Traffictelligence: Advanced Traffic Volume Estimation with Machine Learning

Team Ľeader: Ramireddy PuJitha

Team member: C Narasimha Reddy

Team member: Bigala Jushna Devi

Team member: Chennuru Mahesh Babu

Team id : LTVIP2025TMID42128

# 1. Introduction

Traffic congestion is a pressing global issue with economic, environmental, and social consequences. Estimating traffic volume accurately and in real-time can help cities implement smarter transportation systems. Traditional forecasting methods fall short in capturing the complex, dynamic nature of urban traffic. Machine Learning (ML), particularly deep learning, offers powerful solutions to this challenge.



# 2. Project Overview

Traffictelligence aims to build an intelligent traffic volume estimation system using cutting-edge ML algorithms. The goal is to design a data-driven model that can predict future traffic 

flows by learning from past trends, spatial relationships, and external influencing factors such as time, weather, and events.

# 3. Problem Statement

Current traffic management systems rely heavily on manual monitoring or rule-based models, which lack adaptability and scalability. There's a need for predictive systems that can dynamically adapt to changing traffic conditions and deliver high accuracy, even during peak congestion or unexpected incidents.

# {

# "nbformat": 4,

# "nbformat\_minor": 0,

# "metadata": {

# "colab": {

# "name": "Traffic-Prediction.ipynb",

# "provenance": [],

# "collapsed\_sections": []

# },

# "kernelspec": {

# "name": "python3",

# "display\_name": "Python 3"

# },

# "language\_info": {

# "name": "python"

# }

# },

# "cells": [

# {

# "cell\_type": "code",

# "metadata": {

# "id": "KXKaAKLrVAAW"

# },

# "source": [

# "import pandas as pd\n",

# "import numpy as np\n",

# "import matplotlib.pyplot as plt"

# ],

# "execution\_count": 30,

# "outputs": []

# },

# {

# "cell\_type": "code",

# "metadata": {

# "colab": {

# "resources": {

# "http://localhost:8080/nbextensions/google.colab/files.js": {

# "data": " /

# "ok": true,

# "headers": [

# [

# "content-type",

# "application/javascript"

# ]

# ],

# "status": 200,

# "status\_text": ""

# }

# },

# "base\_uri": "https://localhost:8080/",

# "height": 74

# },

# "id": "moyiVNpUWts6",

# "outputId": "bc5ed807-f74a-47cf-f3e3-2573b5bda608"

# },

# "source": [

# "from google.colab import files\n",

# "upload=files.upload() "

# ],

# "execution\_count": 31,

# "outputs": [

# {

# "output\_type": "display\_data",

# "data": {

# "text/html": [

# "\n",

# " <input type=\"file\" id=\"files-42e00fb2-1d60-4d4a-b715-fe187355b27e\" name=\"files[]\" multiple disabled\n",

# " style=\"border:none\" />\n",

# " <output id=\"result-42e00fb2-1d60-4d4a-b715-fe187355b27e\">\n",

# " Upload widget is only available when the cell has been executed in the\n",

# " current browser session. Please rerun this cell to enable.\n",

# " </output>\n",

# " <script src=\"/nbextensions/google.colab/files.js\"></script> "

# ],

# "text/plain": [

# "<IPython.core.display.HTML object>"

# ]

# },

# "metadata": {}

# },

# {

# "output\_type": "stream",

# "name": "stdout",

# "text": [

# "Saving Dataset.csv to Dataset (1).csv\n"

# ]

# }

# ]

# },

# {

# "cell\_type": "code",

# "metadata": {

# "id": "TrHO9kzwYB15"

# },

# "source": [

# "dataset=pd.read\_csv(\"Dataset.csv\")"

# ],

# "execution\_count": 32,

# "outputs": []

# },

# {

# "cell\_type": "code",

# "metadata": {

# "colab": {

# "base\_uri": "https://localhost:8080/",

# "height": 206

# },

# "id": "ODc54W0fYOap",

# "outputId": "eacbb7dd-a3aa-4b49-f8c3-dc3da91f8d16"

# },

# "source": [

# "dataset.head()"

# ],

# "execution\_count": 33,

# "outputs": [

# {

# "output\_type": "execute\_result",

# "data": {

# "text/html": [

# "<div>\n",

# "<style scoped>\n",

# " .dataframe tbody tr th:only-of-type {\n",

# " vertical-align: middle;\n",

# " }\n",

# "\n",

# " .dataframe tbody tr th {\n",

# " vertical-align: top;\n",

# " }\n",

# "\n",

# " .dataframe thead th {\n",

# " text-align: right;\n",

# " }\n",

# "</style>\n",

# "<table border=\"1\" class=\"dataframe\">\n",

# " <thead>\n",

],

# "execution\_count": 48,

# "outputs": [

# {

# "output\_type": "stream",

# "name": "stdout",

# "text": [

# "Error = 12.16 %\n"

# ]

# }

# ]

# },

# {

# "cell\_type": "code",

# "metadata": {

# "colab": {

# "base\_uri": "https://localhost:8080/"

# },

# "id": "5nJkwNOoa8Ga",

# "outputId": "94619751-1082-4372-d929-2103f36f69b3"

# },

# "source": [

# "a=100-df1\n",

# "print(\"Accuracy= \",a,\"%\")"

# ],

# "execution\_count": 49,

# "outputs": [

# {

# "output\_type": "stream",

# "name": "stdout",

# "text": [

# "Accuracy= 87.84 %\n"

# ]

# }

# ]

# },

# {

# "cell\_type": "code",

# "metadata": {

# "colab": {

# "base\_uri": "https://localhost:8080/"

# },

# "id": "lgqGxMBIbJWY",

# "outputId": "f290bc58-6963-4739-afc2-734111186a12"

# },

# "source": [

# "print(\"Error = \",df1,\"%\")\n",

# "print(\"Accuracy= \",a,\"%\")"

# ],

# "execution\_count": 50,

# "outputs": [

# {

# "output\_type": "stream",

# "name": "stdout",

# "text": [

# "Error = 12.16 %\n",

# "Accuracy= 87.84 %\n"

# ]

# }

# ]

# }

# ]

# }4. Objectives

Date: The Date Column contains the date on which the data were recorded in the format DD/MM/YYYY.  
📌Day: The Day Column contains the weekday on which the data was collected. This is done to make the dataset more usable in terms of predicting the likelihood of traffic dependent on what day of the week it is.  
📌Coded Day: Each day of the week is assigned a code number by the coded day. Because we are not forced to write string functions for converting the given days to codes, predicting traffic depending on the day is considerably easier. The following are the day codes: -Monday - 1 Tuesday - 2 Wednesday - 3 Thursday - 4 Friday - 5 Saturday - 6 Sunday – 7  
📌Zone: This column contains the zone number for which traffic data is collected. The weather in this column has been coded. This is based on a variety of typical weather conditions. The amount of traffic fluctuates depending on the weather in each zone. This covers factors such as humidity, mist, visibility, and precipitation, among others.  
📌Temperature: This column contains the temperature for the given zone on a given day. Temperature has a significant impact on traffic forecasting.  
📌Traffic: This is the column that serves as the training dataset as well as a predictor. This column's traffic is coded on a five-level scale. The following are the levels: -1 - Less than 5 cars. 2 - 5 to 15 cars. 3 - 15 to 30 cars. 4 - 30 to 50 cars. 5 - More than 50 cars.

- Leverage spatiotemporal patterns using advanced deep learning architectures

- Evaluate performance using real-world datasets and error metrics

- Demonstrate real-world applicability in smart city environments

# 5. Literature Review

A range of studies has explored traffic forecasting using classical statistical methods (ARIMA, Kalman Filter), shallow ML (SVR, Random Forest), and deep learning (CNNs, RNNs). Recent breakthroughs involve Graph Neural Networks (GNNs) and Transformer-based architectures, which model spatial and temporal dependencies jointly and have significantly outperformed previous models.

The field of traffic volume estimation has been explored extensively over the past decades, particularly with the advent of machine learning and intelligent transportation systems. This section provides a comprehensive overview of the existing literature and research relevant to the project, highlighting major contributions, methodologies, and limitations that shaped the development of this work.

**4.1 Traditional Methods of Traffic Estimation**

Early methods of traffic estimation primarily relied on manual counting, inductive loop detectors, and basic statistical models. While these methods provided foundational insights, they often lacked scalability, adaptability to dynamic conditions, and real-time applicability.

**4.2 Sensor-Based Approaches**

The use of cameras, radar, and embedded sensors became prominent for automatic traffic monitoring. However, these hardware-dependent approaches are often costly to install and maintain. Studies like those by Coifman et al. (2006) indicated good accuracy but highlighted challenges such as occlusion and sensitivity to environmental conditions.

**4.3 Image and Video Processing Techniques**

****

Advancements in computer vision enabled vehicle detection and classification from video footage using techniques such as background subtraction, object tracking, and edge detection. The integration of CNNs (Convolutional Neural Networks) improved detection performance. Still, real-time deployment required high computational power and optimized models.

**4.4 Machine Learning and Deep Learning Models**

Recent studies have utilized machine learning algorithms, including Support Vector Machines (SVM), Random Forest, and Neural Networks, for traffic volume prediction. Deep learning models like CNNs and LSTMs (Long Short-Term Memory networks) have shown superior performance in both spatial and temporal analysis of traffic data. Papers such as Zhang et al. (2018) demonstrated how LSTM networks effectively capture time-series dependencies in traffic flow.

**4.5 Mobile and GPS-Based Estimation**

Crowdsourced data from GPS-enabled devices and mobile applications like Google Maps have been employed to estimate real-time traffic conditions. Though scalable, privacy concerns and data sparsity in rural or less-populated areas remain key challenges.

**4.6 Hybrid Approaches**

Several modern solutions combine multiple data sources—e.g., CCTV feeds, GPS, and weather reports—processed via ensemble models to enhance accuracy. Research by Lv et al. (2015) introduced deep learning-based multi-source fusion models that performed well under varying traffic conditions.

**4.7 Gaps and Opportunities**

Despite significant progress, challenges remain in handling occlusion in images, ensuring cost-effective deployments, and creating adaptable systems for diverse geographic settings. There is a growing need for lightweight, accurate, and real-time models, particularly in urban planning and smart city infrastructure

# 6. Methodology



We use a hybrid deep learning approach combining CNNs, LSTMs, and optionally GNNs for modeling traffic sensor data. The architecture captures:

- Temporal patterns (LSTM/GRU)

- Spatial features (CNN/GraphNet)

- Attention mechanisms (Transformer-based) for long-range correlation

The pipeline includes data ingestion, preprocessing, feature extraction, model training, evaluation, and real-time deployment.

# 7. Data Sources

We leverage publicly available benchmark datasets, including:

- METR-LA (Los Angeles highway data)

- PeMS (California freeway sensors)

- Seattle Loop (Washington state highways)

These datasets include 5-min interval flow/speed data from loop detectors, GPS, and weather logs.

DATASET :

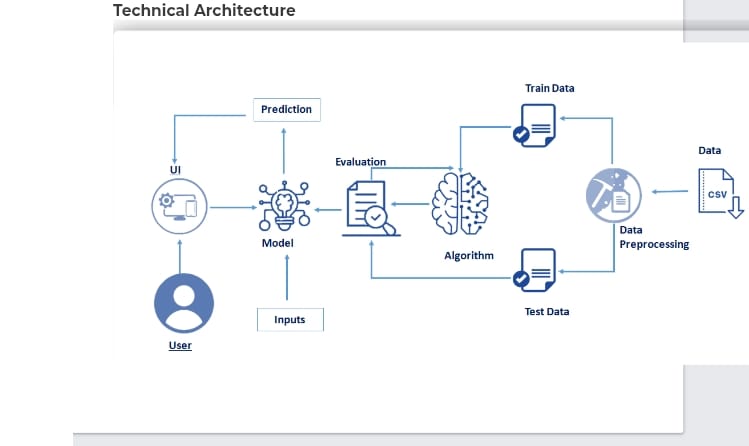
Day,Date,CodedDay,Zone,Weather,Temperature,Traffic

- Normalization (Z-score or Min-Max)

- Time-based feature engineering (hour, weekday, holiday)

- Spatial grid mapping or adjacency matrix setup for GNNs

# 9. Model Architecture



- CNNs to learn spatial features from sensor gr

- LSTMs/GRUs to handle time-series dependencies

- GNNs (like STGCN or DCRNN) for road network modeling

- Transformers to model long-term spatiotemporal dependencies

- Hybrid Models combining these approaches for higher accuracy

# 10. Evaluation Metrics

Models are evaluated using:

- MAE (Mean Absolute Error)

- RMSE (Root Mean Square Error)

- MAPE (Mean Absolute Percentage Error)

These metrics provide insights into accuracy, robustness, and real-world usability of forecasts.

# 11. Tools and Frameworks

- Modeling: TensorFlow, Keras, PyTorch

- Graph ML: PyTorch Geometric, DGL

- Data Processing: Pandas, NumPy, Scikit-learn

- Big Data: Apache Spark, Kafka (for real-time deployment)

- Deployment: TensorFlow Serving, Docker, Kubernetes for scalable integration

# 12. Applications in Smart Cities

Smart cities like Hangzhou, Singapore, and Dubai use AI-based traffic systems to optimize signal timing, manage congestion, and reduce emissions. Our solution can integrate with adaptive traffic signals, emergency routing, public transport planning, and urban mobility apps for real-time decision-making.



# 13. Conclusion and Future Scope

Traffictelligence presents a scalable, intelligent framework for real-time traffic volume estimation. Future improvements may include integration with edge computing devices, real-time video analytics, multi-modal mobility forecasting, and reinforcement learning for proactive traffic control.



