Predicting Passenger Flow Using Graph Neural Networks with Scheduled Sampling on Bus Networks

Baghbani, Asiye; Rahmani, Saeed; Bouguila, Nizar; Patterson, Zachary

DOI
10.1109/ITSC57777.2023.10422701

Publication date
2023

Document Version
Final published version

Published in
2023 IEEE 26th International Conference on Intelligent Transportation Systems, ITSC 2023

2023 IEEE 26th International Conference on
Intelligent Transportation Systems (ITSC)
24-28 September 2023. Bilbao, Bizkaia, Spain

Predicting Passenger Flow Using Graph Neural Networks with Scheduled Sampling on Bus Networks

Asiye Baghbani[1], Saeed Rahmani [2], Nizar Bouguila[1], and Zachary Patterson[1]

2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC) | 979-8-3503-9946-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ITSC57777.2023.10422701

Abstract— **Predicting short-term passenger flows in bus networks is crucial to improving the overall performance of such systems and increasing their attractiveness. This study develops a graph neural network-based framework for multi-step passenger flow prediction specifically designed for bus networks to capture their unique characteristics. We propose the Multi-step Multi-component Graph Convolutional Long Short-Term Memory (Multi-GCN-LSTM) model, which uses 1) a proximity matrix in addition to an adjacency matrix to consider the effects of vehicular traffic and link-level distances; 2) Scheduled Sampling for multi-step prediction, which prevents error propagation across prediction steps; and 3) a novel fusion mechanism for considering time-varying spatial and temporal correlations among passenger flow data based on recent, daily, and weekly travel patterns. This model is validated using real-world data collected from the Laval bus network. Also, benchmarking the established model against state-of-the-art baselines indicated its competency.**

I. INTRODUCTION

Bus transportation systems are essential parts of urban transportation networks. They can reduce the share of private cars and thus reduce traffic congestion, air pollution, and fuel consumption. The relative crowdedness of buses in the network plays a significant role in the optimal operation and increasing the attractiveness of these systems. Knowing about passenger flows, which in this study means the number of people in the bus, on a specific bus or route at a short-term horizon can help decision-makers and transit operators to control the passenger outflows and inflows at bus stops. Moreover, informing passengers about the crowdedness of arriving buses in the network will help them decide when and what services or modes of transport to use.

Accordingly, short-term passenger flow prediction has drawn the attention of researchers [1], [2]. Initial attempts mainly tried to model passenger flows using linear models [3]. However, linear methods showed to be insufficient for capturing complex relationships among such data. Accordingly, nonlinear methods, and especially deep learning techniques, became dominant for short-term passenger flow prediction by utilizing fully connected deep neural networks [2], [4], convolutional neural networks (CNN) [5], LSTM-based models [6], [7], and hybrid frameworks [8], [1].

*This work was not supported by any organization

[1] Asiye Baghbani, Nizar Bouguila, and Zachary Patterson are with Concordia Institute for Information Systems Engineering, Concordia University, 1515 St. Catherine W., Montreal, Canada asiye.baghbani@mail.concordia.ca, nizar.bouguila@concordia.ca, zachary.patterson@concordia.ca

[2] Saeed Rahmani is with the Department of Transport and Planning, Delft University of Technology, Stevinweg 1, 2628 CN Delft, The Netherlands s.rahmani@tudelft.nl

Nevertheless, general deep learning models tend to overlook important characteristics of transportation networks. For example, CNNs, which are widely used for capturing spatial correlations, assume Euclidean spatial relationships among data points. This assumption may not hold in transportation networks, where spatial correlations are usually better represented as graphs rather than Euclidean space [9]. For instance, passenger flows between nearby stops may not exhibit strong correlations if they are not connected by the same route or are serving different areas of the city.

In recent years, a new class of machine learning methods has emerged that enables the translation of deep learning tasks on graph-structured data, which have been also widely applied to passenger flow prediction. The promising results of Li et al. [10] and Han et al. [11] encouraged many researchers to develop graph-based frameworks for passenger flow prediction utilizing multi-graphs [12], [13], utilizing hypergraph concept [14], introducing dynamic graph structures [15], integrating LSTM into GCN [16], [17], applying attention mechanism [18], and using encoder-decoders [19].

However, these studies are also faced with some limitations. Firstly, they have been focused on metro and train systems, neglecting bus networks. Nevertheless, bus systems have unique operational and design characteristics, which require a separate line of research [2], [7]. For instance, they usually share their routes with vehicular traffic, which causes significant uncertainties with regard to their stickiness to pre-defined timetables, and therefore, passenger flows. Moreover, unlike rail-based systems, bus systems are significantly prone to environmental disruptions. In addition, bus networks are often designed at lower operational levels and thus are usually highly interconnected with more stops, multiple connections, and frequent transfer points. This makes bus networks more complex and intricate for analysis and prediction, especially using graph-based methods.

Besides, most GNN studies on bus passenger flow prediction have focused on single-step prediction, which may not be practical for users or operators. A very short-term horizon (e.g., 10 minutes) does not supply the operators with sufficient time for modifying the system, and a longer horizon (e.g., 60 minutes) might be useless for users considering the short time intervals between bus services. Accordingly, an accurate multi-step prediction can supply multiple time resolutions, which is desirable for both end-users and operators. On the other hand, few studies that have investigated multi-step bus passenger flow prediction [20] suffer from a known problem in sequence-to-sequence prediction, which is called exposure bias [21]. This issue is because, in the training phase, the ground truth data is used as the input for *all* steps ahead. However, when forecasting, the model only uses the *predicted* values from the previous steps. Therefore, there is an inconsistency between the training and testing phases, which leads to the propagation of errors throughout the prediction steps [22]. To solve this issue, we apply a scheduled sampling technique [22], to bridge the gap between the training and testing phases and force the model to learn how to correct its mistakes during the training process. To the best of our knowledge, this is the first time that such a mechanism is applied to a short-term passenger flow prediction.

Last but not least, studies have shown that there usually are multiple temporal and spatial correlations among data points [2], [13], which means there might be short-term, mid-term, and long-term patterns in passenger flow data. Accordingly, we have designed a novel fusion mechanism for incorporating the most recent (for instance, hourly) patterns into the prediction task, while considering the daily and weekly variations as well. The idea behind including daily and weekly patterns is that people usually show similar behaviors at specific hours of the day, and specific days of the week.

To summarize the above discussions, the main contributions of this study are:

1) We have developed a novel graph-based deep learning framework for passenger flow prediction, specifically designed for bus systems. In this framework, unique characteristics of bus networks are considered by defining two types of spatial correlations: 1) the connectivity of the stops based on bus lines and routes, 2) and the road network proximity for considering the structure of the road network and the effects of vehicular traffic (accounting for travel time between stops). This structure enables going beyond the simplified assumptions made for metro and rail transit systems.
2) We have designed a specific multi-step prediction mechanism by using scheduled sampling, which prevents the propagation of errors during the multi-step prediction process.
3) We have designed a fusion mechanism for considering the short-, mid-, and long-term temporal patterns in our prediction frameworks.
4) Finally, we have evaluated the performance of the proposed framework by conducting a real-world experiment using the passenger flow data for 467 stops in Laval, Canada, and comparing its outputs with popular baseline models. It is the first time that the performance of a GNN-based model is evaluated on such a high number of stations based on a real dataset.

The remainder of the paper is organized as follows. In the next section, we detail the structure of the proposed framework. In the Experiment section, we introduce the dataset we have used and then apply the proposed framework on this dataset consisting of 467 stops and evaluate its performance compared to some popular baselines. Finally, in the Conclusion, we summarize the findings of the study and suggest recommendations for future studies.

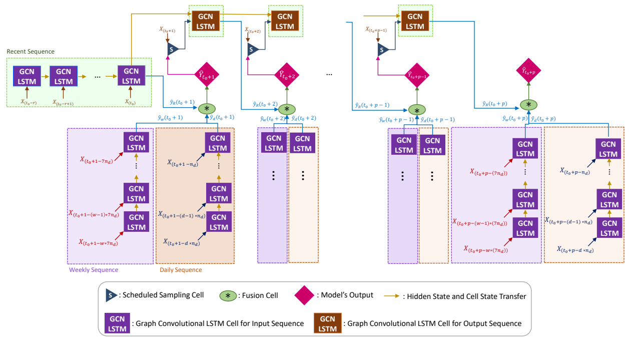
Recent Sequence
GCN LSTM
Weekly Sequence
Daily Sequence
: Scheduled Sampling Cell
: Fusion Cell
: Model's Output
: Hidden State and Cell State Transfer
: Graph Convolutional LSTM Cell for Input Sequence
: Graph Convolutional LSTM Cell for Output Sequence

Fig. 1. The Structure of the Multi-GCN-LSTM Model

II. Methodology

The proposed framework is illustrated in Figure 1. In this framework, the most recent historical data are first fed into the Graph Convolutional LSTM (GCN-LSTM) modules. Simultaneously, the sequences of weekly and daily inputs are fed into similar but separate GCN-LSTM modules. The outputs from these modules are then fused to make a prediction for the next step. This prediction is modified according to a Scheduled Sampling mechanism based on the observed (true) value from the next step. Finally, to make the new prediction, the output from the Scheduled Sampling is combined with a hidden state variable of the most recent sequence, which is the information from previous time steps cells for maintaining long-term dependencies in the input sequence. The whole procedure is repeated to achieve a multi-step prediction. The following section discusses these components in more detail.

A. Graph Definition and Notations

In this study, we have defined a unique graph structure considering the connection between stops, as well as the proximity of the stops based on the travel times of the road network. This enables the model to take into account the effects of vehicular traffic. It is further explained in the following sections.

1) Bus Network Adjacency Matrix: Bus Network Adjacency matrix is defined based on whether two stops are connected by a bus route. Typically, a graph $G = (N, E)$ comprises two sets: a set N with elements called nodes $n_i \in N$, and a set E with elements called edges $(n_i, n_j) \in E$. This study considers bus stops as nodes, and bus routes as edges in a network to represent the connections between the nodes. It considers the bus network as a directed graph since travel along a given route in one direction could be different from the other one, and besides, some stops are active in only one direction. The adjacency matrix describes the connections between nodes in graphs. Considering $A \in R^{(N\times N)}$ as an adjacency matrix, $A_{(i,j)}$ equals one when node i is connected to node j and equals zero otherwise. Moreover, this study considers each node to be connected to itself, so $A_{(i,i)} = 1$.

2) Bus Network Proximity Matrix: In real bus networks, closer stops usually have a greater impact on each other (given that they are connected). This proximity is influenced by the road network structure and vehicular traffic. Accordingly, the Bus Network Proximity Matrix (BNP) is defined to identify close stops for each target stop. This matrix represents the possibility of reaching a particular destination stop from a given origin stop in a certain amount of time. To this end, a distance matrix $Dist \in R^{N\times N}$ is defined, in which each element $Dist_{i,j}$ represents the distance between stops i and j on the real road network. Following this, we will calculate the BNP matrix by using the distance matrix and average bus speed within each edge (1). By using the average speed in the links, our framework accounts for vehicular traffic where buses share their routes with vehicles.

$$BNP_{i,j} = \begin{cases} 1, & \text{if } \; S_{i,j}\Delta t - Dist_{i,j} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In this formula, $S_{i,j}$ represents the average speed of the bus between stops i and j, and Δt stands for the desired time interval between stops. According to the matrix, each element $BNP_{i,j}$ equals one if the passenger can travel from stop i to stop j in Δt and zero otherwise.

B. Short-term Multi-step Passenger Flow Prediction

The purpose is to predict the passenger flows at the stop level for the whole network. $X_t \in R^N$ is defined as a vector of the passenger flows at all nodes in the graph at time t; accordingly, The purpose is to predict the passenger flows at the stop level for the whole network. $X_t \in R^N$ is defined as a vector of the passenger flows at all nodes in the graph at time t; accordingly, the function F(.) is defined that maps the time steps of the historical graph of passenger flows to the graph of passenger flows in the next multiple time steps T_P. Equation (2) defines $F(\cdot)$:

$$\begin{aligned} F([Temporal - Features]; G(N, E, A, BNP)) = \\ [X_{t+1}, X_{t+2}, ..., X_{t+T_P}] \end{aligned} \tag{2}$$

In this formula, N, E, A, and BNP respectively represent the set of nodes, edges, the adjacency matrix, and the proximity matrix, and G represents the bus network graph.

C. Graph Convolutional LSTM Cell

The GCN-LSTM cell includes two main parts: a Graph Convolutional operator and a Long-Short-Term-Memory (LSTM) cell [9]. Graph convolution layers are used to extract spatial information from graph input data. In this model, the Bus Network Graph Convolution (BNGC) operation is defined as the product of the input data, the adjacency matrix, the bus network proximity matrix, and a trainable weight matrix as shown in (3):

$$BNGC_t = (TW_{bngc} \odot A \odot BNP)X_t \tag{3}$$

where, $BNGC_t \in R^N$ is the extracted bus network graph convolution features at time t, $TW_{bngc} \in R^{N\times N}$ is a trainable weight matrix, and $X_t \in R^N$ is the vector of passenger flows for all stops in the network at time t. In addition, $\odot$ represents the Hadamard product operator. According to the definitions, both adjacency (A) and proximity matrices (BNP) are sparse matrices. Therefore $TW_{bngc} \odot A \odot BNP$ is also sparse and contains just 0 and 1. As a result, the trained weight TW_{bngc} can be used to analyze interactive influences among bus network graph stops, which improves model interpretability [9]. The output of the BNGC layer is used as input for the LSTM cell. LSTM cells have four gates: an input gate ig_t, an output gate og_t, a forget gate fg_t, and an input cell state $\tilde{C}_t$ in terms of time step t. Equations (4) to (7) show how to calculate these gates.

$$ig_t = \sigma_g \left(TW_{ig}\cdot BNGC_t + U_{ig}\cdot h_{t-1} + b_{ig}\right) \tag{4}$$

$$og_t = \sigma_g \left(TW_{og}\cdot BNGC_t + U_{og}\cdot h_{t-1} + b_{og}\right) \tag{5}$$

$$fg_t = \sigma_g \left(TW_{fg}\cdot BNGC_t + U_{fg}\cdot h_{t-1} + b_{fg}\right) \tag{6}$$

$$\tilde{C}_t = \tanh \left(TW_{c}\cdot BNGC_t + U_{c}\cdot h_{t-1} + b_{c}\right) \tag{7}$$

Here, weight matrices are split into two groups. The first set of weight matrices, consisting of TW_{ig}, TW_{og}, TW_{fg}, and $TW_C \in R^{N\times N}$, map BNGC outputs to the gates and the input cell state, and the second set, consisting of U_{ig}, U_{og}, U_{fg}, and $U_C \in R^{N\times N}$, are used in the preceding hidden state. Additionally, b_{ig}, b_{og}, b_{fg}, and $b_c \in R^N$ are used as bias vectors. Finally, σ_g is the gate activation function (sigmoid function), $tanh$ is the hyperbolic tangent function, and ”·” represents the matrix multiplication operation. Moreover, to incorporate the neighboring states of each node in the graph, we need to define a cell state gate, which incorporates the previous states of each node and its neighbors. An LSTM cell state gate is thus defined as (8):

$$C^*_{t-1} = TW_N \odot (A \odot BNP) \cdot C_{t-1} \tag{8}$$

The value of TW_N measures the impact of neighboring cell states by multiplying the product of the adjacency and bus network proximity matrices. The cell state is repeatedly inserted into a subsequent time step in order to consider its impact on neighboring cells. The final cell state C_t and hidden state h_t are thus defined as (9) and (10) respectively:

$$C_t = fg_t \odot C^*_{t-1} + ig_t \odot \tilde{C}_t \tag{9}$$

$$h_t = og_t \odot \tanh(C_t) \tag{10}$$

A GCN-LSTM cell's output will ultimately be the hidden state h_T at the last time step T ($\hat{y} = h_T$).

D. Temporal Dependencies and Multi-Component Fusion

In order to capture the varying temporal correlations based on recent, daily, and weekly patterns, the model combines the outputs from the corresponding components using a fusion mechanism (as shown in Figure 1). Here, we consider t_0 as the index of current time in the input dataset, and p the length of the prediction sequence. There are three different input data lengths denoted by r, d, and w; Each represents an integer multiple of p, and respectively is showing the input data length for recent data, daily data, and weekly data. Here, we suppose that the sampling frequency per day is n_d times. Figure 2 is an example of how we build weekly, daily, and recent input data.

- *Recent Data:* This time series data is a segment of historical data just before the period for which the prediction is made. This data is represented as $T_{recent} = \{X_{(t_0-r)}, X_{(t_0-r+1)}, ..., X_{(t_0)}\}$.

- *Daily Periodic Data:* In the daily periodical input data, segments from the past few days were taken into account from the same time slot as the prediction time slot. This data is represented as $T_{daily} = \{X_{(t_0+p-d*n_d)}, X_{(t_0+p-(d-1)*n_d)}, ..., X_{(t_0+p-n_d)}\}$.

- *Weekly Periodic Data:* Segments in the last few weeks at the same time intervals as the prediction period are also included in the proposed framework. This data is represented as $T_{weekly} = \{X_{(t_0+p-w*7n_d)}, X_{(t_0+p-(w-1)*7n_d)}, ..., X_{(t_0+p-7n_d)}\}$.

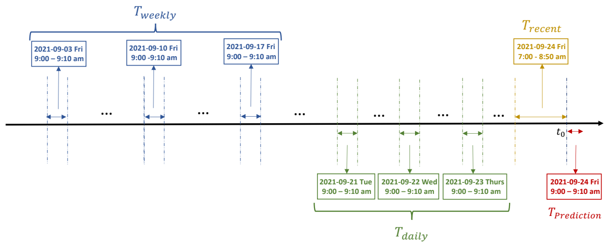
T_{weekly}
2021-09-03 Fri 9:00 – 9:10 am
2021-09-10 Fri 9:00 -9:10 am
2021-09-17 Fri 9:00 – 9:10 am
T_{recent}
2021-09-24 Fri 7:00 - 8:50 am
...
t_0
2021-09-21 Tue 9:00 – 9:10 am
2021-09-22 Wed 9:00 – 9:10 am
2021-09-23 Thurs 9:00 – 9:10 am
T_{daily}
2021-09-24 Fri 9:00 – 9:10 am
$T_{Prediction}$

Fig. 2. An example of constructing segments of time series dataset for weekly, daily, and recent input data

Each of the three components comprises several GCN-LSTM cells, with similar network structures. $\hat{y}_r$, $\hat{y}_d$, and $\hat{y}_w$, respectively, are the outputs of recent, daily, and weekly components. The impacting weights of the three components are different for each of the nodes when the outputs of the different components are fused, and these weights should be determined from the historical data. To learn these weights, we use two convolution layers with Relu activation function. Those outputs are then fused as (11):

$$\hat{Y}_{t_0+1} = W_2 * RELU(W_1 * [\hat{y}_{r(t_0+1)} | \hat{y}_{d(t_0+1)} | \hat{y}_{w(t_0+1)}]) \tag{11}$$

where, $*$ means the convolution operation and $|$ means the concatenation operation.

E. Scheduled Sampling and Training Algorithm

Throughout this section, we explain how the prediction time horizon is extended into multiple time steps in the future. We also utilize scheduled sampling [22]to mitigate the error propagation through prediction steps by bridging the gap between training and inference in sequence prediction. Again, assume t_0 is an index of current time in the dataset, and we intend to predict the passenger flow over the next $p > 1$ timestamps. In each prediction step, if we can build the three time-series segments we introduced in section D (recent, daily, and weekly), we can make the prediction for the next step. These three time-series segments for the first timestamp were introduced in Section D, so the first prediction ($\hat{Y}_{t_0+1}$) would be feasibly estimated.

In the second step, we can still build the daily- and weekly time series segments based on the historical data. Daily-periodic segment would be $T_{daily} = \{X_{(t_0+2-d*n_d)}, X_{(t_0+2-(d-1)*n_d)}, ..., X_{(t_0+2-n_d)}\}$ and weekly-periodic segment would be $T_{weekly} = \{X_{(t_0+2-w*7n_d)}, X_{(t_0+2-(w-1)*7n_d)}, ..., X_{(t_0+2-7n_d)}\}$. However, for the recent data, since we are using the data of the period directly adjacent to which predictions are made, we need to use the data from $t_0 + 1$ ($X_{(t_0+1)}$). However, the ground truth values for this time slot are not available in the inference (test) phase, and thus, the first predicted value $\hat{Y}_{t_0+1}$ is used. This results in a mismatch between the training and testing processes. Alternatively, if we pretend we are testing in the training process (to use $\hat{Y}_{t_0+1}$ instead of $X_{(t_0+1)}$), the first weights update generates some noise, and these errors are propagated throughout time, leading to more errors at subsequent time steps. Thus, the training process can be very unstable.

To solve this issue, we propose the scheduled sampling strategy trying to utilize the best of both approaches presented in the previous paragraph. It means the ground truth data are used in the initial epochs of training, and will gradually be replaced with the predicted values over time. Accordingly, in this study, if we consider i as the number of epochs, for predicting $\hat{Y}_{(t_0+2)}$, we use the ground truth ($X_{(t_0+1)}$) with the probability of ϵ_i, and an estimation coming from the model for previous time step ($\hat{Y}_{(t_0+1)}$) with the probability of $(1-\epsilon_i)^2$. Equation 13 shows how we calculate T_{recent} for the second time step. As it has been shown, in this way, we consider both the training and interface phases, which means if $\epsilon_i = 1$, the model will be trained only based on its true values (training phase), whereas in the case of $\epsilon_i = 0$, it will be trained in the same way that the inference is performed (only using the prediction from last time step).

$$\begin{aligned} T_{recent} = \{ & X_{(t_0-r+1)}, X_{(t_0-r+2)}, ..., X_{(t_0)}, \\ & (\epsilon_i * X_{(t_0+1)} + (1-\epsilon_i)^2 * \hat{Y}_{t_0+1})\} \end{aligned} \tag{12}$$

Different functions are available for scheduling how Epsilon changes over time as training progresses [22]. In this study, we use exponential decay, which is $\epsilon_i = k^i$, and $k < 1$ which is a constant for the expected speed of convergence.

After modeling each of the recent, daily, and weekly components for the second step of the prediction period, similar to (11), we will have $\hat{Y}_{t_0+2} = W_2 * RELU(W_1 *$

$[\hat{y}_{r(t_0+2)}|\hat{y}_{d(t_0+2)}|\hat{y}_{w(t_0+2)}])$. This process will be repeated for the next step of prediction. The parameters for all fusion parts and the main three components of the model (recent, daily-periodic, and weekly-periodic time series) are shared to avoid the over-smoothing of the model.

The loss, after predicting the values for all future steps, will be defined as:

$$Loss = \sum_{n=1}^{p} L(Y_{(t_0+n)}, \hat{Y}_{(t_0+n)}) \tag{13}$$

where, $L(.)$ is a function to calculate the residual between the predicted values $\hat{Y}_T$ and the actual values Y_T. The loss function here is the Mean Squared Error (MSE).

III. Experiments

In this section, we describe a real-world experiment by implementing the proposed framework.

A. Datasets

The dataset used is a subset of the bus network in Laval, Canada from mid-September to mid-October 2021. The type of data is APC and this subnetwork includes one of the busiest routes - route 26 - and all its feeders in Laval. In total, this dataset includes 467 stops and 11 routes. Fig. 3 shows this network. The data covers from 5 AM to 11 PM, aggregated into 10-minutes time intervals and $n_d = 28$, and is divided into 70% training, 20% validation, and 10% testing sets. Also, model parameters that have been used are $r = 12$, $d = 3$, and $w = 3$. Also, stops that are reachable in 15 minutes have been considered close in the BNP matrix.

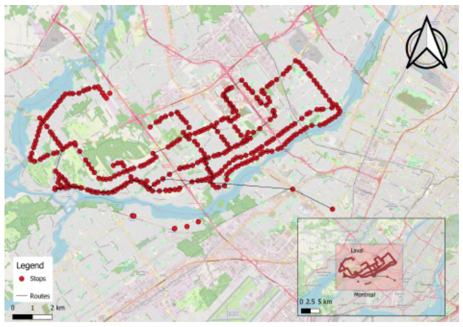

Fig. 3. The modeled network of bus stations and routes – Laval, Canada

B. Experimental Setting

In the proposed model, the dimension of the hidden states in the LSTM cells is considered to be equal to the number of nodes of the graph. The model is trained by minimizing its mean square error where the batch size and initial learning rate are equal to 16 and $10^{(-5)}$, respectively. Besides, since RMSProp overcomes gradient erosions and vanishing, it is considered gradient descent optimization. Evaluation of the model is done by comparing the predicted values with real observations; and comparing the model's performance to that of six other methodologies, including Historical Average (HA), Multi-layer Perceptron model (MLP), the Long Short-Term Memory Model (LSTM), Graph Wavenet [23], STGCN [24], and GMAN [25]. Additionally, we implemented the model without using the Scheduled Sampling method to explore the impact of using this technique.

C. Evaluation Metrics

The Mean Absolute Error -MAE (14) and Root Mean Squared Error - RMSE (15) have been used as the performance metrics in this study.

$$MAE = \frac{1}{n}\sum_{T=1}^{n} |y_T - \hat{y}_T| \tag{14}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{T=1}^{n} (y_T - \hat{y}_T)^2} \tag{15}$$

where, $\hat{y}_T$, y_T, and n represent the predicted value, the actual value, and the number of observations for each stop.

D. Experimental Results

The average MAE and RMSE values for both the proposed model and the baselines are presented in Table I. As can be seen, the proposed model has outperformed all baselines in both metrics. Also, it can be inferred that scheduled sampling has significantly improved the performance of the model. Furthermore, based on MAE and RMSE values, the STGCN and Graph Wavenet models have the weakest performance results as compared to all other models.

TABLE I
Performance comparison between different models

Model	MAE	RMSE
Graph Wavenet	3.208	4.464
STGCN	2.298	4.209
HA	1.716	3.021
LSTM	1.824	2.995
GMAN	1.577	3.304
MLP	1.723	3.511
Multi-GCN-LSTM without Sechedule Sampling	1.697	2.914
Multi-GCN-LSTM	**1.465**	**2.688**

Figure 4 and Figure 5, respectively, present the values of MAE and RMSE for each timestamp of the prediction period. As can be seen, as we move along in the prediction period, the MAE and RMSE values increase for most of the models. This indicates that each model performs better in closer time windows. Moreover, as can be seen in both figures, the MAE and RMSE values of the Multi-GCN-LSTM model in all time periods are lower than the other models, which indicates it has a better performance in individual prediction steps as well. Moreover, it is apparent that the model without scheduled sampling is performing significantly worse than the model with scheduled sampling.

IV. Conclusion

This study develops a novel graph-based deep learning model for the multi-step prediction of passenger flows in bus networks. It combines the adjacency and proximity matrices of the network graph to extract more meaningful spatial

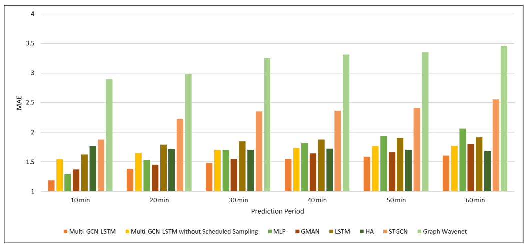

Fig. 4. Comparison of multi-step Prediction - MAE

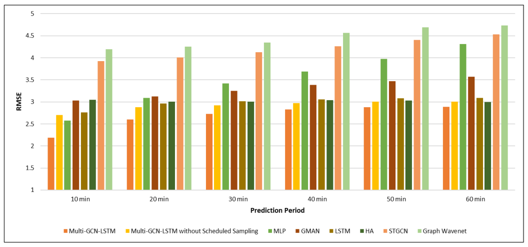

Fig. 5. Comparison of multi-step Prediction - RMSE

features of the network. With the graph convolution operator and the LSTM, not only the spatial but also the temporal aspects of passenger flow are considered. As an additional consideration, we use multicomponent fusion to examine the effects of temporal dependencies on daily and weekly trends in historical data. Moreover, we use a scheduled sampling method to prevent error propagation in multistep predictions. The proposed model is validated against real-world data by comparison with the popular baselines. Future research can include other elements such as weather conditions, events, and constructions. In addition, the origin destinations matrix for different historical periods can also be used to examine how the spatial correlation between stops varies between different periods of time to have a dynamic adjacency matrix.

ACKNOWLEDGMENT

The authors are grateful to BusPas Inc. and Mitacs Accelerate Program for supporting this research and also to Société de Transport de Laval for providing the data.

REFERENCES

[1] D. Luo, D. Zhao, Q. Ke, X. You, L. Liu, D. Zhang, H. Ma, and X. Zuo, "Fine-grained service-level passenger flow prediction for bus transit systems based on multitask deep learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7184–7199, 2020.

[2] L. Liu and R.-C. Chen, "A novel passenger flow prediction model using deep learning methods," *Transportation Research Part C: Emerging Technologies*, vol. 84, pp. 74–91, 2017.

[3] H. Zhai, L. Cui, Y. Nie, X. Xu, and W. Zhang, "A comprehensive comparative analysis of the basic theory of the short term bus passenger flow prediction," *Symmetry*, vol. 10, no. 9, p. 369, 2018.

[4] H. Zhu, X. Yang, and Y. Wang, "Prediction of daily entrance and exit passenger flow of rail transit stations by deep learning method," *Journal of Advanced Transportation*, vol. 2018, 2018.

[5] K. Zhang, Z. Liu, and L. Zheng, "Short-term prediction of passenger demand in multi-zone level: Temporal convolutional neural network with multi-task learning," *IEEE transactions on intelligent transportation systems*, vol. 21, no. 4, pp. 1480–1490, 2019.

[6] S. Hao, D.-H. Lee, and D. Zhao, "Sequence to sequence learning with attention mechanism for short-term passenger flow prediction in large-scale metro system," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 287–300, 2019.

[7] Y. Liu, Z. Liu, and R. Jia, "Deeppf: A deep learning based architecture for metro passenger flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 18–34, 2019.

[8] Y. Jing, H. Hu, S. Guo, X. Wang, and F. Chen, "Short-term prediction of urban rail transit passenger flow in external passenger transport hub based on lstm-lgb-drs," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[9] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, 2019.

[10] J. Li, H. Peng, L. Liu, G. Xiong, B. Du, H. Ma, L. Wang, and M. Z. A. Bhuiyan, "Graph cnns for urban traffic passenger flows prediction," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 29–36, IEEE, 2018.

[11] Y. Han, S. Wang, Y. Ren, C. Wang, P. Gao, and G. Chen, "Predicting station-level short-term passenger flow in a citywide metro network using spatiotemporal graph convolutional neural networks," *ISPRS International Journal of Geo-Information*, vol. 8, no. 6, p. 243, 2019.

[12] J. Zhang, F. Chen, Y. Guo, and X. Li, "Multi-graph convolutional network for short-term passenger flow forecasting in urban rail transit," *IET Intelligent Transport Systems*, vol. 14, no. 10, pp. 1210–1217, 2020.

[13] Y. He, L. Li, X. Zhu, and K. L. Tsui, "Multi-graph convolutional-recurrent neural network (mgc-rnn) for short-term forecasting of transit passenger flow," *arXiv preprint arXiv:2107.13226*, 2021.

[14] J. Wang, Y. Zhang, Y. Wei, Y. Hu, X. Piao, and B. Yin, "Metro passenger flow prediction via dynamic hypergraph convolution networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7891–7903, 2021.

[15] H. Peng, H. Wang, B. Du, M. Z. A. Bhuiyan, H. Ma, J. Liu, L. Wang, Z. Yang, L. Du, and S. P. Wang, Senzhang and, "Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting," *Information Sciences*, vol. 521, pp. 277–290, 2020.

[16] P. Chen, X. Fu, and X. Wang, "A graph convolutional stacked bidirectional unidirectional-lstm neural network for metro ridership prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[17] Y. He, Y. Zhao, H. Wang, and K. L. Tsui, "Gc-lstm: A deep spatiotemporal model for passenger flow forecasting of high-speed rail network," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2020.

[18] Y. Lu, H. Ding, S. Ji, N. Sze, and Z. He, "Dual attentive graph neural network for metro passenger flow prediction," *Neural Computing and Applications*, vol. 33, no. 20, pp. 13417–13431, 2021.

[19] L. Bai, L. Yao, X. Wang, C. Li, and X. Zhang, "Deep spatial–temporal sequence modeling for multi-step passenger demand prediction," *Future Generation Computer Systems*, vol. 121, pp. 25–34, 2021.

[20] S. Rahmani, A. Baghbani, N. Bouguila, and Z. Patterson, "Graph neural networks for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[21] M. Sangiorgio and F. Dercole, "Robustness of lstm neural networks for multi-step forecasting of chaotic time series," *Chaos, Solitons & Fractals*, vol. 139, p. 110045, 2020.

[22] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *Advances in neural information processing systems*, vol. 28, 2015.

[23] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv preprint arXiv:1906.00121*, 2019.

[24] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[25] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 1234–1241, 2020.