CTU Journal of Innovation and Sustainable Development
ISSN 2588-1418 | e-ISSN 2815-6412

DOI:10.22144/ctujoisd.2025.059

Traffic flow prediction using adaptive graph convolutional networks and long short-term memory

Phan Thi Ngoc Han*, Ho Quoc Ngoc, Nguyen Quoc Trung, and Nguyen Thanh Binh

Faculty of Information Technology, Ho Chi Minh City University of Foreign Languages - Information Technology, Viet Nam

**Corresponding author (hanptn@huflit.edu.vn)*

Article info.

Received 15 Jul 2025
Revised 18 Aug 2025
Accepted 3 Oct 2025

Keywords

Graph convolutional networks, long short-term memory, traffic flow prediction, YOLO

ABSTRACT

Traffic congestion is becoming an increasingly serious and challenging issue in major urban areas. This problem not only causes a waste of time and increased fuel consumption but also contributes to environmental pollution and deterioration of residents' quality of life. In this study, a new method of predicting the average speed reported by traffic sensors across the city was proposed. In this method, we make the most of two core models: Graph Convolutional Networks and Long Short-Term Memory. The YOLO model is used to analyze images and video during data collection. By leveraging Graphe Convolution Networks ability to capture spatial information, Long Short-Term Memory capacity to model temporal dynamics, and YOLO's strength in visual object detection, our integrated framework enhances the accuracy of traffic flow predictions at specific locations and time intervals. This comprehensive approach aims to support real-world applications such as adaptive traffic light control, traffic planning support, and congestion alerts. The proposed method outperforms other methods on the Caltrans PeMS dataset.

1. INTRODUCTION

Traffic congestion is a pressing issue in many urban areas around the world, negatively impacting the economy, environment, and quality of life. The rapid growth in urban population, the increasing number of private vehicles, and underdeveloped transportation infrastructure have exacerbated congestion in many countries, including Viet Nam.

Moreover, traditional traffic control methods (signage, fixed-timing traffic lights, and direct traffic police intervention) have become less effective as vehicle volume continues to rise.

Traffic flow forecasting is very important because it helps manage and direct traffic effectively, reduce congestion, support infrastructure planning, enhance the experience of traffic participants, optimize transportation and logistics, reduce traffic accidents, and serve as the foundation for intelligent transportation systems in modern cities.

Therefore, the application of technology for real-time traffic flow prediction is essential to assist regulatory authorities and the public in optimizing travel plans, reducing congestion, and enhancing traffic efficiency.

Traffic forecasting is a crucial component of modern transportation systems, influencing various aspects of traffic management, safety, and urban planning. Accurate predictions of traffic conditions are essential for managing urban mobility, alleviating congestion, and enhancing road safety.

In the context of Intelligent Transportation Systems (ITS), forecasting technologies, particularly

machine learning and deep learning methodologies, have transformed how traffic conditions are predicted. Techniques such as LSTM networks and hybrid models have demonstrated superior capabilities in short-term traffic forecasting, effectively capturing the complex temporal and spatial dependencies of traffic phenomena (Ren et al., 2021; Tran et al., 2022).

Additionally, Gao et al. (2021) emphasize that accurate traffic speed forecasting is integral for providing real-time data, which can significantly improve the decision-making process for both traffic managers and individual travelers. When combined with historical data, such forecasting empowers travelers with predictive insights, allowing for better route choices and, in turn, reducing overall congestion (Awan et al., 2020). This predictive capability is pivotal in developing Ren et al. (2021) integrated transport solutions that cater to the evolving dynamics of urban road networks.

The importance of traffic forecasting extends beyond mere prediction. It encompasses safety enhancements, infrastructure planning, and the overall optimization of urban mobility solutions. As cities continue to grow and traffic conditions evolve, leveraging advanced forecasting techniques will be essential for sustaining efficient and safe transportation systems.

In this study, a new method of predicting the average speed reported by traffic sensors across the city was proposed. In this method, we combine two core models, GCN and LSTM. The YOLO model is used to analyse images and video during data collection. By leveraging GCN's ability to capture spatial information, LSTM's capacity to model temporal dynamics, and YOLO's strength in visual object detection, our integrated framework enhances the accuracy of traffic flow predictions at specific locations and time intervals. This comprehensive approach aims to support real-world applications such as adaptive traffic light control, traffic planning support, and congestion alerts.

This study is divided into five sections. The first section discusses the importance of traffic flow prediction, why it needs to be addressed, and proposes a method to solve the problem. The second section is the related works study. The third section proposed a method to solve the problem of predicting the average speed. The fourth section analyzes the collected dataset as well as the application results of the problem using this dataset. Finally, the conclusion is based on the obtained results and outlines directions for future research.

2. RELATED WORKS

Traffic-related research has been widely developed and expanded in recent years, especially in the area of predicting traffic flow in cities. This task is crucial for building an Intelligent Transportation System (ITS), which aims to address traffic-related problems efficiently. The advancement of artificial neural networks, particularly Graph Convolutional Networks (GCNs), has made the problem of traffic prediction more feasible. In the past, traffic prediction research often relied on traditional statistical methods. For instance, Chandra and Al-Deek (2009) employed the Auto Regression (VAR) model to extract data from historical records in order to predict present values; Wang et al. (2003) also utilized VAR for traffic analysis. Meanwhile, Kumar and Vanajakshi (2015) adopted a traditional time series forecasting method—ARIMA (Autoregressive Integrated Moving Average) — with the goal of predicting city traffic flow. A common limitation of these methods is the assumption of time series stationarity. However, traffic data often exhibits complex spatiotemporal characteristics, which means the stationarity assumption may not be satisfied. Although traditional statistical approaches such as ARIMA, VAR, and similar models have been widely used in time series forecasting tasks, they have not demonstrated high effectiveness when applied to real-world urban traffic systems. In this context, the rapid advancement of machine learning and deep learning has opened up new directions for addressing the challenges of traffic prediction more effectively. Manoel et al. (2009) applied Support Vector Regression (SVR) to exploit traffic data. Building on this, Random Forest models developed by Johansson et al. (2014) showed promising results in capturing temporal patterns through machine learning techniques. Furthermore, Markov models implemented by Zhang et al. (2018) demonstrated effectiveness in identifying traffic patterns from large volumes of historical traffic data. Although these methods generally perform better than statistical approaches due to their ability to capture nonlinear spatio-temporal correlations, their performance remains suboptimal when applied to large road networks with hundreds or even thousands of road links. With the remarkable success of deep learning techniques, there have been numerous efforts to apply deep learning to traffic flow prediction. With the rapid development of deep

learning, researchers have begun applying Recurrent Neural Networks (RNNs); Chung et al. (2014) and Knol et al. (2021) have utilized RNNs for time series prediction. Following that, with the improvement of RNNs, the Long Short-Term Memory (LSTM) network is considered a viable solution for traffic prediction due to its ability to capture long-term dependencies. Yang et al. (2019) and Zhao et al. (2017) demonstrated the effectiveness of LSTM networks. However, LSTM still treats traffic sequences from different road segments as independent data streams, ignoring spatial relationships, and therefore cannot optimize the prediction performance of the entire network. Zhu et al. (2020) combined RNN with GCN to address spatial limitations. However, RNNs suffer from vanishing or exploding gradient problems, therefore Zhao et al. (2019) proposed a T-GCN network combined with GRU to overcome the drawbacks of RNN, such as gradient explosion and vanishing, while integrating graph convolution to extract corresponding spatial and temporal feature, Yu et al. (2017) also proposed a graph convolutional network to extract spatial and temporal graph convolutional features. Although the above studies have achieved many positive results, there are still some challenges that need to be addressed. Notably, most current models assume the traffic network to be an undirected graph, whereas in reality, urban traffic networks are often directed, which significantly affects prediction accuracy.

In addition, the paper also touches on the topic of object detection, a core area in computer vision. Viola and Jones (2001) introduced a real-time face detection classifier. Continuing this development, Dalal et al. (2021) introduced an object detector using HOG features. The first appearance of the object detection model YOLO by Joseph et al. (2016) marked significant advancement in computer vision. In this study, we will perform vehicle detection using the YOLOv8 model due to its stability, fast processing speed, and accuracy for real-time processing.

Traffic flow prediction is a crucial area of research within Intelligent Transportation Systems (ITS), aiming to enhance traffic management, reduce congestion, and improve road safety. The integration of advanced machine learning techniques has marked a significant shift in how traffic flow is analyzed and predicted. This synthesis discusses various methodologies and advancements in traffic flow prediction using machine learning and deep learning approaches, highlighting influential studies in the field. Haghshenas et al. (2023) review the efficacy of multiple machine-learning techniques in traffic flow prediction. They provide both qualitative and quantitative analyses, advancements in urban traffic flow prediction technologies. The authors acknowledge that modern machine learning applications have transformed traditional methods, offering improved efficiencies in traffic management and signal control. Ma et al. (2020) proposed a Multi-Parameter Chaotic Fusion approach for traffic flow forecasting that accounts for various external factors, such as weather conditions and geographic influences. Their findings suggest that understanding these variables can enhance the stability and accuracy of traffic predictions. This analysis fits within a broader framework recognizing that traffic conditions are inherently dynamic and influenced by multifaceted factors. Li et al. (2020) extends with a focus on vehicle counting and traffic parameter estimation within dense traffic scenes. They identify challenges related to accuracy and speed in estimating parameters such as volume and density, key components for real-time traffic management systems. Their findings align with those of Yang et al. (2024), who emphasize the effectiveness of machine learning methods in enhancing traffic flow monitoring through the integration of cloud data.

In the realm of deep learning, innovative models have emerged to tackle the complexities of traffic flow prediction. Li et al. (2021) developed a hybrid deep learning framework integrating wavelet decomposition and convolutional neural networks with long short-term memory networks (CNN-LSTM). Their work emphasizes the importance of accurate long-term predictions to enable better strategic planning for traffic management. Similarly, Huang et al. (2024) proposed an MEA-LSTM model that leverages chaotic characteristics for short-term traffic prediction, enhancing accuracy by understanding upstream and downstream traffic behaviors. Karim and Nower (2024) research into long-term traffic predictions using a Stacked GCN model is particularly relevant, as it addresses the increasing necessity for precise modeling in urban environments where traffic patterns are rapidly evolving. Their work underscores the importance of predictive models that are accurate and applicable to real-world traffic management challenges. Turki and Hasson (2023) focused on employing Artificial Neural Networks (ANN) to estimate hourly traffic flows on motorways, highlighting the potential of deep learning methodologies to capture temporal

patterns and predict future flows accurately. Such methodologies are further supported by Shigemi et al. (2023), who explored the prediction of traffic breakdowns using detector data, illustrating the growing importance of data-driven decision-making in managing congestion.

The landscape of traffic flow prediction is evolving through the advanced integration of machine learning and deep learning methodologies. Continuous research and development in this area enhance predictive accuracy and significantly contribute to the optimization of urban traffic systems and safety strategies.

3. PROPOSED METHOD FOR TRAFFIC FLOW PREDICTION

Traffic flow prediction is a difficult problem and has many challenges such as accuracy, real-time, etc. Real-time prediction depends heavily on hardware, infrastructure and proposed methods. This section proposes a traffic flow prediction method using directed graphs to build a system that combines three deep learning models GCN, LSTM, and YOLO in order to achieve better outcomes. The models GCN, LSTM and YOLO has its own advantages and disadvantages. In this paper, we combine all three models to take advantage of their respective advantages. And each model is used to serve a different purpose in the solving process. Figure 1 presents the combination of three model GCN, LSTM and YOLO with three stages:

(i) Recognizes and object tracks

(ii) Extracting information from data structures

(iii) Time-series data processing

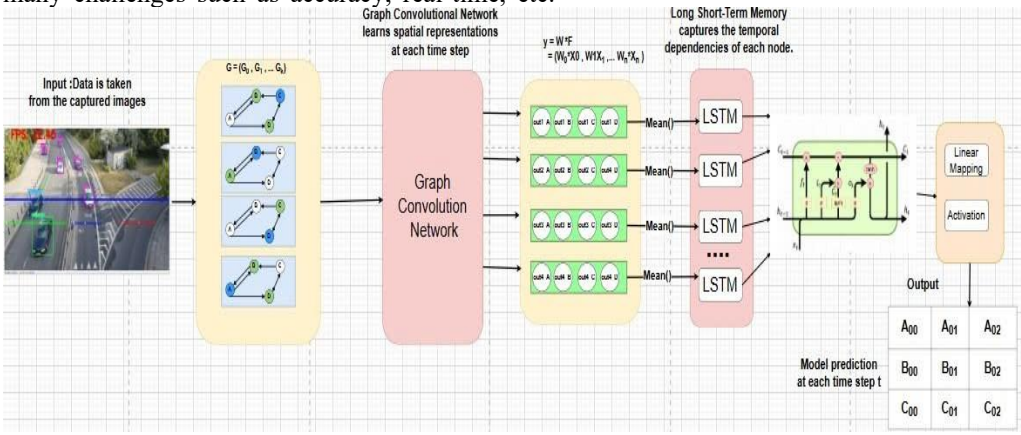

Figure 1. Detailed architecture of the combination of models

3.1. Recognizes and object tracks

The data extracted from the surveillance cameras on the road are videos, which are also the input data of our system. The core of the video is a sequence of image frames. Objects (vehicles, people, etc.) in these frames are detected and localized using the YOLO object model (blue, red, purple boxes in the image).

One of the primary advantages of YOLO is its single-stage detection process, which performs object classification and localization in one go. This architecture significantly reduces the computational load compared to traditional two-stage methods like R-CNN or SSD, which require separate processes for detecting and classifying objects (Guan, 2023; Vaikunth et al., 2024). The efficiency gained from this simplified structure allows YOLO to achieve remarkable real-time processing capabilities, making it well-suited for scenarios where speed is critical. Moreover, the YOLO model has incorporated enhancements that further improve detection accuracy and manage occlusions better than earlier versions.

In these stages, YOLOv8 is used for recognizing and tracking objects. The reason YOLOV8 was chosen for this task is:

– The YOLOv8 framework has emerged as a significant advancement in the field of object detection, offering improvements over its predecessors. This iteration integrates enhancements in efficiency, accuracy, and usability, making it suitable for various applications.

– One key advantage of YOLOv8 is its improved detection speed and accuracy. Compared to earlier iterations, YOLOv8 features refined architectural elements that optimize the processing workflow,

allowing it to maintain a high frame rate while analyzing images.

– Additionally, YOLOv8 maintains a compact and lightweight design, which facilitates its deployment on various platforms.

3.2. Extracting information from data structures

To describe spatial information and interactions between objects, a graph G = (V, E) is used, where:

V: nodes (representing each object, e.g. each vehicle).

E: edges (representing spatial relationships between objects).

This graph stores spatial information and interactions between objects. Each node (object) will have features extracted based on its location and relationship with other nodes. To extract features on G, a Graph Convolutional Networks (GCN) model is used. GCN is a deep learning model specialized in processing graph-structured data (graphs). Unlike CNN (Convolutional Neural Networks), which operate on grid-like data (images), GCNs can work with data that lacks a fixed structure, such as traffic systems. GCNs are widely used in the field of traffic flow prediction because the GCN structure is designed to process graph-based data, and predicting the traffic flow of a city requires a network of sensors and traffic cameras across the entire city. GCNs are particularly effective in extracting information from data structures with complex relationships, where each node not only carries its own information but is also influenced by neighboring nodes.

The GCN network allows for the propagation and aggregation of information from neighboring nodes through graph convolution operations, thereby learning integrated representations that combine both node content and graph structure.

The formula for computing the layers of GCN is as follows (Singh, 2019):

where,

H_i^{l+1}: The hidden layer at level L +1

A: The aggregated value from nodes.

W: The weight at layer l.

GCN learns spatial representations at each time point. Each node (object) will have features extracted based on its location and relationship to other nodes. The formula y (where, y = W·F) shows how weights (W) are multiplied by feature vectors (F) to create a new representation. The representations are pooled (mean or aggregate) to reduce dimensionality and normalize the information. The output of this step retains the most important features for each node.

3.3. Time series data processing

To process time series data, Long Short-Term Memory (LSTM) is proposed for use. LSTM (Long Short-Term Memory) is a variant of Recurrent Neural Networks (RNNs), designed to handle time series data. Thanks to its long-term memory mechanism, LSTM can learn both long-term and short-term trends in the data, helping to overcome the vanishing gradient problem found in traditional RNNs.

LSTM learns temporal dependencies. Each node across video frames is fed into a separate LSTM to learn sequence dynamics. The goal is to recognize the movement or change trend of each object over time. As we can observe, the convolution operation processes neighboring nodes and aggregates their features. For example, at timestep t where t ∈ T = {1, 2, 3, …}, nodes A, B, and C have values of [10, 20, 30] respectively. When we perform convolutio at node A, we consider its two adjacent nodes and aggregate them. If we apply Mean (average) as the aggregation method, we compute the average of these node values: Mean([10, 20, 30]) = 20. This value is considered the feature of a node at a specific time, aggregated to be fed into the Long Short-Term Memory (LSTM) network.

In traditional neural networks, we assume that the inputs are independent; however, in reality, these values are not. For example, today's data may influence the data of subsequent days. RNNs have been effective at modeling and learning from such dependencies.

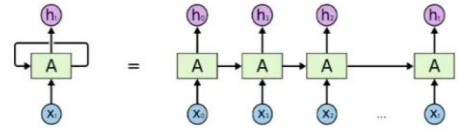

Figure 2. Illustration of the learning process in an RNN

(Source: Bengio et al., 2002)

In the figure 2, X_(0,1,2,3,…t) denotes the input data at time t, A represents the neural network block, h_(0,1,2,3,…t) is the output at time t. Intuitively, at t = 0 (the start of a sequential chain), there is no prior information to learn from. For subsequent steps t = 12,3,…T, at each layer t ≠ 0, there will be a loop in the hidden layers (A) that captures the preceding values.

However, a key weakness of RNNs is their handling of long-term dependencies.

For example, if T = 1000-time steps, during backpropagation to update the model's weights, we must perform Gradient Descent 1000 times. The backpropagation process of RNN:

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \times \frac{\partial h_t}{\partial h_{t-1}} \times \frac{\partial h_{t-1}}{\partial h_{t-2}} \times \ldots \times \frac{\partial h_1}{\partial h_W}$$

where:

∂L : Partial derivative of loss function L

∂W: Partial derivative of weight W

∂ht : Partial derivative of the hidden layer at time t

During each gradient descent step, if we use the sigmoid activation function, its output range is (0, 1). Every back propagation step multiplies the current gradient by a factor such as 0.5, 0.6. Over many steps, these repeated multiplications drive the gradient toward 0, leading to the vanishing gradient problem.

Because of this weakness, plain RNNs are less popular in practice than LSTM networks, which can retain long-term information by design. Thanks to this ability, LSTMs are widely used in time series analysis and NLP tasks.

To handle more complex interactions between nodes or between features, the interaction module (nonlinear interaction) is used. It can include attention or additional connections to model deeper relationships and map the LSTM output through a linear mapping layer. The final output can be either predicted coordinates (e.g. A00, B00, C00… are the locations of each object) or the state (stopped, moving, redirected…) of the object.

4. EXPERIMENT END EVALUATE RESULTS

4.1. Dataset analysis

In this study, public dataset from Caltrans PeMS (Caltrans, "n.d.") is used. PeMS dataset continuously gathers real-time measurements from nearly 40 000 sensors installed along the entire freeway network in California's major urban areas. For this study we selected the "Station 5 Minute" table, in which each sensor uploads a record every five minutes. The data collection period is from 01/01/2025 to 07/03/2025. The dataset includes 16 variables defined as follows:

– Timestamp: Date and start time of the summarized interval. For example, time 08:00:00 indicates that aggregated values contain measurements collected from 08:00:00 to 08:04:59. Note that the "seconds" value is always 0 for five-minute aggregates. Format: MM/DD/YYYY HH24:MI:SS.

– Station: Unique station ID, cross referenced with the metadata files.

– District: County name.

– Direction of Travel: Travel direction (East | West | South | North)

Lane Type:

● CD (Coll/Dist) – Collector/Distributor Lane: Intermediate lanes that distribute traffic flow between mainlines and entrance/exit ramps, typically found at major interchanges.

● CH (Conventional Highway): Standard highway - Roads without median barriers, featuring at-grade intersections (crossroads, traffic signals).

● FF (Fwy-Fwy Connector): Freeway-to-freeway connector - Short roadway segments enabling transitions between two freeways without exiting the freeway system.

● HV (HOV) (High Occupancy Vehicle Lane): Carpool lane - Dedicated lanes for multi-occupant vehicles (e.g., vehicles carrying 2-3+ passengers, buses, or electric vehicles in some regions).

● ML (Mainline): Mainline lanes - Primary freeway lanes carrying the main traffic flow.

● OR (On Ramp): Entrance ramp - Access roads connecting local roads or other routes to the freeway.

– Station Length: Length of roadway covered by the station, in miles/kilometers.

– Samples: Total number of sensor samples received across all lanes.

Observed (%): Percentage of individual lane points at this location that were actually observed (i.e., not imputed/filled values).

– Total flow: Sum of traffic flows over a 5-minute interval across all lanes.

– Avg Occupancy: Average occupancy across all lanes during a 5-minute interval, expressed as a decimal between 0 and 1.

– Avg Speed (mph): Average speed of vehicles over a 5-minute interval.

– Lane N Samples: Number of valid samples obtained for Lane N, where N ranges from 1 to the total number of lanes at that location.

– Lane N: Total traffic volume for lane N during a 5-minute interval, normalized by the number of valid samples.

– Lane N Avg Occ (%): Average occupancy for Lane N, expressed as a decimal between 0 and 1. N ranges from 1 to the total number of lanes.

– Lane N Avg (mph): Weighted average speed for Lane N (weighted by flow). If flow is 0, the arithmetic mean of lane speeds over the 5-minute interval is used. N ranges from 1 to the total number of lanes.

= Lane N Observed: Indicates observed data (1 = observed, 0 = imputed/filled data).

Data extraction challenges included missing data and noisy data. Removing such data improve. accuracy. Processing was done in PyCharm using Python 3.12.2 with these libraries: Patoolib, Os, and Pandas.

This study focuses on the Avg Speed variable from 5-minute sensor intervals, as average speed significantly impacts traffic flow. The dataset contains six lane types: CD, CH, FF, HV, ML, and OR. We analyze total-flow and avg-flow distributions per lane type. These variables help filter noisy sensors during training and identify the data distribution of both total-flow and avg-flow variables for each lane type.

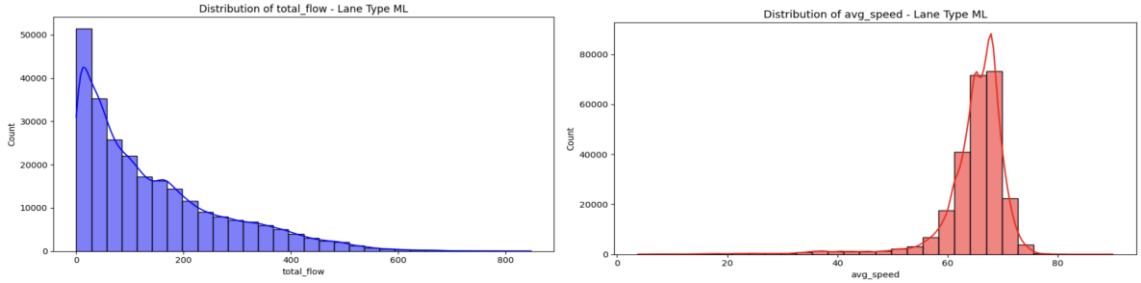

Figure 3. Distribution of total-flow and avg-speed data on ML lanes

Figure 3 shows that ML lane data has the most uniform distribution and the highest data volume compared to other lane types. Therefore, we selected this lane type for model implementation.

4.2. Experiment

Within the scope of this study, YOLO is used to simulate the process of data collection from traffic cameras. Specifically, the system utilizes the YOLOv8 model for the purpose of identifying and detecting objects so that we can collect data from sensors, such as data on average speed, traffic volume, etc., and that data will be sent to the server for analysis and processing.

The operational procedure includes the following key steps:

- Connect to real-time video streams from traffic camera sensors (livestream .m3u8 format).
- Read and process each frame from the video stream.
- Use the YOLO model to detect traffic objects present in each frame.
- Record information about the detected objects, including: vehicle type, confidence score, position, and time of appearance.
- Store or visualize the detection results to simulate the actual traffic data collection process.

By using YOLO, the system can quickly detect and collect object data in real-time conditions, laying the foundation for subsequent processing steps such as traffic flow analysis, speed calculation, or congestion prediction. Figure 4 presents an image captured from a real-time sensor (image cropped from available video).

Figure 4. Image captured from a real-time sensor (image cropped from available video)

To process this, YOLO and its associated tools are utilized, such as the Ultralytics and Supervision libraries. These libraries simplify tasks like drawing bounding boxes, labeling, processing video streams, and visualizing object detection results in real-time.

Since the YOLO model is trained on Facebook's COCO dataset, which can detect approximately 80 types of objects, in this study, we only focus on traffic-related objects such as cars, buses, trucks, etc. Other unrelated objects are excluded to prevent unnecessary or redundant predictions from the model.

ByteTrack, a component of the Supervision library, is responsible for tracking objects to prevent them from being lost during detection. To perform this tracking effectively, we need to configure specific parameters for object tracking as follows:

- Track_activation_threshold =0.25: An object detected by the YOLO model with a confidence score higher than 25% will be tracked.
- lost_track_buffer=30: If an object is lost for 30 consecutive frames, it will be removed from the tracking system. This helps to reduce the issue of losing track when the object is temporarily occluded.
- minimum_matching_threshold=0.8: The minimum similarity threshold (IoU or cosine distance) required to match objects between frames. A higher value makes the system more "selective," reducing the chances of misidentifying objects.

frame_rate=30: This is the video's frame rate, which is crucial for determining the time and speed of moving objects.

- minimum_consecutive_frames=3: An object must be detected for at least 3 consecutive frames to be considered valid.

Next, we need to add annotations to each bounding box of the object, such as the object's name and the confidence score. To do this, we use the BoxAnnotator and LabelAnnotator classes from the Supervision library. Figure 5 presented annotations on each object (image cropped from available video).

Figure 5. Annotations on each object (image cropped from available video)

As we can see, each object predicted by YOLO will be annotated with two values: the class name and the confidence score for the prediction.

Next, we will track the objects to calculate the speed of each object detected by YOLO:

In the image, we can see that each object's bounding box has a line extending behind it. To achieve this, we use the TraceAnnotator class from the supervision library.

The parameters for this object include thickness(the thickness of the motion trace line) and position(the display position, which in this case is set to the bottom center of the bounding box)

Next, we simply calculate the speed of each object based on its tracking path. To calculate the speed of an object, follow these steps:

- Iterate over the objects detected by YOLO.
- Check if the number of stored coordinates for that object is less than half a second's worth (i.e., half the number of frames per second). If True, it means the object has just appeared and doesn't have enough tracking information to calculate speed, so add that object to the label for speed calculation in the next frame.

Obtain the object's coordinates in a single frame, and then compute the distance by subtracting the initial position from the current position across frames.

- Calculate the travel time by dividing the number of frames by the frame rate (fps).
- Calculate the velocity, then multiply by 3.6 to convert from m/s to km/h.

- Add a label that shows the object ID and speed (e.g., label: 5, 32 km/h).

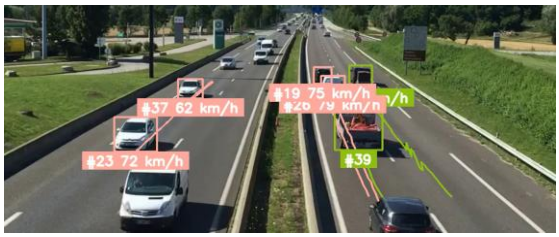

Figure 6. Calculated Speed (image cropped from available video)

Figure 6 presented the results after calculation. We simply need to collect data over a period (e.g., 5 minutes, 10 minutes, or longer) so that the model can predict the average speed of vehicles on a given road segment.

The model training process is conducted to learn optimal parameters that help the model make the most accurate predictions. In this section, the model is trained on a dataset that has been preprocessed, normalized, and divided into two parts: a training dataset, a validation dataset and a testing dataset in the ratio 7:1:2.

The input parameters are as follows:

- In_feat = In_feat = 1: The number of input features, which in this case is 1, meaning only one attribute — avg_speed (average speed) — is selected.
- Epochs = 50: The number of times the model is trained over the entire dataset.
- Input_seqenuce_length = 12: The number of previous time steps used as input for training the model
- Forecast_hoziron = 3: The number of future time steps the model is expected to predict.
- Out_feat: The number of output features generated by the graph convolution.
- Graph_conv_params: Parameters used during the graph convolution process:

+ aggregation_type: The chosen aggregation method is mean, which averages the neighboring node features.

+ combination_type: The selected combination method is concat, which concatenates the features of the current node with its neighbors.

+ activation: No activation function is used in this setup.

+ Optimization method: Adam optimizer is used with a learning rate of 0.01.

+ Loss: The Mean Square Error (MSE) is used to calculate the loss.

The train_dataset has been divided into multiple batches during the data preprocessing step for the model. Each batch in the train_dataset is a tuple of data (Input, Target), where Input has the shape (64, 12, 168, 1) corresponding to (batch_size, input_sequence_length, num_nodes,avg_speed), and Target has the shape (64, 3, 168) corresponding to (batch_size, output_sequence_length, num_nodes).

Validation_data: the data used to evaluate the model's performance during the training process.

Epochs refers to the number of complete passes through the entire dataset for updating the loss.

Callbacks: The Early Stopping callback is set with a patience value of 10, which means that if the loss value does not improve significantly for 10 consecutive epochs, the training process will stop early to ensure the model retains the best-performing weights.

4.3. Evaluate results

First, we need to visualize the Loss function over each epoch. After performing the training process, we obtained the following result. Figure 7 shows the relationship between Training and validation loss. We can observe that the Loss value gradually decreases on both the training and validation sets, indicating that the model is learning effectively from the data. The final result yields a Mean Squared Error (MSE) of 0.2634, which is a relatively good score for this model.

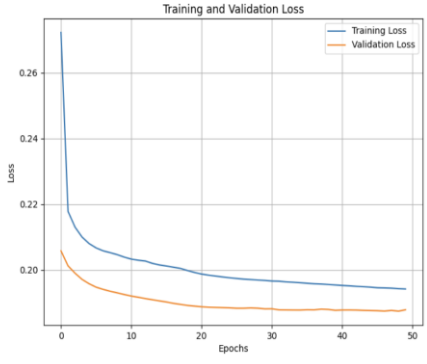

Figure 7. Training and validation loss

Given that the number of sensors is 168, we randomly select three values to visualize the behavior of these sensors. Below is the visualization of these three randomly selected sensors as Figures 8, 9 and 10.

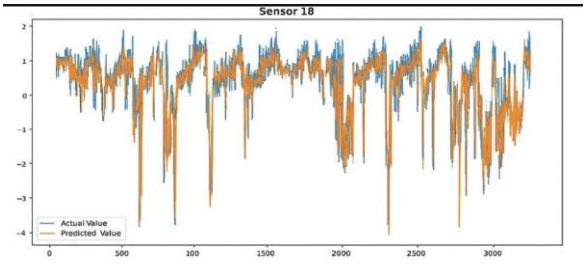

Figure 8. Visualize the behavior of sensor number 18

At sensor number 18, there are some small errors, which could be due to sudden changes on the route that sensor number 18 recorded, leading to some discrepancies.

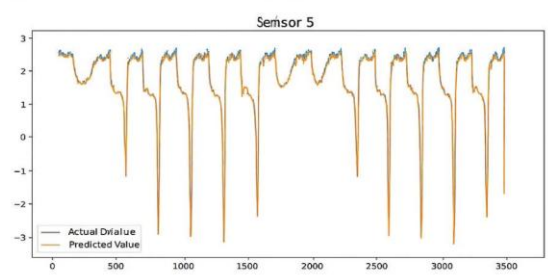

Figure 9. Visualize the behavior of the sensor number 5

At sensor number 5, we can see that the predicted data closely matches the actual data, indicating that the model performs very well.

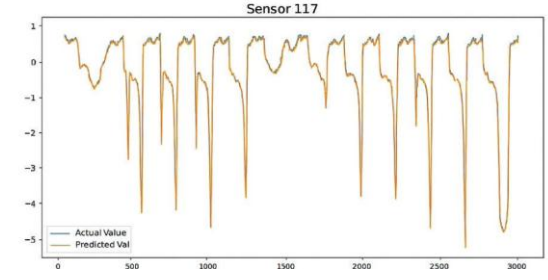

Figure 10. Visualize the behavior of sensor number 117

Similarly, at sensor number 117, we can see that the results are quite accurate compared to the actual data.

To evaluate the objectivity of the results achieved by the proposed method, the results of the proposed method will be compared with the results of other methods such as: ST-GCN (Yu et al., 2017), EIGRN (Ma et al., 2023), GCN-GRU (Karim & Nower, 2024). The comparison results are shown in Table 1. The evaluation scales used are: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

MAE measures the average magnitude of the errors in a set of predictions without considering their direction. It is the average value over the test sample of the absolute difference between the prediction and the actual observation, where all individual differences are weighted equally.

RMSE measures the average difference between the predicted value of a statistical model and the actual value. Mathematically, it is the standard deviation of the residuals. The residuals represent the distance between the regression line and the data points.

Table 1. Comparison of the results of the proposed method with other results

Methods	MAE	RMSE
ST-GCN	2.37	7.56
EIGRN	1.14	2.45
GCN-GRU	3.4033	2.0273
Our Method	**1.115**	**2.057**

From Table 1, our method achieves the lowest MAE (1.115) and RMSE (2.057) on the PEMSD7 dataset, substantially outperforming ST-GCN, EIGRN , and GCN-GRU. This improvement stems from our model's ability to capture, jointly, complex spatio-temporal dependencies through adaptive graph learning and temporal attention mechanisms.

In contrast, ST-GCN relies on fixed graph structures, limiting its spatial flexibility. EIGRN struggles with long-term temporal patterns and is sensitive to noise. GCN-GRU separates spatial and temporal modeling, which weakens interaction learning. These limitations result in significantly higher prediction errors.

The results demonstrate the superiority and robustness of our approach for spatio-temporal forecasting tasks.

5. CONCLUSIONS

Traffic flow prediction is a crucial area of research within ITS, aiming to enhance traffic management, reduce congestion, and improve road safety. The integration of advanced machine learning

techniques has marked a significant shift in how traffic flow is analyzed and predicted. This paper presented a method to predict the average speed reported by traffic sensors across the city by combining two core models, GCN and LSTM. The YOLO model is used to analyze images and video during data collection. Proper data processing plays a critical role in enabling the model to learn effectively and achieve better performance. This research successfully constructed and integrated two models: GCN and LSTM to predict the average speed of vehicles detected by sensors across the city. This integration allows the model to generalize both spatial and temporal relationships inherent in traffic data. Furthermore, the study also applied the YOLO model to simulate the sample data collection process from traffic sensors, providing a clearer understanding of how sensor data is gathered in practice.

However, this proposed method still has limitations, most notably, the model has not yet been deployed in a real-world environment. It has only been trained and evaluated on computers and has not been deployed on a server for live inference or large-scale application. The study does not provide comparisons with other experimental models, such as the combination of GCN with RNN or GCN with GRU. YOLO is only used at a simulation level to emulate the data collection process and has not been fully exploited for its advanced capabilities. The GCN model currently uses a static graph structure. This poses a significant limitation—if a new sensor is added or one of the existing sensors fails, the entire model might need to be retrained, leading to increased computational cost and inefficiency. Furthermore, the study does not address the correlation between motorcycles and cars in the traffic flow, which is particularly relevant for regions like Viet Nam where motorcycles are dominant. The model has not yet been developed using datasets from Viet Nam, which limits its generalizability. Current data is focused primarily on four-wheeled vehicles, whereas in Viet Nam, two-wheeled vehicles account for a significantly larger proportion of traffic.

In the future, integrating more powerful neural networks like GRU may help produce better results, while applying YOLO in real-world environments for real-time prediction. A different GCN model could also be adopted, where the graph structure allows for the addition or removal of unnecessary nodes, thereby reducing computational costs. Furthermore, a long-term goal is to identify the correlation between the speed of two-wheeled and four-wheeled vehicles, which could be particularly useful in regions like Viet Nam where two-wheeled vehicles are more prevalent.

ACKNOWLEDGMENT

This research is funded by Ho Chi Minh City University of Foreign Languages - Information (HUFLIT) under grand number H2024-11.

REFERENCES

Awan, F. M., Minerva, R., & Crespi, N. (2020). Improving road traffic forecasting using air pollution and atmospheric data: Experiments based on LSTM recurrent neural network. *Sensors*, *20*(13), 3749. https://doi.org/10.3390/s20133749

Bengio, Y., Frasconi, P., & Simard, P. (2002 Aug 06). The problem of learning long-term dependencies in recurrent networks. In IEEE International Conference on Neural Networks, San Francisco, CA, USA (pp. 1183-1188) https://ieeexplore.ieee.org/document/298725).

Caltrans. (n.d.). *Source performance measurement system data*. https://dot.ca.gov/programs/traffic-operations/mpr/pems-source

Chandra, S. R., & Al-Deek, H. (2009). Predictions of freeway traffic speeds and volumes using vector autoregressive models. *Taylor and Francis*, *13*(2), 53-72. https://www.tandfonline.com/journals/gits20

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of gated recurrent neural networks on sequence modeling. *Arxiv*, https://doi.org/10.48550/arXiv.1412.3555

Guan, Z. (2023). Real time object recognition based on yolo model. *Theoretical and Natural Science*, 137-143. https://doi.org/10.54254/2753-8818/28/20230450

Haghshenas, S. S., Astarita, V., Guido, G., Seraji, M. H. M. S., Gonzalez, P. A. A., Haghdadi, A., & Haghshenas, S. S. (2023). Assessment of machine learning techniques and traffic flow: Aqualitative and quantitative analysis. *Semantic Scholar*, *3*, 119-129. https://doi.org/10.47852/bonviewjcce32021062

Huang, Z., Zhang, Z., Li, Y., & Zhao, D. (2024). Short-term traffic flow prediction: A method of MEA-LSTM model based on chaotic characteristics analysis. *Journal of Sustainable Built Environment*, *1*(1), https://doi.org/10.70731/yk3fpz22

Johansson, U., Boström, H., Löfström, T., & Linusson, H. (2014). Regression conformal prediction with

random forests. *Springer Nature*, *97*, 55-176. http://dx.doi.org/10.1007/s10994-014-5453-0

Karim, A. A., & Nower, N. (2024). Probabilistic spatio-temporal graph convolutional network for traffic forecasting. *Springer Nature Link*, 54, 7070–7085.

Knol, D., Leeuw, F., Meirink, J. F., & Krzhizhanovskaya, V. (2021 Jun 09). Deep learning for solar irradiance nowcasting: A comparison of a recurrent neural network and two traditional methods. *Computational Science – ICCS 2021 Conference paper. 12746*, (pp. 309-322). Springer Nature Link.

Kumar, S., & Vanajakshi, L. (2015). Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *Springer Nature link*, https://link.springer.com/article/10.1007/s12544-015-0170-8

Li, S., Chang, F., Liu, C., & Li, N. (2020). Vehicle counting and traffic flow parameter estimation for dense traffic scenes. *IET Research, 14*, 1517-1523. https://doi.org/10.1049/iet-its.2019.0521

Li, Y., Chai, S., Ma, Z., & Wang, G. (2021). A hybrid deep learning framework for long-term traffic flow prediction. *IEEE*, *9*, 11264-11271. https://doi.org/10.1109/access.2021.3050836

Ma, C., Sun, K., Chang, L., & Qu, Z. (2023). Enhanced information graph recursive network for traffic forecasting. *Electronics*, *12*(11), 2519, 2519.https://doi.org/10.3390/electronics12112519

Ma, Q., Huang, G. H., & Ullah, S. (2020). A multi-parameter chaotic fusion approach for traffic flow forecasting. *IEEE Xplore*, *8*, 222774-222781. https://doi.org/10.1109/access.2020.3043777

Manoel, C., Jeong, Y. , Jeong, M., & Han, L. (2009). Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Sciencedirct*, *36*(3), 6164-6173. https://doi.org/10.1016/j.eswa.2008.07.069

Ren, C., Chai, C., Yin, C., Ji, H., Cheng, X., Gao, G., & Zhang, H. (2021). Short-term traffic flow prediction: A method of combined deep learnings. *Journal of Advanced Transportation*, *2021*(1), 9928073. https://doi.org/10.1155/2021/9928073

Shigemi, R., Ando, H., Wada, K., & Mukai, R. (2023). Predicting traffic breakdown on expressways using linear combination of vehicle detector data. *Nonlinear Theory and Its Applications, IEICE, 14*(2), 416-427. https://doi.org/10.1587/nolta.14.416

Singh, P. (2019). *Wading through Graph Neural Networks*. https://spraphul.github.io/blog/GCN

Tran, Q. H., Fang, Y., Chou, T., Hoang, T. V., Wang, C., Vu, V. T., Ho, T. L. H., Le, Q., & Chen M. (2022). Short-term traffic speed forecasting model for a parallel multi-lane arterial road using gps-monitored data based on deep learning approach. *Sustainability, 14*(10), 6351. Https://doi.org/10.3390/su14106351.

Turki, A. I., & Hasson, S. T. (2023). Study estimating hourly traffic flow using artificial neural network: A M25 motorway case. *Samarra Journal of Pure and Applied Science*, *5*(1), 47-59. https://doi.org/10.54153/sjpas.2023.v5i1.448

Vaikunth, M., Dejey, D., Vishaal, C., & Balamurali, S. (2024). Optimizing helmet detection with hybrid yolo pipelines: A detailed analysis. *Big Data, IOT and Blockchain Trends 2025*, 83-93. https://doi.org/10.5121/csit.2024.142406

Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (Vol. 1, pp. I-I). IEEE. https://ieeexplore.ieee.org/document/990517

Wang, S., Zhang, X., Li, F., Yu, P. S., & Huang, Z. (2018). Efficient traffic estimation with multi-sourced data by parallel coupled hidden markov model. *IEEE*, *20*(8), 3010-3023. Https://ieeexplore.ieee.org/Xplore/home.jsp

Yang, D., Chen, K., Yang, M., & Zhao, X. (2019). Urban rail transit passenger flow forecast based on LSTM with enhanced long-term features. *SemanticScholar*, *13*(10), 1475-1482. https://doi.org/10.1049%2Fiet-its.2018.5511

Yang, P., Chen, Z., Su, G., & Wang, B. (2024). Enhancing traffic flow monitoring with machine learning integration on cloud data warehousing. *ResearchGate*, *67*, 1-7. Https://doi.org/10.54254/2755-2721/67/2024ma0058

Yu, B., Yin H., & Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *ArXiv*, 3634-3640. https://doi.org/10.24963/ijcai.2018/505

Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J. (2017). LSTM network: A deep learning approach for short-term traffic forecast. *IET intelligent transport systems*, *11*(2), 68-75. http://dx.doi.org/10.1049/iet-its.2016.0208

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., & Lin, T. (2019 Aug 22). T-GCN: A Temporal Graph Convolutional Network for Trafic Prediction. *IEEE Transactions on Intelligent Transportation Systems* (pp. 3848-3858. https://ieeexplore.ieee.org/document/8809901).

Zhu, H., Xie, Y., He, W., Sun, C., Zhu, K., Zhou, G., & Ma, N. (2020). A novel traffic flow forecasting method based on RNN-GCN and BRB. *Journal of Advanced Transportation*, *2020*(1), 7586154. http://dx.doi.org/10.1155/2020/7586154