Adaptive Traffic Signal Control Using Deep Reinforcement Learning and V2I Data

By Emmanuel Kiprotich

Research - Project

Problem Statement

Background:

- Traditional traffic signals operate based on fixed-time plans or simple reactive rules.
- These systems cause unnecessary delays, waste fuel, and increase emissions, especially during fluctuating traffic conditions.

Project Goal:

- Use Deep Reinforcement Learning to dynamically adjust traffic signal timing based on real-time traffic conditions from Vehicle-to-Infrastructure (V2I) communication.
- Objective: Minimize vehicle delays and improve traffic flow efficiency.

Datasets Used

1. **NGSIM Peachtree Dataset:**
    - **Content:** High-resolution vehicle trajectories (positions, speeds) on a real urban corridor in Atlanta.
    - **Purpose:** To extract vehicle arrival patterns, queueing behavior, and realistic traffic dynamics for simulation.

2. **NYC CV Pilot Dataset:**
    - **Content:** Real-world V2I communication messages, including Signal Phase and Timing (SPaT) broadcasts and Basic Safety Messages (BSMs).
    - **Purpose:** To simulate realistic V2I communication events influencing traffic control.

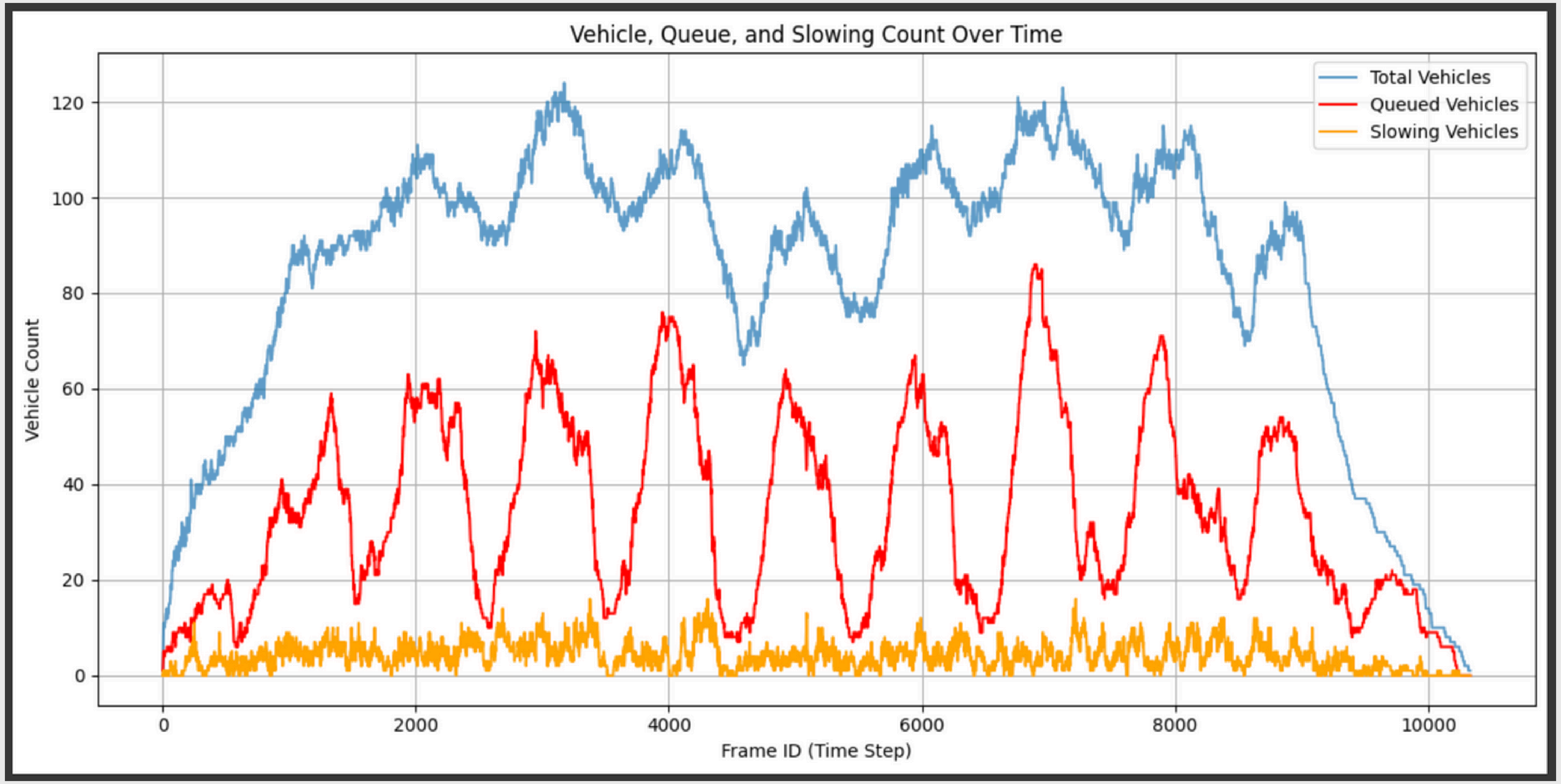
Data Preprocessing

- Cleaned missing or corrupted records.
- **From NGSIM:**
  - Defined queued and slowing vehicles based on speed thresholds.
  - Aggregated features like Vehicle_Count, Queued_Vehicle_Count, Slowing_Vehicle_Count.
- **From NYC CV:**
  - Parsed time-series communication events.
  - Extracted Event_Count, Excessive_Speed_Count, spatial coordinates.

Data Preprocessing

- **From NGSIM: Extracted queued vehicles, slowing vehicles**

```
ngsim_df_clean['is_queued'] = ngsim_df_clean['v_Vel'] <= 2.0  # Fully stopped
ngsim_df_clean['is_slowing'] = (ngsim_df_clean['v_Vel'] > 2.0) & (ngsim_df_clean['v_Vel'] <= 6.0)  # Slowing down
```



Methodology

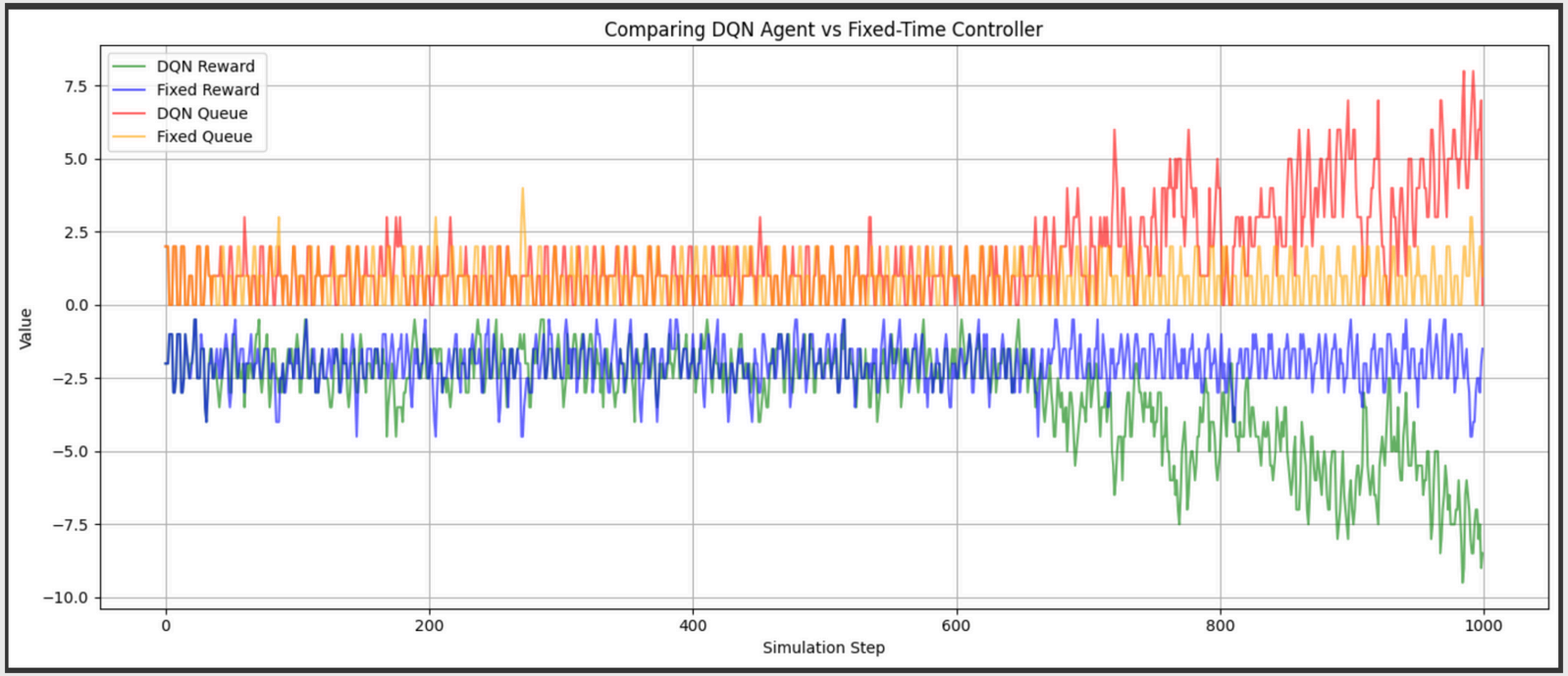
- **Simulator:**
  - SUMO (Simulation of Urban Mobility) + TraCI API for real-time control.
- **Agent Inputs:**
  - Number of queued vehicles.
  - Number of slowing vehicles.
- **Actions:**
  - Maintain or switch the traffic light phase.
- **Reward Function:**
  - Penalize delays and stop-go driving.
  - Reward smoother flows.

Baseline Model: Deep Q-Network (DQN)

- Trained a DQN agent on traffic simulation for 10,000 timesteps.
- Used a basic state-action-reward feedback loop to learn.
- Baseline Result: Improved over a naive fixed-schedule signal.

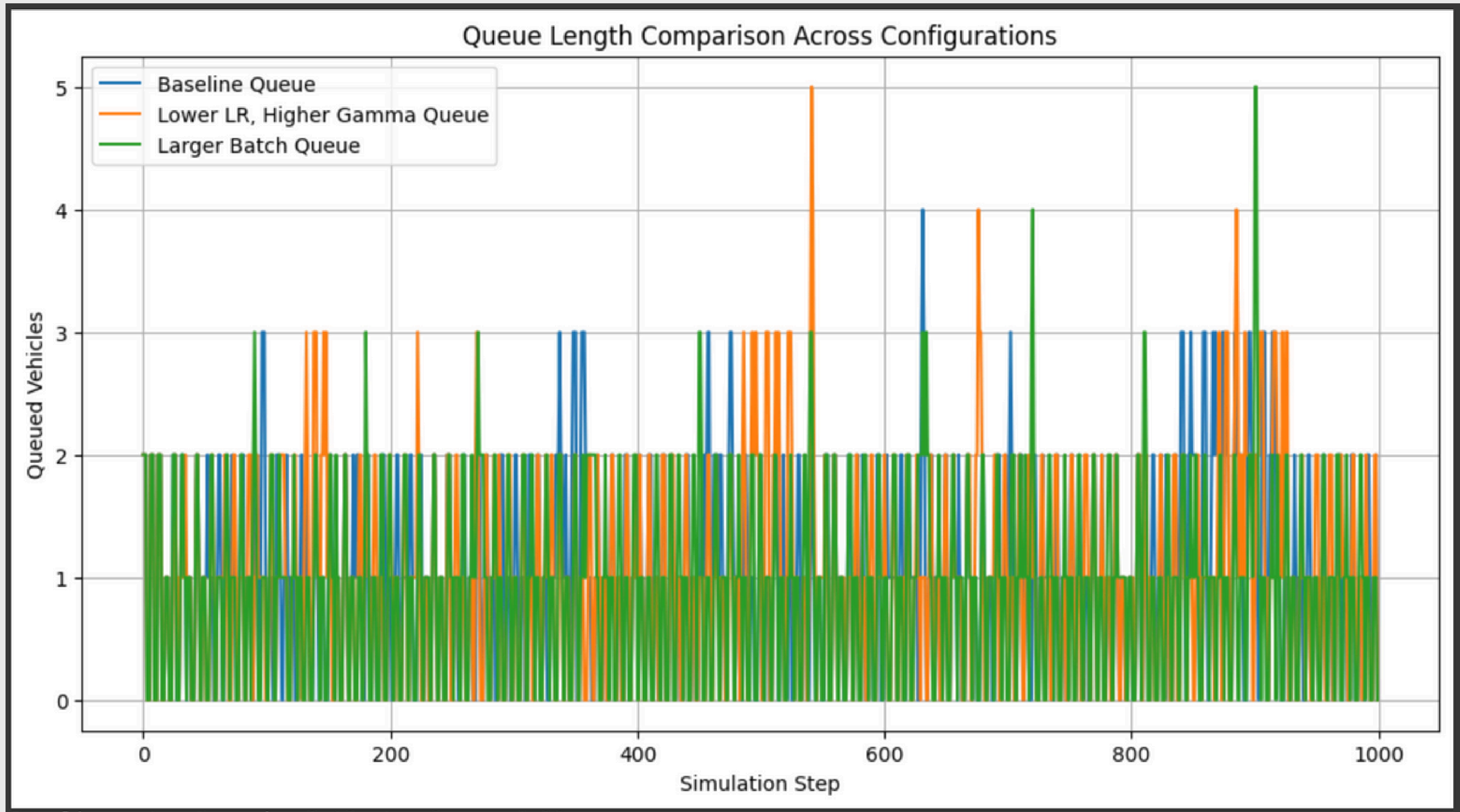
Comparison: Fixed-Time vs DQN

- **Fixed-Time Controller:** Switched signal phase every 20 seconds mechanically.
- **DQN Agent:** Learned to adapt switching dynamically based on live traffic conditions.



Hyperparameter Tuning Results

- **Adjusted:**
  - Learning Rate: 1e-3, 5e-4
  - Gamma: 0.99, 0.995
  - Batch Size: 32, 64
- **Goal:**
  - Improve model stability and generalization.



Baseline → Avg Reward: -2.18, Avg Queue: 0.96
Lower LR, Higher Gamma → Avg Reward: -2.21, Avg Queue: 0.97
Larger Batch → Avg Reward: -2.04, Avg Queue: 0.89

Performance Comparison Table

Config	Avg Reward	Avg Queue
Baseline	-2.18	0.96
Lower LR, Higher Gamma	-2.21	0.97
Larger Batch	**-2.04**	**0.89**

- Larger batch size improved both reward and reduced average queue length.
- Lower learning rate and higher gamma had minimal effect compared to batch size tuning.

Key Findings and Learning

- Larger batch size improved model performance significantly.
- Lower learning rate and higher gamma helped but were less impactful than batch size.
- Realistic datasets (NGSIM, NYC CV) made the simulation environment authentic.
- SUMO-TraCI integration was crucial but sometimes caused connection delays ("Retrying in 1 second").

```
Running config: Lower LR, Higher Gamma
 Retrying in 1 seconds
/usr/local/lib/python3.11/dist-packages/gym/spaces/box.py:128: UserWarning: WARN: Box bound precision lowered by casting to float32
  logger.warn(f"Box bound precision lowered by casting to {self.dtype}")
 Retrying in 1 seconds
 Retrying in 1 seconds
 Retrying in 1 seconds
 Retrying in 1 seconds
```

THANK YOU