Reinforcement Learning for Autonomous Shuttle Simulation

Final Project Log: Legion of Nazgul

Team: Legion of Nazgul

November 9, 2025

Project Overview

This project implements a Deep Reinforcement Learning (DRL) agent for the autonomous navigation of a shuttle within a simulated 10×10 grid map of IIT Kanpur, using a custom Gymnasium environment. The primary goal is to maximize navigation safety and path efficiency using a Deep Q-Network (DQN) with Double DQN enhancements.

Contributors and Timeline

- **Course:** DES646 - AIML for Designers
- **Team:** Legion of Nazgul
- **Duration:** Oct 20 – Nov 9, 2025
- **Members:** Ayush Yadav, Vibhanshu Choudhary, Adiba Khan, Vyom Pratap Singh, Yash Giri

Stepwise Methodology

1. Environment Simulation

- Modeled IITK campus as a 10×10 grid.
- **State Space:** Discrete tuples (x, y) representing location, plus binary flags for obstacles, start, goal.
- **Action Space:** Four possible moves (up, down, left, right).

2. Deep Q-Network Model Architecture

- **Base:** DQN with enhancements:
- **Double DQN** to reduce overestimation.
- **Experience Replay** for decorrelated updates.
- **Target Network Synchronization** to stabilize Q-values.
- **Epsilon Decay** for exploration-exploitation balance.

DQN Update Equation:

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

Double DQN Target:

$$y_{\text{DDQN}} = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a'; \theta), \theta') \tag{2}$$

3. Reward Engineering

Safety-first reward configuration:

- **Collision:** $R_{\text{crash}} = -100$
- **Goal:** $R_{\text{goal}} = 1000$
- Other rewards/penalties as designed in code for intermediate steps.

4. Exploration Decay

Epsilon Decay Function:

$$\epsilon = \max(\epsilon_{\text{min}}, \epsilon_0 \cdot \exp(-kt)) \tag{3}$$

Where ϵ_{min} is the minimum exploration rate, ϵ_0 initial, and k is the decay rate.

5. Visualization

Plotly dashboards for real-time reward and convergence monitoring.

Training and Results

- **Episodes:** 50,000
- **Success Rate:** 92%
- **Hardware:** T4 GPU, 45 min runtime
- **Collision Rate:** 0% (after tuning)
- **Reward Convergence:** Stable after 40,000 episodes.

Reflection & Future Work

Challenges

Sim-to-Real Gap: The 2D grid simplifies real-world navigation, limiting generalization.

Future Extensions

- Add **dynamic obstacles** and complex traffic scenarios.
- Port simulation to a **3D Unity environment** for increased realism.
- Deploy on **TurtleBot** for physical world experiments.

Dependencies & Execution

Python Libraries:

- `gymnasium, numpy, pandas, tensorflow, keras, matplotlib, plotly`

  Execution:

- `python trainagent.py`
- `python visualizeresults.py`

Configurable parameters in `config.json`.

Summary

A robust DRL agent was developed for autonomous navigation with strong safety and efficiency metrics. The work forms a foundation for advanced research and publication in AI-driven transportation robotics.