Enhancing Urban Traffic Management Through Hybrid Convolutional and Graph Neural Network Integration

[1]Karrar S. Mohsin, [2]Jhansilakshmi Mettu, [3]**Chinnam Madhuri**, [4]**Gude Usharani**, [5]**Silpa N and [6]Pachipala Yellamma**

[1]Department of Information Technology, College of Science, University of Warith Al-Anbiyaa, Karbala, Iraq.
[2,3,4]Department of Information Technology, CMR Engineering College, Hyderabad, Telangana, India.
[5]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women, Bhimavaram, Andhra Pradesh, India.
[6]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Andhra Pradesh, Vaddeswaram, India.
[1]karar.sadeq@uowa.edu.iq, [2]jhansi 2023@gmail.com, [3]chinnam.madhuri@gmail.com, [4]gude.usharani@gmail.com, [5]nrusimhadri.silpa@gmail.com, [6]pachipala.yamuna@gmail.com

Correspondence should be addressed to Jhansilakshmi M : jhansi 2023@gmail.com.

Article Info
Journal of Machine and Computing (http://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202404034
Received 06 April 2023; Revised from 17 January 2024; Accepted 05 February 2024.
Available online 05 April 2024.

Abstract – Traffic congestion has made city planning and citizen well-being difficult due to fast city growth and the increasing number of vehicles. Traditional traffic management fails to solve urban transportation's ever-changing issues. Traffic prediction and control systems are vital for enhancing Traffic Flow (TF) and minimizing congestion. Smart cities need advanced prediction models to regulate urban TF as traffic management becomes more complex. This paper introduces a hybrid Convolutional Neural Networks (CNN) and Graph Neural Networks (GNN) model for better real-time traffic management. The hybrid model combines CNNs' spatial feature extraction with GNNs' structural and relational data processing to analyze and predict traffic conditions. Traffic camera images are pre-processed to extract spatial characteristics. Traffic network graph construction is used for structural research. The model accurately captures traffic topology and space. The proposed method sequentially processes spatial data with CNNs and integrates them with GNNs. The final hybrid model is trained on one year of traffic data from diverse circumstances and events. The hybrid model is compared to CNN, GNN, and traditional Traffic Prediction Models (TPM) like ARIMA and SVM utilizing MAE, RMSE, and MAPE. The hybrid GNN+CNN model outperforms benchmark models with lower MAE, RMSE, and MAPE across several prediction intervals.

Keywords – Urban Traffic Management, Deep Learning, Traffic Prediction Models, Traffic Flow Control, CNN.

I. INTRODUCTION

The growth of metropolitan areas around the world has resulted in the emergence of a persistent problem, which is traffic bottlenecks. This has an effect on the infrastructure of the city and the quality of life of its people. According to [1], the annual economic burden of these traffic conductions is tremendous. It ranges into the hundreds of billions of dollars. This is because of decreased productivity, increased fuel expenditures, and raised carbon emissions. Traditional traffic management systems, which are mainly pre-set timers, have been shown to be insufficient in managing the complex and ever-changing nature of modern urban Traffic Flow (TF). These outdated methods are unable to adjust to the ever-changing conditions of the traffic. The effective utilisation of highways is further exacerbated by these instances, which frequently make congestion more severe. The rapid development of the related problems will be predicted in response to the significant boost in vehicle traffic. There is a critical need for a more intelligent, more adaptable approach to controlling TF. Better TF, decreased congestion, and more environmentally conscious cities are all potential results of data mining and ML's ability to provide real-time, dynamic solutions [2]. The future potential of these advancements can be fulfilled by setting them to use.

On an international scale, cities have been developing at a rapid pace, resulting in traffic congestion that has an impact on services provided by the government and the standard of living conditions for residents. According to [3], the annual

expense of traffic congestion is hundreds of billions of dollars. Lower results, higher fuel costs, and more significant greenhouse gases are to responsibility. The variability and complexity of today's city traffic cannot be controlled by conventional Traffic Control Systems (TCS) that rely on schedules that are set. The failure of many of these outdated technologies to adapt to changing traffic conditions causes jams, which contribute to roadway loss [4]. The corresponding issues are going to increase significantly as the number of vehicles increases at an alarming rate. Quickly, a TCS that is both greater in sophistication and more adaptable is required. Better TF, less congestion, and safer cities are all potential results of the use of data mining and ML [5]. The solutions mentioned earlier can be given by using these technological advancements.

With Traffic Congestion Prediction (TCP), smart TF control has advanced. This lets TF systems act, not react. Accurate foresight allows control mechanisms to change signal timings, redirect traffic, or give drivers real-time smartphone advice [6]. Intelligent systems anticipate and prevent congestion, eliminating existing bottlenecks and preventing new ones. This proactive approach improves road infrastructure usage, which streamlines flow and reduces urban traffic difficulties. Various methods have been used to predict and control traffic congestion. These methods include time-series analysis and more advanced machine learning algorithms like Random Forests (RF), Support Vector Machines (SVM), and Neural Networks (NN). Time-series models can anticipate future traffic conditions using historical data but often cannot account for dynamic variables like weather, special events, or road construction [7]. The more flexible ML algorithms may take into consideration multiple variables at once, generating accurate real-time projections. Traffic patterns are complex and constantly changing; thus, even these advanced models have their limits [8].

Predictive TCS is considerably more effective when it is implemented through the utilization of a hybrid modelling approach. These approaches integrate the most advantageous aspects of many methods. Traditional models frequently perform admirably in certain circumstances but do not perform as well when applied to scenarios such as traffic in the real world [9]. When it comes to estimating traffic, hybrid models combine the most compelling aspects of many different approaches. These produce a tool that is both more dependable and versatile. These models can readily combine statistical methods with machine learning algorithms, enabling a more in-depth understanding of data from the past and data that is occurring at the present moment. There has been an important development in the construction of smart cities that has been caused by the use of Convolutional Neural Networks (CNNs) and Graph Neural Networks (GNNs). This is particularly true concerning the management of traffic in real-time. The combination of CNNs, which are well-known for their ability to extract spatial features from visual data [10], and Generative Neural Networks (GNNs), which are outstanding at modelling relational data, results in the creation of a strong method for comprehending and managing TF [11]. It would be feasible for these two adaptable neural network models to make effective use of both the complex network of city roadways as well as the detailed images of space. This results in the creating of a comprehensive and ever-evolving system for weather forecasting and traffic management. This mixed model makes it simpler to forecast traffic conjunctions [12] and improves route planning and traffic signal regulation. It contributes to achieving the primary objective of intelligent transportation systems (ITS). In order to effectively manage the complexity of city traffic, it is essential to implement such an integrated framework that is driven by artificial intelligence. This could result in less congestion and improved overall traffic efficiency in smart cities.

The paper is structured as follows: **Section 2** presents the literature study, **Section 3** presents the methodology, **Section 4** presents the experiment analysis, and **Section 5** concludes the work.

II. LITERATURE REVIEW

The [13] conducted a comprehensive analysis of 165 papers that discussed the application of data mining and machine learning to traffic management. The authors bring to the attention that there are no established procedures in the industry. Researchers, corporations that develop traffic software, and government officials can all benefit from having access to this information. This review demonstrates how essential it is to devise novel approaches to managing traffic—a comprehensive source of information and a catalyst for further investigation.

In [14] conducted a study in 2019 that investigates the challenges associated with intelligent TCS. These challenges include what to do with complex data and requirements that occur in real-time. The researchers aim to develop a control system that is hosted in the cloud and uses DL to forecast how traffic will move and become backed up in the near future. Additionally, their model uses intelligent optimization techniques for real-time control, and simulations demonstrate that it is effectively implemented.

ConvLSTM is a hybrid Convolutional Long Short-Term Memory (LSTM) model that may be used to improve travel time and Origin-Destination (OD) flow estimates. This is another proposal of [15], which was published in 2019. Their model utilized a grid-based approach to demonstrate OD. Because time slots are used to divide the day into sections, it is much simpler to forecast the flow of OD in both the spatial and temporal dimensions. The study demonstrates that the model is superior to other models in terms of its ability to forecast the overall taxi OD flow around the city by utilizing actual taxi data. A study that was conducted in [16] utilized Support Vector Regression (SVR) to develop a more accurate model for TPM in the short term. In the experiments, the error rate was only 3.22%, and the accuracy of the forecasts improved significantly at times when there were many people. The study also improves methods of counting pedestrians, which suggests that feature computations could be enhanced even further in order to provide results that are even more precise.

Many studies have been conducted recently about applying GNNs for the purpose of TPM, particularly concerning Intelligent Transportation Systems (ITS). This is because GNNs are naturally adept at comprehending the intricate ways traffic systems are interdependent over space and time. A comprehensive analysis of the development of GNNs for TPM was carried out by Jiang et al. in the year 2023. They boosted these types of models because they functioned better with graph-structured traffic data and produced highly precise forecasts. The latest developments highlight how study utilises open-source datasets and programming tools to interact with people and develop novel concepts.

GNN methods and Open-Government Data (OGD) have been studied for predicting TF incidents [17]. On traffic information in real time OGD, the Temporal Graph Convolutional Network (TGCN) and Diffusion Convolutional Recurrent Neural Network (DCRNN) prototypes superior to the Historical Average and Autoregressive Integrated Moving Average (HAAIMA). It also demonstrates how OGD helps traffic models for prediction. GNNs are highly successful at studying an intricate network models. "STG4Traffic" analyses Spatial-Temporal GNN for TPM and evaluates how they perform. How to set up and function with GNNs, develop traffic prediction challenges, and define optimisation goals are all discussed thoroughly. The research presented here illustrates how essential it is to develop innovative graph patterns and software programmes for comprehending traffic networks.

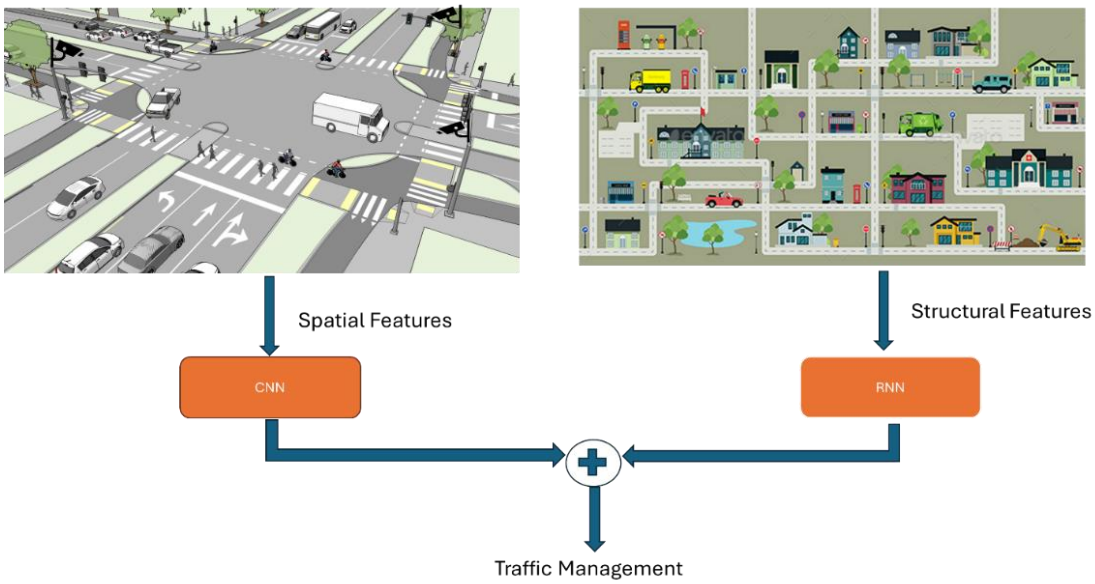

Fig 1. Architecture of Hybrid Model

The literature reviewed in this section highlights GNN developments and current work in TCS decision-making processes. The following Intelligent Transportation System (ITS) demands more intelligent, accurate, and better-performing TPMs, as researchers recommend. The emphasis placed on open-source resources and the incorporation of OGD demonstrates that the challenges associated with TPM can be overcome through collaborative efforts. New research opportunities and practical applications in smart city infrastructures are created as a result of this.

Fig 1 explains the proposed model, which analyzes spatial and structural features using CNN and RNN. The proposed model is explained in detail in the following sections.

III. METHODOLOGY

Convolutional Neural Network

A CNN is one of the most significant advances that has been made in the field of deep learning. However, it was initially conceived in the 1990s and utilized for the purpose of handwriting detection [18]. CNN's victory in the ImageNet competition in 2012, in which it surpassed the top results for the first time, was a significant turning point in the company's history. In this section, an introduction is provided to the four most important parts of CNN. Loss function, convolution, pooling, and total link are the four components that make up this system. Generally speaking, Convolutional Neural Networks, often known as CNNs, are among the most significant artificial intelligence components currently available, particularly in computer vision. The ability of these deep learning models to automatically discover and learn hierarchical patterns in visual data is awe-inspiring. This has impacted how people in various industries investigate and comprehend multimedia. A CNN has been organised inside to approximate how the human eyes perform. There are many layers in the system, and they all do something distinct.

Furthermore, the convolutional layers form the first layers, applying filters to the receiving images and generating feature maps that better display details like forms, textures, and edges. Next are the convolutional layers and the pooling layers. Because they reduce spatial voting, algorithms have a simpler time storing and integrating feature networks. They further decrease the size of the image. As time passes, the framework's predictability reduces due to activation functions,

particularly the Rectified Linear Unit (ReLU). Because of this, the algorithm may detect complicated trends in the data. In the final phases of a CNN, fully connected layers are typical. Tasks like classification and prediction are performed in these layers using the high-level, eliminated features identified in previous layers.

Employing this systematic method, CNNs can analyse and interpret massive volumes of graphical information. **Fig 2** illustrates the CNN layout. A CNN usually comprises numerous convolution layers. An activation function precedes every layer of convolution—a flattening layer and an output layer next.

Convolution

Affine transformation, as described by [19], is the process that neural networks perform consistently: A 2D or 3D matrix vector is supplied as input, then multiplied by a matrix to produce an output. Prior to the production being sent through a non-linearity, a bias vector is often added to the output that is being processed. One can express the concept of picture convolution in a broad fashion by using EQU (1).

$$W \circ M(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} W(i,j)M(x+i,y+j) \tag{1}$$

where W is the convolution filter kernel, and M is the target image. Every element of the filter kernel is considered by $-N \leq i \leq N$ and $-N \leq j \leq N$. The values x and y are the image length and width, respectively.

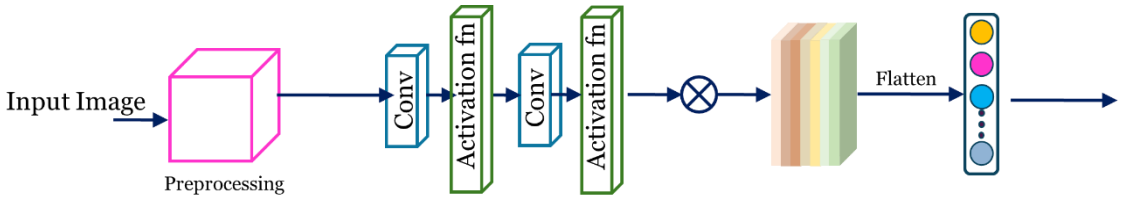

Fig 2. Convolutional Neural Network

Pooling

The pooling layers in the deep learning system [20] ensure that the input remains unchanged even after it is somewhat translated. In the majority of cases, maximum pooling and average pooling are discovered. For max pooling to function, the input is first segmented into patches that do not overlap, and then the highest value from each patch is transmitted. Avgpooling is the process that is used to send out the average value of each patch. Over the course of the input picture, a window is moved, and the contents of the window are then fed into a pooling function. In this manner, pooling is applied. Because the linear combination specified by the kernel is transformed into a distinct function when you share, it functions in a manner that is quite similar to that of convolution. An illustration of the procedure for determining the form of the pooling output can be seen in EQU (2).

$$SP = \left[\frac{M-P}{S}\right] + 1 \tag{2}$$

where SP is the pooling output spatial shape, P is the pooling window size, M is the target image, S is the pooling stride and ‘‘[ ]’’ means floor operation.

Full Connection

CNN, which is particularly successful in identifying and classifying pictures for computer vision, is an essential component that includes a fully connected layer [Liu et al. 2019]. The first step in the CNN method is to perform convolution and pooling, which involves dividing the image into features and examining each feature separately. When this procedure is complete, the output is sent into a neural network structure that is fully connected, which is responsible for making the ultimate classification choice. The generic formulation of full connection can be found in EQU (3), which is displayed below.

$$Z = W^T X + B \tag{3}$$

It is important to note that Z represents the features that need to be sorted, X represents the picture feature, W represents the feature matrix, and B represents the bias matrix. Every node in the full connection layer is connected to every other node in the upper layer. This applies to the whole network. After that, the parts that were extracted from the front edge are assembled by utilizing these connections. The fact that the general completely connected layer is fully connected is why it contains the most significant number of factors.

Loss Function

Cross-entropy (CE) loss is the most common type of loss function used in picture classification [21-22]. For example, this function comes right after the softmax function and takes in both the result of the softmax function and the accurate label.

When figuring out the CE loss, the CE is found by adding up the real and predicted labels for each pair. The formula is shown by the EQU (4) in its most basic form.

$$CE(p,q) = -\sum_x p(x)\log q(x) \tag{4}$$

where CE is the cross entropy of p and q, p(x) is the predicted classification label, and q(x) is the true label.

Graph Neural Networks

Node Representations: In a graph $G = (V, E)$, where E represents the set of edges, V represents the set of nodes, each node $v \in V$ has an initial feature vector $\underline{X_q}$. The GNN learns a representation (embedding) $\vec{H_2}$ for each node that captures the features and structure of its neighbourhood.

Message Passing: GNNs pass messages between nodes, updating each node by collecting messages from its neighbors. For each node v, the update at the $\bar{k}$-th iteration (or layer) is computed as in EQU (5).

$$H_v^{(k)} \equiv \text{UPDATE}^{(k)}\left(H_v^{(k-1)}, \text{AGGREGATE}^{(k)}\left(\left\{H_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)\right) \tag{5}$$

where $H_\tau^{(k-1)}$ is the representation of node v at layer $k-1$, $\underline{\mathcal{N}}(v)$ denotes the set of neighbors of $\bar{v}$, and Aggregate and update are methods that aggregate and update the features of a node, respectively.

Aggregation: The AGGREGATE function gets features from the node's neighbors and combines them in a single vector. A common choice for this function is the mean AGGREGATOR given in EQU (6).

$$\text{AGGREGATE}^{(k)}\left(\left\{H_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) = \frac{1}{\mathcal{N}(v)|}\sum_{u \subset \mathcal{N}(v)} H_u^{(k-1)} \tag{6}$$

This step makes sure that the data from a node's close neighbors is added to the new version.

Update: The UPDATE function then takes the node's current representation and adds the neighbourhood details to make a UPDATE representation. A simple UPDATE mechanism could involve a non-linear transformation, as in EQU (7)

$$H_v^{(k)} = \sigma\left(W^{(k)} \cdot \left(H_v^{(k-1)} \parallel \text{AGGREGATE}^{(k)}\right) + b^{(k)}\right) \tag{7}$$

where $W^{(k)}$ and $b^{(k)}$ are trainable parameters at layer k. σ denotes a non-linear activation function (*e.g.*, ReLU), and || represents concatenation.

Readout Layer: A graph-level description is generated through the integration of the end node symbols using an READOUT function. Tasks like network regression and classification are helped by this. Here is an EQU (8), which denotes the READOUT function.

$$H_G = \text{READOUT}\left(\left\{H_v^{(K)} : v \in V\right\}\right) \tag{8}$$

where K is the sum of iterations or layers and H_G is the end resultant graph representation. Taking the mean of all the node embeddings might result in a simple READOUT function.

IV. PROPOSED METHODOLOGY

Accurate smart city traffic data collection and analysis is essential for real-time TCS, which combines CNNs and GNNs. Applying activation functions like Rectified Linear Units (ReLU) convolutional and pooling layers, the CNN module is trained to extract spatial information from traffic data. The aim is to reduce the download size of the images while maintaining every significant detail intact. Two convolutional layers comprise the CNN structure, with a pooling layer and a ReLU activation function following each.

The framework results in fully connected layers for feature analysis. The inclusion of the GNN feature enables the management of urban traffic networks' topology data. For identifying the link between data that exists between nodes, graph convolutional layers are used. The crossings, road segments, and boundaries illustrated here represent in for real roads. The GNN focuses on graph convolution techniques—which focus on the idea of message passing—to integrate data across various neighbourhoods. To improve the model's concept of linearity, GNNs employ an activation function called ReLU.

The GNN's features of structure and the CNN's spatial data are incorporated in a hybrid layer. The feature vectors of the two networks are merged to generate a complete set of features, which is achieved using conjunction. **Fig 3** demonstrates the fundamental layout of the hypothesised Hybrid CNN-GNN.

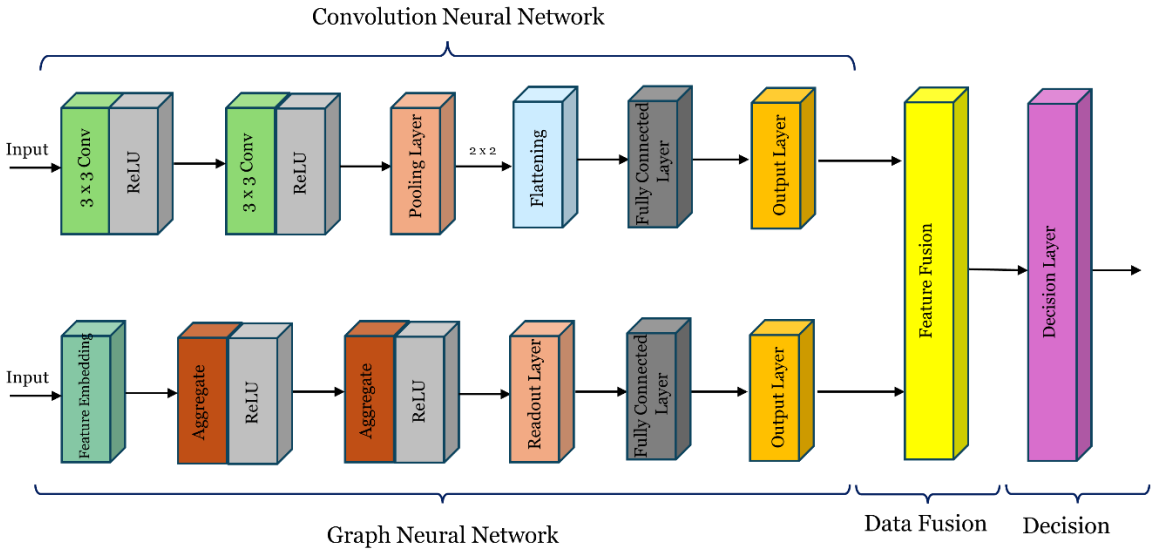

Fig 3. Hybrid CNN-RNN Model

Convolution Neural Network: The CNN for traffic image analysis has two convolutional layers. Each layer could have an increasing number of filters, starting from 32 in the first layer and doubling in subsequent layers to capture more complex features. The kernel size for these layers is 3x3. ReLU (Rectified Linear Unit) is the standard choice for activation in CNNs due to its efficiency and effectiveness in introducing non-linearity, allowing the model to learn complex patterns. ReLU is used as an activation function after convolution. Following each convolutional layer with a pooling layer, the max pooling with a 2x2 filter is used to reduce dimensionality and computational load while retaining essential features. After the convolutional and pooling layers, the architecture includes one or more fully connected layers to interpret the features extracted by the CNN.

Graph Construction and Feature Integration: Given a traffic network graph $G = (V, E)$, where V denotes the set of nodes (intersections, road segments) and E represents the edges (pathways between nodes), each node $v \in V$ and edge $e \in E$ is associated with a feature vector X_v and X_e, respectively. Integrating spatial features from CNNs into this graph can mathematically represent an EQU enhancement of the node feature vectors (9).

$$X_v' = X_v \oplus CNN(I_v) \tag{9}$$

where X_v' is the enhanced node feature vector, X_v is the original node feature vector, $CNN(I_v)$ represents the CNN-extracted features for the visual data I_v corresponding to node v, and $\oplus$ denotes the feature fusion operation, such as concatenation.

GNN Architecture for Feature Processing: The GNN processes the graph through a series of layers, where each layer l updates the features of node v based on its features and the aggregated features from its neighbours $\mathcal{N}(v)$. The feature update mechanism for node v in layer $l + 1$ can be expressed as EQU (10) and EQU (11)

$$H_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \text{AGGREGATE}\left(\downarrow^{(l)} : u \in \mathcal{N}(v)\right) + b^{(l)}\right) \tag{10}$$

$$H_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \text{AGGREGATE}\left(\left\{H_u^{(l)} : u \in \mathcal{N}(v)\right\}\right) + b^{(l)}\right) \tag{11}$$

where $H_v^{(l)}$ is the feature vector of node v at layer $l, W^{(l)}$ and $b^{(l)}$ are the trainable weight matrix and bias vector for layer l, σ is a non-linear activation function (e.g., ReLU), and AGGREGATE is a function aggregating features from the neighbors of v.

The $AGGREGATE$ function is the mean aggregator, EQU (12).

$$\text{AGGREGATE}\left(\left\{H_u^{(l)} : u \in \mathcal{N}(v)\right\}\right) = \frac{1}{|\mathcal{N}(v)|} \sum_{u \subset \mathcal{N}(v)} H_u^{(l)} \tag{12}$$

This function computes the average of the feature vectors of the neighbors of v, capturing the local structural information around v.

The final output for node v after L layers of GNN processing can be used to predict traffic conditions, such as congestion levels C_v, at node v, EQU (13).

$$C_v = f\left(H_v^{(L)}\right) \tag{13}$$

Here, f is a prediction function. A fully connected layer is followed by a SoftMax and $H_v^{(L)}$ is the final representation of node v after L layers of GNN processing. The GNN will be trained to minimize a loss function $\mathcal{L}$, which measures the difference between the predicted traffic conditions $\hat{C}_v$ and the actual conditions C_v, EQU (14).

$$\mathcal{L} = \sum_{v \subset V} \ell\left(\hat{C}_v, C_v\right) \tag{14}$$

where ℓ is a suitable loss function, mean squared error. The GNN component captures and processes the structural complexities of the traffic network. It leverages physical connectivity and dynamic traffic states to inform real-time traffic management decisions in smart cities.

CNNs effectively process the structural and relational data inherent in urban traffic networks. The GNN model captures not only the spatial relationships between nodes in the traffic network (e.g., intersections, roads) but also the temporal dependencies of TF over time. This dual focus allows a more nuanced understanding of traffic patterns, facilitating accurate predictions of traffic conditions and potential congestion.

For a Hybrid GNN-CNN model tailored for real-time traffic management in smart cities, the algorithm would focus on leveraging both the CNN's capability to extract spatial features from traffic images and the GNN's strength in processing the structural data of traffic networks.

Algorithm: *Hybrid GNN-CNN for Traffic Management*

Input:

- I : Traffic camera images across different locations and times.
- $G(V, E)$: Graph representing the traffic network, where V denotes vertices (intersections, road segments) and E denotes edges (pathways between vertices).
- F_v : Initial feature vectors for each vertex $v \in V$, including TF, speed, and density.
- N : Number of training epochs.
- lr : Learning rate.
- M Number of neighbourhoods or zones for the GNN model.

Output:

- $S_{t,\text{ Hybrid}}$: Predicted traffic conditions for each vertex in the graph at time t.

Parameter Initialization:

Initialize CNN model parameters Θ_{CNN}, GNN model parameters Θ_{GNN} and initialize hybrid model parameters Θ_{Hybrid}.

Steps:

Preprocessing:

- For each image i in I, preprocess i to get uniform size and scale. The processed images are given as I'.

Spatial Feature Extraction with CNN:

- For each i' in I', use CNN to extract spatial features: $F_{i',\text{ spatial}} = CNN(i'; \Theta_{CNN})$.

Graph Construction and Feature Augmentation:

- Construct graph $G(V, E)$ from traffic network data.
- For each vertex $v \in V$, augment F_v with corresponding $F_{i',\text{ spatial}}$ to form enhanced features F'_v.

Structural Feature Processing with GNN:

- Use GNN to process $G(V, E)$ with end features F'_v, updating vertex representations:

$$H_v = GNN(F'_v; \Theta_{GNN}).$$

Hybrid Model Integration:

- Integrate CNN and GNN outputs using a fusion technique (e.g., feature concatenation, weighted sum) to form a comprehensive traffic condition representation for each vertex v :

$$S_{v,\text{ Hybrid}} = \text{Fusion}\left(H_v; \Theta_{\text{Hybrid}}\right).$$

Training:

- For each epoch $n \in \{1, \ldots, N\}$:
- Update $\Theta_{CNN}, \Theta_{GNN}$, and Θ_{Hybrid} by minimizing a loss function $L\left(S_{v,Hybrid}, S_{v,\text{ actual}}\right)$, where $S_{v,\text{ actual}}$ is the actual traffic condition.
- Adjust lr as needed based on optimization strategy.

Prediction:

- For real-time traffic images and network data, repeat steps 1-5 to predict $S_{t,\text{ Hybrid}}$ for each vertex at time t.

The notations used in algorithms are defined as follows:

- $CNN(i'; \Theta_{CNN})$: Function representing the CNN with parameters Θ_{CNN} applied to image i'.

- $GNN(F_v'; \Theta_{GNN})$: Function representing the GNN with parameters Θ_{GNN} applied to enhanced vertex features F_v'.
- Fusion $\left(H_v; \Theta_{\text{Hybrid}}\right)$: Fusion technique integrating CNN and GNN outputs with hybrid model parameters Θ_{Hybrid}.
- $L\left(S_{v,\text{ Hybrid}}, S_{v,\text{ actual}}\right)$: Loss function comparing predicted traffic conditions $S_{v,Hybrid}$ against actual conditions $S_{v,\text{ actual}}$.

This algorithm outlines the integrated approach of using CNN for spatial feature extraction and GNN for structural data processing, culminating in a hybrid model that leverages both for enhanced real-time traffic management in smart cities. The hyperparameters for the GNN and CNN components of the model:

Table1. CNN Hyperparameters

CNN	
Hyperparameter	Values
Convolutional Layers	3
Kernel Size	3 X 3 for all layers
Strides	1 for all convolutional layers
Pooling Size	2 X 2 soft max pooling
Batch Size	32
Learning Rate	0.0005
Epochs	50
Activation Function	ReLU for intermediate layers, SoftMax for output layer
Dropout Rate	0.5
Loss Function	RMSE

Table 2: GNN hyperparameters

GNN	
Hyperparameter	Values
Graph Convolutional Layers	2
Aggregation Function	Mean aggregation
Batch Size	32
Learning Rate	0.001
Epochs	50
GNN Hidden Units	64
Activation Function	ReLU for intermediate layers, SoftMax for node classification
Dropout Rate	0.2
Loss Function	RMSE

Table 3: Hybrid CNN- GNN hyperparameters

Hybrid CNN-GNN	
Hyperparameter	**Values**
Fusion Technique	Concatenation
Combined Layers	2 Fully Connected Layers
FC Layer Units	64
Final Activation Function	SoftMax
Optimization Algorithm	Adam
Early Stopping	Patience: 10 epochs

Table1 Show in CNN Hyperparameters. **Table 2** Show in GNN hyperparameters. This **Table 3** Show in outlines the hyperparameters of your hybrid GNN-CNN model and provides a clear and concise overview of the model's configuration.

V. EXPERIMENTAL ANALYSIS

The research was done on a computer system running Linux and used an AMD Ryzen 9 3950X CPU and an NVIDIA Tesla V100 GPU. The whole software stack was made up of Python and TensorFlow 2.5 software. Between May 1, 2022, and April 30, 2023, traffic information was gathered for a year. This produced a file with 242,034 records. From the dataset, 80% of the data is partitioned as training data and 20% as testing dataset.

From **Table 1,** you can see the parameters that were used to train the suggested model. In particular, the analysis was done for three different time periods in the dataset: i) 15 min, ii) 30 min, and iii) 45 min. To find out if the hybrid LSTM-BSTS model can help predict traffic, it is measured using three main criteria, which are listed below:

Definitions:
$S_{t,predicted}$ The predicted congestion score at time t.
$S_{t,actual}$ The actual observed congestion score at time t.

Mean Absolute Error (MAE): The MAE is computed as EQU (16).

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{N} \left| S_{t,predicted} - S_{t,actual} \right| \tag{16}$$

MAE calculates the average size of the discrepancies between forecasted and real congestion scores without considering their direction. This technique presents an easy and precise method for assessing the accuracy of the model's predictions.

Root Mean Square Error (RMSE): The RMSE is defined as EQU (17).

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \left(S_{t,predicted} - S_{t,actual} \right)^2} \tag{17}$$

By scaling them prior averaged values, the RMSE provides greater weight to more significant errors. When significant errors can be particularly aggravating, this comes in helpful.

Mean Absolute Percentage Error (MAPE): MAPE is calculated using the formula:

$$\text{MAPE} = \frac{100}{N}\sum_{t=1}^{N} \left| \frac{S_{t,predicted} - S_{t,actual}}{S_{t,actual}} \right| \tag{18}$$

In order to clarify the scale of the errors when compared to the actual traffic results, the MAPE presents the outcome's error level as expressed as a percentage.

Performance Comparison
In order to evaluate how well the recommended hybrid LSTM-BSTS model can predict congestion, it is compared to various baseline models, such as:

1. Auto-Regressive Integrated Moving Average model (ARIMA)
2. RF
3. LSTM
4. GNN
5. CNN
6. Hybrid GNN-CNN

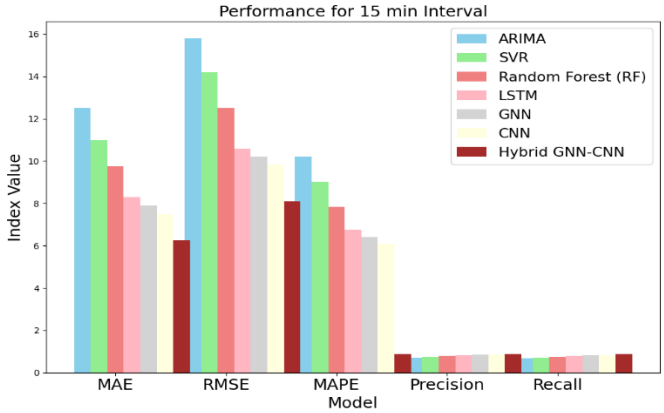

Fig 4. Performance on a 15-Minute Interval

The Hybrid GNN-CNN model is superior to different approaches, as demonstrated in **Fig 4** illustrates the traffic prediction for 15-minute intervals. The ARIMA model's MAE of 12.5 indicates a significant improvement in predicting accuracy, which is outperformed by the proposed Hybrid GNN-CNN with its 6.25 MAE. This dramatically improves forecast accuracy.

With an RMSE of 8.1, the Hybrid GNN-CNN model again outperformed the other model. This was much lower than the ARIMA model's 15.8 RMSE. This score highlights the model's ability to reduce significant errors and its outstanding predictive consistency. MAPE is a percentage of error of the actual values. A lower MAPE suggests better measurement accuracy. With 0.89% MAPE, the Hybrid GNN-CNN model outperforms compared to the other models. This contrasts

with the ARIMA model's 10.2% MAPE and exceeds CNN's 6.1%. The Hybrid GNN-CNN model's precision and recall of 0.89 and 0.87 show it performs better. These figures show that the model strikes an outstanding balance between TP and FN. This is far better than the ARIMA model, which scored 0.72 in precision and 0.68 in recall.

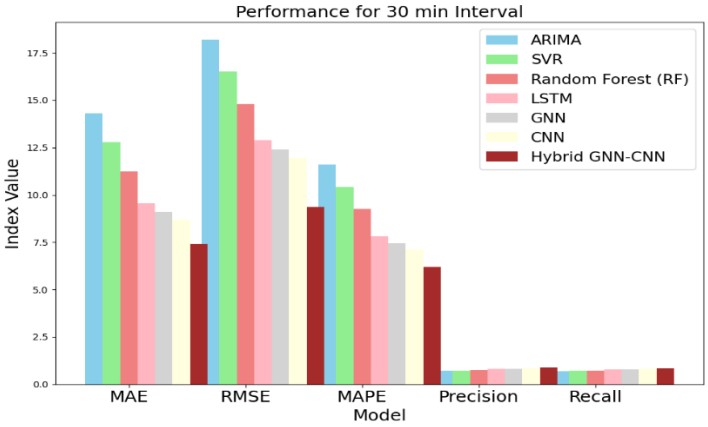

Fig 5. Performance on 30-Min Interval

A multilevel evaluation of the 30-minute data collection of the Traffic Prediction Model (TPM) is presented in **Fig 5.** The Hybrid GNN-CNN approach is superior to the other models in TFP with a Mean Absolute Error (MAE) of 7.4. The ARIMA system's MAE was 14.3; the present model is a great deal superior. Root Mean Square Error (RMSE), a statistic emphasising higher-level errors, is 9.35 and the smallest for the Hybrid GNN-CNN model. With a RMSE of 18.2, the ARIMA framework is less effective in mitigating prediction errors. An additional essential proof of error compared to actual figures is MPE or MAPE. Hybrid GNN-CNN approaches have a lesser MAPE of 6.2% than ARIMA algorithms, which increase by approximately a double value, at 11.6%. The most significant precision and recall, at 0.89 and 0.86, accordingly, were obtained by the hybrid GNN-CNN model. Tests like these demonstrate that the framework is helpful for predicting traffic and can spot when conditions are optimum for transportation. With recall and accuracy of 0.67 and 0.7, accordingly, the ARIMA approach falls short when compared to the rest of the models, demonstrating that it needs development in terms of sensitivity and precision.

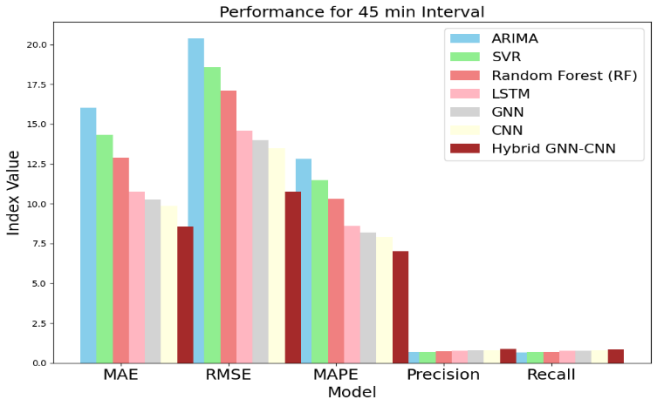

Fig 6. Performance on 45-Min Interval

Fig 6 shows that the 45-minute TPM comparison generated significant and noteworthy results. The hybrid GNN-CNN model outperformed all other models in all categories. The lowest Mean Absolute Error (MAE) of 8.55 indicated substantial TF estimation accuracy. Hybrid GNN-CNN again outperformed its competitors in Root Mean Square Error (RMSE), which penalizes more significant errors more harshly, with 10.75, indicating its robustness to prediction variance. RMSE punishes more essential errors. A MAPE of 7% demonstrates excellent forecast accuracy in proportion to actual values, while the ARIMA model has 12.8%. This strengthens the Hybrid GNN-CNN model's 7% MAPE advantage. Precision and recall measures, which assess model categorization, showed a similar tendency. The Hybrid GNN-CNN model had the highest accuracy (0.87) and recall (0.85) ratings, demonstrating its prediction relevance and completeness, unlike the ARIMA model, which got 0.68 precision and 0.65 recall.

VI. CONCLUSION AND FUTURE WORK

Studies indicate that the Hybrid GNN-CNN method improves ARIMA and Machine Learning (ML) techniques including SVR, Random Forest (RF), LSTM, and each GNN and CNN. ARIMA is a conventional model. The high degree of precision has been demonstrated by the Hybrid GNN-CNN model's periodic reduced MAPE, RMSE, and MAE scores over

a period of time. The Hybrid GNN-CNN system is reliable and precise, which is essential for real-time traffic management decision. Good accuracy as well as recall show that the framework is able to recognise practical traffic situations. This guarantees the framework's precise forecasting data is accurate and beneficial. The implementation of adaptive Traffic Control Systems (TCS) is required to respond to evolving traffic conditions to minimise travel congestion. The present research highlights the value of the hybrid CNN+GNN in order to take benefit of their distinct features. GNNs assess temporal and structure-based information from smart city transportation systems, while CNNs derive spatial features from images. Combining the two methods into a Hybrid GNN-CNN design provides a powerful tool to measure traffic flow's heterogeneity. This practical method might significantly affect future urban traffic control generations.

Data Availability
No data was used to support this study.

Conflicts of Interests
The author(s) declare(s) that they have no conflicts of interest.

Funding
No funding agency is associated with this research.

Competing Interests
There are no competing interests.

References:
[1] E. Weisstein, “Making MathWorld,” The Mathematica Journal, vol. 10, no. 3, Aug. 2007, doi: 10.3888/tmj.10.3-3.
[2] L. Xu, A. Krzyzak, and C. Y. Suen, “Methods of combining multiple classifiers and their applications to handwriting recognition,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 22, no. 3, pp. 418–435, 1992, doi: 10.1109/21.155943.
[3] M. Xin and Y. Wang, “Research on image classification model based on deep convolution neural network,” EURASIP Journal on Image and Video Processing, vol. 2019, no. 1, Feb. 2019, doi: 10.1186/s13640-019-0417-8.
[4] P. N. Druzhkov and V. D. Kustikova, “A survey of deep learning methods and software tools for image classification and object detection,” Pattern Recognition and Image Analysis, vol. 26, no. 1, pp. 9–15, Jan. 2016, doi: 10.1134/s1054661816010065.
[5] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, “Learning Pooling for Convolutional Neural Network,” Neurocomputing, vol. 224, pp. 96–104, Feb. 2017, doi: 10.1016/j.neucom.2016.10.049.
[6] S. Liu et al., “Magnetic Anomaly Detection Based on Full Connected Neural Network,” IEEE Access, vol. 7, pp. 182198–182206, 2019, doi: 10.1109/access.2019.2943544.
[7] Y. Zhang, K. Shang, Z. Cui, Z. Zhang, and F. Zhang, “Research on Traffic Flow Prediction at Intersections Based on DT-TCN-Attention,” Sensors, vol. 23, no. 15, p. 6683, Jul. 2023, doi: 10.3390/s23156683.
[8] J. Cheng, G. Li, and X. Chen, “Research on Travel Time Prediction Model of Freeway Based on Gradient Boosting Decision Tree,” IEEE Access, vol. 7, pp. 7466–7480, 2019, doi: 10.1109/access.2018.2886549.
[9] N. O. Alsrehin, A. F. Klaib, and A. Magableh, “Intelligent Transportation and Control Systems Using Data Mining and Machine Learning Techniques: A Comprehensive Study,” IEEE Access, vol. 7, pp. 49830–49857, 2019, doi: 10.1109/access.2019.2909114.
[10] V. Myilsamy, S. Sengan, R. Alroobaea, and M. Alsafyani, “State-of-Health Prediction for Li-ion Batteries for Efficient Battery Management System Using Hybrid Machine Learning Model,” Journal of Electrical Engineering & Technology, vol. 19, no. 1, pp. 585–600, Jun. 2023, doi: 10.1007/s42835-023-01564-2.
[11] D. Zhang, G. Luo, and J. Li, “Traffic Spatial-Temporal Prediction Based on Neural Architecture Search,” Proceedings of the 18th International Symposium on Spatial and Temporal Data, Aug. 2023, doi: 10.1145/3609956.3609962.
[12] P. Brimos, A. Karamanou, E. Kalampokis, and K. Tarabanis, “Graph Neural Networks and Open-Government Data to Forecast Traffic Flow,” Information, vol. 14, no. 4, p. 228, Apr. 2023, doi: 10.3390/info14040228.
[13] W. Jiang, J. Luo, M. He, and W. Gu, “Graph Neural Network for Traffic Forecasting: The Research Progress,” ISPRS International Journal of Geo-Information, vol. 12, no. 3, p. 100, Feb. 2023, doi: 10.3390/ijgi12030100.
[14] W. Li, X. Wang, Y. Zhang, and Q. Wu, “Traffic flow prediction over muti-sensor data correlation with graph convolution network,” Neurocomputing, vol. 427, pp. 50–63, Feb. 2021, doi: 10.1016/j.neucom.2020.11.032.
[15] M. Bai, Y. Lin, M. Ma, P. Wang, and L. Duan, “PrePCT: Traffic congestion prediction in smart cities with relative position congestion tensor,” Neurocomputing, vol. 444, pp. 147–157, Jul. 2021, doi: 10.1016/j.neucom.2020.08.075.
[16] M. Gollapalli et al., “A Neuro-Fuzzy Approach to Road Traffic Congestion Prediction,” Computers, Materials & Continua, vol. 73, no. 1, pp. 295–310, 2022, doi: 10.32604/cmc.2022.027925.
[17] W. Ju et al., “A Comprehensive Survey on Deep Graph Representation Learning,” Neural Networks, vol. 173, p. 106207, May 2024, doi: 10.1016/j.neunet.2024.106207.