **Environment:** Highway-Env 2D simulation
- **Objective:** Train agents to handle complex driving scenarios with intelligent decision-making



Problem Statement

Challenges in Autonomous Driving:

1. Limited adaptability to dynamic traffic patterns
2. Poor decision-making in complex scenarios

Limitations of existing approaches:

- Rule-based systems: Cannot handle unexpected situations
- Supervised learning: Needs extensive labeled data, struggles with edge cases

Our solution:

Use Reinforcement Learning to train agents that learn optimal driving policies through interaction, improving both safety and efficiency in diverse driving environments.

Reinforcement Learning Algorithms

DQN (Deep Q-Network):

DQN is a value-based Reinforcement Learning algorithm that uses a neural network to approximate the Q-value function, enabling an agent to learn optimal actions in complex environments with discrete action spaces.

Why DQN?

- It learns from experience by remembering past actions and rewards (experience replay).
- It chooses actions using an ε-greedy strategy — sometimes exploring, sometimes picking the best-known action.
- It uses a target network to make learning more stable.

In Our Project:

- DQN helps the car decide when to change lanes, speed up, or slow down.
- It learns to drive safely and efficiently by getting rewards for good driving and penalties for risky behavior.

Evaluation Metrics

We evaluate our RL agents using the following key performance indicators:

1. **Collision Avoidance Rate (Safety):**
   Measures how effectively the agent avoids collisions throughout each driving episode.
2. **Travel Time and Speed Maintenance (Efficiency):**
   Evaluates whether the agent maintains a safe yet optimal speed while minimizing total travel time.
3. **Scenario Completion Rate (Robustness):**
   Tracks how often the agent successfully completes different driving scenarios, including high-density traffic and merges.
4. **Training Convergence Speed (Learning Efficiency):**
   Indicates how quickly the algorithm learns an optimal driving policy during training (measured by reward stabilization).

State Representation

State Function – A state function(st) represents the current situation of the agent in the environment at time t.

$$st = [pt, vt, lt, Nt]$$

At each time step t, the agent observes a state vector composed of:

- p_t: Position of the vehicle
- v_t : Velocity (current speed) of the vehicle
- l_t : Lane the vehicle is currently in
- N_t : Information about neighboring vehicles, including positions and velocities of nearby cars

These features provide the context for decision-making and are fed into the DQN model.

Action Space (a_t)

Action Space (a_t) –
The agent operates in a discrete action space with 5 possible high-level driving actions:

$$a_t \in \{\text{LANE\_LEFT}, \text{LANE\_RIGHT}, \text{FASTER}, \text{SLOWER}, \text{IDLE}\}$$

1. **LANE_LEFT:** Change to the left lane if safe
2. **LANE_RIGHT:** Change to the right lane
3. **FASTER:** Accelerate to increase speed
4. **SLOWER:** Decelerate to reduce speed
5. **IDLE:** Maintain current speed and lane

These actions allow the agent to make strategic driving decisions such as overtaking, maintaining safety, and adjusting to traffic. The DQN learns the optimal action in each state to maximize future rewards.

Reward (R)

Reward Function –

The reward function balances safety and driving efficiency:

$$R(s_t , a_t , s_{t+1}) = w_1 (-\alpha \cdot d^2_{collision}) + w_2 (\beta_1 \cdot v_t + \beta_2 \cdot I_{fast\_lane} - \beta 3 \cdot I_{lane\_change})$$

❖ **Collision penalty ($-\alpha \cdot d^2_{collision}$):**

- The agent is penalized for being too close to other vehicles to ensure safety.

❖ **Speed reward ($\beta_1 \cdot v_t$):**

- Encourages the agent to maintain a good speed for efficient driving.

❖ **Lane preference:**

- $\beta_2 \cdot I_{fast\_lane}$ (– reward)
- $-\beta 3 \cdot I_{lane\_change}$ (– penalty)
- Rewards the agent for staying in the fast lane and penalizes unnecessary lane changes.

❖ **Weights (w_1 & w_2):**

- Adjust the importance of safety vs. efficiency in training.

This explanation makes it easy to understand how your agent learns to drive smart and safe on the highway.

Optimal Policy ($\pi^*(s)$)

optimal policy function–

The optimal policy π^*(s) tells the agent the best action to take in any given state. It aims to maximize the total expected future reward over time.

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

This equation means:

❖ $\pi^*(s)$ is the optimal policy function

- It tells the agent which action to take in state(s)to get the highest future reward.

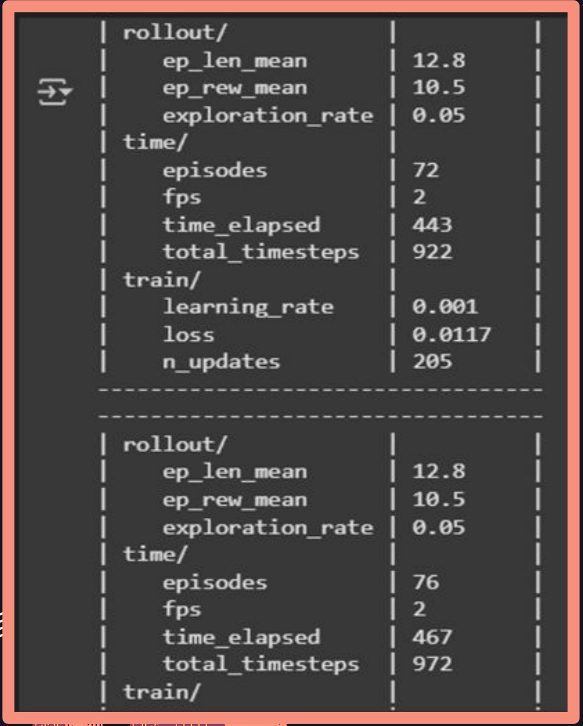
❖ $Q^*(s,a)$ is the optimal action-value function

- It gives the maximum expected cumulative reward for taking action(a) in state(s), and then always following the optimal policy.

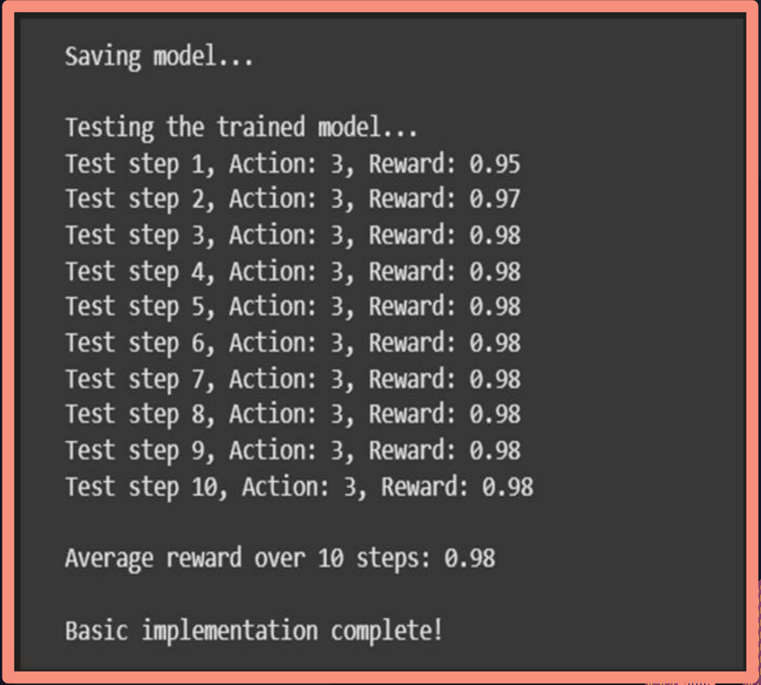
So, the optimal policy guides the agent to make the most rewarding decisions at each state, enabling it to navigate the environment efficiently and achieve long-term goals.

Training Results

1. Our initial DQN training shows promising results even with limited training:

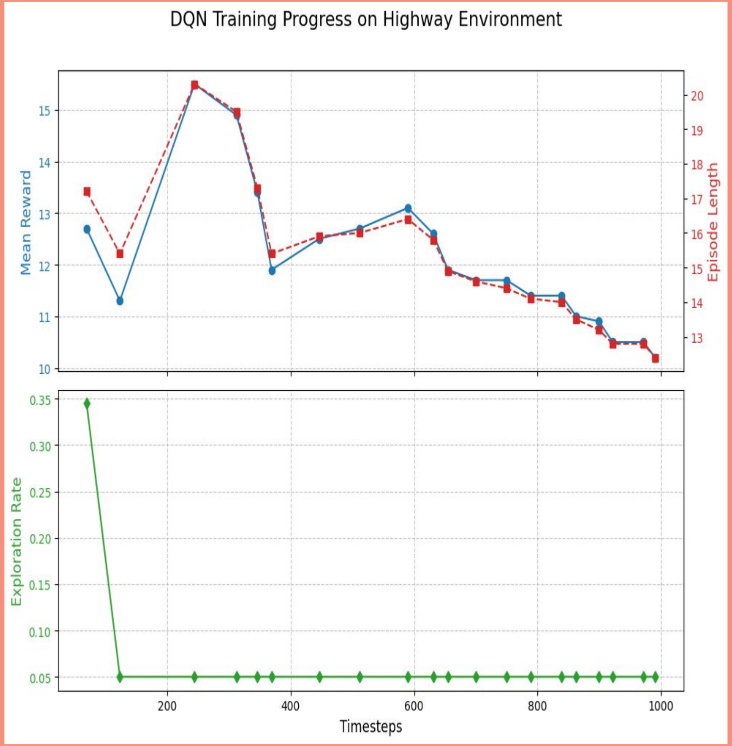


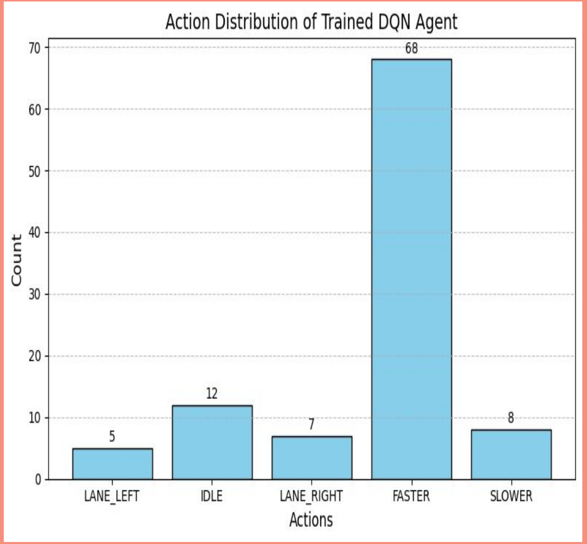
```
| rollout/            |          |
|    ep_len_mean      | 12.8     |
|    ep_rew_mean      | 10.5     |
|    exploration_rate | 0.05     |
| time/               |          |
|    episodes         | 72       |
|    fps              | 2        |
|    time_elapsed     | 443      |
|    total_timesteps  | 922      |
| train/              |          |
|    learning_rate    | 0.001    |
|    loss             | 0.0117   |
|    n_updates        | 205      |
----------------------------------
----------------------------------
| rollout/            |          |
|    ep_len_mean      | 12.8     |
|    ep_rew_mean      | 10.5     |
|    exploration_rate | 0.05     |
| time/               |          |
|    episodes         | 76       |
|    fps              | 2        |
|    time_elapsed     | 467      |
|    total_timesteps  | 972      |
| train/              |          |
```



```
Saving model...

Testing the trained model...
Test step 1, Action: 3, Reward: 0.95
Test step 2, Action: 3, Reward: 0.97
Test step 3, Action: 3, Reward: 0.98
Test step 4, Action: 3, Reward: 0.98
Test step 5, Action: 3, Reward: 0.98
Test step 6, Action: 3, Reward: 0.98
Test step 7, Action: 3, Reward: 0.98
Test step 8, Action: 3, Reward: 0.98
Test step 9, Action: 3, Reward: 0.98
Test step 10, Action: 3, Reward: 0.98

Average reward over 10 steps: 0.98

Basic implementation complete!
```

Training Results





Challenges Faced

We evaluate our RL agents using the following key performance indicators:

1. Environment Configuration Issues

- Initially faced errors while launching the Highway-Env simulator due to missing or incompatible packages.
- **Solution:** Resolved by properly installing dependencies and configuring the environment settings.

2. Training Instability

- Observed that the model's performance plateaued early, with little reward improvement over episodes.
- **Solution:** Tuned hyperparameters such as learning rate, epsilon decay, and batch size to enhance exploration and learning.
- **Future scope:** Implement a dynamic learning rate scheduler to improve long-term training stability.

3. Algorithm Behavior Bias

- Agent tended to overuse the "FASTER" action, indicating reward function sensitivity.
- **Solution:** Planning to experiment with modified reward structures to balance aggressive vs. safe driving behavior.

Expected Final Outcomes

1) Comprehensive Algorithm Comparison

- Analyze and compare the performance of DQN, PPO, and A2C across various driving scenarios (highway, merge, intersection) using standardized metrics.

2) High Safety Performance

- Achieve at least 90% collision avoidance rate, ensuring agents can drive safely in complex and dynamic environments.

3) Reward Function Analysis

- Investigate how different reward structures influence agent behavior, and refine them to balance speed, safety, and efficiency.

4) Scenario-Specific Algorithm Recommendations

- Identify which RL algorithm performs best for each scenario and explain why (e.g., PPO for merges, DQN for highways).

5) Result Visualization and Reporting

- Present training curves, performance graphs, and behavior snapshots for each agent to demonstrate learning progress and outcomes.

Thank you