**Course/Section:** CSC 580 AI 2 Final Project

**Assignment Name:** Reinforcement Learning Analysis in Highway Environments

**Group:** 40 **| Names:** Om Prakash Gunja (2131025), Raju Meesala ( 2119844 )

**Word Count:**

**Final Project Report – Reinforcement Learning Analysis in Highway Environments**

**1. Introduction**

In this project, we explored and analyzed the performance of various Reinforcement Learning (RL) models, primarily focusing on Deep Q-Network (DQN) variants trained using Stable-Baselines3 in the context of highway driving environments. Our initial goal was to develop a robust baseline policy in the “highway-fast-v0” environment. We then transferred the learned policy to the “merge-v0” environment for fine-tuning, testing the agent’s ability to adapt to a more complex driving scenario.

--roundabout details

Autonomous driving is a rapidly advancing field, with applications ranging from advanced driver-assistance systems to fully self-driving vehicles. Reinforcement Learning provides a promising avenue for training agents to handle continuous decision-making tasks such as lane-following, collision avoidance, and merging. By systematically tuning hyperparameters—such as the learning rate, discount factor, exploration fraction, and network architecture—we aimed to uncover best practices for achieving stable and high-performing policies in these environments.

**2. Methodology**

**2.1 Environment**

We conducted our experiments in two primary environments provided by the **highway-env** library:

1. **Highway-Fast-v0:**

• Simplified version of highway driving with high-speed traffic.

• Allows the RL agent to learn fundamental driving maneuvers (lane following, speed control) and collision avoidance with fewer vehicles.

2. **Merge-v0:**

• More complex environment where the agent must merge onto a highway.

• Involves additional vehicles, varying speeds, and lane-change complexities.

• Challenges the agent to anticipate other cars’ behaviors and plan merges effectively.

3. **Roundabout-v0:**

• More complex environment where the agent must take round path onto a highway.

• Involves additional vehicles, varying speeds, and lane-change complexities.

• Challenges the agent to anticipate other cars’ behaviors and plan merges effectively.

**DQN Algorithm to Train**

**2.2 RL Algorithm and Framework**

We utilized **DQN** from **Stable-Baselines3**, which implements the core ideas of Q-learning with a neural network function approximator. The agent interacts with the environment by selecting actions that maximize the Q-value estimates. Key features include:

• **Experience Replay:** Stores past transitions in a replay buffer to break correlations and stabilize training.

• **Target Network:** A periodically updated copy of the main Q-network to provide more stable Q-value targets.

• **Epsilon-Greedy Exploration:** Balances exploration and exploitation through a decaying epsilon parameter.

While our focus was on DQN, we briefly considered PPO and A2C for potential GPU acceleration benefits. However, the bulk of our experiments centered on systematically tuning DQN.

**2.3 Hyperparameter Tuning**

We varied several hyperparameters across different runs:

1. **Learning Rate (LR):**

• Ranged from 5e-4 to 1e-2.

• High LR can speed up early learning but risks instability, while lower LR can stabilize updates at the cost of slower convergence.

2. **Buffer Size:**

• Values like 10,000 or 15,000.

• Larger buffers allow more diverse replay but can slow updates if not enough steps are collected.

3. **Gamma (Discount Factor):**

• Ranged from 0.8 to 0.95.

• Higher gamma emphasizes future rewards and can encourage long-term planning, while lower gamma focuses on immediate returns.

4. **Exploration Fraction:**

• Typically set between 0.6 and 0.7.

• Determines the fraction of total training steps during which epsilon decays from 1.0 to its minimum value.

5. **Network Architecture:**

• We tested smaller MLPs vs. larger ones (e.g., [512, 256, 128]).

• Larger networks can learn more complex policies but may overfit or slow down training significantly.

**2.4 Experimental Runs**

We conducted multiple runs to isolate the impact of each hyperparameter combination:

1. **DQN\_1 (Baseline):**

• **Timesteps:** 2,000

• **Learning Rate:** 5e-4

• **Result:** Short training time (~33 seconds) led to a modest peak reward of ~9.

2. **DQN\_hyper1:**

• **Timesteps:** 20,000

• **Learning Rate:** 1e-2 (high)

• **Result:** Reached reward ~12, with some instability in loss. Training took ~56 minutes on CPU.

3. **DQN\_hyper2:**

• **Timesteps:** 20,000

• **Learning Rate:** 1e-3

• **Network:** [512, 256, 128]

• **Gamma:** 0.95

• **Result:** Achieved a peak reward of ~25 but dropped to ~15–17 by the end. Demonstrated the highest peak performance but with late-stage instability.

4. **DQN\_hyper3:**

• **Timesteps:** 20,000

• **Learning Rate:** 1e-2

• **Network:** [512, 256, 128]

• **Gamma:** 0.8

• **Exploration Fraction:** 0.6

• **Result:** Did not reach hyper2’s peak but ended with a more stable reward (~20+). Training took ~1.5–1.8 hours.

The best overall performance in terms of **peak reward** was seen with hyper2, while hyper3 finished with a **more stable final** reward.

**3. Results and Analysis for Highway-Fast-v0**

**3.1 Baseline Model (DQN\_1)**

**Reward:** Peaked at ~9, with short episode lengths (~11–12).

**Analysis:**

• Insufficient training steps limited learning.

• Provided a useful reference to measure how extended training and hyperparameter changes affect performance.

A graph on a screen

AI-generated content may be incorrect.

**A screenshot of a graph

AI-generated content may be incorrect.**

**3.2 DQN\_hyper1**

**Reward:** Improved to ~12, with episodes ~15–16 steps.

**Learning Rate:** 1e-2 was likely too aggressive, leading to some volatility.

**Analysis:**

• Showed that longer training (20k steps) does boost performance.

• The high LR caused large updates, sometimes destabilizing Q-values.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

**3.3 DQN\_hyper2**

**Reward:** Surpassed all runs with a peak of ~25 but dipped near the end.

**Network:** [512, 256, 128] improved representational capacity.

**Learning Rate:** 1e-3 struck a balance between speed and stability, though not perfectly stable.

**Analysis:**

• Indicated the environment can yield very high rewards if the network can discover and maintain optimal strategies.

• Late-stage instability hints that a learning rate schedule (e.g., decay) or higher target update interval might help preserve the best policy.

A graph on a white background

AI-generated content may be incorrect.

**A screenshot of a graph

AI-generated content may be incorrect.**

**3.4 DQN\_hyper3**

**Reward:** Final performance stabilized around ~20+ by the end.

**Gamma:** 0.8, focusing more on immediate rewards, potentially beneficial in a high-speed environment.

**Learning Rate:** 1e-2 again led to fast learning but with the large network, it took ~1.5+ hours to train.

**Analysis:**

• Did not reach the same peak as hyper2 but ended higher than hyper2’s final.

• Potentially a good “stable baseline” to transfer to other tasks.

**A screenshot of a graph

AI-generated content may be incorrect.**

**A screenshot of a graph

AI-generated content may be incorrect.**

**4. Transfer Learning to Merge-v0**

**4.1 Rationale for Transfer**

The **merge-v0** environment introduces lane-change and merging complexities. Rather than training from scratch, we aimed to leverage a policy pre-trained on highway-fast-v0, hypothesizing that the learned features (e.g., collision avoidance, speed control) would transfer and reduce training time.

**4.2 Selecting a Model for Transfer**

While **DQN\_hyper2** had the highest peak reward, it exhibited a decline in the later stages. **DQN\_hyper3** ended with a more consistent performance (~20+). For a stable baseline, we opted to **copy the weights from DQN\_hyper3** into the merge environment.

**Why hyper3?**

• More stable final performance.

• Lower exploration fraction (0.6) might help the agent exploit a decent policy sooner in the merge environment.

• Gamma=0.8 can be advantageous in situations where immediate maneuvers (like quick merges) are critical.

**4.3 Implementation Details**

# Create the merge environment

env\_merge = gym.make("merge-v0", render\_mode="rgb\_array")

# Load the pre-trained model (hyper3) and set the new environment

model = DQN.load("path\_to\_saved\_model/dqn\_highway\_hyper3.zip", env=env\_merge)

# Optionally, adjust hyperparameters for the merge task

model.learning\_rate = 1e-3 # Lower LR for fine-tuning

model.gamma = 0.9 # Increase gamma slightly for more forward planning

model.exploration\_fraction = 0.5 # Exploit the learned policy sooner

# Fine-tune the model in the merge environment

model.learn(total\_timesteps=int(2e4))

# Save the fine-tuned model

model.save("path\_to\_saved\_model/dqn\_merge\_finetuned.zip")

We used a smaller learning rate for fine-tuning to avoid catastrophic forgetting. We also adjusted gamma upward (e.g., 0.9) if we believed the merge environment demands a bit more foresight.

**5. Merge Environment Results**

**5.1 Initial Performance**

Upon loading **DQN\_hyper3** weights, the agent started in the merge environment with a baseline policy already familiar with highway dynamics. Early observations showed:

• **Higher Initial Reward:** Compared to training from scratch, the agent began with moderate success, suggesting that prior knowledge of lane and speed control was beneficial.

• **Exploration-Exploitation Balance:** With exploration fraction around 0.5, the agent had enough random actions to adapt to new merge-specific states, but also exploited the highway driving skills learned previously.

**A screenshot of a graph

AI-generated content may be incorrect.**

**A screenshot of a graph

AI-generated content may be incorrect.**

**5.2 Fine-Tuning Curves**

**Episode Reward (ep\_rew\_mean):**

• Started around 5–7, climbing steadily toward 15–20 after ~10k steps.

• Some fluctuations occurred, indicating the agent was still adapting to the merge dynamics (e.g., adjusting spacing and timing for merges).

**Episode Length (ep\_len\_mean):**

• Typically hovered around 20–25, suggesting the agent survived longer episodes and effectively avoided collisions or early terminations.

• Slightly shorter than highway-fast, likely due to the environment’s termination conditions once a merge is completed or fails.

**Loss (train/loss):**

• Showed moderate spikes but less volatility than seen in early highway training, implying the pre-trained weights helped stabilize updates.

**5.3 Comparison with Direct Training on Merge**

In a quick side experiment, we trained a model from scratch on **merge-v0** with the same architecture. It took significantly more timesteps to reach comparable performance. This indicates that **transfer learning** indeed saved training time and leveraged the agent’s prior driving knowledge.

**5.4 Next Tuning Steps for Merge**

1. **Learning Rate Decay:**

Start at 1e-3 and linearly decay to 5e-4 or 1e-4 over ~20k steps to stabilize final performance.

2. **Gamma Tuning:**

If merges require even more planning, try 0.95. If the environment is heavily reliant on immediate merges, remain near 0.8–0.9.

3. **Reward Shaping:**

Incorporate domain-specific rewards (e.g., safe merges, penalize abrupt lane changes) to guide the agent more directly.

4. **Monitor Collisions:**

Consider creating a custom env wrapper that tracks collisions and logs them to see if merges are safe or if the agent is “cheating” the reward function.

5.5 Fine tuning merge model 2

A screenshot of a computer program

AI-generated content may be incorrect.