A Graph Neural Network Framework for Imbalanced Bus Ridership Forecasting

Samir Gupta*, Agrima Khanna*, Jose Paolo Talusan*, Anwar Said*,
Dan Freudberg†, Ayan Mukhopadhyay*, Abhishek Dubey*
*Vanderbilt University
Nashville, TN, USA
†WeGo Public Transit
Nashville, TN, USA

***Abstract*—Public transit systems are paramount in lowering carbon emissions and reducing urban congestion for environmental sustainability. However, overcrowding has adverse effects on the quality of service, passenger experience, and overall efficiency of public transit causing a decline in the usage of public transit systems. Therefore, it is crucial to identify and forecast potential windows of overcrowding to improve passenger experience and encourage higher ridership. Predicting ridership is a complex task, due to the inherent noise of collected data and the sparsity of overcrowding events. Existing studies in predicting public transit ridership consider only a static depiction of bus networks. We address these issues by first applying a data processing pipeline that cleans noisy data and engineers several features for training. Then, we address sparsity by converting the network to a dynamic graph and using a graph convolutional network, incorporating temporal, spatial, and auto-regressive features, to learn generalizable patterns for each route. Finally, since conventional loss functions like categorical cross-entropy have limitations in addressing class imbalance inherent in ridership data, our proposed approach uses focal loss to refine the prediction focus on less frequent yet task-critical overcrowding instances. Our experiments, using real-world data from our partner agency, show that the proposed approach outperforms existing state-of-the-art baselines in terms of accuracy and robustness.**

***Index Terms*—Public Transit, Big Data, Graph Neural Network, Time Series, Occupancy Prediction, Data Sparsity**

I. Introduction

The transportation sector has a massive environmental impact it contributes 24% of the total carbon dioxide emissions and in the United States, it accounts for 33% of the total greenhouse gas emissions [1, 2]. To address the environmental impact, the US government has increased the budget allocated to the public transportation sector — 42% annual increment from 2022 to 2026 as compared to 2016 to 2021 [3]. Public transportation is at the center of this vision as buses are the cheapest and most accessible form of transit across the country. Travel by bus results in fewer greenhouse gasses per passenger mile in comparison to a typical single-occupancy car [4]. However, public transit is only effective when it becomes a preferred mode of transport over private vehicles.

With this motivation in mind, there has been a significant increase in research to improve public transport systems and bring in additional consumers, thus guiding advancement in urban mobility. One approach to improving the transit experience is to reduce potential overcrowding through predicting demand and thus occupancy, which can inform downstream decisions, for example, routing. With the help of predictions made by these models, transportation operators can plan to shorten the gap between two consecutive buses during peak times and prevent overcrowding. However, the available data is inherently noisy and sparse, with the majority of high ridership incidents occurring very infrequently. This "heavy tail problem", is shown in Figure 1, where it is evident that trips with higher occupancy numbers are exceedingly rare.

Thus, significant efforts are being made to employ spatial and temporal characteristics in forecasting ridership in public transportation networks through the application of machine learning and deep learning techniques as they can learn complex abstractions from data [5, 6, 7]. **(a)** Liu et al. [8] depict the city's road network with a static graph, but their method overlooks the bus network's dynamic nature thus, adopting a dynamic graph strategy addresses this limitation; **(b)** Certain methodologies approach the task as a regression issue, which results in the "heavy tail problem" or they fail to utilize a loss function conducive to better learning of low-frequency classes; **(c)** Talusan et al. [9] tackle the problem of trip level day-ahead prediction, however, determining the occupancy at the stop level can help transit agencies alter the route patterns and headway to cope with high demand and improve serviceability.

Challenges and Contributions: We introduce a day-ahead, stop-level ridership prediction for public bus transit. Day-ahead prediction helps transit agencies determine the demand at every stop for the next day. We predict the ridership at every stop in a route over a time window. We make use of data sources such as Automatic Passenger Counting (APC) and General Transit Feed Specification (GTFS). However, ridership prediction of public bus transit from these data sources can be quite challenging. We have applied robust data pre-processing techniques to clean the data to remove the noise present in the datasets. To reduce the heavy tail problem, we convert the regression problem into a classification task, mitigating sparsity issues by transforming continuous output variables into discrete categories (Low, Medium, High, Overload). This approach simplifies the learning task, as the model focuses on distinguishing between predefined classes instead of predicting precise continuous values. Having discrete cate-

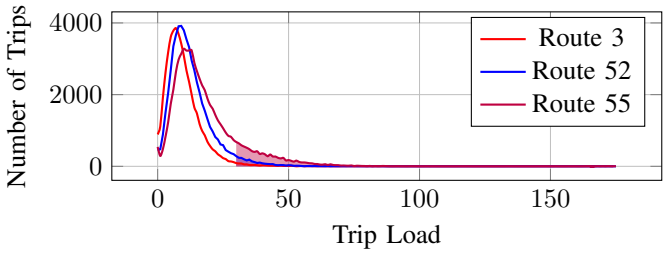

Fig. 1: Trips with > 30 riders are in the heavy tail.

gories also provides greater interpretability for transit agencies for example, knowing the 'high' and 'overload' categories can directly guide decisions on dispatching more buses or adjusting routes. We address sparsity by making use of state-of-the-art solutions for class imbalance such as the focal loss function [10] which makes the model more sensitive to sparse classes, to overcome class imbalance.

We then convert the transit networks into distinct graphs to leverage their structure and apply graph convolutional networks (GCNs) to the task of ridership prediction. By exploiting the graph topology of bus routes, where nodes represent bus stops and edges denote the path taken by the bus for a given route in a time window, GCNs are uniquely equipped to learn and predict the patterns of bus ridership across different time windows and routes. Through our experiments, we show how we solve the problems of sparsity and noise present in our data sources.

Organization: This paper is divided into the following sections: Section II describes the related work. In Section III we formulate the problem statement, Section IV highlights the data collection and pre-processing process. Section V explains the approach used in this paper. Section VI gives a brief on the results obtained. Section VII details the comparison between our model and its variants, highlighting the impact of different configurations. Finally, Section VIII discusses the future scope of the work presented in this paper.

II. Related Work

There has been a growing interest in developing predictive models for trip-level and stop-level ridership prediction of public bus transit. Talusan et al. [9] demonstrate the impact of altering the time window on day-ahead ridership forecasts utilizing XGBoost. They show how each feature influences the prediction, highlighting the substantial role that temporal attributes such as *hour* and *month* play in shaping the overall forecast accuracy. Wood et al. [7] and Yan et al. [5] employ a Random Forest Classifier (RFC) model for predicting real-time bus ridership and modeling the demand of ride-sourcing services. Their research showcases the effectiveness of RFC in handling models based on origin-destination (OD) pairs and time-series data for accurate predictions.

Numerous studies have been conducted on public transit occupancy prediction using a plethora of deep learning techniques, such as long-short term memory (LSTM) and multi-task learning (MTL). Jiao et al. [11] utilize an LSTM model to predict bus occupancy levels during the Covid-19 pandemic.

TABLE I: Description of Symbols

Symbol	Description
$\mathcal{R}$	Set of Routes
r	A single route $\in \mathcal{R}$
$\mathcal{T}$	Set of Trips
t	A single trip $\in \mathcal{T}$
$\mathcal{S}$	Set of Stops
s	A single stop $\in \mathcal{S}$
$\mathcal{W}$	Set of Time windows in a day
w	Time window during a day $\in \mathcal{TI}$
d	Historical Date
X	Input feature space
Y	Output space
$\hat{f}$	Objective function
θ	Set of parameters
θ^*	Optimal set of parameters
L	Loss function

Bin Zulqarnain et al. [12], adopt an alternative perspective on the occupancy prediction challenge, theorizing and showing that the integration of related tasks like occupancy and delay can improve the forecasting accuracy for both aspects.

Liu et al. [8] utilize GraphSAGE [13] for short-term traffic speed forecasting within an urban road network, emphasizing its application in dealing with missing link speed data. The framework involves processing vehicle trajectory data and constructing a segmented network, where a data recovery algorithm imputes missing speed data by considering nonlinear spatial and temporal correlations. This approach offers advantages like avoiding artificial settings required by most graph algorithms and facilitating a more intuitive and efficient representation and analysis of traffic speeds. However, when applied to public transit, it does not directly address dynamic edges, which is a limitation compared to the proposed approach that aims to capture the dynamic nature of the bus transit network more accurately.

III. Problem Statement

To satisfy the mobility demands in large urban cities, transit systems must operate efficiently. Hence, public bus transit systems must predict and monitor ridership levels. Overcrowding might prevent potential passengers from boarding the bus, lowering their willingness to use public bus transit systems. Accurately predicting ridership levels a day ahead enables transit agencies to allocate resources, enhance passenger experience, and take preventive measures if demand fluctuates. Thus, our problem is to precisely forecast the maximum ridership levels for a given stop, a day in advance.

In a public transit system, buses move along a series of designated routes $\mathcal{R}$, where $r_i \in \mathcal{R}$ consists of a total of k trips $\{t_1, t_2, \cdots, t_k\} \in \mathcal{T}$. An individual trip $t \in \mathcal{T}$ denotes a bus traveling in a single direction for one leg of a round trip. For every trip t^j we have a set of n sequentially ordered stops represented by $\{s_1^j, s_2^j, \cdots, s_n^j\} \in \mathcal{S}^j$. For our forecasting problem, we divide the day into 30-minute time windows $\mathcal{W}$, where $|\mathcal{W}| < 48$. Thus, for a route $r \in \mathcal{R}$ consisting of $\{s_1^j, s_2^j, \cdots, s_n^j\} \in \mathcal{S}_w^j$ stops in a time window $w \in \mathcal{W}$, our

goal is to forecast the occupancy at each stop $s_n^j \in \mathcal{S}^j$ in the future.

Let X represent the input feature space, where each element $x \in X$ corresponds to an individual data instance, precisely x denotes the set of features for a stop $s_n^j \in \mathcal{S}_w^j$ at the time window w and date d, where d refers to the historical date. Let Y represent the output space $\{Low, Medium, High, Overload\}$. Our objective is to learn a function $\hat{f} : X \to Y$ such that $\hat{f}(x')$ gives a prediction for the unseen input x' at time window w and date $d+1$, where $d+1$ refers to the date in the future. This function $\hat{f}$ is parameterized by a set of parameters θ in Table I. The learning process involves determining the optimal set of parameters θ^* that minimizes the loss function L:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{n} L(\hat{f}(x_i, \theta), y_i)$$

IV. Data Collection And Processing

The analysis and improvement of bus transportation systems is heavily data-driven. This section describes our data sources; the spatial and temporal characteristics present in our dataset; processes used for cleaning and augmentation of our dataset; and lastly, the features used for our graph neural network. Table II contains a list of all the features that were used, as part of the graph neural network model, as well as our baselines.

A. Datasets

We use Automatic Passenger Count and General Transit Feed Specification data for the WeGo Public Bus Transit system in Nashville, TN, spanning from January 2022 to March 2023. Below is a summary of the data sources:

Automatic Passenger Counting (APC): Provides information on the number of people that enter and exit the bus at each stop in a trip (also called ons and offs). The actual ridership was recorded by infrared door sensors on the bus. Each entry in APC serves as a journal and logs the current state of the bus at each stop in a trip, scheduled arrival time, scheduled departure time, and actual arrival and departure times.

General Transit Feed Specification (GTFS): This is a static data prepared by the transit agency that details the schedule of all the trips and routes taken by the buses for a certain period [14]. Other information such as geometric routes, scheduled arrival time and scheduled departure time are used to determine the scheduled headway along with traffic data.

Traffic: We use road segments present in traffic data and match the stops within these segments using latitude and longitude coordinates. We also join on temporal properties, with traffic data having a frequency of five minutes.

Weather: We acquired the weather data from Darksky [15]. The weather data was matched with APC based on the geographic locations of the stops and temporal attributes such as year, month, day, and hour.

Holidays: These are based on publicly available federal and state calendars that list school breaks and public holidays.

B. Data Cleaning and Augmentation

Raw APC data from buses is noisy and typically contains missing values for some attributes. We identified many entries with missing scheduled times and imputed these data points by merging them with GTFS. We proceed to remove entire trips with inaccurate or dirty entries. We perform cleaning and filtering at the trip level even though APC data is collected at the stop level; APC data captures events each time a bus arrives at a stop. Our approach involves identifying undesirable entries based on specific criteria, followed by removing entire trips associated with those entries [9]. Finally, we merge the resulting data with other datasets (weather, traffic, and holidays).

C. Feature Engineering

We selected a total of 14 features from the APC, GTFS, traffic, weather, and holiday datasets. Among these features, we one hot encoded three of them — *route direction name*, *is holiday*, and *is weekend*. We converted four categorical features into numbers using a label encoder — *time window*, *stop sequence*, *year* and *month*. We numerically encoded the remaining seven features using a min-max scaler [16]. We also introduce an auto-regressive feature, *past* 3-*week load*, that contains information regarding the average occupancy at the stop over the past three weeks within the same time window, this feature helps the model get information regarding how public bus transit ridership over the past three weeks affects the current ridership.

From the data, we select only trips that: **(1)** have route directions *to* and *from* downtown; **(2)** do not include any disruptions; and **(3)** do not have the past 3-week load feature as a null value. Finally, we bin the occupancy data based on the discussions with WeGo Public Transit, to be able to identify instances where buses have high occupancy or are overloaded.

- Low: 0% to 33% of total vehicle capacity.
- Medium: 33% to 66% of total vehicle capacity.
- High: 66% to 100% of total vehicle capacity.
- Overload: 100%+ of total vehicle capacity.

V. Methodology

In this section, we present our methodology for predicting ridership. We define how we translate bus networks into graphs and how we use graph neural network models. We show how focal loss increases sensitivity to heavy tail problems. Finally, we enumerate hyperparameter search and bootstrap processes.

A. Representing Bus Networks as Graphs

We create a unique graph for each of the three routes, $\{3, 52, 55\}$, that encapsulates the unique characteristics and dynamics of each route. Our analysis divides the day into 48 distinct time intervals, each spanning 30 minutes, to cover daily dynamics. As the headway for buses ranges between 15 to 60 minutes; a 30-minute interval ensures that all buses within this headway range are captured without significant loss of detail that could result from averaging information across different buses in the same route. We model the bus

TABLE II: Data Features and Sources

Dataset	Features	Source	Frequency	Type	Description
Transit	Route ID	APC	N/A	Temporal	Route ID of the current route
	Route direction name	APC	N/A	Spatio-temporal	Name of route direction
	Year	APC	N/A	Temporal	The year of the trip
	Month	APC	N/A	Temporal	The month of the trip, 12 unique values
	Time Window	Derived	N/A	Temporal	Window of time of the day (each window is 30 min each)
	Stop sequence	APC	N/A	Spatio-Temporal	Captures the order and context of stops within a given route
	Distance traveled	GTFS	N/A	Spatial	The distance covered from the previous stop to the current stop
	Occupancy (Prediciton Output)	Derived	N/A	Spatio-temporal	Total occupancy at the trip (after alights and boards)
Traffic	Average Speed	Traffic	5 minutes	Spatio-temporal	Median speed observed on a specific road segment
	Congestion	Traffic	5 minutes	Spatio-temporal	Measure of vehicle travel density on major roadways
Weather	Precipitation Intensity	Darksky	1 hour	Spatio-temporal	Recorded Precipitation Intensity
	Humidity	Darksky	1 hour	Spatio-temporal	Recorded water vapor present in the air
	Temperature	Darksky	1 hour	Spatio-temporal	Recorded Temperature measured in Fahrenheit
Holidays	School breaks	Calendar	1 day	Temporal	Scheduled school breaks and holidays in a calender year
	National holidays	Calendar	1 day	Temporal	National holidays
Auto-regressive	Past Week Load	Derived	Variable	Spatio-temporal	Average of the occupancy at the stop over the past three weeks within the same time window

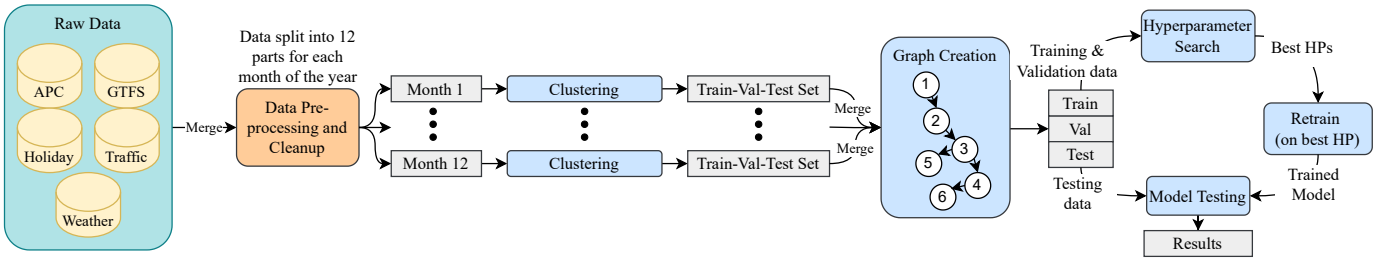

Fig. 2: Workflow diagram showing the GCN model pipeline for occupancy prediction, from data preprocessing and clustering to hyperparameter optimization and model testing.

transit network for individual routes using an approach that combines static nodes with dynamic edges for a dynamic graph representation [17]. Nodes represent all the stops in the route and edges represent the path along which the bus travels for a route in a given time window of 30 minutes. We create a single graph for each time window in the day for each route. Our methodology employs a dynamic graph model to account for the variable patterns of bus routes, which evolve due to factors like road closures, construction, and adjustments made to enhance service in specific regions.

Thus, for a route $r \in \{3, 52, 55\}$, we create a digraph that consists of nodes and edges denoted by $G_r^w = (N_r^w, E_r^w)$, where G_r^w represents a graph for a route r and time window $w \in \mathcal{W}$, where $|\mathcal{W}| < 48$; N_r^w depicts all the stops (nodes) in route r from January 2022 to March 2023 throughout our dataset, E_r^w represents the (edges) path along which the buses travel for a route r in the time window w. Thus two nodes are connected by an edge if a bus passes through those stops in the corresponding time window.

B. Graph Neural Network Models

Once we convert the routes in a public transit network into graph representations, we use these graphs as input for a machine learning model to forecast bus ridership. We use GCN layers to leverage the structure of the graph to perform convolutions directly on the graph. The essential operation of a GCN involves aggregating feature information from a node's neighbors and combining it with the node's own features to generate a new representation for the node. This aggregation

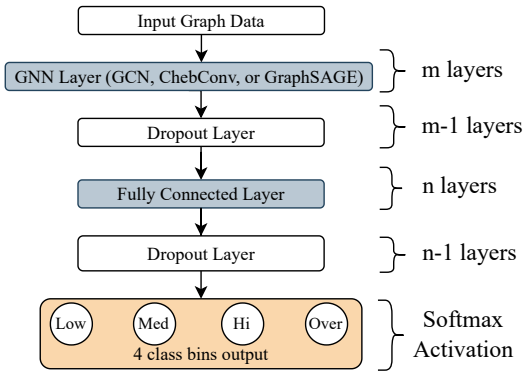

Fig. 3: Generalized graph neural network (GNN) architecture for node classification.

is key for the ridership task, as this helps the model learn the effect of neighboring stops on the occupancy of the current stop. The layer-wise propagation rule for a GCN is given by eq. (1) [18], where X' is the output feature matrix after applying the graph convolution, where each row represents the updated features of a node; σ denotes a non-linear activation function, in our work ReLU ($\sigma(x) = \max(0, x)$), is applied element-wise to the result of the graph convolution operation; $\widetilde{D}$ is the diagonal degree matrix; X is the input feature matrix; θ is the weight matrix associated with the layer.

$$X' = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}X\theta) \tag{1}$$

TABLE III: Percentage of Stops by Class for Each Route

Class / Bin	Route 3	Route 52	Route 55
Low Occupancy (Bin 0)	93.74%	83.61%	79.90%
Medium Occupancy (Bin 1)	5.36%	13.81%	14.96%
High Occupancy (Bin 2)	0.67%	2.07%	3.58%
Overload Occupancy (Bin 3)	0.23%	0.51%	1.55%

Finally, we create a unique model for each route. Each input branch is followed by m GCN layers, in every GNN layer we have x hidden channels. Following the m GNN layers we include n fully connected (dense) layers, and each fully connected layer contains y hidden neurons. To prevent the model from overfitting, each GNN and dense layer has a dropout layer (except the final GNN and dense layer), hence our model has $n + m - 2$ dropout layers and z dropout rate. Finally, we have a single output layer with softmax as the activation function. We determine the values of $m, n, x, y,$ and$z$ through hyperparameter search. Fig. 2 and 3 depict the GNN model framework and the architecture used in our work respectively.

C. Dealing with Imbalanced Data

We use the focal loss function [10] to address the class imbalance present in ridership classes. This approach is particularly effective when dealing with imbalanced datasets, as it modifies the standard cross-entropy loss to apply a weighting factor to the loss function. This factor decreases the loss contribution from easy examples and amplifies the loss for hard, misclassified examples, thus focusing more on the minority classes which are harder to predict but more critical. We determine the weighting factor using the formula mentioned in eq. (2), where α_i is the weighting factor for class $i \in \{0 : Low, 1 : Medium, 2 : High, 3 : Overload\}$ and $Instance_i$ is the number of instances of class i present in the dataset.

$$\alpha_i = \frac{\text{Total Instances}}{\text{Instances}_i} \tag{2}$$

In our case, the 'High' and 'Overload' categories represent the minority classes with significantly less data compared to the 'Low' and 'Medium' classes as depicted in table III. The focal loss function is given in eq. (3), where p_i is the model's estimated probability for the class with label i, α_i is the weighting factor for class i to balance class frequencies as represented in eq. (2), and γ is a focusing parameter that adjusts the rate at which easy examples are down-weighted.

$$FocalLoss(p_i) = \alpha_i(1 - p_i)^\gamma \log p_i \tag{3}$$

D. Hyperparameter Search and Bootstrap

We implemented a hyperparameter search strategy utilizing 10-fold cross-validation (CV). We leveraged Bayesian optimization [19], a probabilistic model-based optimization technique for the hyperparameter search process. We also use bootstrap aggregation to capture the reliability and stability of our models. We conducted three bootstrap iterations, using 10-fold cross-validation (CV) within each bootstrap sample. We have reported the range of values for each hyperparameter in table IV. We utilize Ray [20], an open-source framework that enables simple and efficient parallelization of tasks to manage hyperparameter searches across multiple bootstrap samples.

TABLE IV: Hyperparmeters for Day Ahead - Stop level GNN models

Hyperparameter Name	Range
Epochs	[300, 600]
Patience	[50, 200]
Learning Rate	[0.0001, 0.1]
GNN Layers	[1, 3]
GNN hidden channels	[1, 128]
Chebyshev Filter	[1, 5]
Dense layer count	[1, 3]
Dense hidden channels	[1, 256]
Dropout Rate	[0.0,0.5]
Random Seed	[0, 300]

VI. Results and Discussion

In this section, we provide an evaluation of our proposed framework and compare the results against several state-of-the-art baselines. We conducted our experiments on 36-core Intel Core i9-10980XE with NVIDIA GeForce RTX 3090 (24GB) and 125 GB RAM, and 96-Threads Intel Xenon Gold 6240R and 192 GB RAM, provided by the Chameleon testbed [21].

A. Baselines

We evaluate the proposed method's effectiveness for ridership forecasting by considering the following state-of-the-art baselines:

- **ChebConv** [22]: This method leverages Chebyshev polynomials as convolutional filters to capture localized graph patterns, enabling efficient learning of node representations.
- **GraphSAGE** [8]: This method exploits the node features and topological structure of each node's neighbor to generate new node representations.
- **Random Forest Classifier** [5, 7]: This method builds numerous decision trees during the training phase and integrates their predictions to boost overall accuracy and prevent overfitting.
- **XGboost** [9]: This method employs an ensemble of decision trees, refined iteratively via gradient boosting.

B. Experimental Setup

Our focus is on routes leading *to* and *from* the downtown area, motivated by the observation that overcrowding predominantly occurred in that area, with routes outside this area experiencing minimal overall trips. Consequently, from a total of 35 unique bus routes, there are only 21 that cover the most densely populated areas of the city. Among these, we choose the three routes, $\{3, 52, 55\}$, based on the highest quarterly ridership [23, 24]. In our study, we systematically partitioned the dataset into three distinct subsets: training, validation, and testing. To closely mimic real-world operational scenarios and ensure external validation of our results, we constructed the validation and test sets to reflect monthly variations in

TABLE V: Mean and standard deviation for standard and weighted F1-score, precision, recall, and MCC for all models, tested on Routes 3, 52, and 55 data from January 2022 to March 2023.

Model	Route 3				Route 52				Route 55			
	F1	Precision	Recall	MCC	F1	Precision	Recall	MCC	F1	Precision	Recall	MCC
GCN	79.3 ± 0.69	**93.5 ± 0.12**	70.8 ± 1.09	22.6 ± 0.67	70.0 ± 0.57	84.6 ± 0.13	63.3 ± 0.77	29.6 ± 0.46	72.1 ± 0.48	**81.7 ± 0.15**	67.2 ± 0.65	34.9 ± 0.54
CHEB	72.5 ± 5.82	93.1 ± 0.38	61.7 ± 7.30	16.5 ± 1.41	69.1 ± 4.65	84.4 ± 0.42	63.0 ± 5.99	29.7 ± 3.72	71.2 ± 1.28	81.4 ± 0.19	66.0 ± 1.75	33.5 ± 1.58
GSAGE	77.4 ± 1.08	**93.5 ± 0.13**	68.1 ± 1.70	21.3 ± 1.07	67.8 ± 5.56	**84.7 ± 0.21**	61.3 ± 6.52	29.2 ± 3.68	69.3 ± 1.82	81.0 ± 0.50	63.4 ± 2.43	30.6 ± 1.13
RFC	**93.6 ± 0.20**	93.2 ± 0.20	94.2 ± 0.21	**40.7 ± 1.32**	**84.9 ± 0.05**	84.4 ± 0.11	85.7 ± 0.10	**45.1 ± 0.32**	75.2 ± 0.94	76.8 ± 3.00	77.2 ± 0.24	34.0 ± 0.88
XGB	93.5 ± 0.16	93.2 ± 0.16	**94.6 ± 0.10**	38.2 ± 0.80	84.8 ± 0.12	84.2 ± 0.15	**86.6 ± 0.09**	44.1 ± 0.46	**81.8 ± 0.15**	80.9 ± 0.17	**83.4 ± 0.06**	**44.9 ± 0.39**
Model	$\hat{F1}$	$\hat{Precision}$	$\hat{Recall}$	$\hat{MCC}$	$\hat{F1}$	$\hat{Precision}$	$\hat{Recall}$	$\hat{MCC}$	$\hat{F1}$	$\hat{Precision}$	$\hat{Recall}$	$\hat{MCC}$
GCN	**50.5 ± 1.12**	51.6 ± 1.22	**51.3 ± 0.65**	**35.4 ± 0.81**	**52.9 ± 0.42**	54.3 ± 1.06	**52.8 ± 0.30**	**37.4 ± 0.50**	**57.7 ± 0.90**	**57.8 ± 0.97**	**57.7 ± 0.97**	**43.7 ± 1.33**
CHEB	47.6 ± 3.05	48.7 ± 3.07	48.3 ± 2.29	31.5 ± 2.85	50.6 ± 1.48	53.6 ± 2.02	51.3 ± 1.61	36.0 ± 2.19	56.7 ± 0.13	56.8 ± 0.23	56.9 ± 0.04	42.6 ± 0.05
GSAGE	**50.5 ± 1.21**	**52.0 ± 0.94**	51.1 ± 0.62	35.2 ± 0.67	51.3 ± 1.54	**55.7 ± 1.97**	51.7 ± 1.79	36.6 ± 2.23	55.1 ± 1.31	55.5 ± 1.57	55.4 ± 0.83	40.7 ± 1.04
RFC	36.8 ± 0.56	49.0 ± 0.44	42.1 ± 0.51	25.5 ± 0.36	38.2 ± 0.75	49.2 ± 1.37	43.7 ± 0.61	27.0 ± 0.85	45.8 ± 3.41	48.5 ± 7.97	52.6 ± 2.25	39.0 ± 2.71
XGB	30.9 ± 2.56	46.3 ± 2.03	37.7 ± 1.65	20.0 ± 1.91	31.0 ± 0.93	47.7 ± 2.05	38.7 ± 0.64	20.5 ± 1.02	44.0 ± 0.28	49.9 ± 0.22	46.9 ± 0.19	30.7 ± 0.31

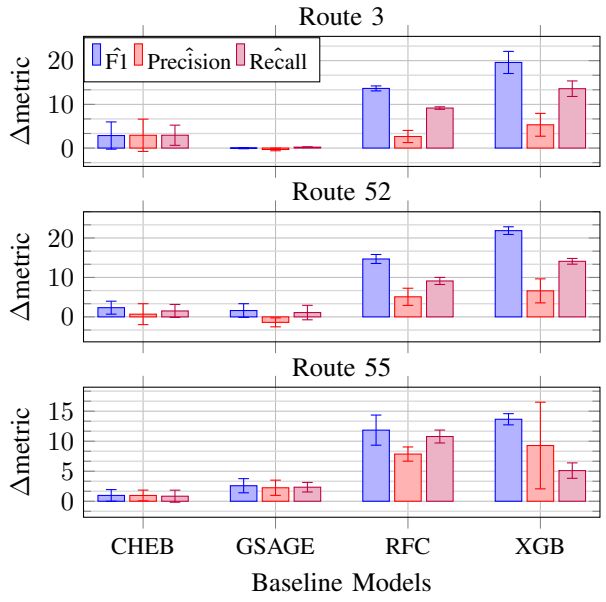

Fig. 4: Relative performance of baseline models in terms of weighted $\hat{F1}$, $\hat{precision}$, and $\hat{recall}$ against our GCN model for Routes 3, 52, and 55. Positive values indicate metrics where GCN outperforms the baseline methods.

ridership patterns. This approach acknowledges the temporal dynamics inherent in public transit usage, which can fluctuate based on factors like the day of the week and seasonal changes. For each month, we first clustered the data (using spectral clustering) based on key features: *is weekend*, *number of patterns*, *number of trips*, and *time window of first and last trip*. From each cluster, we uniformly sampled one day at random to form the validation set and similarly sampled a different day for the test set. This clustering aims to capture the intrinsic structures and recurring patterns within the data. By repeating this process for three iterations, we created our train-validation-test samples for each bootstrap iteration. We used the train-validation sets for hyperparameter search with K-fold validation, as described in the previous sections. Finally, after obtaining the best hyperparameters, we trained the models on both the train and validation sets, obtained the final model, and reported the results.

C. Discussion

We addressed the challenge of predicting bus occupancy across multiple routes with varying levels of data availability, specifically focusing on the sparse classes 'high' and 'overload'. Our approach utilizes a GCN model with separate models trained for each route to accurately capture route-specific characteristics. The performance of our GCN model was compared to state-of-the-art baselines.

$$\Delta\text{metric} = \text{metric}_{GCN} - \text{metric}_{method} \tag{4}$$

The GCN model coupled with focal loss demonstrated superior performance, particularly in identifying the 'overload' classes, where data is inherently sparse. Despite the limited label availability for this class, the model achieved significantly better weighted $\hat{F1}$ scores and Matthews Correlation Coefficient ($\hat{MCC}$) values as described in fig. 4 and table V. The MCC is particularly effective for models' evaluation on imbalanced datasets as it offers a balanced measure capable of effectively handling varying class sizes [16]. The performance metrics in fig. 4 are quantified by the percentage difference from the mean value with the GCN model as reference, as defined in eq. (4). Error bars represent the variance, indicating the consistency of GCN's relative performance across different evaluations. Models with a positive value are outperformed by our approach while those with negative values show models that outperform GCN. Our model is outperformed when evaluated using conventional metrics, however it demonstrates superior performance upon analysis with weighted metrics. This discrepancy arises because weighted metrics account for class imbalance by assigning greater importance to the accurate classification of underrepresented classes. Consequently, although other models may excel in predicting outcomes for classes with higher label counts, they fall short in classifying instances of the sparse 'overload' class.

Focal loss contributes to improved performance by reducing the relative loss for well-classified examples and putting more emphasis on correcting misclassified data points, effectively preventing the overwhelming number of 'normal' class samples from diluting the contribution of the sparse 'overload' class during training. This targeted approach ensures that our model does not become biased towards the majority class and improves the robustness of predictions across all classes. This success is also attributed to the GCN's ability to capture

TABLE VI: Mean and standard deviation for standard and weighted F1-score, precision, recall, and MCC, for our GCN model and its variants tested on Routes 3, 52, and 55 data of March 2023.

	Route 3				Route 52				Route 55			
Model	F1	Precision	Recall	MCC	F1	Precision	Recall	MCC	F1	Precision	Recall	MCC
GCN	68.8 ± 3.60	**93.1 ± 0.70**	56.8 ± 4.64	18.4 ± 0.79	65.3 ± 5.50	79.6 ± 0.77	60.0 ± 7.10	28.7 ± 4.36	68.3 ± 2.50	80.0 ± 0.65	62.8 ± 2.92	32.3 ± 1.53
GCN-C	**92.0 ± 0.53**	91.8 ± 0.56	**92.4 ± 0.54**	**36.5 ± 2.54**	**79.4 ± 1.04**	79.5 ± 0.87	**79.9 ± 1.36**	**41.4 ± 2.01**	**78.8 ± 1.10**	79.9 ± 0.76	**78.0 ± 1.33**	**43.2 ± 0.64**
GCN-A	86.3 ± 3.50	92.5 ± 0.90	82.4 ± 5.75	30.6 ± 1.12	64.8 ± 7.84	**79.8 ± 0.36**	59.4 ± 9.45	28.6 ± 7.01	61.7 ± 7.55	**80.1 ± 0.71**	55.4 ± 8.64	27.5 ± 5.44
Model	$\widehat{\text{F1}}$	$\widehat{\text{Precision}}$	$\widehat{\text{Recall}}$	$\widehat{\text{MCC}}$	$\widehat{\text{F1}}$	$\widehat{\text{Precision}}$	$\widehat{\text{Recall}}$	$\widehat{\text{MCC}}$	$\widehat{\text{F1}}$	$\widehat{\text{Precision}}$	$\widehat{\text{Recall}}$	$\widehat{\text{MCC}}$
GCN	**54.5 ± 3.29**	**59.2 ± 6.33**	**54.7 ± 3.71**	**41.3 ± 6.26**	46.8 ± 2.80	50.9 ± 0.45	47.7 ± 2.62	31.1 ± 3.08	**57.6 ± 0.22**	**58.4 ± 0.30**	**57.7 ± 0.26**	**43.7 ± 0.37**
GCN-C	28.5 ± 4.74	33.9 ± 8.75	37.1 ± 2.86	18.6 ± 4.10	39.1 ± 2.31	46.4 ± 1.28	44.0 ± 1.95	27.1 ± 2.45	52.6 ± 1.49	52.9 ± 0.89	54.3 ± 1.52	39.5 ± 1.91
GCN-A	50.7 ± 11.8	55.8 ± 18.2	53.8 ± 8.99	40.2 ± 12.0	**47.9 ± 4.32**	**51.5 ± 2.88**	**48.4 ± 4.81**	**32.0 ± 6.15**	55.5 ± 2.17	57.9 ± 1.31	55.8 ± 1.74	41.7 ± 2.14

the complex topological structures of the data, enabling it to recognize patterns that indicate 'overload' even from a small number of examples.

Unlike models that may require a substantial volume of samples to learn effectively, our model's architecture allows it to amplify the signal from the scarce 'overload' labels. This capacity stems from the strength of graph-based learning in situations where traditional models might struggle due to the sparsity of critical labels. Additionally, the GCN model's consistent performance across different routes indicates its robustness and suggests that it is well-suited in the context of unbalanced class distribution. Our model achieves an average accuracy of 71% for route 55, 46% for route 52, and 34% for route 3 across all bootstrap iterations. The ability to accurately predict 'overload' scenarios is of high practical relevance, as it enables better management of bus fleet operations and improves passenger experience by preventing overcrowding.

VII. Ablation Study

The ablation study was conducted to assess the influence of certain components and methodological decisions on the performance of our graph neural network models in the task of bus ridership prediction. For this study, we focused on the data from a single month, March 2023, to maintain consistency and control in our experimental comparisons. In table V, GCN with focal loss (denoted as GCN) is compared to models with varying aspects in this study: **(a)** Loss Function (GCN-C): We assessed the effect of changing the loss function from Focal Loss to Categorical Cross-Entropy; **(b)** Model Structure (GCN-A): We evaluated the difference between modeling each bus route separately (our approach) and aggregating all routes: $\{3, 52, 55\}$ into a single model.

Overall, we have a significant improvement by using Focal loss as our loss function, wherein, we see more than 10% improvement for routes with highly imbalanced data across all metrics. From fig. 5, we observe that our method of deploying a single model per route excels over the all-routes approach for 2 routes; however, for Route 52, our model is surpassed. This discrepancy can likely be attributed to the reduced variance in our model's predictions, as depicted in table V, suggesting a more consistent performance across different scenarios. The observed shortfall in performance for Route 52 may stem from the limited data available for training our model, which restricts its learning capacity and adaptability to the unique patterns of this specific route. Moreover, our approach offers additional advantages over the all-routes approach, allowing

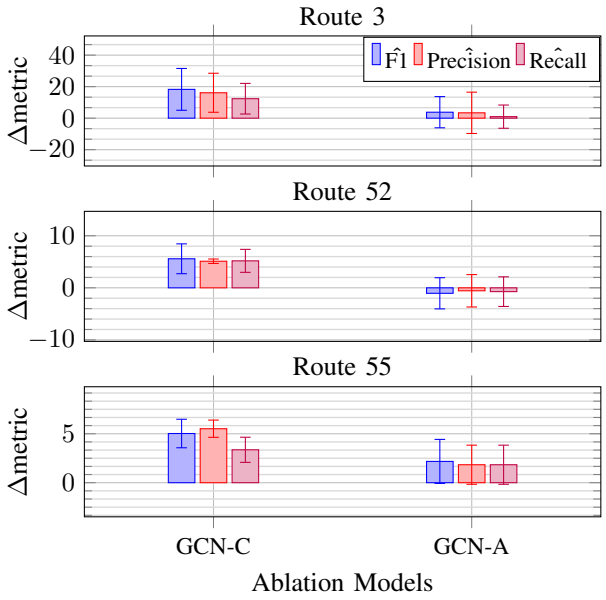

Fig. 5: Relative performance of ablation study models in terms of weighted $\widehat{\text{F1}}$, $\widehat{\text{precision}}$, and $\widehat{\text{recall}}$ against our GCN model for Routes 3, 52, and 55. Positive values indicate metrics where GCN outperforms the baseline methods.

transit agencies the flexibility to modify, add, or remove routes without affecting the predictions for other routes. This flexibility can extend to adjusting route patterns and scheduled times, providing agencies with greater control and adaptability in their operations.

VIII. Conclusion

We introduce an innovative Graph Neural Network framework for the task of forecasting bus ridership in scenarios where data distribution is highly imbalanced. Our framework encapsulates the dynamic spatial interconnections inherent to bus transit networks through the utilization of dynamic graphs. A pivotal component of our approach is the adoption of focal loss, which effectively addresses the issue of class imbalance. Comparative analyses with state-of-the-art baselines, on real-world bus transit data, demonstrate the superior predictive power of our GCN model for stop level day ahead ridership forecasting.

Acknowledgment

This material is based upon work sponsored by the National Science Foundation under Award Numbers 1952011 and

2238815, and by the Federal Transit Administration COVID-19 Research Grant under Federal Award Identification Number TN-2021-015-00. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF), the Federal Transit Administration, or the Department of Energy. Results presented in this paper were obtained using the Chameleon Testbed supported by the NSF.

REFERENCES

[1] S. Awaworyi Churchill, J. Inekwe, K. Ivanovski, and R. Smyth, "Transport infrastructure and CO2 emissions in the OECD over the long run," *Transportation Research Part D: Transport and Environment*, vol. 95, 2021.

[2] Climate action | US department of transportation. [Online]. Available: https://www.transportation.gov/priorities/climate-and-sustainability/climate-action

[3] Government spending on public transportation and other infrastructure | congressional budget office. [Online]. Available: https://www.cbo.gov/publication/58086

[4] T. Hodges and United States. Federal Transit Administration. Office of Policy Development, "Public transportation's role in responding to climate change [january 2010]," Federal Transit Administration, Tech. Rep., 2010. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/17277

[5] X. Yan, X. Liu, and X. Zhao, "Using machine learning for direct demand modeling of ridesourcing services in chicago," *Journal of Transport Geography*, vol. 83, 2020.

[6] K.-F. Chu, A. Y. S. Lam, and V. O. K. Li, "Deep multi-scale convolutional LSTM network for travel demand and origin-destination predictions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, 2020, conference Name: IEEE Transactions on Intelligent Transportation Systems.

[7] J. Wood, Z. Yu, and V. V. Gayah, "Development and evaluation of frameworks for real-time bus passenger occupancy prediction," *International Journal of Transportation Science and Technology*, vol. 12, no. 2, 2023.

[8] J. Liu, G. P. Ong, and X. Chen, "Graphsage-based traffic speed forecasting for segment network with sparse data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, 2022.

[9] J. P. Talusan, A. Mukhopadhyay, D. Freudberg, and A. Dubey, "On designing day ahead and same day ridership level prediction models for city-scale transit networks using noisy APC data," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022.

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[11] F. Jiao, L. Huang, R. Song, and H. Huang, "An improved STL-LSTM model for daily bus passenger flow prediction during the COVID-19 pandemic," *Sensors*, vol. 21, no. 17, 2021, number: 17 Publisher: Multidisciplinary Digital Publishing Institute.

[12] A. Bin Zulqarnain, S. Gupta, J. P. Talusan, D. Freudberg, P. Pugliese, A. Mukhopadhyay, and A. Dubey, "Addressing apc data sparsity in predicting occupancy and delay of transit buses: A multitask learning approach," in *2023 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2023.

[13] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018.

[14] Pioneering open data standards: The GTFS story. [Online]. Available: https://beyondtransparency.org/part-2/pioneering-open-data-standards-the-gtfs-story/

[15] The Dark Sky Company, LLC, "Dark Sky API," https://darksky.net/dev, 2012–2021, accessed: March 15, 2023.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.

[17] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs."

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks."

[19] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014–.

[20] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training."

[21] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

[22] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering."

[23] Nashville Metropolitan Transit Authority, "Final board book 02/22/2023," 2023. [Online]. Available: https://www.wegotransit.com/assets/1/27/MTA_Board_Book_2.22.2024_FINAL.pdf?1876

[24] ——, "Final board book 08/24/2023." [Online]. Available: https://www.wegotransit.com/assets/1/27/Final_Board_Book_8.24.2023.pdf?1687