AI TRAFFIC MANAGEMENT

Submitted for

Artificial Intelligence Machine Learning CSET301

Submitted by:

(E23CSEU2366) KRISH RAJPUROHIT

(E23CSEU2100) YUVRAJ SINGH

Submitted to

DR. YAJNASENI DASH

Jan-May 2025

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

INDEX

Abstract

In recent years, Urban areas often face significant traffic congestion, especially at intersections where multiple routes converge. Traditional traffic management systems rely on pre-set traffic light timings, which may not adapt well to fluctuating traffic conditions. This can lead to increased waiting times, fuel consumption, and emissions.

Introduction

An AI-based traffic management system can dynamically adjust traffic light timings based on real-time traffic data, improving traffic flow and reducing congestion. Expected Solution: Problem statement is to develop a smart, AI-based traffic management system that can monitor traffic conditions in real-time and adapt traffic light timings accordingly. The system should be capable of handling heavy traffic from multiple directions and optimizing traffic flow to minimize delays and improve overall efficiency.

Related Work

- The project is implemented in Python, leveraging libraries such as NumPy for numerical operations, PyTorch for the neural network, and Pygame for visualization.
- The main classes are:
    - DQNetwork: Defines the neural network architecture.
    - ReplayBuffer: Implements the experience replay mechanism.
    - DQNAgent: Encapsulates the DQN algorithm, including action selection, learning, and model management.
    - Car: Represents individual vehicles in the simulation.
    - TrafficLight: Manages the state of traffic lights at the intersection.

Methodology

1. Environment Simulation:

- **The intersection is modeled as a 4-way junction with traffic coming from North, South, East, and West.**
- **Each direction can have up to 10 cars waiting (max_cars = 10, can be changed as per need).**
- **The maximum waiting time for a car is set to 100 time steps (max_waiting_time = 100).**
- **The simulation uses a dynamic car passing rate:**
    - **Base pass rate: 1 car per time step**
    - **Maximum pass rate: 3 cars per time step**
    - **Acceleration factor: 0.5 (increases pass rate over time)**
- **New cars are added to each queue with a 30% probability per time step (1-3 cars).**

2. DQN Architecture:

- **The Deep Q-Network uses a neural network implemented with PyTorch.**
- **Network structure:**
    - **Input layer: 8 neurons (4 for car counts, 4 for average waiting times in each direction)**
    - **Hidden layer: 24 neurons with ReLU activation**
    - **Output layer: 2 neurons (representing the Q-values for the two possible actions)**
- **The network is trained using Adam optimizer with a learning rate of 0.001.**
- **Loss function: Mean Squared Error (MSE)**

3. State Representation:

- **The state is an 8-dimensional vector:**
    - **First 4 elements: Number of waiting cars in each direction (N, S, E, W)**
    - **Last 4 elements: Average waiting time of cars in each direction**

4. Action Space:

- **The agent can take two possible actions:**
    - **Action 0: Set North-South traffic lights to green (East-West to red)**

        - **Action 1: Set East-West traffic lights to green (North-South to red)**

5. **Reward Function:**
    - **The reward is calculated as: R = 1000 * cars_passed - 0.1 * total_waiting_time**
    - **Additional penalty of -10 * total_waiting cars if the action changes the current traffic light state**
    - **This reward structure encourages the agent to maximize throughput while minimizing waiting times**
6. **Epsilon-Greedy Exploration:**
    - **Initial epsilon: 1.0 (100% random actions)**
    - **Minimum epsilon: 0.01 (1% random actions)**
    - **Epsilon decay rate: 0.995 (multiplied after each replay)**
7. **Experience Replay:**
    - **Replay buffer size: 2000 experiences**
    - **Batch size for learning: 64 experiences**
8. **Training Process:**
    - **The agent is trained for 10 episodes, each consisting of 200 time steps.**
    - **After each action, the agent stores the experience (state, action, reward, next_state, done) in the replay buffer.**

- **The agent performs a learning step (replay) after each action if there are enough samples in the replay buffer.**

9. DQN Update:

- **The agent uses a single network for both selecting and evaluating actions (vanilla DQN).**
- **Q-value update follows the equation: Q(s,a) = Q(s,a) + α * (r + γ * max(Q(s',a')) - Q(s,a))**
- **Where α is the learning rate, γ is the discount factor (set to 0.9), and s' is the next state.**

10. Visualization:

- **The system uses Pygame for real-time visualization of the intersection.**
- **Roads are drawn as gray rectangles with white lane markings.**
- **Cars are represented as colored rectangles with wheels and windows.**
- **Traffic lights are drawn as circles (red or green) at each approach to the intersection.**
- **Car counts for each direction are displayed on the screen.**

11. Testing:

- **After training, the agent is tested on a dataset of 100 randomly generated traffic scenarios.**
- **Each test scenario runs for a single time step, and**

the total reward is calculated.

  - **There's a 2-second delay between scenarios for easier visualization.**

12. **Model Persistence:**

  - **The trained model is saved after each episode using PyTorch's save functionality.**
  - **The saved model includes the neural network's state dict, optimizer state, and current epsilon value.**

Hardware/Software Required

Hardware

- For trying out locally, Any computer with an i5 processor or equivalent.

Software

- Numpy ==1.19.2
- Torch ==1.10.0
- Pygame ==2.1.2

Conclusions

The traffic system tells and helps in maintaining the security and wellness of people across the worldwide. It helps in also reducing crimes and road rage which can be seen in a large number across this city and also the in our country.

Future Scope

1. Incorporating additional features such as fees deduction.
2. Experimenting with deep learning techniques for higher accuracy.
3. Developing a user-friendly interface for wider accessibility.

GitHub Link of our Project:

https://github.com/krishhraj/AI-TRAFFIC-MANAGEMENT