Smart Flow- Advanced Traffic Light Management System

1. Introduction & Problem Statement

Real-World Problem

Urban traffic congestion is a major challenge, leading to increased travel times, fuel consumption, and environmental pollution. Traditional traffic light systems operate on **fixed time intervals**, which are inefficient in handling **dynamic traffic conditions**. This results in unnecessary waiting times and imbalanced traffic flow across different directions.

Importance of an Adaptive System

An **AI-driven traffic light management system** can:

- Dynamically **adjust signal durations** based on real-time traffic density.
- **Reduce waiting time** for vehicles by optimizing green light durations.
- **Improve traffic flow efficiency** by balancing congestion across lanes.
- Adapt to different traffic scenarios such as peak hours and light traffic conditions.

Dataset Overview

To develop the system, a dataset containing real-time traffic data has been used. The dataset consists of:

- **Time of Day:** Timestamp at 60-second intervals.
- **Vehicle Density:** Normalized traffic density for four directions (North, South, East, and West).
- **Current Green Duration:** Duration of the green signal for the active direction.

This data serves as the foundation for training a machine learning model to predict optimal green light durations based on traffic density variations.

---

2. Data Exploration & Preprocessing

Data Structure

The dataset includes the following key features:

- **time_of_day**: Representing the hour and minute of the recorded data.
- **vehicle_density_north, vehicle_density_south, vehicle_density_east, vehicle_density_west**: Continuous values normalized between 0 and 1.
- **current_green_duration**: The duration of the last green light cycle in seconds.

Data Preprocessing

To ensure high-quality input for the machine learning model, the following preprocessing steps were performed:

1. **Feature Engineering:**
   - The `time_of_day` feature was transformed using sine and cosine functions to capture its cyclic nature.
     df['hour_sin'] = np.sin(2 * np.pi * df['hour'] / 24)

     df['hour_cos'] = np.cos(2 * np.pi * df['hour'] / 24)
2. **Normalization:**
   - Vehicle density values were normalized using `MinMaxScaler` to scale them between 0 and 1.

     MinMaxScaler().fit_transform(density_columns)
3. **Target Engineering:**
   - The target variable, `optimal_green_duration`, was calculated as:

     df['target'] = (df[density_cols].sum(axis=1) / 4) * 60

The dataset was then split into **80% training data** and **20% testing data** using a time-aware split to prevent data leakage.

---

3. Model Implementation & Evaluation

Model Selection

Given the sequential nature of traffic data, a **Long Short-Term Memory (LSTM) network** was chosen to model the time-dependent relationships between past and future traffic patterns. The model captures both short-term fluctuations and long-term traffic patterns.

Model Architecture

The LSTM model consists of:

- Input Layer: 7 features (time features, vehicle densities, current green duration)
- Two LSTM layers (64 and 32 units) to capture temporal dependencies.
- Fully connected layers for decision-making.
- Output Layer: A single neuron predicting the next **optimal green duration**.

Training Process

- **Optimizer:** Adam (learning_rate = 0.001)
- **Loss Function:** Mean Squared Error (MSE)
- **Batch Size:** 64 samples
- **Epochs:** 50 (early stopping applied to prevent overfitting)

---

4. Results & Insights

Key Observations

- The model effectively **adjusts green light durations** in response to traffic density variations.

- During peak hours (e.g., 8 AM - 9 AM), the model **allocates longer green times** to high-density directions.
- Late at night, with lower traffic, green durations are **evenly distributed** to prevent unnecessary delays.

Performance Evaluation

- The LSTM-based model was tested against rule-based and fixed-time approaches. Preliminary results indicate **better optimization of waiting times** and **fair distribution** of green durations.
- The model prevents traffic buildup by dynamically allocating **fair waiting times** across all lanes.

---

5. Challenges & Future Improvements

Implementation Challenges

1. **Data Collection Limitations**: Real-world sensor data may have inconsistencies that require additional filtering techniques.
2. **Cold Start Problem**: Initial deployment requires a stabilization period before the model adapts effectively.
3. **Hardware Constraints**: Real-time inference needs to be optimized for low-latency execution (<100ms response time).

Future Enhancements

1. **Reinforcement Learning Integration**:
    - Use Deep Q-Learning to dynamically adjust green light durations based on real-time rewards (e.g., minimizing total waiting time).
2. **Emergency Handling System**:
    - Detect emergency vehicles and dynamically adjust signals to provide priority clearance.
3. **Multi-Intersection Coordination**:
    - Extend the system to synchronize multiple intersections for city-wide traffic flow optimization.
4. **Edge Deployment**:
    - Deploy the model on embedded systems like Raspberry Pi for real-world testing.

---

6. Conclusion

This project demonstrates the feasibility of an **AI-powered traffic light system** capable of dynamically adjusting green light durations based on real-time traffic density. The model effectively reduces waiting times, improves traffic flow, and can be further enhanced through reinforcement learning and multi-intersection coordination. With further improvements, this system can be deployed in smart cities to optimize urban traffic management.
