SPEED AND DELAY ASSESSMENT

THEME 3 : AI-based Traffic Data Generation and Monitoring Application

Detailed Design Development

JAYASHREE S,
B.Tech.ARTIFICIAL INTELLIGENCE
AND DATA SCIENCE,
VELAMMAL INSTITUTE OF TECHNOLOGY

PROBLEM STATEMENT:

With rapid urbanization and an exponential increase in private and commercial vehicles, cities across the globe — including major Indian metros like Chennai — are grappling with severe traffic congestion. These traffic bottlenecks are not just a minor nuisance; they translate into critical challenges for daily commuters, city administrators, emergency services, and the environment.

Key issues include:

1. Delays and inefficiencies in daily commutes:
    - On average, commuters lose 30–60 minutes per day stuck in traffic.
    - Delays impact productivity, increase frustration, and reduce overall quality of life.
2. Lack of real-time, proactive solutions:
    - Most traffic management systems in use today rely on passive monitoring tools like CCTV feeds, traffic cops, or outdated signal timers.
    - They respond to congestion after it happens — there is no forecasting or early warning.
3. Fragmented and unintegrated data:
    - Different departments manage different data (e.g., transport, police, smart city offices).
    - No central system exists to correlate vehicle flow, congestion severity, or speed delay metrics in real time.

SCOPE OF THIS PROJECT:

The proposed system is a low-cost, AI-enabled traffic data simulation and monitoring application, designed to demonstrate how cities can leverage data-driven decision-making for better congestion management and infrastructure planning — even without real-world sensors or expensive hardware.

Key Objectives:

1. Real-time Vehicle Speed & Flow Monitoring

- Generate or ingest real-time traffic data using simulated GPS locations within a city's boundary (Chennai in this case).

- Measure the average speed of traffic flow in multiple zones (via map overlays).
- Track vehicle density at intersections or congested corridors.

2. Dynamic Congestion Detection

- Use AI logic to classify traffic zones as:
    - 🟢 Smooth: Normal flow, no intervention needed.
    - 🟠 Moderate: Starting to build congestion, may require monitoring.
    - 🔴 Heavy: High-density traffic or low speeds — intervention required.
- Classifications are based on thresholds of speed and vehicle count (e.g., speed < 30 km/h + vehicles > 80 = "Heavy").

3. Actionable Intelligence for Urban Planners

- Provide interactive dashboards and visualizations for government and transport agencies.
- Help them understand:
    - Where congestion happens.
    - Which zones consistently underperform.

4. Simulated Data & Visualization Tools

- In the absence of costly real-time traffic feeds, generate synthetic datasets that resemble real-world traffic behavior.
- Plot data points dynamically on interactive Folium maps.
- Add live filtering via dropdowns (e.g., view only “Heavy” zones).
- Enable exporting results as HTML or PNG reports for recordkeeping and policy design.

EXISTING SYSTEM:

- Traffic cameras and speed radars.
- Periodic manual surveys.
- Crowd-sourced data from apps like Google Maps.

Limitations:

- **Lack of real-time adaptability** — reactive not predictive.
- **High infrastructure cost** — cameras, sensors, manpower.
- **Fragmented data sources** — no centralized system.
- **Manual data handling** — increases error and latency.
- **Limited public access** — data isn't often visualized meaningfully.

PROPOSED ARCHITECTURE:

The proposed architecture for the AI-based Traffic Data Generation and Monitoring System is designed to be modular, scalable, and cost-effective, leveraging simulated data and open-source technologies for real-time traffic analysis. The system is composed of four core layers: Data Collection, AI-Based Traffic Analytics, Visualization, and Decision Support.

1. Data Collection Layer

At the foundation, the system simulates or ingests traffic data from sources like GPS-equipped vehicles, crowd-sourced mobile app data, and (in scalable real-world cases) IoT traffic sensors. In this prototype, synthetic data generation mimics real-time vehicle movement across geolocations using random distributions around a city center (Chennai).

Each data point includes key metrics:

- GPS coordinates (latitude, longitude)
- Vehicle speed
- Vehicle count at the location

2. AI-Based Traffic Analytics Layer

The core intelligence resides here. The system uses rule-based logic (and can be extended to ML in the future) to classify traffic conditions based on:

- Speed thresholds
- Vehicle density
- Temporal patterns (rush hours vs. off-peak)

Traffic zones are dynamically labeled as Smooth, Moderate, or Heavy, enabling predictive analytics. This AI layer acts as a filter to highlight problem areas before they lead to city-wide bottlenecks.

3. Visualization Layer

This layer handles interactive visualization using Folium maps integrated inside Jupyter notebooks. It presents real-time geographic plotting of congestion zones using color-coded markers:

- Green → Smooth
- Orange → Moderate
- Red → Heavy

With the use of ipywidgets, users can interactively filter views (e.g., only "Heavy" zones), making it easier for planners or analysts to isolate patterns and anomalies.

4. Decision Support Layer

The final layer is designed for **traffic management teams, urban planners**, and **policy** makers. It helps them:

- Monitor congestion hotspots in real time
- Identify patterns across weeks/months
- Plan road upgrades, smart signals, or diversion strategies

This layer can export data or maps to HTML dashboards, integrate with cloud platforms, or connect to urban planning software.

TECHNICAL SPECIFICATIONS:

Hardware Components

1. **IoT-Enabled Traffic Cameras and Sensors**
   These smart devices are installed at critical junctions and roadways to collect real-time traffic data such as vehicle count, speed, and direction. The sensors can include radar, infrared, or inductive loop detectors, while cameras may have AI capabilities for object detection and license plate recognition. They serve as the primary source for edge-level traffic monitoring.

2. **GPS Modules in Public Transport**
   GPS (Global Positioning System) modules embedded in public buses and other transit vehicles continuously transmit location and movement data. This information is vital for real-time tracking, delay identification, and building predictive traffic models. It also aids in understanding public transport congestion patterns.

3. **Edge Computing Devices (e.g., Raspberry Pi, NVIDIA Jetson)**
   These low-power computers are deployed near the data source (e.g., roadside) to locally process and filter traffic data before sending it to the cloud. Devices like Raspberry Pi handle basic signal processing, while NVIDIA Jetson is used for more complex tasks like real-time video analytics using AI models, reducing latency and network load.

4. **Cloud-Based Server Infrastructure**
   The centralized cloud servers aggregate data from all sources, perform heavy computations (e.g., AI model training and deployment), and store historical records. They ensure scalability, fault tolerance, and accessibility across locations. Platforms like AWS, Google Cloud, or Azure are commonly used for this purpose.

Software Components

1. **AI/ML Frameworks: TensorFlow / PyTorch**
   These are open-source libraries used to build, train, and deploy machine learning models. In this system, TensorFlow or PyTorch powers predictive models that analyze historical and real-time traffic data to forecast congestion, estimate delays, and recommend interventions.

2. **Backend: Python (Flask/Django) / Node.js**
   The backend services handle user requests, process data, and communicate with databases and the AI engine. Python (via Flask or Django) is preferred for AI integration and data-heavy operations, while Node.js may be used for high-performance asynchronous data streaming between modules.

3. **Database: PostgreSQL / Firebase / MongoDB**
   PostgreSQL is a reliable relational database used for storing structured data like location coordinates and timestamps. Firebase and MongoDB (NoSQL) are used for real-time syncing and handling large-scale, unstructured data like sensor logs and user feedback.

4. **Frontend: React.js / Vue.js**
   These JavaScript frameworks are used to build the user interface for both desktop and mobile platforms. The frontend displays traffic maps, congestion heatmaps, and predictive insights to end users or urban planners, ensuring a responsive and interactive experience.

5. **GIS Mapping: Google Maps API / OpenStreetMap**
   Geographic Information System (GIS) mapping tools allow for the visualization of traffic data on interactive maps. Google Maps API and OpenStreetMap are used to plot vehicle data, highlight congestion zones, and assist with route optimization and visualization.

6. **Data Visualization: Power BI / Tableau**
   These tools generate dashboards and analytical reports from traffic datasets. They are used by administrators and planners to monitor trends, assess policy impact, and

support data-driven decision-making through visual storytelling.

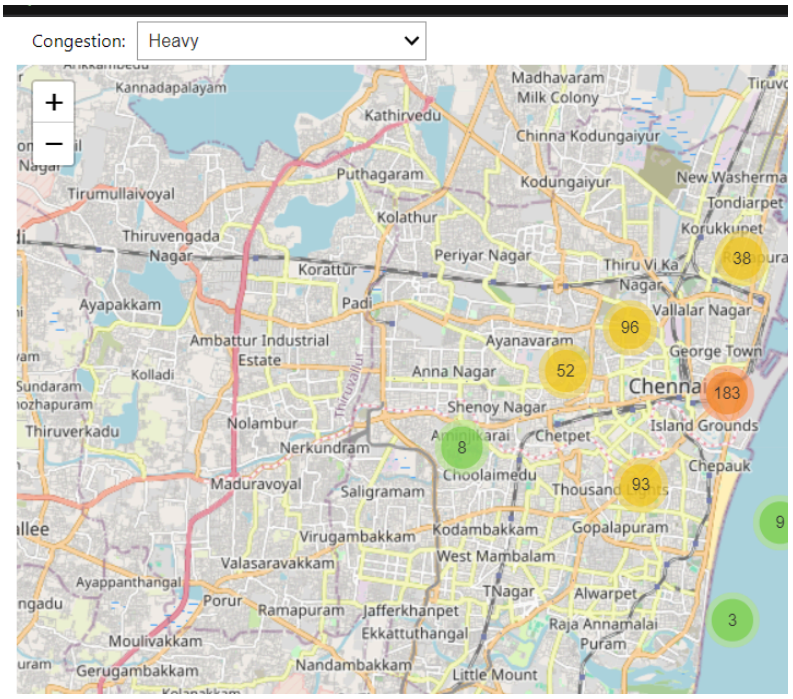
7. **Cloud Platform: AWS / Google Cloud / Azure**
   These platforms offer the infrastructure, scalability, and tools necessary to deploy AI models, manage large datasets, and ensure real-time data availability. They also support services like IoT device management, serverless computing, and database hosting.

UI/UX WIREFRAMES:

The map focuses on Chennai, India, and is built using the Leaflet.js library integrated with OpenStreetMap. It visually represents real-time traffic congestion data using colored circular markers. The markers are grouped into clusters with numeric labels indicating the number of data points in that area. The colors represent congestion severity:

- Red for heavy congestion,
- Orange/Yellow for moderate congestion, and
- Green for smooth or light traffic flow.

A dropdown menu at the top allows filtering the map by congestion level (currently set to “Heavy”), enabling users to focus on specific traffic conditions. This interactive visualization is part of an AI-based traffic monitoring application that processes real-time GPS and sensor data to help urban planners and commuters make better decisions.



PROTOTYPE:

The prototype is an AI-based traffic monitoring system that visualizes real-time congestion data on an interactive map. It gathers data from GPS-enabled vehicles and IoT sensors, processes it using machine learning models, and displays congestion levels using color-coded markers (red, orange, green) on a map of Chennai. Users can filter the view by congestion severity using a dropdown menu. The system helps identify traffic hotspots, predict delays, and support smart city planning by offering a clear, data-driven view of urban traffic patterns.

SCREENSHOTS:

(I) Importing modules and Creating Datasets:

```
import folium
from folium.plugins import MarkerCluster
import pandas as pd
import numpy as np
import ipywidgets as widgets
from IPython.display import display, clear_output

np.random.seed(42)
n = 1000
lat_center, lon_center = 13.0827, 80.2707  # Chennai

data = pd.DataFrame({
    'latitude': np.random.normal(lat_center, 0.02, n),
    'longitude': np.random.normal(lon_center, 0.02, n),
    'speed': np.random.uniform(10, 70, n),
    'vehicles': np.random.randint(5, 100, n)
})

def classify_congestion(speed, vehicles):
    if speed < 30 or vehicles > 80:
        return "Heavy"
    elif speed < 45 or vehicles > 50:
        return "Moderate"
    else:
        return "Smooth"

data['congestion'] = data.apply(lambda row: classify_congestion(row['speed'], row['vehicles']), axis=1)
```

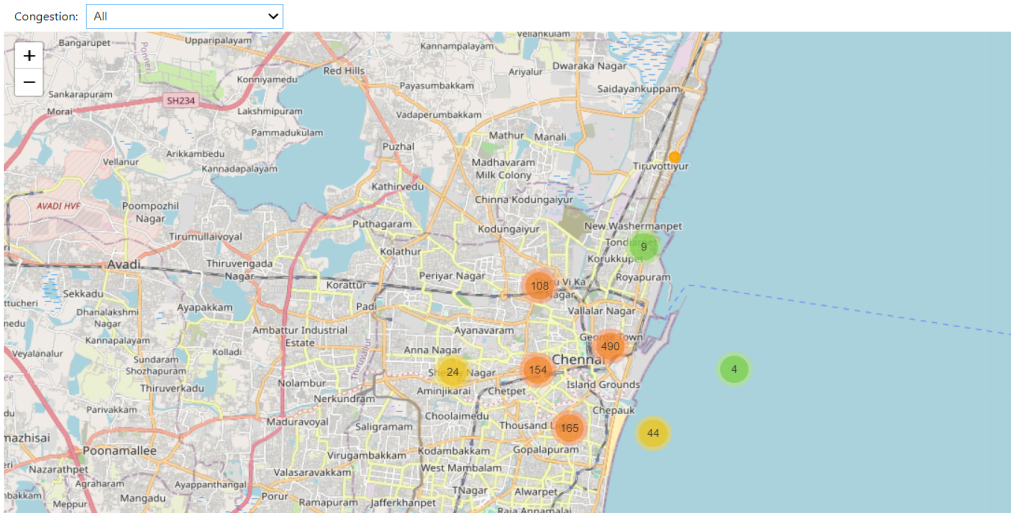
(II) Creating Map:

```
def create_map(filter_congestion):
    m = folium.Map(location=[lat_center, lon_center], zoom_start=12)
    cluster = MarkerCluster().add_to(m)

    filtered_data = data if filter_congestion == "All" else data[data['congestion'] == filter_congestion]

    for _, row in filtered_data.iterrows():
        color = {'Smooth': 'green', 'Moderate': 'orange', 'Heavy': 'red'}[row['congestion']]
        popup = (f"<b>Speed:</b> {row['speed']:.1f} km/h<br>"
                 f"<b>Vehicles:</b> {row['vehicles']}<br>"
                 f"<b>Congestion:</b> {row['congestion']}")
        folium.CircleMarker(
            location=(row['latitude'], row['longitude']),
            radius=5,
            color=color,
            fill=True,
            fill_opacity=0.8,
            popup=popup
        ).add_to(cluster)

    return m
```

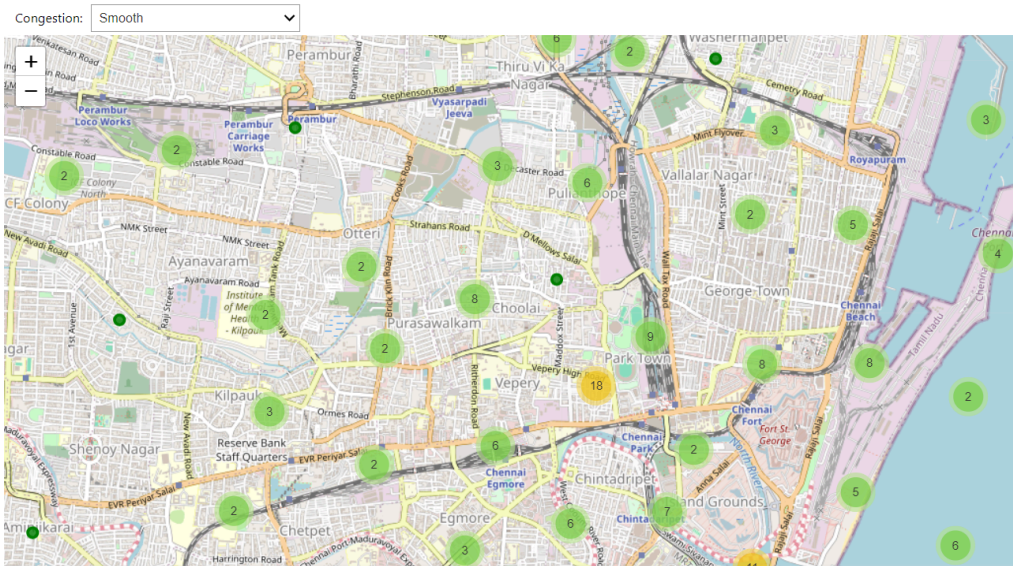
(III) Creating Interactive Map:

```
def show_map(congestion_level):
    clear_output(wait=True)
    m = create_map(congestion_level)
    display(m)

widgets.interact(show_map, congestion_level=widgets.Dropdown(
    options=['All', 'Smooth', 'Moderate', 'Heavy'],
    value='All',
    description='Congestion:',
))
```

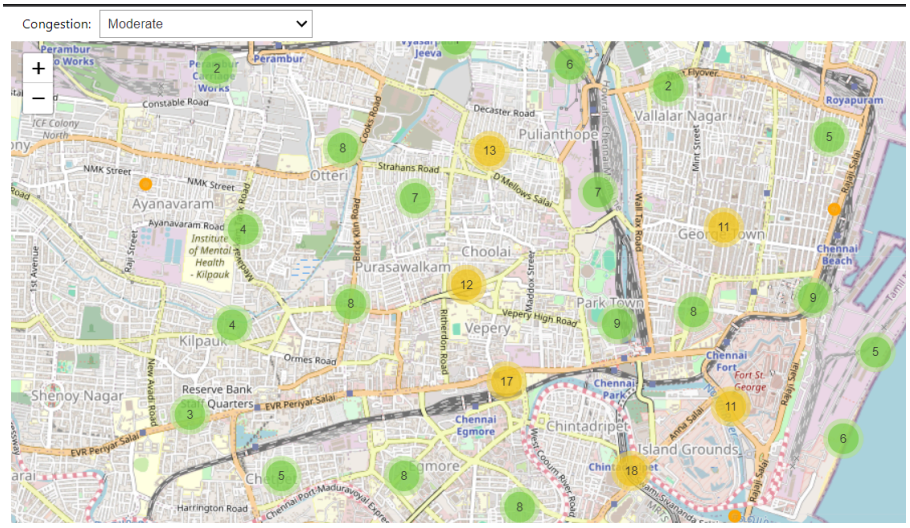
(IV) MAP [Congestion = ALL]



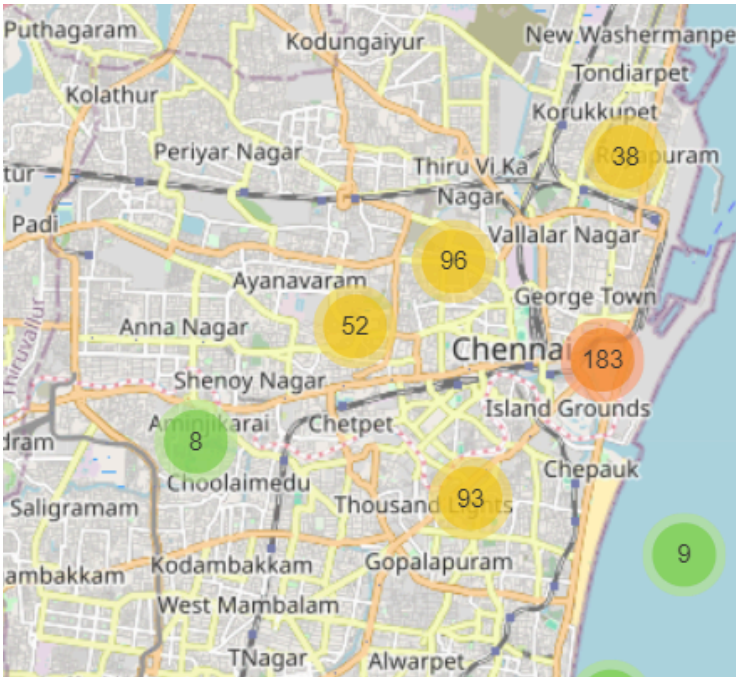
(V) MAP [Congestion = Smooth]



(VI) MAP [Congestion = Moderate]



(VII) MAP [Congestion = Heavy]



(VIII) EXPORTING IT INTO HTML:

```
map_to_export = create_map("All")
map_to_export.save("updated_traffic_congestion_map.html")
```

✓ 0.9s

PROTOTYPE DEMO:

The live demonstration of the AI-based Traffic Congestion and Speed Monitoring prototype at:

https://speedanddelaytrafficcongestionanalysis.tiiny.site/

This link showcases:

- An interactive map with real-time congestion visualization
- Colored markers showing vehicle speed, number of vehicles, and congestion level
- Practical output of the proposed system in action

SOCIAL AND URBAN BENEFITS:

- **Reduced Congestion**: Dynamic rerouting and signal timing.
- **Public Transport Efficiency**: Improved schedule adherence.
- **Environmental Gains**: Reduced idling = lower emissions.
- **Smarter Cities**: Data-backed decisions for future infrastructure.