TO WHAT EXTENT CAN A GRAPH CONVOLUTIONAL NEURAL NETWORK BE USED TO PREDICT PASSENGER INFLOW?

TENZIN RINCHEN DORJEE

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

2061555

COMMITTEE

ir. Federico Zamberlan
dr. Boris Čule

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

January 20, 2023

ACKNOWLEDGMENTS

This thesis has been a long time coming and has definitely put me to the test. It has been a journey of learning about data science, but even more so about having to learn myself. It is not without the support, guidance and enthusiasm of my supervisor Federico Zamberlan that I could have finished my thesis. His dedication to his students is unmatched, better supervisors are hard to come by. I would also like to thank my family for being there for me, especially my parents but also my sisters.

TO WHAT EXTENT CAN A GRAPH CONVOLUTIONAL NEURAL NETWORK BE USED TO PREDICT PASSENGER INFLOW?

TENZIN RINCHEN DORJEE

Abstract

As cities grow bigger and denser, the need for sustainable urban planning has increased with climate change already impacting urban life. A well-functioning public transport network is fundamental in a sustainable city, minimising congestion and reducing emissions. Correctly forecasting public transport demand is thus a socially relevant task. Traditional methods have a hard time addressing both the spatial and temporal nature of this problem, often dealing with the two in isolation instead of simultaneously. This paper focuses on the recent rise of Graph Neural Networks, which are neural networks adapted to handle graph-like data structures natively. GNNs currently achieve state-of-the-art performances on various prediction and classification tasks, among which in the traffic network domain. In this paper a Graph Convolutional Network (GCN) has been trained on bus passenger inflow data in Montevideo (Uruguay), spanning a period of five months with open data published by the bus operating companies (STM). Comparing the results with the established gradient boosted tree ensemble method XGBoost shows that the GCN outperforms in almost every metric. GCN performs better in the short term and drops off only a little when using a bigger prediction window, whilst the performance of XGBoost decreases a lot more in the long term. Comparatively GCN also tends to have less extreme prediction errors.

1 INTRODUCTION

Cities are growing at a rapid rate, with an increasing world population concentrating themselves more densely than before (Duranton & Puga, 2014). Consequently, the number of urban road users have risen, whilst space is limited in the city. At the same time, the world is dealing with a severe climate change crisis. Climate change is one of the biggest challenges to current society, affecting multiple systems in transition such as 'energy; land, ocean, coastal and freshwater ecosystems; urban, rural and infrastructure; and industry and society' (IPCC, 2022). In the same report by IPCC a growing realisation is established that these systems are very much interlinked and must all be tackled when trying to slow down climate change and when trying to minimise its impact. With these two important trends in mind, cities are consciously starting to implement sustainability efforts in their urban planning (Wamsler, Brink, & Rivera, 2013). A common trait of these efforts is the focus on reducing (future) congestion, as congestion makes for an increase in harmful emissions such as carbon dioxide whilst also increasing travel times for inhabitants. The encouragement of a modal shift towards bicycles and public transport is often central in these plans combatting congestion (Miller, de Barros, Kattan, & Wirasinghe, 2016).

A well-functioning public transport network is fundamental in a sustainable city (Miller et al., 2016). However, a popularly used public transport network can also become the victim of congestion issues if poorly planned. As such, poorly planned public transport becomes a waste of public funds, harming the city's sustainability efforts and clogging other transport modes (Miller et al., 2016). In order to plan a public transport network properly, demand must be forecasted accordingly – stations forecasted to be in high demand can then be serviced more often. Ensuring a proper demand forecast for the public transport network is thus socially relevant. The increase in available data over the past years allows for new planning instruments to arise and for improvements to existing methods to help plan a public transport network. This recent increase in data availability is dubbed as the Digital Revolution (Allam, 2020) and is illustrated by the countless of sensors embedded in the urban environment such as road sensors. It gives way for new data-based techniques and provides, along with pressing challenges for the urban environment, the backdrop for this research.

Traffic demand forecasting is both a problem of temporal and spatial nature (Miglani & Kumar, 2019). In particular, predicting demand in traffic networks is a graph problem. Graphs are a way of representing non-Euclidean data as a structure that depict existing relationships between objects as edges between nodes respectively. Unlike traditional machine learning (ML) methods, graph neural networks (GNNs) are designed to fully capture the associations a graph structure entails. In short, this is done by a 'message passing scheme that propagates information of nodes to its neighbours' (Munikoti, Agarwal, Das, Halappanavar, & Natarajan, 2022). However, due to its relatively recent introduction, comparatively not much literature exists on the application of GNNs on traffic flow prediction for public transport networks.

Current literature about GNN application on bus network data mostly focuses on Europe and Asia, whilst having significantly less data than four months' worth (Han et al., 2019; J. Zhang, Chen, Guo, & Li, 2020). Previous research on Montevideo's public transport networks only incorporated non-ML spatiotemporal analysis (Massobrio & Nesmachnow, 2020), highlighting the gap in literature this research aims to fill. The main research question flows out of the aforementioned gap in literature:

> *To what extent can a Graph Convolutional Neural Network be used to predict passenger inflow?*

In order to answer this question, the following two sub research questions are formulated:

RQ1 *To what extent are there differences between short term prediction performance vs long term prediction performance of Graph Convolutional Neural Network?*

RQ2 *How does the prediction performance of a Graph Convolutional Neural Network compare to a conventional shallow learning model like a gradient boosted tree ensemble method (XGBoost)?*

These questions will be answered using a dataset spanning a 5-month period, from February 2022 until June 2022. The dataset covers transactional data of the unified public bus transport system in Montevideo, Uruguay and is published by the Metropolitan Transportation System (STM) of Montevideo (Sistema de Transporte Metropolitano, 2022).

2 LITERATURE REVIEW

2.1 *The domain of traffic flow prediction*

The issue of predicting passenger inflow on public transport networks can be seen as a classic time series analysis regression problem (Miglani & Kumar, 2019). Time series data is a set of data points that are collected over a period of time. This makes it different from other data, which is usually a single snapshot of data from a particular moment in time. Public transport usage is often-times habitual; an employee taking the bus and train will likely do so every working day at similar, set times depending on his schedule. These patterns can be discovered when working with time series data, which can allow for historical lags to have predictive power.

Additionally, traffic flow prediction is also very much a spatial problem (Barthélemy, Barrat, & Vespignani, 2007). Traffic flows are inherently spatial of nature as traffic quite literally flows from one place to another. The flow is subject to a variety of geographical factors such as the road layout, the population density and topography: a trip will take longer if there is no direct road to a destination up in the mountains. Traffic flow data from places that have similar spatial conditions are likely to have predictive power for the flow at hand (Lan et al., 2022).

The problem is both spatial and temporal, but these two sides do not exist in isolation of each other. These two dimensions are at play at the same time and interact with each other. For example, if an accident occurs and causes congestion at a main road, the connected side rides are likely to be more congested too until a large enough period of time has passed in which the accident has been dealt with. Combining these dimensions allow for such patterns to be extracted and exploited, where neighbouring historical lags are of predictive value too. Finding these patterns is key to an accurate prediction, but with limited domain knowledge it can become hard for traffic engineers to find spatial and temporal correlations for huge traffic datasets (Miglani & Kumar, 2019).

Moreover, traffic flow prediction has become increasingly data-driven, as sensors have become more wide-spread in the physical environment (Ma, Sheng, Jin, Ma, & Gao, 2018). An example of such traffic sensors are sensors installed along roads in The Netherlands by the Dutch National Warehouse (Melnikov, Krzhizhanovskaya, Boukhanovsky, & Sloot, 2015). These sensors register cars and are capturing continuous data on speed, intensity and travel time. However, the rise in available data is not just limited to these sensors that might fit the more traditional idea of capturing data. Smartphone data, loop detectors, social media sentiment, local weather forecasts and bus transaction systems are all instances of other ways that are enabled by the emergence of 'smart cities' (Jiang & Luo, 2022). It is this abundance of data created by the extensive adoption of both conventional and unconventional data sources that has propelled the data-focused shift in traffic flow analysis.

2.2 *Classic statistical models and shallow learning models*

Time series forecasting has traditionally been a topic of interest in the field of statistics and econometrics. Simple methods (such as seasonal naïve and simple exponential smoothing) and more intricate methods such as ETS and ARIMA are well-known and well-researched by now (Hewamalage, Bergmeir, & Bandara, 2021). This is not without reason, traditional univariate methods have long outperformed other methods at many forecasting competitions. A key advantage of these traditional univariate methods is that they perform well without the use of a lot of training data, whilst also requiring less parameters to be set (Bandara, Bergmeir, & Smyl, 2020). More sophisticated machine learning algorithms find individual time series often to be too short to be modelled, whereas the simpler, parametric models are more robust to noise. The more complex, non-parametric model thus might not have enough data to fit their parameters, whereas the simpler, parametric models can fall back on their corresponding assumptions (performance relies on how well these assumptions are met). A topical example where these assumptions on (temporal) dynamics do not hold well is the case of a traffic flow with irregular fluctuations.

However, classical univariate time series models have some shortcomings too. Each time series is modelled individually, which means no patterns between time series can be discerned (Hewamalage et al., 2021). This is far from ideal when it comes to traffic demand forecasting, as spatially proximate traffic flow tends to be very useful for the predicted task at hand: neighbouring roads are more likely to be busy too if a road is experiencing heavy congestion. The models thus only take into account the temporal dimension. This issue might be mitigated if the individual time series are long enough for the model to pick up sufficient temporal patterns, but this is often not the case.

The above-mentioned disadvantage of univariate time models becomes a serious issue when viewed in the light of the previously-stated increase in available data streams in the city. Shallow machine learning models do not face such limitations and are able to train on all time series globally, thus able to capitalise on the increase in data streams. They are also capable of dealing with non-linearity in these features describing these time series – this is valuable as research has shown that when it comes to traffic flow prediction, traffic characteristics are indeed non-linear (Hewamalage et al., 2021). This is in contrast to multi-variate models, which are only capable to describe linear relationships between these often non-linear spatial characteristics (Jiang & Luo, 2022). Shallow learning entails most of the machine learning models proposed before 2006 (including shallow neural networks with only on one hidden layer) (Xu, Zhou, Sekula, & Ding, 2021). Examples of such models are support vector machines, KNN, random forest, gradient boosting and hidden Markov models.

2.3 *Deep learning models and Graph Neural Networks*

More recently, deep learning models are employed to take full advantage of both the increase in available data streams and the increase in quality of said data sources. Deep learning models are a branch of machine learning utilising neural networks with multiple hidden layers, simultaneously extracting features when training the model (Xu et al., 2021). These models are also well-equipped to deal with the increase in different data streams since they too handle high-dimensional data (and the subsequent non-linearity in said dimensionality). However, unlike shallow learning models, deep learning models have no trouble extracting correlations in long, sequential data (such as long time series) (Hewamalage et al., 2021). A recurrent neural network (RNNs) is a type of deep learning architecture popularly used for demand forecasting. They contain hidden layers composed of recurrently connected nodes, enabling networks to preserve sequential information and accumulate knowledge from subsequent time steps, which is retained through a feedback loop. LSTMs are a type of RNNs that are more suitable for long-term sequences since it avoids long-term dependency problems that standard RNNs have (Xu et al., 2021).

A deep learning architecture that is particularly well-suited to the domain of traffic demand forecasting, is the graph neural network (GNN). GNNs are neural networks that work with graph structures (Jiang & Luo, 2022). Graphs are a means of capturing a complex system of objects and their connections, with graph data representing that structure as nodes and edges. This is especially useful in the traffic flow domain as spatial information is often hidden in non-Euclidian structures such as a road traffic network. This is in contrast to regular convolutional neural networks (CNN), that model the local spatial information by dividing the plane into a grid and performing a convolutional operation on neighbouring grid cells. By doing so CNNs are unable to model non-Euclidian topological data such as subway networks (e.g. a CNN can't easily process each subway stop having a different number of neighbours). GNNs achieve state of the art results and are a novel and exciting field, with Fan et al. (2020) finding all key traffic forecasting milestones after 2019 to involve GNNs.

A GNN model contains 'a message passing scheme that propagates feature information of the nodes to its neighbours until a stable equilibrium is reached' (Munikoti et al., 2022). Roughly three different types of GNNs can be discerned: graph convolutional network (GCN), GraphSAGE and graph attention network (GAT). Graph convolutional networks are similar to regular CNNs in the way that it uses convolution with shared weights over the filters. However, for GCNs spectral convolution must first transform the non-Euclidian data structure into the Fourier space. The message passage scheme is based on the full adjacency matrix for GCNs, which can be inefficient as it is not generalisable to graphs of different sizes (Munikoti et al., 2022). GraphSAGE uses spatial convolution as opposed to spectral, which means that Fourrier transforms are no longer used as it depends on the properties of k local neighbours. In the case of GraphSAGE, this entails

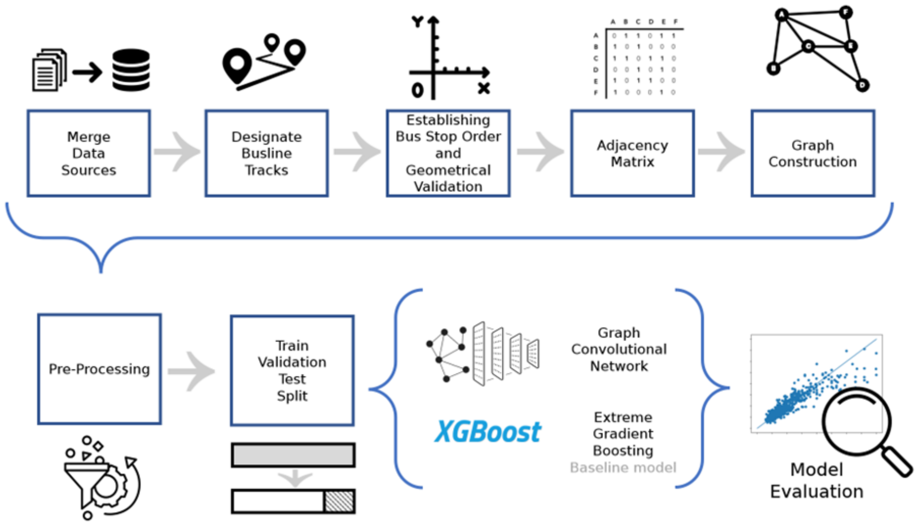
Merge Data Sources
Designate Busline Tracks
Establishing Bus Stop Order and Geometrical Validation
Adjacency Matrix
Graph Construction
Pre-Processing
Train Validation Test Split
Graph Convolutional Network
XGBoost
Extreme Gradient Boosting
Baseline model
Model Evaluation

Figure 1: Methodology pipeline

learning local embeddings of nodes – making it more computationally efficient compared to GCNs. The last major type is graph attention networks, which does not assume the weights of neighbouring nodes to target nodes to be predetermined (GCN) or identical (GraphSAGE). Instead, GAT makes use of attention to learn the relative weights between two connected nodes (Munikoti et al., 2022).

Implementations of such graph neural networks have seen promising results when it comes to traffic demand forecasting for public transport networks. J. Zhang et al. have implemented a graph convolutional network (GCN) combined with a three dimensional convolutional neural network (3D-CNN) on the Beijing subway system using only five weeks' worth of data (2020). They saw their GCN 3D-CNN implementation perform slightly better than other GNN implementations, whilst GNN implementations as a whole performed better than other baseline models. Other research has found positive results for the Shanghai metro network using little over four weeks of data (Han et al., 2019) Current literature about GNN application on bus network data mostly focuses on Europe and Asia, whilst having significantly less data than four months' worth.

3 METHOD

Firstly, the dataset and the necessary pre-processing will be described. Secondly, the Graph Convolutional Network model and the XGBoost model will be described in terms of specification and implementation. The overall pipeline is visualised in Figure 1).

3.1 *Data and pre-processing*

The dataset used in this paper is manually constructed from different datasets published by the Metropolitan Transportation System (STM) of Montevideo. These datasets cover hourly passenger inflow, bus line and bus stop data spanning from February 2022 until June 2022 (Sistema de Transporte Metropolitano, 2022). The passenger inflow is calculated using monthly transaction data ('viajes.csv'). As STM only charges fare based on the origin stop, no destination bus stop is recorded. This is the reason why this research paper only focuses on passenger inflow. Two shapefiles describe the bus stops (`'v_uptu_paradas.shp'`) and bus tracks (`'v_uptu_lsv.shp'`). The first shapefile is used to source the bus stop coordinates and the second shapefile is used to identify which bus stops are attended by which bus line. Shapefile is a format for storing geospatial vector data and is able to describe geometries.

The data sources have been published online by STM in a longer-running open data initiative by the city of Montevideo (Scrollini, 2014). Montevideo has an estimated population of over 1.3 million inhabitants, which is almost 40% of the Uruguayan population (Massobrio & Nesmachnow, 2020). This is in stark contrast to the geographical size of the city, covering 0.3% of Uruguay's surface. This highlights the importance of the STM, a unified system of multiple private bus companies all servicing Montevideo (born out of a wish to modernise the city's public transportation).

These data sources were combined into one graph dataset following the pre-processing steps of Guzmán López, who constructed a graph dataset using the same sources. This dataset is included and briefly evaluated in the Pytorch Geometric Temporal paper (Rozemberczki et al., 2021). Due to the huge size of the dataset and computing limitations only the eleven most popular bus lines are taken into consideration, accounting for more than 25% of the bus trips taken in Montevideo. These eleven bus lines were: 103, 185, 145, G, 183, 306, 163, 137, 405, 110, 546. These bus lines are visualised in Figure 2). Some of the relevant and utilised Python modules are GeoPandas, Shapely and Networkx.

The pre-processing starts by merging all the monthly transaction csv data into one dataset. The coordinates of bus stops and other geometries in the shapefiles are subsequently rounded to two decimals to avoid unintended behaviour by Shapely later on: when coordinates are defined to a too high precision, Shapely has trouble performing a union on geometries, returning a 'non-noded intersection' intersection error. The resulting merged dataset consists of trip id, datetime, number of passengers and bus variant and bus stop. Then all the different variants of bus lines in the shapefile are compared, and the longest variant is designated as being

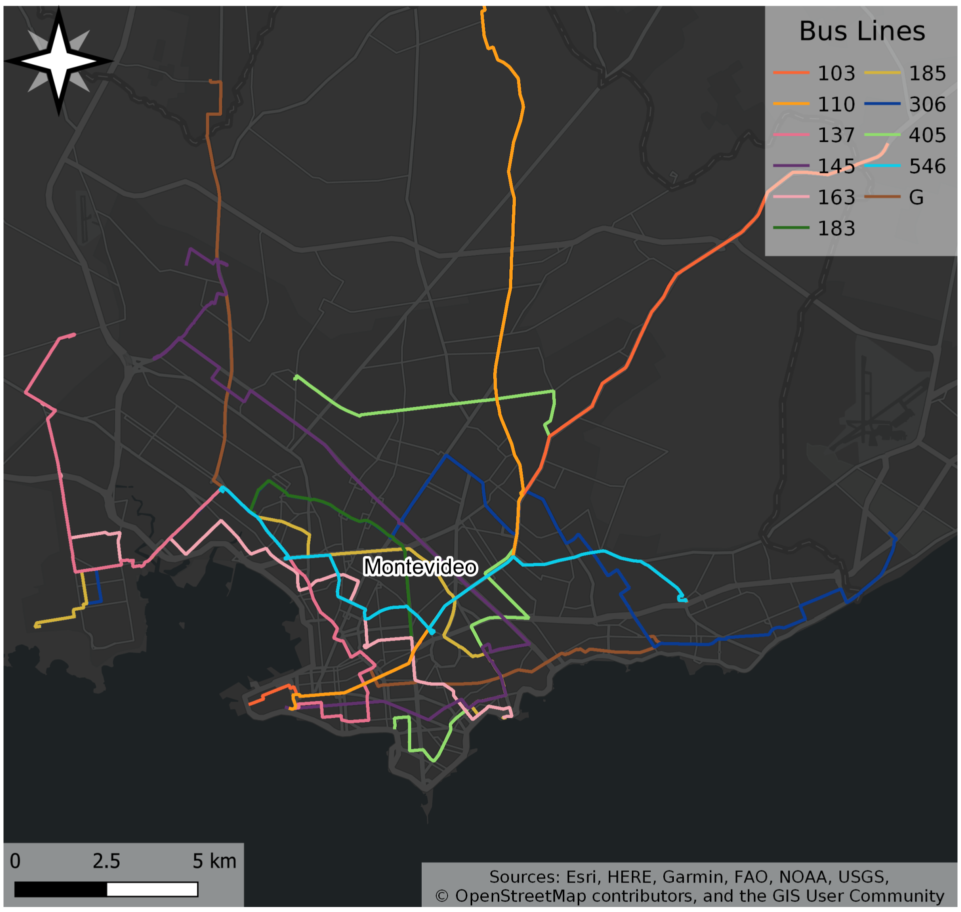
Bus Lines
103
185
110
306
137
405
145
546
163
G
183
Montevideo
0
2.5
5 km
Sources: Esri, HERE, Garmin, FAO, NOAA, USGS,
© OpenStreetMap contributors, and the GIS User Community

Figure 2: Transport network of the eleven most popular bus lines in Montevideo

the de facto bus line track. In other words, sometimes a bus line might not always travel the full route it could take but instead only service the most popular bus stops and leave out more remote bus stops instead. This paper uses the variant that services the most bus stops as it will filter out the least trip data. The bus stop coordinates are then matched with the bus line geometries, which at the same time serves as a validator for the given geometries (all eleven bus lines passed this check). An adjacency matrix of bus stops is then constructed, which is a file containing the distances between them. These distances are not the distances measured in a straight line from bus stop to bus stop, but are instead calculated using the bus line geometries (and are thus a representation of the route driven). A feature matrix is also built (using the adjacency matrix and the merged dataset), aggregating the trip data to a time step of one hour. This feature matrix consists of a list with a number of passengers checking in at the bus stop per time step. The adjacency matrix is then used to extract a topology and build the graph in Networkx, whilst node features are extracted from the feature matrix.

The resulting graph consists of 722 nodes and 737 edges, representing the bus stops and trip segments between those bus stops respectively. This is a relatively sparse network, indicating that most bus stops are visited by only a few bus lines (if not one). The latter might be a result of only using a limited subset of the most popular bus lines. The node features are `'bus_stop'` (containing the bus stop id), `'in_degree'` (the number of edges connected to the nodes), 'lon' (the longitude), 'lat' (the latitude) and 'y' (list of the number of passengers getting on per aggregated time step/per hour). The edge feature is the 'weight', which is the distance between nodes (bus stops). This distance represents the distance of the route travelled, and not the distance measured in a straight line between bus stops. At this point the data pre-processing is finished and the graph representation is saved to a JSON-file, which is available in the repository and can be used for further research.

3.2 *Defining the Graph Convolutional Network*

In order to answer the research question *'To what extent can a Graph Convolutional Neural Network be used to predict passenger inflow?'*, a simple graph convolutional network is defined. This network consists of a graph convolutional layer after which batch normalisation is applied, followed by another graph convolutional layer and batch normalisation. The network is then concluded by a linear layer and batch normalisation, another linear layer and a final linear layer producing the prediction per time step for the prediction window. All the layers use a rectified linear unit (ReLU)

Table 1: The different specifications used when specifying the models

The number of historical lags used	24, 168, 336
The number of values to be predicted	1, 24, 72, 168
The maximum number of neurons in a convolutional layer	64, 128, 256

activation functions, which allows for the non-linearity in the traffic characteristics it tries to model. The inability to produce negative values is no problem considering the predicted output cannot be negative as well, as there is no such thing as a negative passenger inflow (outflow is not measured).

The GCN model will be tuned using several settings, after which the scores will be compared. In order to answer the first sub research question *'To what extent are there differences between short term prediction performance vs long term prediction performance?'*, differently sized training and prediction windows will be used. These values are shown in Table 1, and every combination of past and future values will be explored (3 times 4 equals 12 combinations total). Each value represents a number of time steps, with a time step corresponding to one hour of aggregated data. It is worthy to note that 24 time steps equal to a window of a day, 72 time steps to three days, 168 time steps equal to a window of a week and 336 time steps to two weeks. The depth of the model is defined as mentioned earlier, but the width of the model depends on the number of parameters in each layer. Table 1 also shows a list of parameter values which will be explored for each of the aforementioned 12 combinations as well (resulting in 36 model specifications). The parameter value corresponds to the number of neurons used in the first GCN layer (which corresponds to the output width). The second GCN layer will then reduce the width to half of the previous layer, after which the linear layers will not change the width of the model until the final layer (which will change it to the number of prediction targets: the number of nodes times the length of the prediction window).

The GCN model will thus be explored 36 times (for all the combinations of different training window, prediction window and model width). This will be done using an Adam optimiser with a learning rate of 0.001. Each model specification will be run for 150 epochs or until the last two epochs average a negative trend in validation score (measured using the value). In practice this results in most models being trained for around 20 to 25 epochs. A graph depicting a scatter plot of predicted values and actual values is saved after the model is stopped, along with a graph depicting the change in training and validation error over the past epochs.

The GCN model will thus be explored 36 times (for all the combinations of different training window, prediction window and model width). This

will be done using an Adam optimiser with a learning rate of 0.001. Each model specification will be run for 150 epochs or until the last two epochs average a negative trend in validation score (measured using the r-squared value). In practice this results in most models being trained for around 20 to 25 epochs. A graph depicting a scatter plot of predicted values and actual values is saved after the model is stopped, along with a graph depicting the change in training and validation error over the past epochs.

It must be noted that before the graph can be fed into a defined graph convolutional network, the graph is loaded into the PyTorch Geometric dataloader. Firstly, the graph is read from the JSON-file and converted into a NetworkX graph. Then the graph nodes are relabeled, as the current nodes are named after the bus stop names given by STM. These original names are integers (e.g. bus stop 4756) and can thus cause confusion when trying to differentiate between label names and calling of ordered nodes. Secondly, the node and edge features are transformed to arrays, which are subsequently turned into Tensors. The dataset is then split into a train, validation and test dataset using `70%, 20% and 10%` of the dataset respectively, after which the different datasets are fed into the Pytorch Geometric dataloader. The train-validation-test split is done keeping the temporal order intact, so the oldest `70%` of the observations are used as training data, the next `20%` as validation data and the most current `10%` as test data. This split is illustrated in Figure 3.

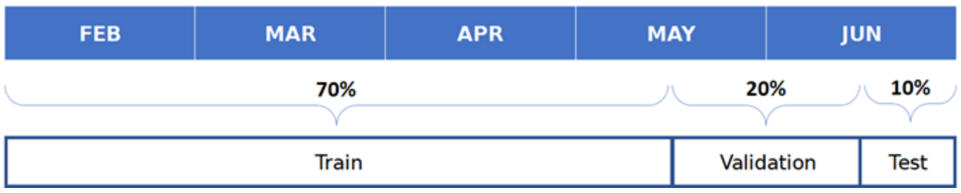

Figure 3: The train-validation-test split

Until this point the data consisted of hourly aggregates of passenger inflow (per bus stop). Now every aggregate is turned into a separate time series, using aggregates of previous time steps as lags. This process is illustrated in Figure 4.

3.3 *Defining the XGBoost model*

XGBoost is a gradient boosting algorithm and falls under shallow machine learning (L. Zhang, Bian, Qu, Tuo, & Wang, 2021). It will be implemented in part to answer the second sub research question: *'How does the prediction performance of a Graph Convolutional Neural Network compare to a conventional shallow learning model like a gradient boosted tree ensemble method (XGBoost)?'*.

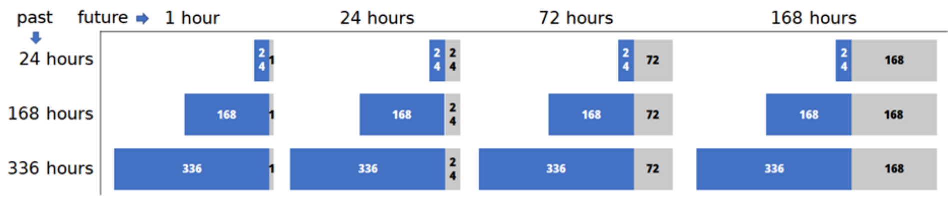

Figure 4: The transformed observation per lag, prediction window specification

As an ensemble method it combines several weak classifiers into a strong classifier, capable of non-linearity. Another benefit of XGBoost is that it is not likely to overfit and works without the need for a lot of data. It is chosen as a baseline model for the reasons above, with the ML model performing well in multiple traffic flow research papers according to a structural literature review conducted by Razali et al. (2021). The model will be specified using the combinations of historical lags and length of prediction windows (see Table 1), which total to 12 combinations. The XGBoost will use the `'gpu_hist'` tree method to enable faster training on the graphical card, doing so using 500 trees. The maximum depth was set to 5.

Before feeding the data into the XGBoost model, the data must be transformed from the graph-structure. The data is initially read from the JSON-file, after which they undergo the same relabelling as in the GCN-model for the sake of consistency. The data is then reshaped so each instance contains the past lags as feature data and future values as target values. This is the same process as in Figure 4. It is worth noting that after the train-test split, the test set and is the exact same size as for the GCN (as is the associated time period). A higher number of lags considered or a higher prediction window does decrease the number of observations in the test set: the oldest observations in the overall dataset are not included (as they do not have the appropriate lags) and the most current observations in the test period are excluded as there are no newer observations to compare the predicted timesteps to. This means that there is a slightly different training or test set between every lag-prediction window combination, but they are exactly the same for the relevant GCN-model and the XGBoost model. Comparing between-model performance and between-parameter performance thus does not come with extra caveats other than the usual limitations.

3.4 *Evaluation methods*

Three different types of metrics are used to evaluate the performance of the models (GCN and XGBoost) the Mean Absolute Error (MAE), the Root Mean Squared Error (RMSE) and the R^2-score.

$$MAE = 1/n \ \sum_{(} i = 1)^n \ |Y \ i - \widehat{Y_i} \ |$$

$$RMSE = \sqrt{\tfrac{1}{n} \sum_{i=1}^n \left(Y_i - \hat{Y}_i\right)^2} \ = \sqrt{MSE}$$

$$R^2 = 1 - \frac{\sum_{i=1}^n \left(\hat{Y}_i - Y_i\right)^2}{\sum_{i=1}^n \left(\bar{Y} - Y_i\right)^2} = 1 - \frac{MSE}{Var(Y)}$$

MAE measures the average absolute difference between the predicted values and the actual values. It is in the same unit as the original data (in our case the number of passengers boarding at a bus stop), so it is easy to interpret. RMSE is the square root of the mean squared errors (MSE). By squaring the errors RMSE is more sensitive to outliers than MAE which is more robust. RMSE is also in the same units as the original data, but the interpretation is slightly less intuitive than MAE. A lower MAE and RMSE indicates a better performing model, with a perfect prediction yielding the minimum score of 0. The R^2-score represents the proportion of the variance in the dependent variable that is predictable from the independent variable. As such, it is affected by a different scaling of the data. It is also sensitive to the number of predictor variables, with the R^2-score almost always increasing when adding a predictor variable (at the risk of inflating the score). A R^2-score of 0 indicates that the model does not perform better than taking the mean of the original data, whereas a higher score indicates a better model fit (with a maximum R^2-score of 1).

4 RESULTS

Firstly, the results of the GCN will be compared for all the different specifications. Secondly, the results of the XGBoost model will be presented.

4.1 *Graph Convolutional Network results*

The Graph Convolutional Network has been implemented 36 times using 3 different historical lags, 4 different prediction windows and 3 different parameter settings. This resulted into the following R^2-scores, visible in Table 2. Moreover, Figure 5 shows the different scatter plots of predicted and actual values for the twelve different time step combinations using only the best-performing parameter setting. A more comprehensive figure containing all 36 possible combinations is included in the appendix **(see figure Z, Appendix).** The R^2-score ranges from 0.330 to 0.816 (the worst-performing and best-performing combination of parameter, lags and prediction window respectively), the MAE-score from 2.001 to 3.043 and the RMSE-score from 4.475 to 7.809.

One of the findings from the results in Table 2 is that the greatest model width (a maximum 256 neurons in a layer) resulted the majority of the time in a higher prediction performance. Out of the twelve time-steps specifications, seven specifications performed the best when using the highest maximum number of neurons in the convolutional layers. This was the case when trying to predict a window of a single time step using 24 lags, 168 steps using 168 lags and for a window of 1, 24 and 72 steps using 336 lags. The second-greatest model width (a maximum of 128 neurons in a layer) was the best-performing for four specifications, and the minimum width performed the best only once. In general the performance difference is relatively small as the R^2-scores are quite close to each other. Only sporadically one of the three parameter settings result in a larger R^2-score drop – e.g. predicting a window of 24 time steps using 336 lags and a max width of 256 neurons results in a score of 0.395 when the other two settings score 0.598 (64 neurons) and 0.637 (128).

When comparing each time-step specification using its respective best performing parameter, it becomes clear that there is a diverse range of prediction scores (Figure 5). The lowest prediction performance with a R^2-score of 0.460 can be found for trying to predict the biggest window (168) using the most historical lags (336), whilst the highest R^2-score (0.812) is for the smallest window size (1) using the smallest number of historical lags (24). Two trends seem to appear: i. the smaller the prediction window, the higher the accuracy and ii. the smaller the number of historical lags used for training, the higher the accuracy. The only exception to this is the biggest prediction window using 24 lags, which performs slightly better than the second biggest window using 24 lags (R^2-scores of 0.678 and 0.671 respectively).

Number of predicted time steps	Number of historical lags	Maximum number of neural nodes in a single layer	R^2-scores	MAE	RMSE
1	24	64	**0.808**	**2.079**	**4.678**
1	24	128	**0.816**	**2.037**	**4.582**
1	24	256	**0.804**	**2.001**	**4.719**
1	168	64	0.763	2.049	4.662
1	168	128	0.762	2.052	4.672
1	168	256	0.782	2.007	4.475
1	336	64	0.712	2.211	4.946
1	336	128	0.723	2.238	4.855
1	336	256	**0.589**	**2.736**	**5.912**
24	24	64	0.706	2.365	5.687
24	24	128	0.715	2.342	5.601
24	24	256	0.728	2.227	5.465
24	168	64	0.695	2.222	5.726
24	168	128	0.601	2.479	6.033
24	168	256	0.720	2.410	5.052
24	336	64	0.598	2.250	5.712
24	336	128	0.637	2.221	5.424
24	336	256	**0.395**	**2.842**	**7.008**
72	24	64	0.651	2.488	5.907
72	24	128	0.667	2.435	5.764
72	24	256	0.671	2.396	5.735
72	168	64	0.662	2.362	5.618
72	168	128	0.673	2.351	5.526
72	168	256	0.688	2.301	5.399
72	336	64	0.556	2.465	6.115
72	336	128	0.516	2.550	6.383
72	336	256	0.516	2.505	6.385
168	24	64	0.649	2.453	5.964
168	24	128	**0.434**	**3.043**	**7.233**
168	24	256	0.678	2.341	5.455
168	168	64	0.655	2.515	5.932
168	168	128	0.664	2.458	5.855
168	168	256	0.612	2.516	6.292
168	336	64	0.409	2.809	7.335
168	336	128	0.330	3.005	7.809
168	336	256	0.460	2.723	7.013

Table 2: Evaluation metrics for every Graph Convolutional Neural Network specification

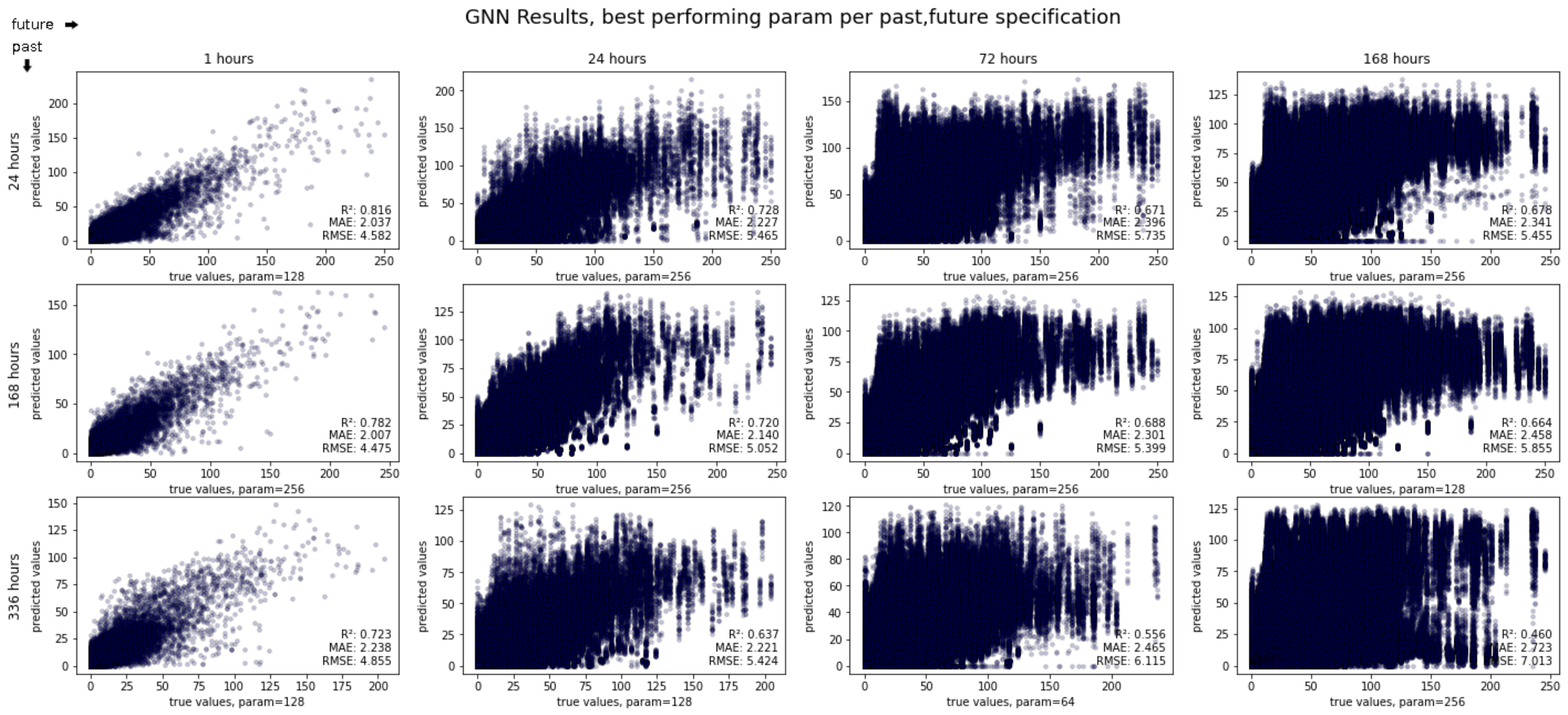

Figure 5: Error plots with corresponding R^2, MAE and RMSE-scores for the Graph Convolutional Network with the best-performing parameter settings

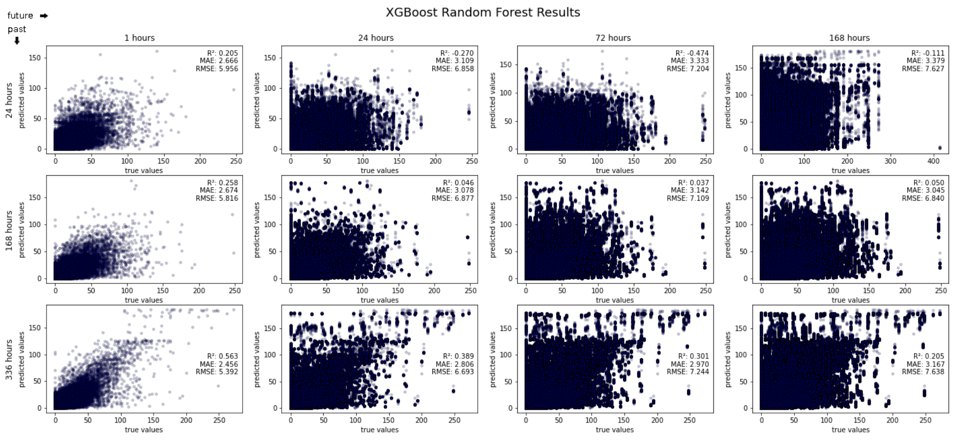

Figure 6: Error plots with corresponding R^2, MAE and RMSE-scores for the XGBoost-model

A visual inspection of the error plots also reveals another finding: the diagonal curve seems to flatten further along on the x-axis. In other words, high passenger inflows are often underestimated in the predictions. The reverse is also true, a lot of smaller values are overestimated, hence the 'fuller' plot. However, the indicated flattening of the (initially fuller) curve shows that there's an inferred upper boundary to the predictions that does not exist in practice, resulting in the longer x-axis of true values. Concludingly, passenger inflow above a certain threshold will always be underestimated. It must be noted that this is a lot less pronounced for the plots with the smallest prediction window (1 hour).

When examined closely, the plots in Figure 5 also show vertical white lines indicating an absence of data (multiple vertical asymptotes). A reason for this behaviour could be that the output of the GNN model is continuous, as opposed to the integers it should be (passengers aren't divisible, so the predicted inflow can't have decimals).

4.2 *XGBoost results*

The XGBoost-model has been implemented 12 times using the same 3 historical lags and 4 prediction windows, resulting in the R^2-scores presented in Table 3. The highest GCN R^2-score (of the three different parameters) has been included for each time step specification for the sake of convenience. Figure 6 shows the different scatter plots of predicted and actual values for the twelve different time steps predicted by XGBoost. The R^2-score ranges from -0.474 to 0.563, the MAE-score from 2.456 to 3.379 and the RMSE-score from 5.392 to 7.638.

Number of predicted time steps	Number of historical lags	XGBoost			Best-performing relevant GCN set-up		
		R^2-score	MAE	RMSE	R^2-score	MAE	RMSE
1	24	**0.205**	**2.666**	**5.956**	**0.816**	**2.037**	**4.582**
1	168	0.258	2.674	5.816	0.782	2.007	4.475
1	336	**0.563**	**2.456**	**5.392**	**0.723**	**2.238**	**4.855**
24	24	**-0.270**	**3.109**	**6.858**	0.728	2.227	5.465
24	168	0.046	3.078	6.877	0.720	2.140	5.052
24	336	**0.389**	**2.806**	**6.693**	**0.637**	**2.221**	**5.424**
72	24	**-0.474**	**3.333**	**7.204**	0.671	2.396	5.735
72	168	0.037	3.142	7.109	0.688	2.301	5.399
72	336	**0.301**	**2.970**	**7.244**	**0.556**	**2.465**	**6.115**
168	24	-0.111	3.379	7.627	0.678	2.341	5.455
168	168	0.050	3.045	6.840	0.664	2.458	5.855
168	336	**0.205**	**3.167**	**7.638**	**0.460**	**2.723**	**7.013**

Table 3: XGBoost evaluation metrics for the different time-step combinations in comparison to the relevant best-performing GCN-model

When looking at Figure 6 and Table 3 a trend seems to arise, the smallest prediction window of 1 hour performs the best for each number of historical lags according to all three evaluation metrics. This is underlined by the rather stark drop-off in when increasing the prediction window to a size bigger (24 hours): the R^2-scores drop from 0.205 to -0.270, 0.258 to 0.046 and 0.563 to 0.389 (for lags of 24, 168 and 336 respectively). Another observation is that the more historical lags used, the better the accuracy score. This holds for all the model specifications except for predicting a window 168 time steps using 168 lags. The latter does do slightly worse (as initially expected) than when using 336 lags, but only when evaluated using MAE and RMSE – the R^2-score is actually lower when using 336 lags.

5 DISCUSSION

The main focus of this research was to improve the demand forecasting of a public transport network by exploring the implementation of a Graph Convolutional Neural Network. This goal was formulated into the following main research question:

> *To what extent can a Graph Convolutional Neural Network be used to predict passenger inflow?*

In order to answer this research question, the following two sub-questions were devised:

RQ1 *To what extent are there differences between short term prediction performance vs long term prediction performance of a Graph Convolutional Neural Network?*

RQ2 *How does the prediction performance of a Graph Convolutional Neural Network compare to a conventional shallow learning model like a gradient boosted tree ensemble method (XGBoost)?*

The results of the GCN revealed multiple findings. One of the findings indicated that a smaller prediction window yielded a better performance, holding true for every combination of prediction window and historical lags but one (going from a prediction window of 72 hours to 168 with 24 lags saw a minor performance increase instead). This finding was consistent for all evaluation metrics except once for MAE, as it slightly decreased (instead of increasing) going from a prediction window of 1 hour to 24 hours with 336 lags. In general the trend of increased performance with a shorter prediction window held up irrespective of the maximum number of parameters. A second finding is that performance according to the RMSE and R^2-score decreases when considering a larger number of historical lags. However, the MAE does not show this trend, which highlights the difference in evaluation metrics. Considering that RMSE and R^2-score punish larger errors, the trend shows that taking into account a smaller number of historical lags lead to predictions that might not necessarily be wrong less often, but are likely to be wrong to a smaller degree. Combining this with MAE-scores sometimes staying the same or increasing (instead of always decreasing), it is not possible to assert whether the frequency of errors occurring decreases when taking into account less lags. However, it is possible to assert that the nature of the errors changes when changing the number of lags considered. Lastly, the error plots reveal that higher true values are more likely to be underestimated to such an extent that the error plotted seem to flatten along the x-axis (the estimated values do not increase proportionally in true values). Passenger flow above a certain threshold seems to always be underestimated. This effect seems more noticeable for long term prediction with the threshold being more pronounced for a larger prediction window. The aversion to large errors could be the result of the loss function of the GCN, which aims to minimise the Mean Squared Error (MSE). MSE is sensitive to larger errors (similar to RMSE and the R^2-score, both of which incorporate MSE) as the errors are squared before their mean is calculated. Overall, these were the three main differences observed between long and short term prediction using a Graph Convolutional Neural Network model.

Comparing the results of the XGBoost model with the GCN revealed multiple findings as well. First of all, XGBoost has a lower R^2-score and a higher MAE and RMSE for every lag-prediction window specification when compared with the relevant GCN specification (using the best-performing parameter setting). Secondly, the trend of a larger prediction window resulting in a worse performance holds true for XGBoost too, seemingly affecting it even stronger. The drop-off in performance is especially pronounced when moving from a prediction window of 1 hour to 24 hours, whilst the GCN does not drop off notably more than when moving from 24 to 72 hours or from 72 to 168 hours. Thirdly, XGBoost seems to perform better when considering more historical lags. This is unlike the GCN, which saw a change in the nature of errors at most, but not a noticeable improving trend. It must be remarked that the difference in performance in MAE-score is nearly always at least 0.6 or higher, which translates to a difference in error of over a half passenger in hourly inflow. In the scale of a bus capacity this does not seem that large, perhaps the nature of the error is more distinctive. Overall, there are multiple differences in the prediction performances of a GCN model and a XGBoost model, ultimately indicating that the GCN model is more suitable in predicting passenger inflow for a public transport network.

The results are relatively in line with current literature that shows GCN are the current state-of-the-art models and achieve performance as such. That XGBoost has a harder time predicting is thus expected, however the extent to which the R^2-scores are lower is surprising, since XGBoost is one of the shallow deep learning methods found to predict well on its own (Razali et al., 2021). The finding that the inclusion of more historical lags do not increase GNN performance is in line with literature as well, as the more distant lags are likely to be less predictive for forecasting with underlying patterns having changed in the meantime (Bandara et al., 2020). In addition, literature does show that short-term prediction is harder than long-term prediction when it comes to traffic forecasting, with literature also showing similar results for GNNs (Cui et al., 2022).

This current study comes with several limitations and constraints. One of the limitations is that the only features are the previous time lags in combination with the spatial information from a network. With IOT enabling data streams from a variety of sources, a model implemented with said features would give more practical insight. However, this paper does reflect the differences between more barebone implementations, which is also useful. Furthermore, the geospatial error analysis is limited which might obscure some relationships in the data. Additionally, the dataset has been limited to only the most popular bus lines to restrict the amount of computing resources needed. Depending on the underlying differences in data the prediction performance might increase or decrease, potentially changing the findings when considering less popular bus lines as well.

Another data limitation this research paper faced was the absence of outflow data. Outflow data is usually present and has the possibility to reveal a lot of spatiotemporal patterns in public transport trips, whereas now the model had to infer from inflow data only.

These limitations also signal potential avenues for future research, such as comparing model performance using additional features from a wide range of data sources. Examples are weather data, traffic (GPS) data and neighbourhood data (exploring the spatial dimension). Future research could also implement a more rigorous spatial error analysis. Moreover, they could investigate the effect of different graph neural networks using a different number and type of network layers such as GraphSAGE.

6 CONCLUSION

In this paper the performance of a Graph Convolutional Neural Network on predicting the passenger inflow of a bus transport network is explored. The short and long-term prediction performance is evaluated using various numbers of historical lags considered, various prediction windows and different layer widths. A bigger prediction window yielded less accurate results, whereas considering a bigger number of historical lags did not reliably increase or decrease the mean absolute error. However, the nature of the errors did change, with an increase in lags considered resulting in less large errors. This is accompanied with the inflow of passengers being structurally more underestimated for higher prediction windows. Furthermore, having the maximum number of parameters in a layer resulted in the highest performance in seven out of twelve model specifications, but further research is needed to come to conclusive insight.

The performance of the GCN has also been compared using a gradient boosted tree ensemble method (XGBoost). The GCN performed better than the XGBoost for every lag and prediction window combination. XGBoost saw a larger prediction window result in a worse prediction performance in an even more pronounced manner. The drop-off in performance was especially notable when predicting more than 24 hours. However unlike the GCN, XGBoost saw including more historical lags improve performance using all metrics.

Concludingly, these findings indicate that a Graph Convolutional Neural Network is a more than suitable model to predict passenger inflow on a bus network. It provides an improvement in prediction performance over the established gradient boosted tree ensemble method XGBoost in multiple ways. Future research exploring this topic further seems exciting and salient, both from a scientific and social point of view.

REFERENCES

Allam, Z. (2020). Data as the New Driving Gears of Urbanization. In Z. Allam (Ed.), *Cities and the Digital Revolution: Aligning technology and humanity* (pp. 1–29). Cham: Springer International Publishing. Retrieved 2023-01-18, from https://doi.org/10.1007/978-3-030-29800-5_1 doi: 10.1007/978-3-030-29800-5_1

Bandara, K., Bergmeir, C., & Smyl, S. (2020, February). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, *140*, 112896. Retrieved 2022-12-04, from https://www.sciencedirect.com/science/article/pii/S0957417419306128 doi: 10.1016/j.eswa.2019.112896

Barthélemy, M., Barrat, A., & Vespignani, A. (2007, March). The role of geography and traffic in the structure of complex networks. *Advances in Complex Systems*, *10*(01), 5–28. Retrieved 2022-12-02, from https://www.worldscientific.com/doi/abs/10.1142/S021952590700091X (Publisher: World Scientific Publishing Co.) doi: 10.1142/S021952590700091X

Cui, Y., Zheng, K., Cui, D., Xie, J., Deng, L., Huang, F., & Zhou, X. (2022, February). METRO: A Generic Graph Neural Network Framework for Multivariate Time Series Forecasting. *Proc. VLDB Endow.*, *15*(2), 224–236. Retrieved from https://doi.org/10.14778/3489496.3489503 (Publisher: VLDB Endowment) doi: 10.14778/3489496.3489503

Duranton, G., & Puga, D. (2014, January). Chapter 5 - The Growth of Cities. In P. Aghion & S. N. Durlauf (Eds.), *Handbook of Economic Growth* (Vol. 2, pp. 781–853). Elsevier. Retrieved 2022-09-30, from https://www.sciencedirect.com/science/article/pii/B9780444535405000057 doi: 10.1016/B978-0-444-53540-5.00005-7

Fan, X., Xiang, C., Gong, L., He, X., Qu, Y., Amirgholipour, S., ... He, X. (2020, December). Deep learning for intelligent traffic sensing and prediction: recent advances and future challenges. *CCF Transactions on Pervasive Computing and Interaction*, *2*(4), 240–260. Retrieved 2022-12-05, from https://doi.org/10.1007/s42486-020-00039-x doi: 10.1007/s42486-020-00039-x

Han, Y., Wang, S., Ren, Y., Wang, C., Gao, P., & Chen, G. (2019, June). Predicting Station-Level Short-Term Passenger Flow in a Citywide Metro Network Using Spatiotemporal Graph Convolutional Neural Networks. *ISPRS International Journal of Geo-Information*, *8*(6), 243. Retrieved 2022-09-30, from https://www.mdpi.com/2220-9964/8/6/243 (Number: 6 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/ijgi8060243

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021, January). Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting*, *37*(1), 388–427. Retrieved 2022-12-04, from https://www.sciencedirect.com/science/article/pii/S0169207020300996 doi: 10.1016/j.ijforecast.2020.06.008

IPCC. (2022). Summary for Policymakers. In H. O. Pörtner et al. (Eds.), *Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* (p. In Press). Cambridge, UK: Cambridge University Press. Retrieved from https://www.ipcc.ch/report/ar6/wg2/downloads/report/IPCC_AR6_WGII_SummaryForPolicymakers.pdf (Type: Book Section)

Jiang, W., & Luo, J. (2022, November). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, *207*, 117921. Retrieved 2022-12-02, from https://www.sciencedirect.com/science/article/pii/S0957417422011654 doi: 10.1016/j.eswa.2022.117921

Lan, S., Ma, Y., Huang, W., Wang, W., Yang, H., & Li, P. (2022, June). DSTAGNN: Dynamic Spatial-Temporal Aware Graph Neural Network for Traffic Flow Forecasting. In *Proceedings of the 39th International Conference on Machine Learning* (pp. 11906–11917). PMLR. Retrieved 2022-12-02, from https://proceedings.mlr.press/v162/lan22a.html (ISSN: 2640-3498)

Ma, D., Sheng, B., Jin, S., Ma, X., & Gao, P. (2018). Short-Term Traffic Flow Forecasting by Selecting Appropriate Predictions Based on Pattern Matching. *IEEE Access*, *6*, 75629–75638. (Conference Name: IEEE Access) doi: 10.1109/ACCESS.2018.2879055

Massobrio, R., & Nesmachnow, S. (2020, January). Urban Mobility Data Analysis for Public Transportation Systems: A Case Study in Montevideo, Uruguay. *Applied Sciences*, *10*(16), 5400. Retrieved 2022-09-29, from https://www.mdpi.com/2076-3417/10/16/5400 (Number: 16 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/app10165400

Melnikov, V. R., Krzhizhanovskaya, V. V., Boukhanovsky, A. V., & Sloot, P. M. A. (2015, January). Data-driven Modeling of Transportation Systems and Traffic Data Analysis During a Major Power Outage in the Netherlands. *Procedia Computer Science*, *66*, 336–345. Retrieved 2022-12-02, from https://www.sciencedirect.com/science/article/pii/S1877050915033888 doi: 10.1016/j.procs.2015.11.039

Miglani, A., & Kumar, N. (2019, December). Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges. *Vehicular Communications*, *20*, 100184. Retrieved 2022-09-29, from https://www.sciencedirect.com/science/article/pii/S2214209619302311 doi: 10.1016/j.vehcom.2019.100184

Miller, P., de Barros, A. G., Kattan, L., & Wirasinghe, S. C. (2016, April). Public transportation and sustainability: A review. *KSCE Journal of Civil Engineering*, *20*(3), 1076–1083. Retrieved 2022-09-30, from https://doi.org/10.1007/s12205-016-0705-0 doi: 10.1007/s12205-016-0705-0

Munikoti, S., Agarwal, D., Das, L., Halappanavar, M., & Natarajan, B. (2022, June). *Challenges and Opportunities in Deep Reinforcement Learning with Graph Neural Networks: A Comprehensive review of Algorithms and Applications.* arXiv. Retrieved 2022-09-29, from http://arxiv.org/abs/2206.07922 (arXiv:2206.07922 [cs]) doi: 10.48550/arXiv.2206.07922

Razali, N. A. M., Shamsaimon, N., Ishak, K. K., Ramli, S., Amran, M. F. M., & Sukardi, S. (2021, December). Gap, techniques and evaluation: traffic flow prediction using machine learning and deep learning. *Journal of Big Data*, *8*(1), 152. Retrieved 2023-01-06, from https://doi.org/10.1186/s40537-021-00542-7 doi: 10.1186/s40537-021-00542-7

Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., ... Sarkar, R. (2021). PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (pp. 4564–4573). New York, NY, USA: Association for Computing Machinery. Retrieved 2022-09-29, from https://doi.org/10.1145/3459637.3482014 doi: 10.1145/3459637.3482014

Scrollini, F. (2014, July). *Open cities : the case of Montevideo* (Working Paper). Retrieved 2022-12-02, from https://idl-bnc-idrc.dspacedirect.org/handle/10625/55362 (Accepted: 2016-01-13T18:26:41Z)

Sistema de Transporte Metropolitano. (2022). *Montevideo Bus (STM) Datasets.* Intendencia de Montevideo. Retrieved 2022-09-15, from https://catalogodatos.gub.uy/dataset?q=stm

Wamsler, C., Brink, E., & Rivera, C. (2013, July). Planning for climate change in urban areas: from theory to practice. *Journal of Cleaner Production*, *50*, 68–81. Retrieved 2023-01-18, from https://www.sciencedirect.com/science/article/pii/S095965261200652X doi: 10.1016/j.jclepro.2012.12.008

Xu, Y., Zhou, Y., Sekula, P., & Ding, L. (2021, May). Machine learning in construction: From shallow to deep learning. *Developments in the Built Environment*, *6*, 100045. Retrieved 2022-12-

05, from https://www.sciencedirect.com/science/article/pii/S2666165921000041 doi: 10.1016/j.dibe.2021.100045

Zhang, J., Chen, F., Guo, Y., & Li, X. (2020). Multi-graph convolutional network for short-term passenger flow forecasting in urban rail transit. *IET Intelligent Transport Systems*, *14*(10), 1210–1217. Retrieved 2022-09-29, from https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-its.2019.0873 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-its.2019.0873) doi: 10.1049/iet-its.2019.0873

Zhang, L., Bian, W., Qu, W., Tuo, L., & Wang, Y. (2021, April). Time series forecast of sales volume based on XGBoost. *Journal of Physics: Conference Series*, *1873*(1), 012067. Retrieved 2022-12-07, from https://dx.doi.org/10.1088/1742-6596/1873/1/012067 (Publisher: IOP Publishing) doi: 10.1088/1742-6596/1873/1/012067

APPENDIX A

If you have nothing to append: remove this. You can do a page referral for these, like: Appendix A (page 26).