Traffic Flow Prediction Using Graph Neural Network

Prisha Gupta
Amrita School of ArtificialIntelligence,
Amrita Vishwa Vidyapeetham
Coimbatore, India

Vedant Maheshwari
Amrita School of Artificial Intelligence,
Amrita Vishwa Vidyapeetham
Coimbatore, India

Dr. Kritesh Kumar Gupta
Amrita School of Artificial Intelligence,
Amrita Vishwa Vidyapeetham Coimbatore, India

***Abstract*—Traffic prediction is a complicated task in transportation management. Here we are using *GNN (Graph Neural Network)* and LSTM *(*long short-term model*) to* solve our problems in predicting the flow of traffic.**

More accurate and precise predictions are made possible by this approach, which improves the model's capacity to identify complex patterns within traffic data. Processing large structures is another way that the LSTM helps to more precise and reliable predictions are made possible by this approach. A further factor in the LSTM's efficiency in handling different input value quantities is its processing capacity for large structures.

Promising results in traffic flow prediction are shown by the suggested LSTM-based technique, indicating its potential for practical applications in traffic management. By employing certain activation functions, the model gains more flexibility and is better equipped to handle a variety of traffic situations. By offering a solid answer to the problem of inconsistent input values in traffic flow prediction, the research described in this paper advances the field of precise transportation systems.

Keywords – Traffic flow prediction, Model development, LSTM (*long short-term model*), GNN (Graph Neural Network).

I. INTRODUCTION

Problem and Purpose Statement

We consider Traffic congestion as a normal intrusion in our daily life, but we face many negative consequences because of this congestion it affects our economy in its means, as it takes people valuable time. The accumulation of heavy traffic causes pollution and global warming, inefficiency in punctuality of some logistical companies depended on deliveries and transport, an increase of fuel consumption, accidents etc.

Traffic prediction helps us overcome the consequences of traffic congestion. It helps us to predict the congestion occurring. It helps the user make precise decisions, not only for the travelers but also for the authorities like The Intersection Traffic Signal Control Problem (ITSCP), Ministry of Road Transport and Highways, etc., to plan for better economic development.

The need of Artificial Intelligence (AI) and Machine Learning (ML):

The main goal of AI in transportation is to predict traffic density and ensure a safe and pleasing traffic flow. AI can analyze the given vast historical and real-time data and also will be able to recognize the correlations and patterns which may not be found in traditional ways. It is beneficial in providing enhanced public safety by taking proactive measures, excels in resource optimization by making good use of the resources for precise and enhanced performance of the model.

ML algorithms in Al are capable of adapting to continuous real-time and new data. ML's analysis of the historical data gives us the scope to observe the trends, patterns and any other abnormal activities in the traffic. The AI and ML's efficiency, adaptivity, optimization, and analyzation make the model more comfortable for the users preventing human error.

Literature Review

In this comprehensive review, we embark on a thorough exploration of the intricate relationship between traffic congestion and the methodologies employed for forecasting and predicting traffic flow to foster economic growth. A detailed analysis is presented, providing insight into the methodical incorporation of cutting-edge technologies including Artificial Intelligence (AI), Machine Learning (ML), Graph representation, and Graph Neural Networks (GNN). The synthesis of these advanced tools and techniques contributes significantly to our understanding of traffic dynamics and paves the way for innovative solutions to tackle congestion-related challenges.

The investigation delves into the diverse array of databases that serve as valuable repositories of information, enabling a multifaceted analysis of traffic patterns. Noteworthy databases such as Traffic sensor data, PeMS, Social media feeds, and OpenStreetMap come under scrutiny, revealing their instrumental role in shaping strategies for mitigating traffic congestion. By leveraging these rich datasets, researchers gain insights into the intricate web of factors influencing traffic, ultimately aiming to devise interventions that not only alleviate congestion but also foster economic prosperity. This interdisciplinary approach underscores the interconnected nature of urban mobility and economic development.

As the project unfolds, a pivotal focus emerges on the practical challenges associated with dataset selection, Graph Neural Network (GNN) application, and the intricate process of code development and model training. The narrative expands to acknowledge the inherent complexities of training diverse datasets and models, emphasizing the need for ongoing research to explore varied methodologies and longitudinal approaches. This multifaceted exploration positions the endeavor as not just a solution to immediate congestion issues but also as a stepping stone toward a more profound comprehension of the intricate dynamics that govern traffic flow and economic growth.

II. METHODOLOGY

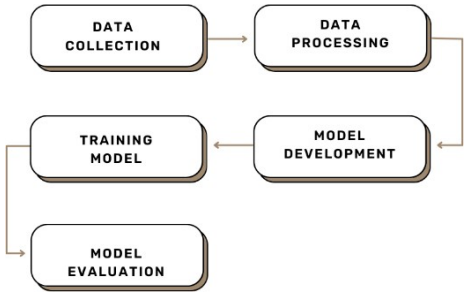


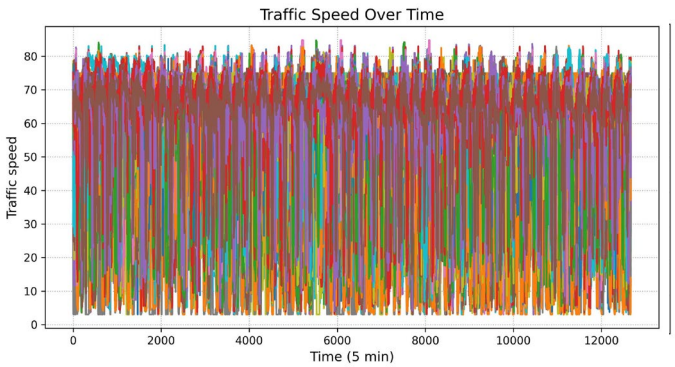
Fig. 1. Flow Diagram

A. Data collection and pre-processing:

The medium version of the PeMSD7 dataset is the one we will use for this application. Using the Caltrans Performance Measurement System (PeMS), traffic speed data from 39,000 sensor sites was gathered during the weekdays in May and June 2012 to create the original dataset. In the medium form, we will only take into account 228 stations spread throughout California's District 7. This dataset aggregates the 30-second speed values these stations produce into 5-minute periods.

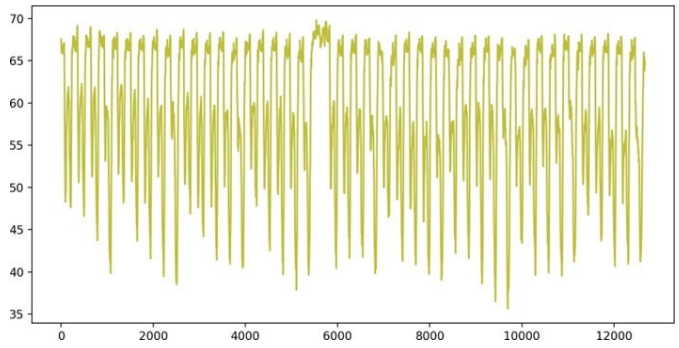
B. Building the graph

After the data had been pre-processed, the graph needed to be constructed. In a graph, the junctions or regions of the road network were portrayed as nodes, and the connections between them were represented as edges. The graph was made by identifying the nodes and edges in the road network and transforming them into a graph representation. Using this dataset, our initial goal is to illustrate the traffic speed evolution. Time series forecasting is a classic because features like seasonality can be very useful. However, non-stationary time series may require additional processing steps prior to their utilization.

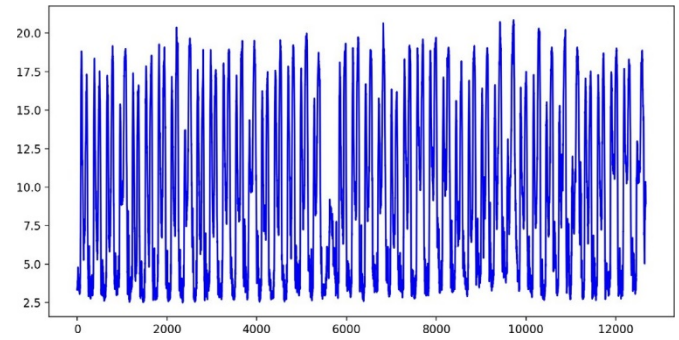


Graph1: Dataset Visualization

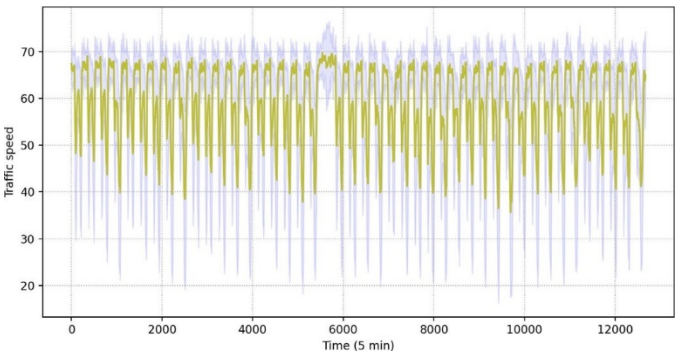
The Graph 1 shows the Dataset Visualization of the trend of traffic on junctions over the years. It gives a visual representation of the number of vehicles that pass through different junctions. The different colors in thegraph represent the different junctions. Unfortunately, the data is too noisy for this approach to provide any useful information. Alternatively, the data relating to a few sensor sites could be plotted. It might not, however, be an accurate representation of the complete dataset. Plotting the mean traffic speed with standard deviation is an additional option. In this manner, we have the capability to see a summary of the dataset.



(a) Mean Graph



(b) Standard Deviation Graph



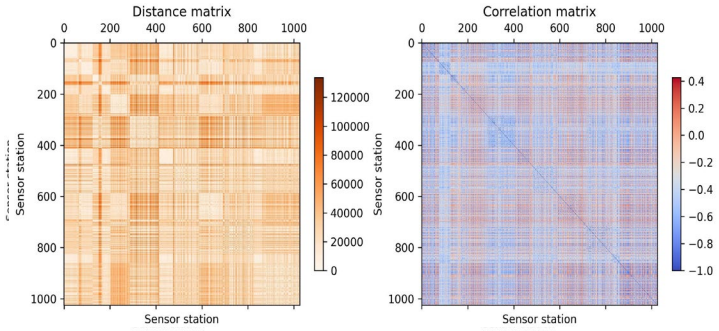
Graph 2: Mean graph and Standard deviation graphs

A visual depiction of the variations in traffic patterns throughout the year and at different times of the day is given by Graph 2, which displays the Mean and Standard Deviation graphs. These graphs can provide valuable insightsinto traffic patterns that can be used to improve transportation planning and management.

The distance matrix represents the pairwise distance between a set of points, the difference, and the correlation matrix represents the pairwise correlations between variables in the data set, the strength and direction of linear relationships. The correlation coefficient varies between [-1 1], where -1 means a perfect negative correlation, 1 means a perfect positive correlation.

After analyzing both subplots, we can conclude that a long distance between stations means that the correlation coefficient is low and, in the subplot, it shows darker lines, and a short distance between stations means that the correlation coefficient is high and in the subplot. -graph shows brighter lines.

There are significant spikes in the variability of the traffic speed. This makes sense given that the sensor stations are dispersed throughout California's District 7, where some traffic may be held up for certain sensors while not for others.



Graph 3- Distance and correlation matrices

Graph 3 represents the Plotting Correlation (Pearson correlation). Plotting the correlation between the speed readings from each sensor will allow us to confirm that. We can also compare it to the separations between each station in addition to that. Nearby stations ought to show comparable values more frequently than far-off ones.

The Pearson correlation coefficients for every sensor station are then determined. The speed matrix needs to be transposed; otherwise, we will obtain the correlation coefficients for each time step. In order to make it easier to compare the two plots, we finally invert them.

It's interesting to see that strong correlations between stations persist even at large distances between them (and vice versa).

C. Processing the Dataset:

Making a temporal graph out of the tabular dataset is the initial stage. Thus, we must first make a graph from the unprocessed data. Put otherwise, we need to meaningfully connect the many sensor stations. The distance matrix, which is fortunately available to us, appears to be a useful tool for connecting the stations.

The adjacency matrix can be computed from the distance matrix in a number of ways. We will compute a weighted adjacency matrix using a more sophisticated processing method. We use the following formula to calculate weights between 0 (no connection) and 1 (strong connection), rather than binary values.

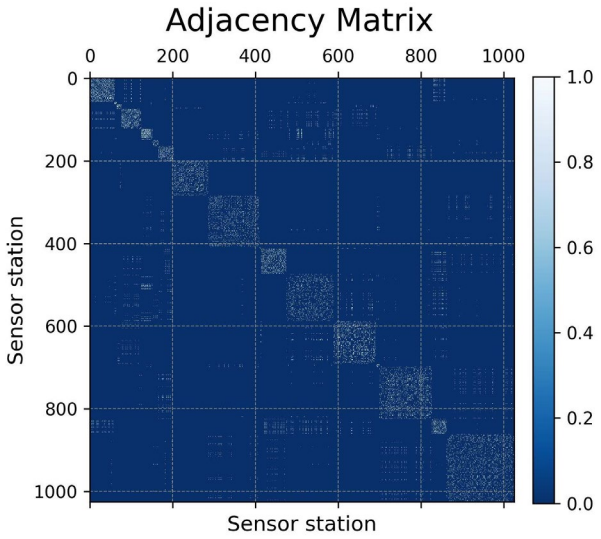
$$W_{ij} = exp\left(-\frac{d_{ij}^2}{\sigma^2}\right), i \neq j \; and \; exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) \in 0, otherwise$$

a. We develop a function that accepts three arguments, the distance matrix and the two thresholds σ^2 and ϵ, to compute the adjacency matrix. We compute d^2 by dividing the distances by 10,000, in order to scale it down for the normalization.

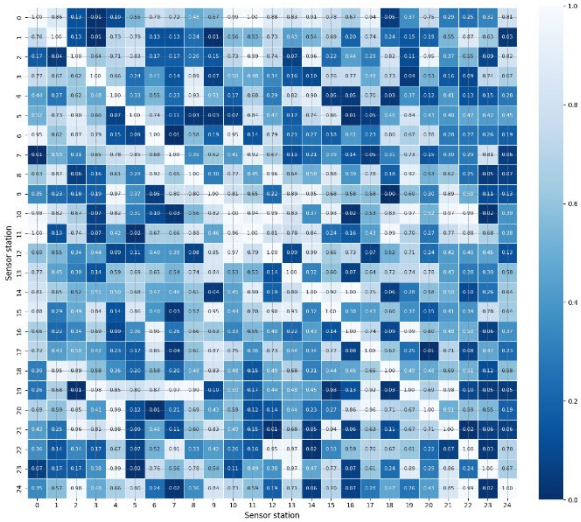
b. We want weights when their values are greater than or equal to ϵ, otherwise, they should equal zero. The outcomes are True or False statements depending on whether the weights are more than or equal to ϵ. This is why we need a mask of ones to convert it back into 0 and 1 values. We multiply it a second time so that we only obtain the real values of weights that are greater than or equal to ϵ.

Example of the adjacency matrix that we obtain.

To better visualize it we make a matrix graph, shown below.



Graph 4- Adjacency matrix



Graph 5- Heat map

The heatmap illustrates the adjacency relationships between different sensor stations. In the context of the code, these sensor stations could represent nodes in a network.

Darker shades (e.g., dark blue) represent stronger connections (higher values in the adjacency matrix), while lighter shades (e.g., light blue) indicate weaker or no connections.

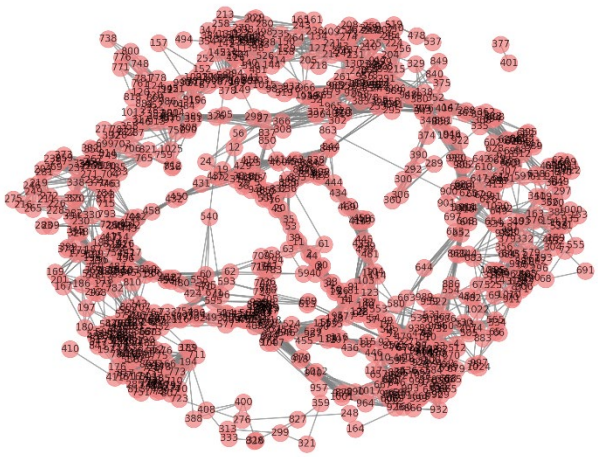
Numerical annotations within each cell of the heatmap display the specific values in the adjacency matrix. These values quantify the degree of association or connectivity between pairs of sensor stations.

Diagonal elements are set to 1, indicating that each sensor station is perfectly connected to itself.

C. Graph plotting of the processed data

Using *networkx*, we can also easily plot it as a graph. Since the links in the current situation are binary, we can simply take into account any weight greater than 0.

Although edge labels may be used to represent these values, this would make the graph very hard to read. Because they are so close to one another, a large number of nodes are in fact interconnected.



Graph 6 – The PeMS-M dataset as a graph (every node represents a sensor station)

Yet, despite that, we can distinguish several branches that could correspond to actual roads.

D. Z-score Normalization:

With a graph in hand, we can now concentrate on the time series component of this issue. Normalizing the speed numbers is the first step in preparing them for feeding into a neural network.

To determine how far a data point deviates from the mean, we use the Z-Score value. In technical terms, it calculates the standard deviations that are above or below the mean. It is within the range of -3 to +3 standard deviations.

We use the basic formula for the normalization;

$$y = \frac{x-\mu}{\sigma}$$

where y is new value of the normalized data.

Table 1: Normalized Speeds

[[1.16337385	0.81204374	…	0.86833647	0.64952888]
[1.29184496	0.93622448	…	0.7713368	0.65704686]
[1.35179814	0.80249137	…	0.70897987	0.49916923]
…				
[0.63235995	0.74517718	…	0.61198019	0.67208283]
[0.48675936	0.87891029	…	0.7574797	0.73222669]
[0.10991079	0.8502532	…	0.82676518	0.65704686]]

Edge Indices:
(array([0, 0, 0, ..., 1025, 1025, 1025]),
array([2, 8, 11, ..., 986, 1004, 1013]))

These numbers have the proper standardization. They may now be used to produce time series for every node.

III. ML VALIDATION

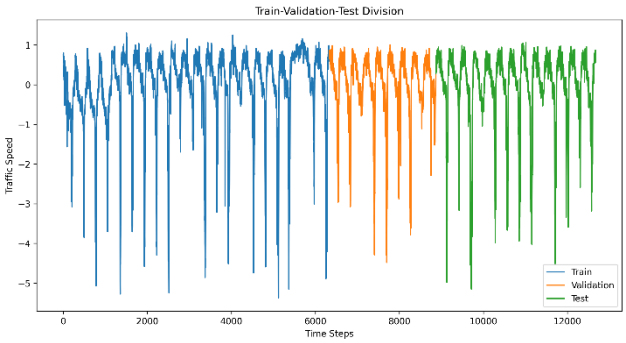
After we pre-processed the data, it is essential that we verify that whether the model that we are working with is giving the required result. So here the ML validation comes into play. It ensures whether the model can make accurate predictions on the unseen data or not. It ensures that the model is not just learning the predictions of an input data but can also give an accurate prediction for any new set of data, thus preventing the problem of over fitting.

In the process of training an ML model we experimented with various methods and techniques to get the desired result. We compared the output of various methods and selected the model that gave us the best possible result.
The steps involved in ML Validation are as follows:

1. Data Splitting and Normalization

We began by splitting our dataset into training, validation and test sets in a ratio of 50%, 20%, and 30% respectively.
Allocating 50% of the dataset to the training set allows our model to learn various meaningful patterns.
The validation set 20% helps in tuning the hyperparameters, selecting models and preventing overfitting.
The test size that is 30%, provide a more reliable result of the model's performance.

So basically, we have used a 70-30 split which keeps a balance between all datasets.



Graph 7- Train-Validation-Test Data Split

The division is as follows:

Table2. The division of dataset

	Samples	Features
Train set size	6336	26
Validation set size	2534	26
Test set size	3802	26

2. Creating TensorFlow Datasets

We then construct the dataset needed to solve our forecasting issue. This is one way to formulate the forecasting problem: provided a series of road speed valuesat times

t+1, t+2, t+3, t+4 ..., t+T

We intend to forecast the values of the roads speed in the future for times

t+T+1, t+T+2, t+T+3, t+T+4 ..., t+T+h.

Thus, for every time t, we have T vectors of size N as inputs to our model and h vectors of size N as targets, where N is the number of roads.

We use the Keras built-in function timeseries_dataset_from_array(). The function create_tf_dataset() takes as input a numpy.ndarray and returns a tf.data.Dataset. In this function input_sequence_length=T and forecast_horizon=h.

Assume forecast_horizon= 3 in the multi_horizon argument. The model will forecast for time steps t+T+1, t+T+2, t+T+3 if multi_horizon=True.
However, if multi_horizon=False, the target will have form (T, 1) since the model will only forecast for time step t+T+3.
We got the output as:

To predict the speed 3- time ahead of time, we take the last 12 speed measurements recorded on each road:
Inputs Shape: (64, 12, 26, 1)
Here
64 is the batch size, i.e., we have 64 samples
12 is the input sequence length, i.e., Each sample in the batch has a sequence of 12-time steps.
26 is the number of features for each time step.
1 represents each feature at a time step.
Targets Shape: (64, 3, 26)
Here:
64 is the batch size
3 is the target sequence length, i.e., Each sample in the batch has a target sequence of 3-time steps.
26 is the number of features for each time step in the target sequence.

3. Roads Graph Neural Network

The road segment distance is available in the PeMSD7 dataset. Using these distances, the graph adjacency matrix needs to be created.

It can be done by the formula as stated above in the Methodology section.

The adjacency matrix returned by the function compute_adjacency_matrix() is boolean, with 1 denoting the presence of an edge between two nodes. To store the graph's data, we employ the following class.

We got the number of nodes: 26 and number of edges: 52.

4. Graph convolution layer And Network architecture

The GraphConv custom layer is designed to implement a graph convolution operation within a neural network. This layer is specifically tailored for graph-based neural network architectures where nodes are connected by edges, representing relationships in a graph structure.

The graph convolution layer performs the following steps:

Layer initialization followed by defining Layer parameters and then finally Aggregation Method.

1. The representations of the nodes are computed in self.compute_nodes_representation() by multiplying the input features by self.weight.

2. The messages from the aggregated neighbors are computed in self.compute_aggregated_messages() by first aggregating the neighbors' representations and then multiplying the results by self.weight.

3. The layer's final output is calculated in self.update() by merging the nodes representations and the neighbours' aggregated messages.

5. LSTMGC Layer: Graph Convolution + LSTM + Dense

The LSTMGC layer is a layer that combines a graph convolutional layer (GraphConv) with a Long Short-Term Memory (LSTM) layer and a dense layer.

1. This layer utilizes a 'GraphConv' layer to perform Graph Convolution on the input data. It captures the node-wise relationship in the graph structure.

2. The output from the previous step is passed to LSTM layer to capture temporal (time) dependencies.

3. The LSTM takes only 3D tensor as input. This output is then passed on to the dense layer togive the final output.

6. Graph Neural Network Model Training

We then define a Keras model for time series forecasting, with a LSTMGC layers. The model is then compiled with a specified optimizer and loss function in preparation for training.
Here we set the
- Input features = 1
- Batch size = 64
- Epochs = 50
- Input sequence length = 12
- Multi-Horizon = False
- Output Features = 10
- LSTM unit = 64
- Forecast Horizon = 3

Keras Input layer is defined with the shape *(input_sequence_length, graph.num_nodes, in_feat)*

Model Compilation
RMSprop optimizer with a learning rate of 0.0002 is chosen. Mean Squared Error (MSE) is used as the loss function.

Model Summary
- Model Name = model
- Number of layers = 2

Table3. Model Summary

Model: "model"		
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 12, 25, 1)]	0
lstmgc (LSTMGC)	(None, 3, 25)	17101
Total params: 17101 (66.80 KB)		
Trainable params: 17101 (66.80 KB)		
Non-trainable params: 0 (0.00 Byte)		

Layers:

1. Input Layer (input_1)
   Output Shape: (None, 12, 26, 1) (None represents the batch size, 12 is the input sequence length, 26 is the number of nodes, and 1 is the input feature dimension.)

2. LSTMGC Layer (lstmgc)
   Output Shape: (None, 3, 26) (None represents the batch size, 3 is the output sequence length, and 26 is the number of nodes.)
   Trainable Parameters: 17,101
   Parameters:
   -Total Trainable Parameters: 17,101
   -Total Non-Trainable Parameters: 0
   -Total Parameters: 17,101 (66.80 KB)

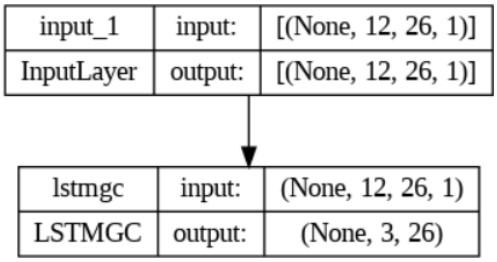


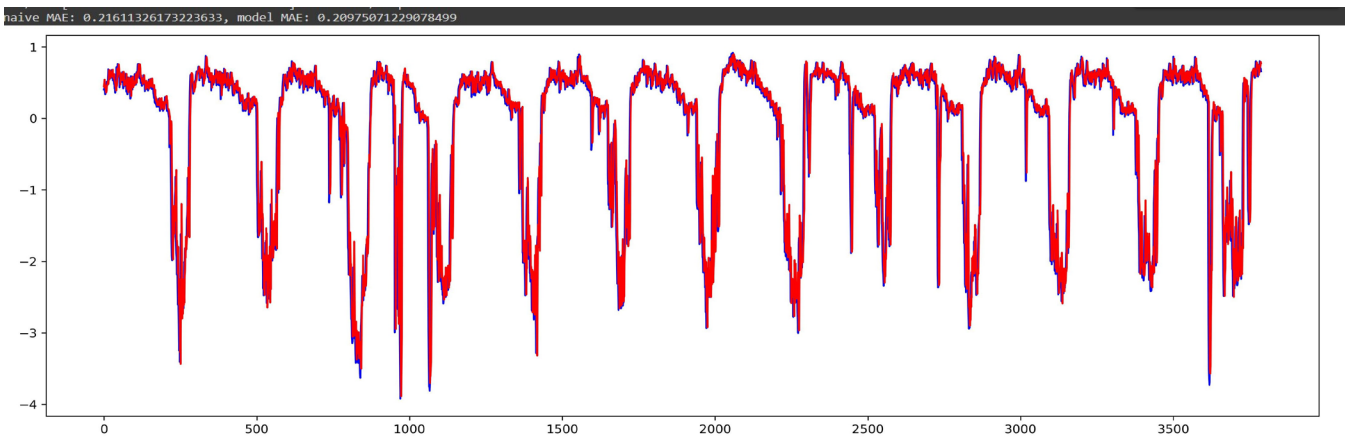
Fig. 2. Flow Diagram

Model Training

The *model.fit* function initiates the training process, during which the model's parameters are adjusted to minimize the specified loss function. The training progress is continuously monitored. And at the end of each epoch, and the model's performance on the validation dataset is evaluated.

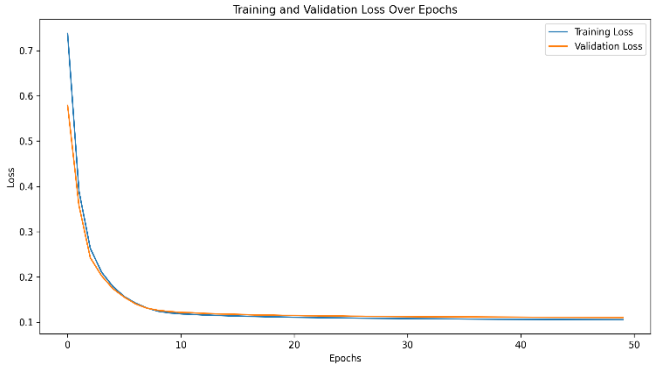
7. Making forecasts on test set

- We plotted a graph between actual v/s predicted values and also verified that using a scatter plot.



Graph. 8. Between actual and predicted values

In the result as we can say that the loss value keeps on decreasing at successive epochs, hence it continues to all 50 epochs which was defined the beginning.



Graph 9: Training and Validation Loss over Epochs = 50

- The model is trained over 50 epochs

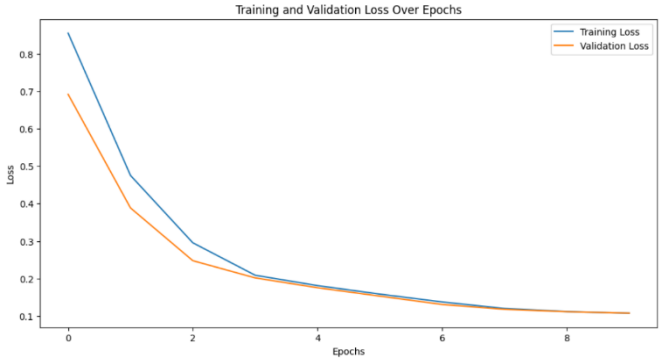
- For each epoch, the training loss and validation loss are reported.

  - The training starts with a relatively high loss, and over epochs, both the training and validation loss decrease.

  - The decreasing loss values indicate that the model is learning and improving its performance.

- The loss values seem to converge, indicating that the model's performance stabilizes

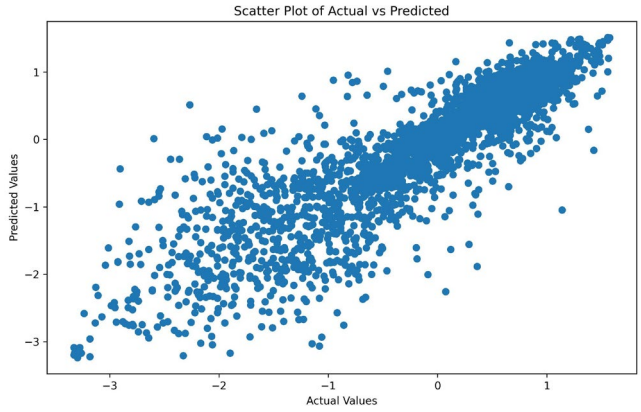
- The final training loss is 0.0873, and the final validation loss is 0.0902



Graph 10: Training and Validation Loss over Epochs = 1

IV. IMPLEMENTATION AND RESULTS

The project on traffic prediction using GNN (graph neural network) achieved good results with a root mean squared error as shown in *Table2*.



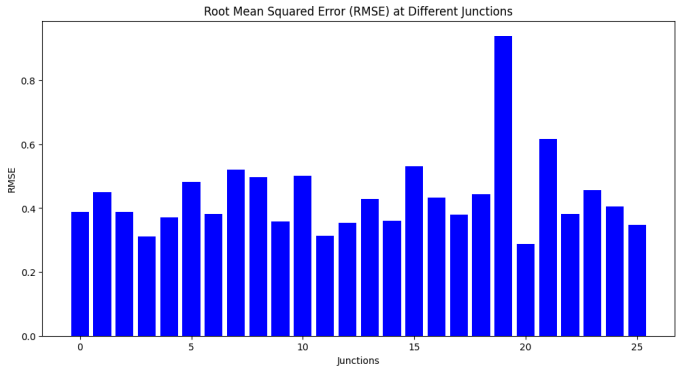
Graph. 11.Scatter plot Between actual and predicted values

The RMSE values vary across different junctions, indicating varying levels of predictive accuracy.

- Junctions with **lower RMSE values** (e.g., Junction 4, Junction 21) suggest that the model performs relatively well in capturing the patterns for those specific locations
- Junctions with **higher RMSE values** (e.g., Junction 20, Junction 22) might indicate challenges in accurately predicting the patterns for those locations. Investigating these specific cases can provide insights into potential model improvements.
- **Junction 20** stands out with a **relatively high RMSE** of **0.9395**. This might require special attention, as it indicates a larger discrepancy between predicted and actual values for this junction.
- **Junction 21** has a **low RMSE** of **0.2876**, suggesting good performance in predicting traffic patterns for this specific location.

Table 3. The division of dataset

JUNCTIONS	RSME VALUES
Junction 1	0.3884
Junction 2	0.4491
Junction 3	0.3885
Junction 4	0.3102
Junction 5	0.3718
Junction 6	0.4829
Junction 7	0.3813
Junction 8	0.5207
Junction 9	0.4973
Junction 10	0.3573
Junction 11	0.5012
Junction 12	0.3127
Junction 13	0.3540
Junction 14	0.4284
Junction 15	0.3609
Junction 16	0.5311
Junction 17	0.4320
Junction 18	0.3793
Junction 19	0.4437
Junction 20	0.9395
Junction 21	0.2876
Junction 22	0.6158
Junction 23	0.3821
Junction 24	0.4567
Junction 25	0.4046



Graph 12. RSME plot at different junctions

V. CONCLUSION

The developed model, a hybrid of Graph Convolutional Layers and LSTM units, demonstrated the ability to learn complex relationships within the traffic network. The adjacency matrix, derived from route distances, facilitated the incorporation of spatial dependencies, while the LSTM component addressed temporal aspects. The training process involved careful tuning of hyperparameters, to optimize model performance.

The model's performance was evaluated using Root Mean Squared Error (RMSE), providing insights into its accuracy across individual junctions. Notably, Junction 20 emerged as a focal point for further investigation due to its comparatively high RMSE.

In conclusion, this project signifies a valuable contribution to the field of traffic flow prediction, presenting a model capable of learning and predicting patterns across multiple junctions. The variation provides a roadmap for iterative improvements, laying the groundwork for enhanced predictive accuracy and real-world applicability in traffic management and planning scenarios.

VI. REFERENCES

[1] V. Ahsani, M. Amin-Naseri, S. Knickerbocker, and A. Sharma, "Quantitative analysis of probe data characteristics: Coverage, speed bias and congestion detection precision," Journal of Intelligent Transportation Systems, vol. 23, no. 2, pp. 103–119, Oct. 2018, doi: 10.1080/15472450.2018.1502667.

[2] Y. Hou, Z. Deng, and H. Cui, "Short-Term Traffic Flow Prediction with Weather Conditions: Based on Deep Learning Algorithms and Data Fusion," Complexity, vol. 2021, pp. 1–14, Jan. 2021, doi: 10.1155/2021/6662959.

[3] Y. Hou, Z. Deng, and H. Cui, "Short-Term Traffic Flow Prediction with Weather Conditions: Based on Deep Learning Algorithms and Data Fusion," Complexity, vol. 2021, pp. 1–14, Jan. 2021, doi: 10.1155/2021/6662959.

[4] W. Zeng, K. Wang, J. Zhou, and R. Cheng, "Traffic flow prediction based on hybrid deep learning models considering missing data and multiple factors," Sustainability, vol. 15, no. 14, p. 11092, Jul. 2023, doi: 10.3390/su151411092.

[5] J. Bai et al., "A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting," ISPRS International Journal of Geo-information, vol. 10, no. 7, p. 485, Jul. 2021, doi: 10.3390/ijgi10070485.

[6] S. Sepasgozar and S. Pierre, "Network Traffic Prediction Model considering road traffic parameters using artificial intelligence methods in VANET," IEEE Access, vol. 10, pp. 8227–8242, Jan. 2022, doi: 10.1109/access.2022.3144112.

[7] S. Suhas, V. Kalyan, M. R. Katti, B. V. A. Prakash, and C. Naveena, "A Comprehensive Review on Traffic Prediction for Intelligent Transport System," Ieee, Mar. 2017, doi: 10.1109/icraect.2017.33.

[8] H. R. Deekshetha, A. V. S. Madhav, and A. K. Tyagi, "Traffic prediction using machine learning," in Lecture notes on data engineering and communications technologies, 2022, pp. 969–983. doi: 10.1007/978-981-16-9605-3_68.

[9] M. Shaygan, C. Meese, W. Li, X. Zhao, and M. Nejad, "Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities," Transportation Research Part C: Emerging Technologies, vol. 145, p. 103921, Dec. 2022, doi: 10.1016/j.trc.2022.103921.

[10] B. Abdulhai, H. Porwal, and W. Recker, "Short-Term traffic flow prediction using Neuro-Genetic algorithms," Journal of Intelligent Transportation Systems, vol. 7, no. 1, pp. 3–41, Jan. 2002, doi: 10.1080/713930748.

[11] M. Akhtar and S. Moridpour, "A review of traffic congestion prediction using Artificial Intelligence," Journal of Advanced Transportation, vol. 2021, pp. 1–18, Jan. 2021, doi: 10.1155/2021/8878011.

[12] V. Ahsani, M. Amin-Naseri, S. Knickerbocker, and A. Sharma, "Quantitative analysis of probe data characteristics: Coverage, speed bias and congestion detection precision," Journal of Intelligent Transportation Systems, vol. 23, no. 2, pp. 103–119, Oct. 2018, doi: 10.1080/15472450.2018.1502667.

[13] F. Aljuaydi, B. Wiwatanapataphee, and Y. Wu, "Deep Learning-Based Prediction Models for Freeway Traffic Flow under Non-Recurrent Events," 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT), May 2022, doi: 10.1109/codit55151.2022.9803892.

[14] B. Alsolami, R. Mehmood, and A. Albeshri, "Hybrid Statistical and Machine Learning Methods for Road Traffic Prediction: A review and tutorial," in EAI/Springer Innovations in Communication and Computing, 2019, pp. 115–133. doi: 10.1007/978-3-030-13705-2_5.

[15] J. S. Angarita-Zapata, A. D. Masegosa, and I. Triguero, "A Taxonomy of Traffic Forecasting Regression Problems from a Supervised learning perspective," IEEE Access, vol. 7, pp. 68185–68205, Jan. 2019, doi: 10.1109/access.2019.2917228.

[16] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-Dependent Pre-Trained deep neural networks for Large-Vocabulary speech recognition," IEEE Transactions on Audio, Speech, and Language Processing, vol. 20, no. 1, pp. 30–42, Jan. 2012, doi: 10.1109/tasl.2011.2134090.

[17] E. I. Vlahogianni, M. G. Karlaftis, and J. Golias, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach," Transportation Research Part C: Emerging Technologies, vol. 13, no. 3, pp. 211–234, Jun. 2005, doi: 10.1016/j.trc.2005.04.007.

[18] A. Elfar, A. Talebpour, and H. S. Mahmassani, "Machine Learning approach to Short-Term Traffic Congestion Prediction in a Connected Environment," Transportation Research Record, vol. 2672, no. 45, pp. 185–195, Sep. 2018, doi: 10.1177/0361198118795010.

[19] A. Elfar, A. Talebpour, and H. S. Mahmassani, "Machine Learning approach to Short-Term Traffic Congestion Prediction in a Connected Environment," Transportation Research Record, vol. 2672, no. 45, pp. 185–195, Sep. 2018, doi: 10.1177/0361198118795010.

[20] M. M. Fischer and S. Gopal, "ARTIFICIAL NEURAL NETWORKS: a NEW APPROACH TO MODELING INTERREGIONAL TELECOMMUNICATION FLOWS*," Journal of Regional Science, vol. 34, no. 4, pp. 503–527, Nov. 1994, doi: 10.1111/j.1467-9787.1994.tb00880.x.

[21] Y. Jia, J. Wu, and M. Xu, "Traffic Flow Prediction with Rainfall Impact Using a Deep Learning Method," Journal of Advanced Transportation, vol. 2017, pp. 1–10, Jan. 2017, doi: 10.1155/2017/6575947.

[22] F. Kunde, A. Hartenstein, S. Pieper, and P. Sauer, "Traffic Prediction using a Deep Learning Paradigm.," EDBT/ICDT Workshops, Jan. 2017, [Online]. Available: http://ceur-ws.org/Vol-1810/BIGQP_paper_03.pdf

[23] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," Ieee, Oct. 2016, doi: 10.1145/2996913.2997016.

[24] E. I. Vlahogianni, M. G. Karlaftis, and J. Golias, "Short-term traffic forecasting: Where we are and where we're going," Transportation Research Part C: Emerging Technologies, vol. 43, pp. 3–19, Jun. 2014, doi: 10.1016/j.trc.2014.01.005.

[25] N. G. Polson and V. Sokolov, "Deep learning for short-term traffic flow prediction," Transportation Research Part C: Emerging Technologies, vol. 79, pp. 1–17, Jun. 2017, doi: 10.1016/j.trc.2017.02.024.