Traffic Forecasting Using Graph Neural Network and LSTM

Vaishnavi Pawar - A61
Computer Science
KIT,Kolhapur
Email:vaishnaviipawarr@gmail.com

Bharti Pawar-A59
Computer Science
KIT,Kolhapur
Email: bhartipawar277@gmail.com

Shreya Jadhav-A26
Computer Science
KIT,Kolhapur
Email: shreyajadhav3112@gmail.com

***Abstract*—For urban planning and transportation management, traffic forecasting is essential. In order to improve the accuracy of traffic prediction, this study blends Graphic Neural Networks (GNNs) and Long Short-Term Memory (LSTM) networks. While LSTMs model the temporal patterns in traffic data, GNNs capture the intricate spatial dependencies inside road networks. By combining these models, a more complete and precise forecasting method is provided for the prediction of traffic flow and congestion.**

Using LSTM neural networks,it illustrates time series forecasting for traffic data. The data is loaded and preprocessed, training sequences are generated, LSTM models are defined, and they are trained using various data splitting techniques. The models are assessed, forecasts are made using them, and the outcomes are displayed and contrasted. Future research may include enhanced loss functions, ensemble models, online learning, hyperparameter tuning, and feature engineering. There is also opportunity for real-time deployment, multivariate forecasting, and data quality enhancement. The project's usefulness for traffic management and urban planning may be improved by localized models, environmental impact analysis, and user-friendly interfaces.

I. Introduction to Progress Report

We have built a solid foundation using Long Short-Term Memory (LSTM) neural networks for time series forecasting throughout this research, with a focus on predicting traffic patterns in particular. We started our adventure with data pretreatment, where we discovered how crucial operations like data loading and formatting are. Then, after learning the subtleties of model architecture and layer specifications, we set out to build LSTM models specifically designed for time series forecasting. We then divided the data, trained the models, and assessed their performance using measures like Root Mean Square Error (RMSE). Our skill set was enhanced by the capacity to produce both single-step and multi-step forecasts. We also explored data visualization because we understood how important it was for understanding and presenting models. We've identified future directions as we get to a conclusion.

A. GitHub Basics

GitHub is a platform used for version control and live sharing of resources related to project development. It helped us to track changes in our code and work on projects with teammates efficiently. We created a public repository to store our code and shared with team members.

Creating a Repository: A GitHub repository is a fundamental place which is the central governing body where we can manipulate the project files. It's like a box that holds all the resources for a specific project. We created a repository named Traffic.

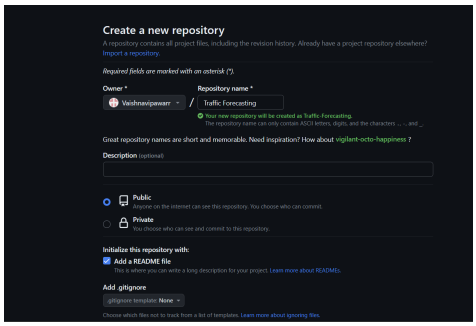

Fig. 1. Creating repository

Creating a Branch: GitHub provides branching to work developers on their projects simultaneously. It allows developers to create their own branch through which they can add their own code without disturbing the previous version of the code. We used branching when we wanted to make different changes in our code without disturbing the earlier version of our project code. We created a branch in GitHub repository and then after completing one phase of our code we switched our existing branch with the original branch using the checkout command. Lastly pushed the required branch to GitHub.

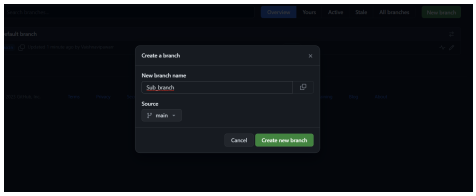

Fig. 2. Branching

Making and Committing Changes: In the local copy of the code on our machine we made required changes according to how our project proceeded. We modified or created new files according to our needs. With the help of the 'git add' command we staged our files for committing. When all the changes were staged, with the help of 'git commit' command we committed

the required changes to the repository. Whenever required we used commit messages to highlight the changes we made.

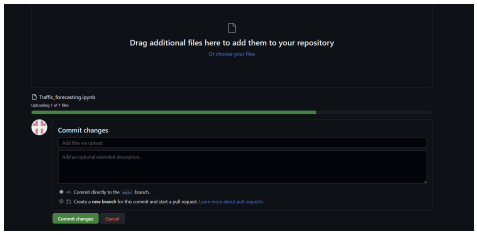

Fig. 3. Pushing Repository and comitting changes

Opening a Pull Request: A pull request in GitHub signifies that the changes made by the developer can be merged with the original source code files to achieve the betterment of the project. After committing our changes and pushing the specific branch, we created a pull request on our GitHub repository for the recently created branch. In the pull request branch we described the necessary changes that were needed to be made in the final project files with a proper suitable title.

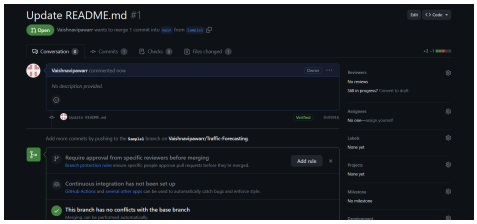

Fig. 4. Pull request

Merging Your Pull Request: As a project manager or administrator, the pull requests can be viewed in the pull requests tab of the respective repository. If the changes made by the developers are essential then the pull request can be merged into the main branch of the repository. The previous files and new changes in the files are visible to the administrator of the repository. We analysed the changes and discussed them with team members and merged them if they are necessary.

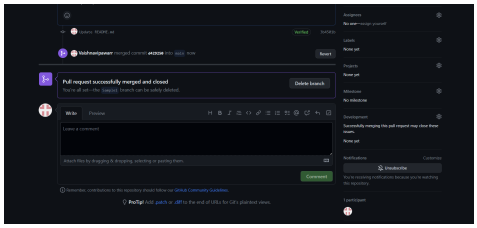

Fig. 5. Merging pull request

Familiarizing Yourself with Key Terms: Following are some key terms that we learnt during the internship:

- **GitBash** : GitBash is a command line interface that can be used to run and execute git commands on the Windows operating system. Using the Unix-like shell environment developers can interact with the git and perform different tasks.
- **Fork** : Forking in GitHub allows users to create the identical copy of the repository under the GitHub account where users can make changes in the source code files. It enables collaborative development. It allows users to enhance the features of the project.
- **Cloning** : Cloning the repository in GitHub refers to creating the local copy in the user's own personal computer. It can be only used for personal use of the user. It doesn't change the original repository contents as the user doesn't have the access to the original repository.

B. Deep learning Problem Selected

The project's chosen deep learning challenge is time series forecasting, with a special emphasis on predicting traffic patterns. In order to solve important urban transportation problems, it is intended to use Long Short-Term Memory (LSTM) neural networks to assess historical traffic data, model patterns, and generate precise predictions for future traffic flow and congestion.

1) Methodology: Time series forecasting with a focus on predicting traffic patterns is the chosen deep learning issue. Long Short-Term Memory (LSTM) neural networks are used in the selected methodology to model and forecast temporal relationships. Through the use of data preprocessing, sequence generation, model training, evaluation, and visualization, this method produces precise traffic forecasts and insights that may be used for traffic management and planning.

2) Results and Discussion: Time series forecasting, with a focus on predicting traffic patterns using Long Short-Term Memory (LSTM) neural networks, was the deep learning topic chosen for this study. The project's outcomes include a successful training of LSTM models to anticipate traffic data, an assessment of model performance, and the identification of areas requiring further research. The focus of the debate is on prospective enhancements and approaches for raising the precision and usefulness of traffic forecasting systems, hence boosting their value for traffic management and urban planning.

II. Final Report Content

Throughout this project, we have established a strong base employing Long Short-Term Memory (LSTM) neural networks for time series forecasting, with a focus on foreseeing traffic patterns in particular. Our journey began with data pretreatment, during which we learned how important processes like data loading and formatting are. We then set out to create LSTM models specifically made for time series forecasting after learning the nuances of model architecture and layer specifications. The models were then trained, the data was divided, and their performance was evaluated using metrics like Root Mean Square Error (RMSE). The ability to create single-step and multi-step forecasts has improved our skill set. Since we were aware of how crucial data visualization was for comprehending and presenting models, we also looked into it. We've

III. Introduction

- **Q1.** What problem are we trying to solve?
  Traffic forecasting using Graph Neural Networks (GNNs) and Long Short-Term Memory (LSTM) networks aims to address the persistent challenges associated with urban traffic management and transportation logistics. The primary problem it seeks to solve is the prediction of future traffic conditions accurately. This includes forecasting traffic congestion, travel times, road occupancy, and other vital parameters within a city's transportation network.
  By leveraging GNNs, this approach seeks to capture complex spatial dependencies within the traffic network. It accounts for how different road segments, intersections, and nodes interact, enabling more precise predictions of traffic flow and congestion patterns. LSTM networks, on the other hand, tackle the temporal dimension by modeling how traffic conditions change over time, accounting for daily and seasonal variations.
  Ultimately, the goal is to provide actionable insights for urban planners, transportation authorities, and commuters. Accurate traffic forecasting helps optimize traffic flow, reduce congestion, enhance safety, and improve resource allocation in a city's transportation system, contributing to more efficient and sustainable urban mobility.
- **Q2.** Why is this an Important Problem?
  Traffic forecasting is of paramount importance due to its profound impact on various aspects of modern society. Here are several key reasons why traffic forecasting is a crucial problem to address:
  1. Congestion Mitigation: Accurate traffic predictions enable authorities to proactively manage and alleviate congestion, reducing travel times and enhancing the overall quality of life for commuters.
  2. Safety Improvement: Forecasting helps identify potential traffic hotspots and areas prone to accidents, allowing for targeted safety measures and reduced accident rates.
  3. Resource Optimization: Efficient traffic forecasting aids in the optimal allocation of transportation resources, such as buses, trains, and traffic signal timing, leading to cost savings and reduced environmental impact.
  4. Urban Planning: City planners rely on traffic forecasts to design transportation infrastructure, make informed land-use decisions, and promote sustainable urban development.
  5. Economic Impact: Traffic congestion costs economies billions of dollars annually in lost productivity. Forecasting helps minimize these economic losses by improving traffic management and logistics.
  6. Environmental Benefits: Reduced congestion and efficient traffic flow lead to lower emissions, contributing to environmental sustainability and cleaner air quality.
  In summary, traffic forecasting plays a pivotal role in enhancing transportation efficiency, safety, and sustainability, making it a vital problem to address for the well-being of communities and economies.
- **Q3.** How is this a challenging Problem?
  Traffic forecasting using graph neural networks (GNNs) and Long Short-Term Memory (LSTM) networks is a challenging problem due to several factors:
  Complex Spatial Dependencies: Traffic data typically exhibits complex spatial dependencies, with road segments, intersections, and regions influencing each other's traffic conditions. GNNs are designed to capture these dependencies by modeling the graph structure of the road network, but defining an accurate and efficient graph representation can be challenging.
  Temporal Dynamics: Traffic conditions change over time due to various factors such as rush hours, accidents, weather conditions, and special events. LSTMs are good at modeling temporal dynamics, but incorporating them into a GNN-LSTM hybrid model while ensuring efficient training and prediction is non-trivial.
  Data Sparsity and Missing Values: Traffic data can be sparse and often contain missing values, which can make it difficult to train accurate models. Dealing with missing data and imputing values appropriately is a challenge in traffic forecasting.
  Scalability: Real-world road networks can be vast, containing thousands of road segments and intersections. Designing a scalable GNN-LSTM model that can handle large-scale networks while maintaining prediction accuracy is challenging.
  Noise and Outliers: Traffic data can be noisy, and outliers (unusual events like accidents or road closures) can have a significant impact on predictions. Developing models that are robust to noise and capable of detecting and handling outliers is crucial.
- **Q4.** What are existing solutions?
  As of my last knowledge update in September 2021, several existing solutions and research efforts were combining Graph Neural Networks (GNNs) and Long Short-Term Memory (LSTM) networks for traffic forecasting:
  ST-MetaNet: ST-MetaNet is a model that uses GNNs to capture spatial dependencies among traffic sensors and LSTMs to model temporal patterns. It employs a meta-learning approach to adapt to different cities, making it versatile across urban environments.
  Graph WaveNet: This model combines GNNs with WaveNet, an advanced deep learning architecture for time series data. It learns to generate traffic forecasts at different time horizons, providing fine-grained predictions.
  DCRNN (Diffusion Convolutional Recurrent Neural Network): DCRNN uses GNNs to model the spatial dependencies of traffic sensor data and LSTMs to capture temporal patterns. It has been widely used for traffic forecasting tasks and exhibits strong performance.
  Graph Convolutional LSTM (GC-LSTM): GC-LSTM integrates GNNs with LSTMs to model both spatial and temporal aspects of traffic. It has demonstrated effectiveness in handling large-scale urban road networks.
  Gated Graph Convolutional LSTM (GGCL): GGCL com-

bines Gated Graph Convolutional Networks (GGCNs) and LSTMs to predict traffic speed. The model effectively handles dynamic road networks and varying traffic conditions.

Graph Attention Networks (GAT) with LSTMs: Researchers have also explored using GAT, a variant of GNNs that incorporates attention mechanisms, in combination with LSTMs for traffic forecasting. GAT helps capture fine-grained spatial dependencies.

These solutions address the challenges of traffic forecasting by harnessing the capabilities of both GNNs and LSTMs to model complex spatial and temporal patterns in traffic data. Keep in mind that advancements in this field may have occurred since my last update, so it's a good idea to check the latest research papers and implementations for the most recent developments in traffic forecasting using GNNs and LSTMs.

- **Q5.** What is the Core Idea of your project?
  The core idea of traffic forecasting using a Graph Neural Network (GNN) and Long Short-Term Memory (LSTM) network is to leverage the power of neural networks to model and predict complex traffic patterns in a road network.

  GNNs are employed to capture the spatial dependencies among road segments and intersections by representing the road network as a graph. This allows the model to understand how traffic conditions in one part of the network affect others. LSTMs, on the other hand, handle the temporal dynamics of traffic, accounting for changes over time due to factors like rush hours or accidents.

  By combining these two neural network architectures, the model can effectively predict future traffic conditions at various locations and time intervals. This approach enables more accurate and dynamic traffic forecasts, benefiting transportation management, route planning, and congestion mitigation efforts.
- **Q6.** Your Idea is sufficient w.r.t current existing solutions? What aspects they solve? What Not?
  Traffic forecasting using LSTM and Graph Neural Network (GNN) is a powerful approach that addresses several aspects of traffic prediction. It effectively tackles:

  Spatial Dependencies: GNNs capture complex spatial dependencies in road networks, enabling the model to understand how traffic conditions in one area affect others.

  Temporal Dynamics: LSTMs model temporal patterns in traffic data, allowing the prediction of how traffic evolves over time due to factors like rush hours or accidents.

  Graph Representation: They provide an efficient way to represent and process road network data, making it suitable for large-scale urban environments.

  However, while LSTM-GNN hybrids are powerful, they may not fully address the following aspects:

  Data Quality: They may struggle with noisy or incomplete data, requiring pre-processing and data cleaning.

  Real-Time Prediction: Achieving low-latency predictions for real-time applications can be challenging.

  Generalization: Models might not generalize well across different cities or regions with distinct traffic patterns.

  Interpretability: Understanding why the model makes specific predictions can be difficult, which may be crucial for certain applications.

  In summary, while LSTM-GNN models are robust and versatile for traffic forecasting, they may require additional considerations and domain-specific adaptations for specific use cases and data challenges.

IV. Literature Survey

The project benefited greatly from the insightful information and context that the literature survey for this study provided. It involves a thorough analysis of already published papers and articles on the topics of traffic prediction, LSTM networks, and time series forecasting. The poll covered a range of strategies, techniques, and industry-recognized best practices. It allowed us to advance and add to this body of knowledge by providing a framework for comprehending the state of the art in LSTM-based time series forecasting. By incorporating the survey's findings into our model design and methodology, we were able to conduct thorough, up-to-date research.

V. Problem Statement

Given a dataset about time series data of number of vehicles each hour in four different junctions, forecast,

1) the next day's number of vehicles in each junction (single-step forecasting),

2) the next several days' number of vehicles in each junction (repeated single-step forecasting)using deep learning approach (LSTM)

A. Objectives

The objective of the given problem statement is to use a deep learning approach, specifically Long Short-Term Memory (LSTM) neural networks, to forecast the number of vehicles in each of the four different junctions for the following tasks:

1) Single-Step Forecasting: Predict the number of vehicles for the next day in each junction. This involves providing a single prediction for each junction's vehicle count one day into the future.

2) Repeated Single-Step Forecasting: Predict the number of vehicles for the next several days in each junction. This involves making multiple single-step predictions for each junction over several consecutive days into the future.

3)The primary evaluation metric for these predictions is the Root Mean Squared Error (RMSE), which measures the accuracy of the forecasts by quantifying the square root of the average squared differences between the predicted values and the actual values in the time series data.

In summary, the goal is to leverage deep learning, specifically LSTM models, to provide accurate forecasts of vehicle counts for different junctions, helping to predict both short-term (next day) and medium-term (several days ahead) trends in the data.

VI. Methodology

Flowchart/Block:

Traffic Forecasting is an example of time series forecasting using a Long Short-Term Memory (LSTM) neural network. Here's a simplified flowchart of what the code does:

1.Data Preparation:

- Mounts Google Drive to access data. - Loads a CSV file containing traffic data into a Pandas DataFrame. - Converts the "DateTime" column to a datetime format. - Resamples the data to create daily traffic data.

2.Data Sequence Generation:

- Defines a function to create sequences and targets for time series data. - Splits the data into sequences with a specified training window (lookback) and prediction window.

3.LSTM Model Definition:

- Defines a simple LSTM neural network model with one LSTM layer and two fully connected layers. - This model is designed for time series forecasting.

4.Training the LSTM Model:

- Splits the data into training and testing sets using random and sequential methods. - Trains two separate LSTM models with different data split approaches. - Records and prints the loss at each training epoch.

5.Model Evaluation and Forecasting:

- Defines functions for making one-step and multi-step forecasts using the trained models. - Evaluates the trained models' performance using RMSE (Root Mean Squared Error). - Generates forecasts for future time points.

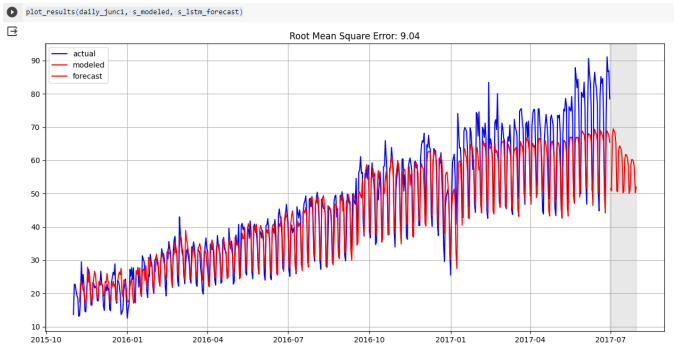

Fig. 6. Traffic Forecasting output

6.Visualization:

- Plots the original time series data (actual), the modeled data (predicted), and the forecasted data for comparison. - Calculates and displays the Root Mean Square Error (RMSE) as a measure of the model's accuracy.

The Project aims to demonstrate how to use LSTM neural networks for time series forecasting and provides a visual comparison of the model's predictions with the actual data.

Algorithm/ Pseudo-code, Mathematical functions.

Algorithm/ Pseudo-code

//Load and Prepare Data

1. Load traffic data from Google Drive into a DataFrame.

//Data Preprocessing

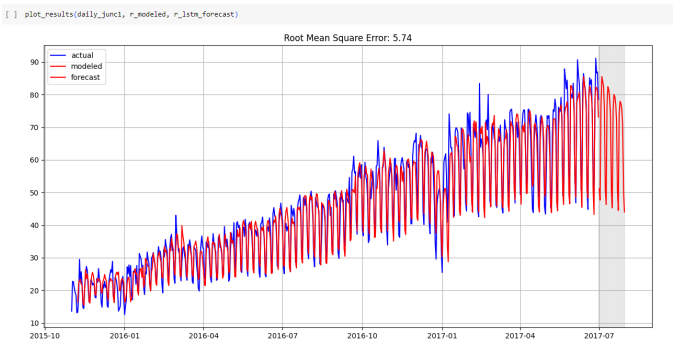

Fig. 7. Traffic Forecasting output

2. Import necessary libraries.

3. Clean and preprocess the data, setting DateTime as the index.

//Time Series Analysis

4. Select data for a specific traffic junction (Junction 1).

5. Resample the data into daily intervals for time series analysis.

//Sequence Generation

6. Define a function to create sequences and their target values.

7.Generate sequences with a training window (lookback) and a prediction window (n pred).

//Dataset Preparation

8. Create a custom dataset for PyTorch to work with sequences.

9. Split the dataset into training and testing subsets.

//Define a Simple LSTM Model

10. Define a basic LSTM neural network model for time series forecasting.

//Train the LSTM Model

11. Train the LSTM model using training data and record training losses.

//Model Evaluation and Forecasting

12. Evaluate the model's performance using RMSE and generate forecasts for future time points.

//Visual Comparison

13. Plot the original time series, modeled data, and forecasts to compare results.

//Calculate RMSE for Model Evaluation

14. Define functions for RMSE and other error metrics.

15. Calculate and display RMSE between the original and modeled data.

//Repeat the Above Steps for a Second Model

16. Create a second LSTM model and train it on the same dataset.

17. Evaluate the second model's performance and generate forecasts.

//Plot Results and Compare Models

18. Plot the results for both models and compare their performance.

Mathematical functions.

1.Time Series Data Manipulation:

Loading and preprocessing the time series data, which involves reading data from a CSV file and converting date-time information into a usable format.

2.Data Sequence Generation: generate sequences:A function that creates sequences (input data) and their corresponding target values for training the model. It defines the training window (how many past steps to consider) and the prediction window (how many future steps to predict).

3.LSTM Model Architecture:

SimpleLSTM: A class that defines the architecture of the Long Short-Term Memory (LSTM) neural network used for time series forecasting. This includes defining the input size, hidden size, and number of LSTM layers, as well as fully connected layers.

4.Training the LSTM Model:

Training the LSTM model involves an optimization process. The code defines an optimizer (Adam), a loss function (Mean Squared Error), and uses data loaders to manage batches of data for training.

5.Model Evaluation and Forecasting: Functions like one step forecast and n step forecast are used for making predictions with the trained models. These functions take historical data as input and produce forecasts.

6.Model Evaluation Metrics: Mean absolute percentage error: Calculates the mean absolute percentage error between true and predicted values. root mean squared error: Computes the root mean squared error, which measures the model's prediction accuracy.

7.Data Visualization:

The code includes functions to plot the results for comparison. It visualizes the actual time series data, the modeled (predicted) values, and the forecasted values. The Root Mean Square Error (RMSE) is also calculated and displayed as a measure of accuracy.

These mathematical functions and operations together enable the training, evaluation, and visualization of LSTM-based time series forecasting models.

VII. Results and Analysis

A. Data Loading and Preprocessing:

The code loads traffic data from a CSV file and preprocesses it.

B. Data Resampling:

The traffic data is resampled to daily intervals to create a time series.

C. Sequence Generation:

The code defines a function to generate sequences of historical traffic data and their corresponding target values for training.

D. LSTM Model Definition:

A simple LSTM neural network model is defined for time series forecasting. It takes historical sequences as input and predicts future values.

E. Model Training:

Two LSTM models are trained, one using random data splitting and another using sequential data splitting. Training involves optimizing the models to minimize Mean Squared Error (MSE).

F. Model Evaluation:

The code calculates and reports the training Mean Squared Error (MSE) for both models.

G. Model Forecasting:

The trained models are used to make one-step-ahead and multi-step-ahead forecasts for future traffic data.

H. Model Evaluation Metrics:

Metrics such as Root Mean Squared Error (RMSE) are used to evaluate the accuracy of the model forecasts.

I. Results Visualization:

The code visualizes the actual traffic data, the modeled (predicted) values, and the forecasted values. It calculates and displays the RMSE between the actual and modeled data.

J. Comparison of Models:

The code compares the performance of the two models by plotting their results. the code trains LSTM models to forecast future traffic data based on historical information. The results are evaluated using RMSE, and the performance of two different data splitting techniques is compared. The code provides insights into how well the models can predict future traffic patterns, which can be useful for traffic management and planning.

K.

VIII. Challenges Faced

Our Problem Statement was to do Traffic Forecasting using Graph Neural Network and LSTM.We faced various challenges while implementing this project.at first we tried implementing it using tenserFlow Library which resulted didn't give us acurate results.Then we did some analysis and got to that we can use FbProphet library as well as Pytorch for forecasting.First, we implemented it using Fbprophet library but didn't get expected result.So,we tried using Pytorch.The results were almost as expected,we faced few issues but we successfully fixed them and implemented our problem statement.

IX. Learning and Insights

This internship helped us gain knowledge in various aspects.It made us familiar with various technologies like GitHub,Google Collab and Overleaf which are essential to learn.We also learned about various concepts of machine learning and learn to implements various concepts practically.We gained practical knowledge about implementation of various tools which we weren't aware.This internship has definitely taken us one step further inn the filed of data science and machine leaning.

X. Future Work

Feature Engineering:

Explore additional relevant features that could improve the accuracy of traffic forecasting. These features might include weather data, holidays, special events, or road construction schedules. Hyperparameter Tuning:

Optimize the hyperparameters of the LSTM model, such as the number of hidden layers, the learning rate, and the batch size. Fine-tuning these parameters can lead to better model performance. Ensemble Models:

Consider combining multiple models, such as LSTM and other time series forecasting techniques like ARIMA, to create ensemble models. Ensemble methods often lead to more robust and accurate predictions. Online Learning:

Implement an online learning approach, where the model is continuously updated with new data. This can be valuable for real-time traffic prediction and adapting to changing traffic patterns. Anomaly Detection:

Integrate anomaly detection techniques to identify unusual traffic patterns or incidents. This can help in improving the model's ability to handle unexpected events. Visualization Tools:

Develop interactive data visualization tools to provide insights into traffic patterns and model predictions. Visualizations can be useful for both stakeholders and the general public. Advanced Loss Functions:

Experiment with advanced loss functions that are more suitable for time series forecasting tasks. Custom loss functions can be designed to address specific challenges in traffic prediction. Multivariate Forecasting:

Extend the project to handle multivariate time series forecasting, where multiple variables (e.g., traffic at multiple junctions, weather data) are considered simultaneously for more comprehensive predictions. Real-Time Deployment:

Work on deploying the model in a real-time system, such as a traffic management application, to provide immediate traffic forecasts and suggestions for optimizing traffic flow. Data Quality Improvement:

Focus on data quality by addressing missing data, outliers, and noise in the dataset. Clean and reliable data can significantly enhance model performance. Localized Models:

Build localized traffic forecasting models that consider specific conditions for different regions or cities. Traffic patterns can vary widely, so region-specific models might be more accurate. Evaluate Environmental Impact:

Assess the environmental impact of traffic flow and use traffic forecasting to optimize routes to reduce fuel consumption and emissions. User-Friendly Interfaces:

Develop user-friendly interfaces for policymakers, traffic managers, and the public to access traffic forecasts and insights easily. These potential improvements and directions can extend the project's impact and make it more useful for traffic management, urban planning, and improving overall transportation systems.

XI. Conclusion

The code first mounts Google Drive to access a CSV file containing traffic data.It loads the data into a Pandas DataFrame and performs some basic data exploration.The code focuses on traffic data for Junction 1.It resamples the data into daily intervals and performs some data visualization.A function is defined to create sequences of historical traffic data and their corresponding target values for training.A simple LSTM neural network model is defined for time series forecasting. The model architecture includes LSTM layers and fully connected layers.Two LSTM models are trained using different data splitting techniques (random and sequential).Training includes defining an optimizer and a loss function and iterating over epochs to optimize the model.The code calculates and reports the training Mean Squared Error (MSE) for both models. The trained models are used to make one-step-ahead forecasts for future traffic data.Metrics like Root Mean Squared Error (RMSE) are used to evaluate the accuracy of the model forecasts.The code visualizes the actual traffic data, the modeled (predicted) values, and the forecasted values.It calculates and displays the RMSE between the actual and modeled data. The code compares the performance of the two models by plotting their results. In summary, this code demonstrates how to build and train LSTM models for time series forecasting, with a focus on traffic data. It provides insights into the ability of the models to predict future traffic patterns and compares the results of different data splitting techniques. The primary goal is to create accurate forecasts for traffic management and planning.

Acknowledgment

It's important to acknowledge that this code appears to be a demonstration of a time series forecasting pipeline using LSTM models. The code might need some refinements and adjustments for specific use cases, but it serves as a foundation for time series forecasting tasks. In summary, the code leverages LSTM models to forecast future traffic data based on historical information, evaluates their performance, and visualizes the results. This kind of analysis can be valuable for traffic management and planning.

Individual Contribution

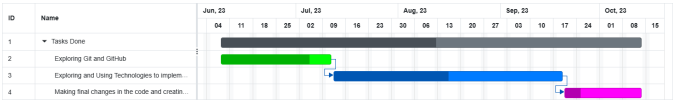

Fig. 8. Gantt Chart

Vaishnavi Pawar: Library Research,Problem Solving and Analysis and Report.

Bharti Pawar: Algorithm Research,Code Examination,Architecture Design,Report.

Shreya Jadhav:Initial Analysisand Research and Data Cleaning and Transformation.

IMPLEMENTED/BASE PAPER

Paper Name: A survey on long short-term memory networks for time series prediction

Author Name: Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, Michael Weyrich.

Conference:14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME ‘20.

Published Year:2015.

REFERENCES

[1] Hochreiter S, Schmidhuber J (1997a) Long short-term memory. Neural Comput 9:1735–80

[2] Sang C, Pierro MD (2019) Improving trading technical analysis with tensorflow long short-term memory (LSTM) neural network. J Finance Data Sci 5(1):1–11