STATION-GRAPH: BIKE FLOW TRAFFIC PREDICTION USING SPATIO-TEMPORAL GRAPH CONVOLUTIONAL NEURAL NETWORK

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Darshit A. Pandya

FALL
2019

STATION-GRAPH: BIKE FLOW TRAFFIC PREDICTION USING SPATIO-TEMPORAL GRAPH CONVOLUTIONAL NEURAL NETWORK

A Project

by

Darshit Atulkumar Pandya

Approved by:

_____, Committee Chair
Dr. Haiquan Chen

_____, Second Reader
Dr. Xuyu Wang

_____
Date

Student: Darshit A. Pandya

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

_____, Graduate Coordinator _____
Dr. Jinsong Ouyang Date

Department of Computer Science

Abstract

of

STATION-GRAPH: BIKE FLOW TRAFFIC PREDICTION USING SPATIO-TEMPORAL GRAPH CONVOLUTIONAL NEURAL NETWORK

by

Darshit Pandya

These days, when the concept of vehicle sharing has been exploited by people a lot, it becomes necessary to predict the flow of the vehicles to match the demand with the availability. Nowadays, in metros like San Francisco and New York, peak hour traffic problem has been a major concern for most of the daily commuters and residents. Bikes are an easy solution for the short commutes and hence should be made available in the designated stations all day round. This project visits the problem of predicting the bike flow at station-level using a novel architecture of Spatial-temporal Graph Convolutional Network. In this project, we consider two types of relations, the temporal patterns and the spatial correlation between the stations to predict the next hour rides. The graph constructed as an input to the model has nodes which are bike stations and edges with weights as the distance between the stations. We consider the graph to be a fully connected graph. The input graph along with the temporal sequences serves as an input to the model with complete convolutional structures. Using the Euclidean haversine distance algorithm, we calculate the great circle distance between the stations in miles. Unlike normal time series analysis, we use distance graph as a relationship matrix between each of the stations.

The model was evaluated using the open sourced ford go (now Lyft) bike demand dataset of City of San Francisco with more than 3 million trips data to predict the next hour ride at both station-level and cluster-level. Extensive experiments validate that our STGCN-based model outperformed the baseline methods in terms of RMSE in three different settings, i.e., at the top 10 most popular stations, at the cluster level, and at the station level.

_____, Committee Chair
Dr. Haiquan Chen

_____
Date

ACKNOWLEDGEMENTS

I am grateful to Dr. Haiquan Chen for providing me with the opportunity to complete my project under his guidance. I would like to extend my thanks to him for his continuous guidance and support throughout the complete journey. I would also like to thank Dr. Xuyu Wang to review my master's project report. I am grateful to Department of Computer Science and the university for their trust and faith in my success.

My parents, my brother and my friends constantly motivated me throughout my journey of completing Masters degree. I could not have achieved the feat without their constant faith and trust.

.

TABLE OF CONTENTS

LIST OF TABLES

Tables Page

LIST OF FIGURES

Figures Page

Chapter 1

INTRODUCTION

Today, when the traffic problems are reaching its peak due to population rise and exponential addition of vehicles on road, people are found stuck in traffic on average 2 hours of their day. To overcome this rampant issue of traffic, the urban population are inclining towards using bicycles to commute to their work, go shopping, attend nearby local event or do any recreational activity in near periphery to avoid getting stuck into traffic. Considering the popularity, cycling infrastructure and the community support have grown substantially. Major companies belonging to ride hailing and transportation services are investing heavily in bicycle services these days. Currently there are two type of bicycle services that are run by organizations, docked bikes and dock-less bikes. A comparative study shows that docked bicycles has a better customer base due to its reliability and availability. Hence, it becomes important to make the consumers available with the bikes whenever and wherever required, at the nearest station, at the right time. An average of 3.5% of the commuters in United States have started bicycling to work since 2012 [1]. Major of the work focusing on demand prediction is considering the time series data for analysis. In present, the demand prediction is conducted using two major methodologies: (1) temporal pattern analysis through time-series data where the prediction is completely dependent on temporal data [2-4], for e.g. the number of trips for a particular hour, frequency of trips w.r.t time, (2) a graph based approach where station-level weighted graph is constructed for relationship analysis along with the time series data. Compared to other methods which effectively captures the patterns of Euclidean data, graph methods are

powerful in capturing the relationship between the nodes through linkages [5]. The previous temporal analysis approach adapted by Zeng et al. has certain limitations as the work done by missed out considering the effect of one station's bike flow pattern over the other connected stations in the prediction process [6]. There are comparatively very few graph network implementations for demand prediction due to the complexity involved in handling the graph data. As this field is currently under a lot of research, there are new architectures that are discovered for handling the graph data. However, the current state-of-the-art bike flow prediction has a higher complexity as it involves a graph convolutional network, a LSTM network and an autoencoder network to form the complete architecture.

1.1 Our Contributions.

The contributions of our work are as follows:

- We propose a novel architecture of Spatial-temporal Graph Network for the problem of predicting the bike flow at both station-level and cluster-level. The framework eliminates the need to implement the LSTM network and auto-encoder network explicitly after doing spatial analysis using a graph convolutional network. ST-GCN model is able to combine features from both spatial and temporal domains. The ST-GCN architecture is designed to process the graph-structured temporal data jointly.
- Our approach enables the user to predict the demand at all the stations at the same time rather than predicting one station one by one. Additionally, we construct a single graph for all the stations i.e. a distance connectivity graph unlike most of the

state-of-the-art which use multiple feature graphs to do relationship analysis between the stations.

- We designed our model architecture to predict the bikes demand for the next hour as the future window considering multiple history windows. Our model supports a history window of 24 hours, 72 hours and 168 hours which helps in understanding the flow pattern in a better way. Additionally, we consider both in-flow and out-flow of bike flow traffic to find the deficit in availability with respect to the predicted demand.
- Extensive experiments performed on our proposed architecture along with the deep learning methods on the real-world bike ride dataset validate that our proposed architecture outperforms the state-of-the-art model in terms of accuracy and effectiveness.

The following chapters unveils the in-detailed flow of the project. Chapters 2 presents details about the architecture of Station-graph with Chapter 3 discussing the spatial modeling using graph convolution network in detail, and Chapter 4 talks about the temporal pattern modeling. Chapter 5 highlights the comparative experiments along with discussing about the Google Cloud Platform (GCP) setup and data preparation flow. Chapter 6 identifies the related work and discusses them. The final chapter, Chapter 7 culminates the report with conclusion and future work.

Chapter 2

OVERVIEW OF STATION-GRAPH

Station-Graph is a completely convolutional architecture implemented using a spatial-temporal graph convolutional neural network. A spatial-temporal graph convolutional block is made up of two parts, (1) a graph convolutional network which is responsible for spatial analysis and (2) a gated recurrent unit which is responsible for temporal convolutions. A spatial-temporal graph convolutional block is a sandwiched structure with a 1-D gated recurrent unit followed by a graph convolutional block and a 1-D gated recurrent unit again.

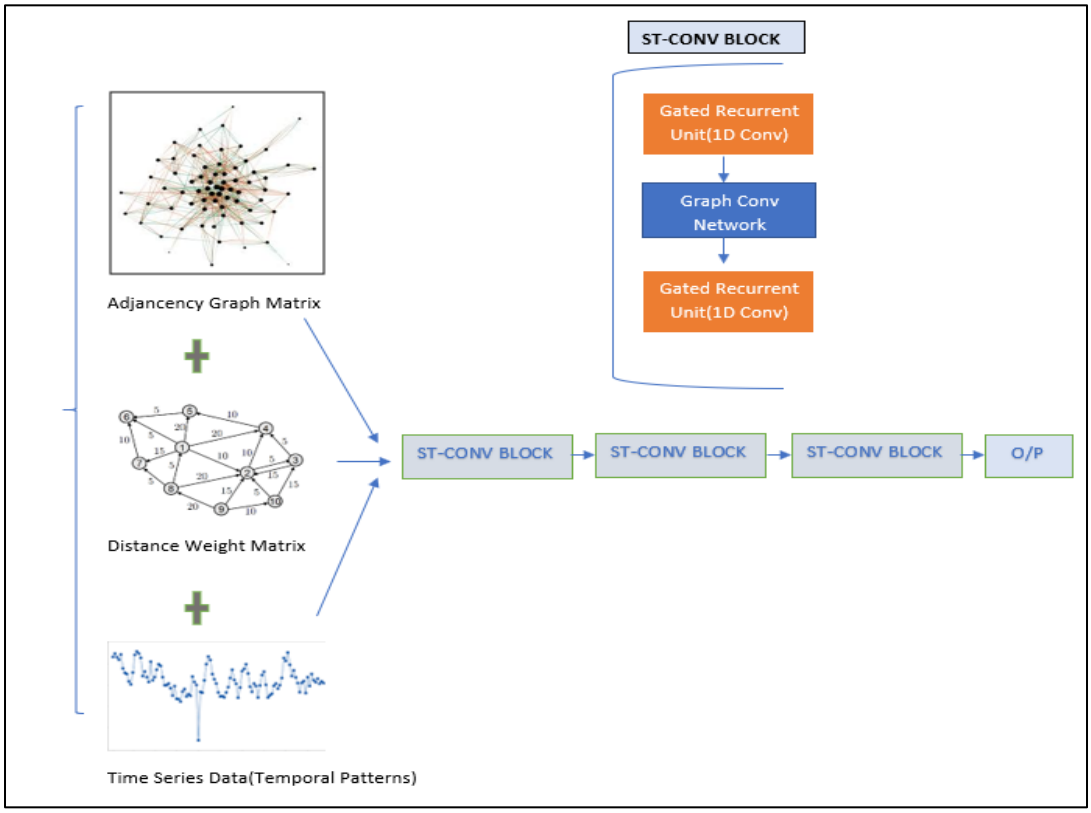

Figure 1: Station-Graph network architecture

As shown in Figure 1, a single spatial-temporal convolutional block consists of two discrete parts. Using both the graph convolution and the temporal convolutions, we can find the useful spatial perspective features and the most essential temporal features at the same time. Additionally, we reduced the overhead of applying auto-encoder unlike the network architecture adapted by Chai et al. which makes our architecture efficient in terms of time and scalability [7]. Hence our architecture will be able to handle the huge networks in comparatively efficient way. We will discuss each of the part in brief in the current chapter and explaining details of it in the upcoming chapters.

3.1. Spatial Dependency Modeling

While considering the graph network, the connection between the edges along with the features of the nodes holds the utmost importance. In the first phase, we modeled two type of correlations among the different stations for spatial modeling namely the connectivity correlation as neighborhood and the weight correlation in form of distance between the stations. We then normalized the weight matrix along with 1^{st} order approximation to handle the sparsity issue. Stacking multiple graph convolutional networks acts similar to a fixed-kernel convolution looking at kernel(k) – 1 neighborhood values. We then applied graph convolution operation using the kernel K_c on the whole flow matrix. As we stacked multiple spatial-temporal graph convolutional block, multiple graph convolutional networks were stacked indirectly.

3.2. Temporal Correlation Modeling

In the next phase of the data modeling, we considered the time-series historical data sampled over the span of 2 years i.e. from Jan 2017 to May 2019. To handle the time-series data, we used the Gated Recurrent Neural Network as the initial layer in a spatial-temporal block to model the correlations amongst the bike ride observations. The temporal convolutional gating mechanism involves the information from the connected stations which is obtained during the graph convolution operation. The temporal convolution is designed in order to get a single output element for each of the station. The in-detailed description of the temporal convolution is covered in Chapter 4.

Chapter 3

SPATIAL DEPENDENCY MODELING – GCN

In this section, we will go in depth with the spatial dependency modeling performed using a graph convolutional neural network. Spatial dependency modeling extracts the relationship between different stations of the network. Construction of a graph has two perspectives namely spectral perspective and spatial perspective. We are considering the spatial perspective for this project. In this project, we propose a single graph convolution in form of distance graph to model spatial dependency. Along with the distance graph, we also consider the connectivity graph between different stations. The proximity matrix helps to identify the stations of interest w.r.t a particular station.

3.1 Neighborhood:

Considering the spatial proximity, we defined the two stations adjacent with a value of 1 else 0. Our dataset consists of total of 430 stations and hence we formed a 430 X 430 grid using Equation 1.

$$Adj(Vi, Vj) = 1 \text{ if } V_i \text{ is connected to } V_j \text{ else } 0 \quad \ldots\ldots\ldots (1)$$

A station is declared to be connected to other station if there has been any ride between them till date else are considered disconnected. Consideration of both inflow and outflow makes the neighborhood connectivity graph an almost fully connected undirected graph.

3.2 Distance Relationship:

Every ride has a start station and a destination station marked using their respective latitude and longitude co-ordinates. Using the haversine formula, we constructed another graph *W* with the value of (V_i, V_j) as the distance between station i and station j. Haversine formula can be represented as:

$$a = sin^2(\varphi B - \varphi A/2) + cos\,\varphi A * cos\,\varphi B * sin^2(\lambda B - \lambda A/2) \quad \ldots\ldots\ldots (2)$$

where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6371 kms or 3958 miles). Using the adjacency matrix generated above, we calculated a degree matrix *D* defining the total number of stations connected to a particular station. The graph convolutional operation can be expressed in below equation.

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \quad \ldots\ldots\ldots (3)$$

For certain cases where the origin station is same as the destination station, a normal adjacency matrix won't suffice. Hence, to enable considering the rides from the same origin and destination station, we applied self-loops on the adjacency matrix considering the Equation 4.

$$A' = D^{-1}A + I \quad \ldots\ldots\ldots (4)$$

Following several universal implementations of the spatial-temporal graph convolutional network [7, 8], we defined our graph convolution operation as in [7].

In the convolution operation, C_i and C_o are the size of input and output feature maps respectively and Θ is the kernel parameter. For each time step t, the graph convolution with the same kernel Θ is applied on the input.

The spatial layer that is bridged between the two temporal layers in the ST-Conv block makes the spatial-propagation faster through temporal convolutional layer. We applied layer normalization within each of the spatial-convolutional block to avoid overfitting of the features. It modeled spatial dependencies by feature extraction through closeness relationship. With a small reception field, the feature extraction will get focused on close regions, i.e., neighbors that are more closely connected to each other through the distance matrix.

Chapter 4

TEMPORAL PATTERNS MODELING – GRU

In this project, we considered the data sampled over the span of 2 years which is equivalent to 3M ride observations and hence there are ample scope to encounter vanishing gradients. Additionally, the popularity of Recurrent Neural Network (RNN) for solving the time-series problems, they were not adaptive towards the dynamic changes in the flow prediction. To overcome this and address the issue of long term dependencies, we applied Gated Recurrent Unit which is a type of Recurrent Neural Network for temporal patterns modeling. The input to the GRU will be historical data i.e. the ride counts observations as per the station concatenated with output of GCN block i.e. the information to the related stations. We used 1-D GRU to model the co-relations amongst the ride count at different time-steps. The gating mechanism is designed to map the given input to a single output value. The final temporal convolutional layer along with the fully connected layer is implemented to get the output in form a single-step value of next hour ride count.

Chapter 5

EXPERIMENTAL RESULTS

In this section, we empirically evaluated the performance of Station-Graph considering the FordGo Bike Dataset ride information as the ground truth. In the upcoming subsections, we discuss about the details of the models implemented for doing a comparative study.

5.1. Data Preparation

Graph Methods require the data to be in the form of graph matrices. The real-world trip data that we are trying to experiment with for this project is not in a compatible format and hence it becomes important to convert the data into graph format. A graph can be represented using a *(V, E, W)* tuple where *V* stands for the nodes (stations), *E* stands for the set of edges and *W* represents the weight of the edges. Using these, certain matrix representations are generated, adjacency matrix(connectivity), weight matrix (relationship) and feature matrix. An example of an undirected graph and its adjacency matrix is shown in Figure 2.

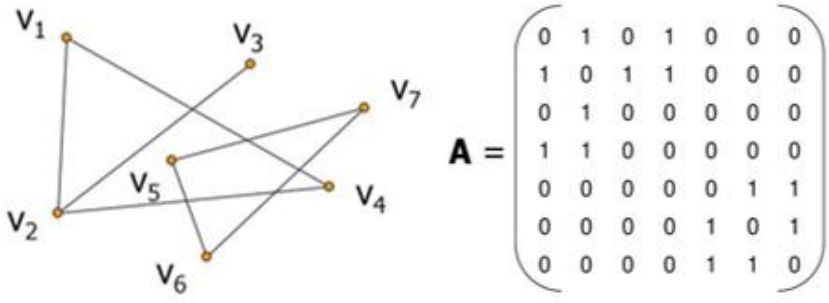

Figure 2: Sample graph representation

Implementation of graph methods as opposed to non-geometric deep learning methods is difficult because of the hidden complexity of data generation and cleaning [8]. Hence, in the next subsections we will cover the data generation process in detail.

5.1.1 Creation of Adjacency Graph and Weight Graph

As per the terminology, the adjacency matrix, sometimes also called the connection matrix, of a simple labeled graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether y_i and v_j are adjacent or not [9]. In our project, vertices stand for the stations/cluster of stations. Our project works on both station-level and cluster-level. In the upcoming subsection, we show the generated graphs for station-level and cluster-level prediction.

a. Station-level:

In our dataset, we had nearly 430 different stations across two metropolitan cities, San Francisco and Oakland. A short visualization of the stations' distribution is demonstrated in Figure 3 where every dot represents a docked bike station.

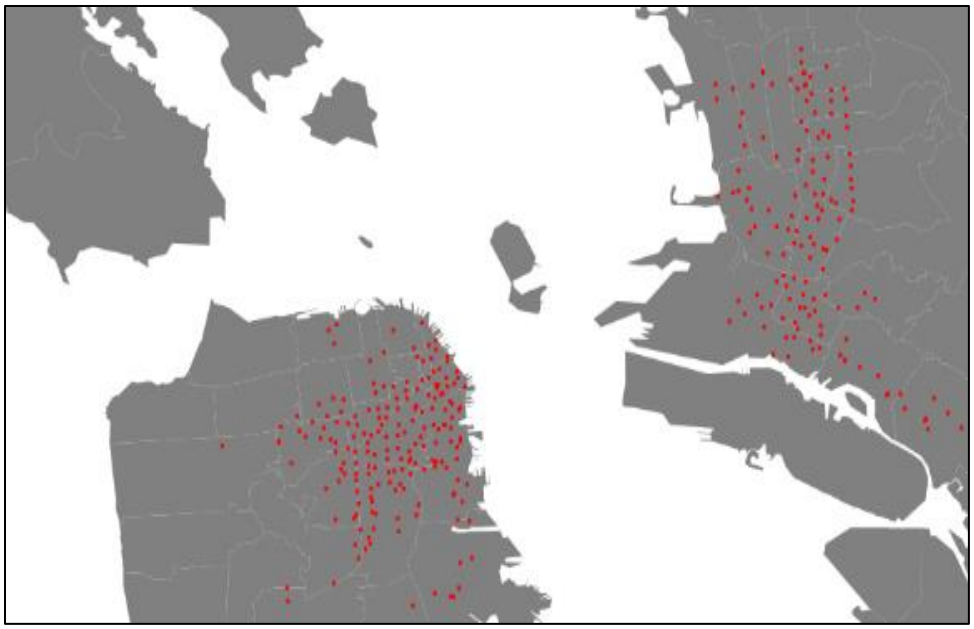

Figure 3: Distribution of stations across two cities

Considering the total available stations, we created a matrix of 430 X 430 dimensions where every value indicates if the row station and column station are connected to each other. If there had been any ride from station X to station Y, we considered it as a connected station.

After discovering the connectivity between the stations, the distance between the stations was calculated using the Haversine great circle distance formula. Additionally, in the connectivity matrix, we considered the self-loops and hence the diagonal values are marked as 1. Self-loops help to analyze and consider the rides which started and ended in the same station.

b. Cluster-level:

The stations in Figure 3 are quite close to each other and hence, in order to execute a better comparative study of bike ride demand, we combined several stations into one area. The selection of stations into a specific cluster is dependent upon the distance threshold between the stations. We considered 0.3 miles as the distance threshold while maximum of 20 stations under one area. Using a classic and powerful density-based clustering algorithm DBSCAN, we combined the stations into clusters. The generated clusters are demonstrated in Figure 4.

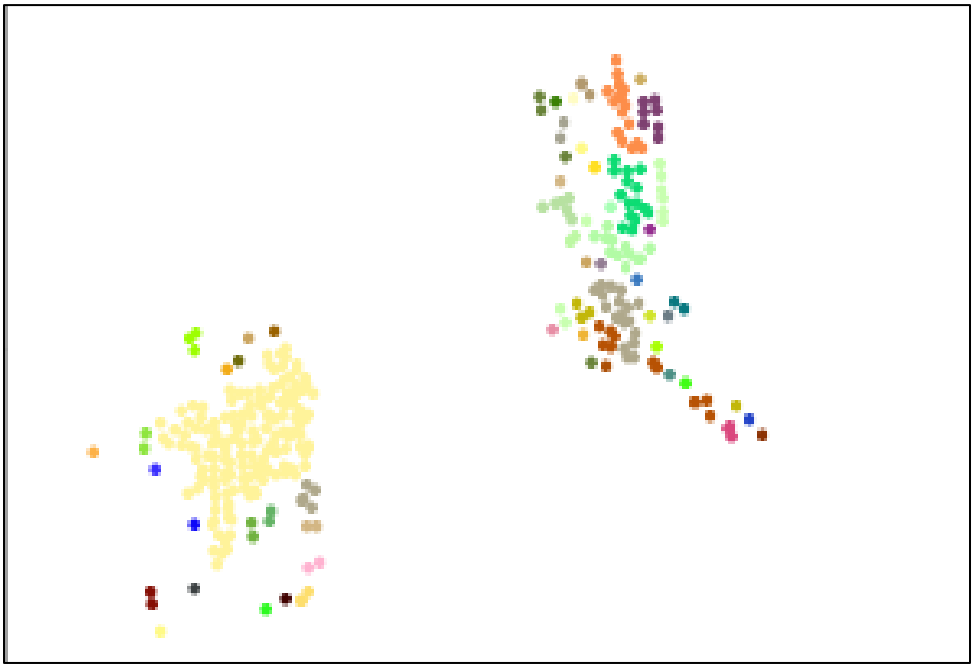

Figure 4: Clustered stations using DBSCAN

After clustering, we found 76 total areas from 430 stations. A resultant graph matrix of 76 X 76 dimensions was constructed by converting the station ids to cluster ids. The distance between two areas was calculated by taking the distance between the centroids of each of the cluster into consideration.

5.1.2 Creation of time-series data from ride data

For this project, we considered the rides count on hourly basis, and hence we grouped the rides as per each hour of the day. In some cases, where the ride data wasn't available for a particular hour like morning 5 am or similar, we considered the value as 0.

Applying the above two steps, we generated an adjacency matrix, a weight matrix, and a cleaned time series data in a proper format for sequencing and applying convolutions.

5.2. Setting up a Google Cloud Platform environment in this project

We conducted experiments on Google Cloud Platform (GCP) to avoid the need to have a locally installed GPU. This section serves as a quick guide to setting up the GCP environment to guide the future users. In order to begin the setup, we need to instantiate and set up a project as shown in Figure 5 and Figure 6.

Step 1: Setting up a project

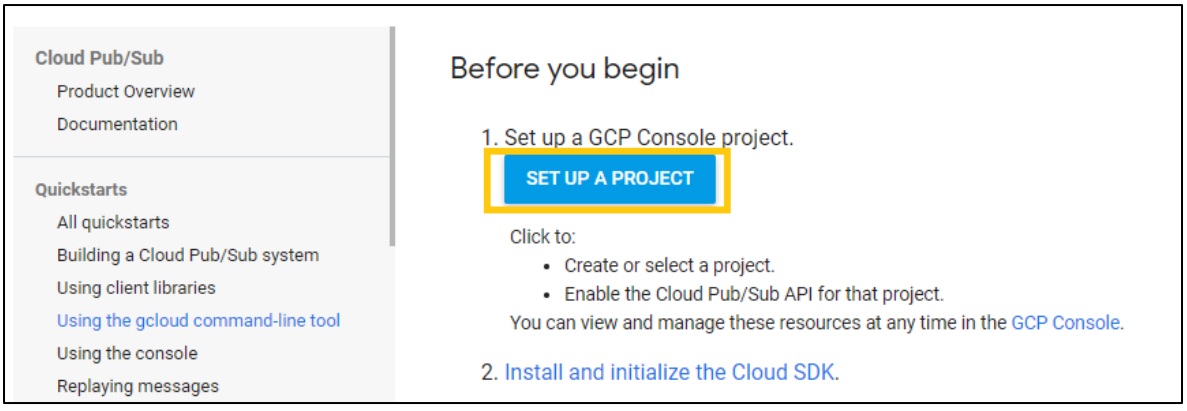

Figure 5: Setting up a GCP console project

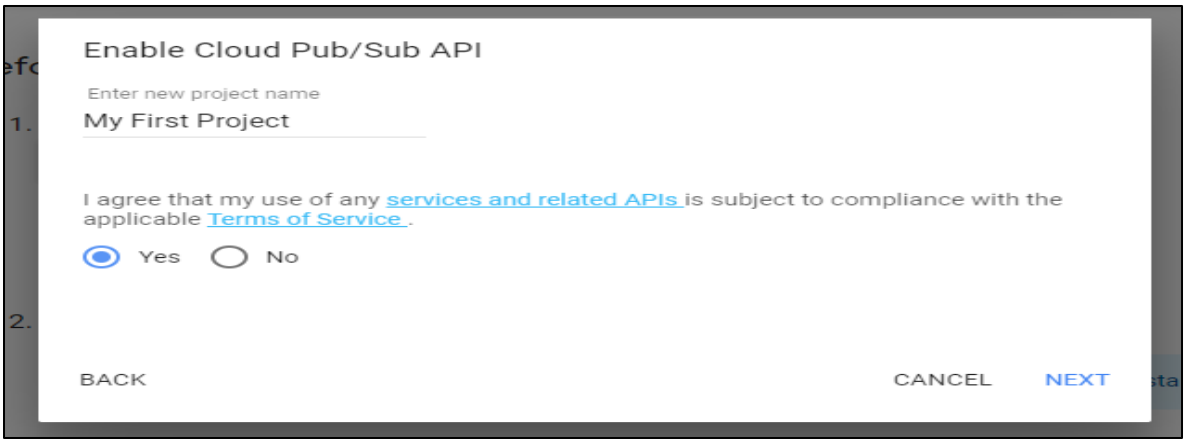

Figure 6: Setting up a GCP console project - 2

Step 2: Configure the compute engine virtual machine instance

To set up a virtual machine, we need to set up a compute engine on GCP and allocate the required resources to it as shown in Figure 7 and Figure 8.

Figure 7: Setting up the compute engine

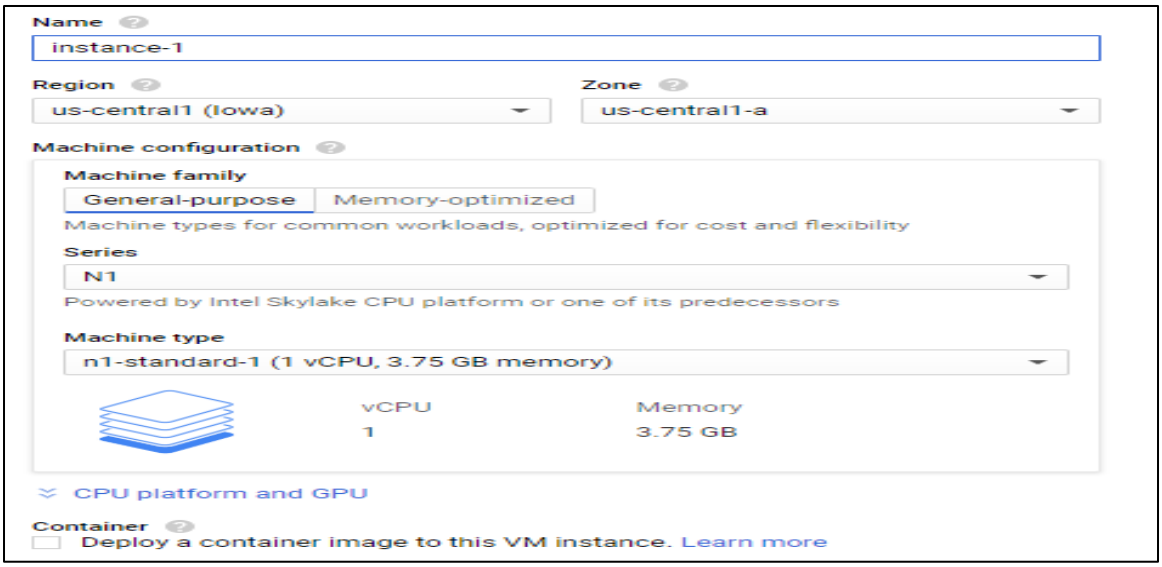

Figure 8: Setting up the compute engine - 2

Step 3: Create a boot disk space and the backend operating system

On the same compute engine page, as shown in Figure 9, select the boot disk "Deep Learning Image: TensorFlow…" for inbuilt support of machine learning libraries and TensorFlow framework.

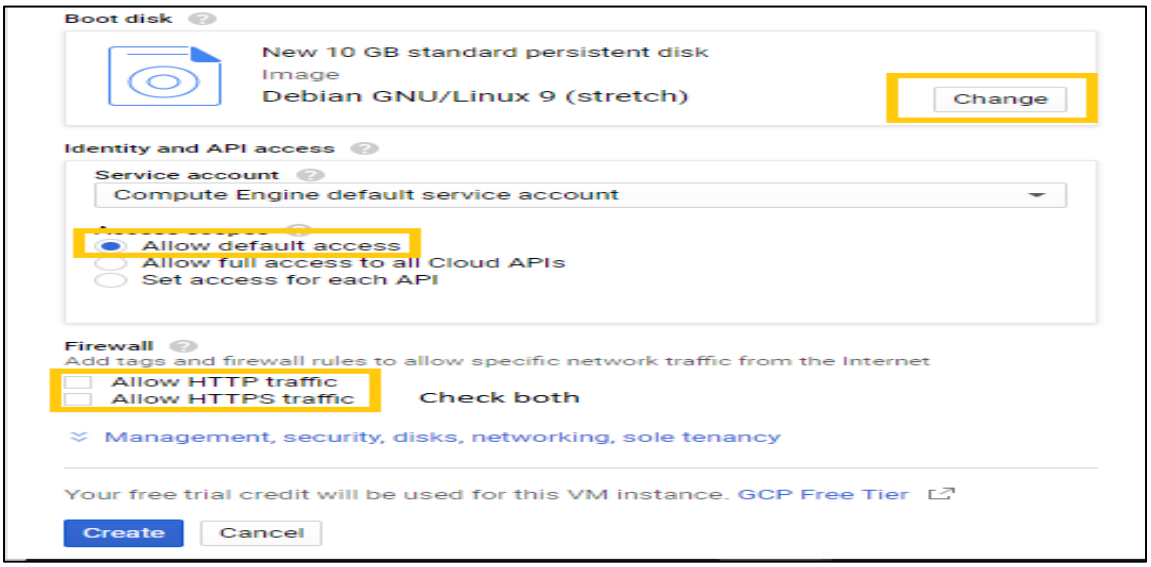

Figure 9: Setting up the compute engine - 3

Step 4: Accessing the virtual machine instance from command line

As shown in Figure 10, click on SSH -> browser window to start the command line console.

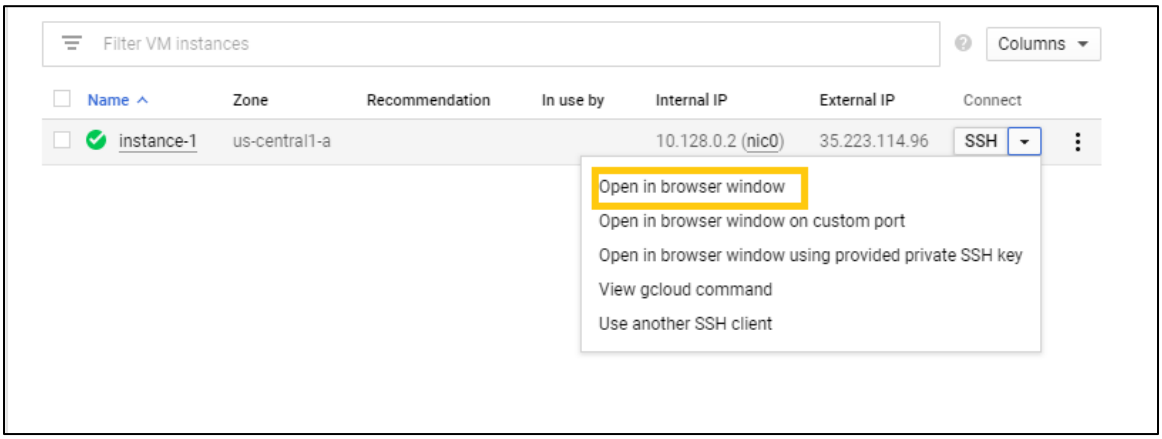

Figure 10: Accessing command line console

5.3. Time-Series Modeling

Most of the demand prediction tasks today are carried out using the standard time-series models. To understand the importance and power of graph methods and doing a comparative study, we first consider the prediction as a pure time-series problem. We implemented the following traditional time-series and deep learning sequencing models:

1. ARIMA (*A*uto-*R*egressive *I*ntegrated *M*oving *A*verage)
2. SARIMA (*S*easonal *A*uto-*R*egressive *I*ntegrated *M*oving *A*verage)
3. LSTM (*L*ong *S*hort-*T*erm *M*emory)

5.3.1 ARIMA Model:

ARIMA model is a class of statistical models that are used for analyzing and forecasting the time-series related data. The parameters to this model needs understanding of the individual terms, Auto-Regression, Integrated and Moving Average. Auto-Regression uses the dependent relationship between a given observation and 'n' number of

lagged observations. Integrated part in the model means differencing the given time-series to make it stationary. Moving average part does a dependency check between the observation and residual error from lagged observations. Each of the term has a parameter associated in the algorithm. It can be represented as (p, d, q) in a respective order. The time-series we dealt with were stationary as visible in Figure 11 and hence we avoided differencing the data.

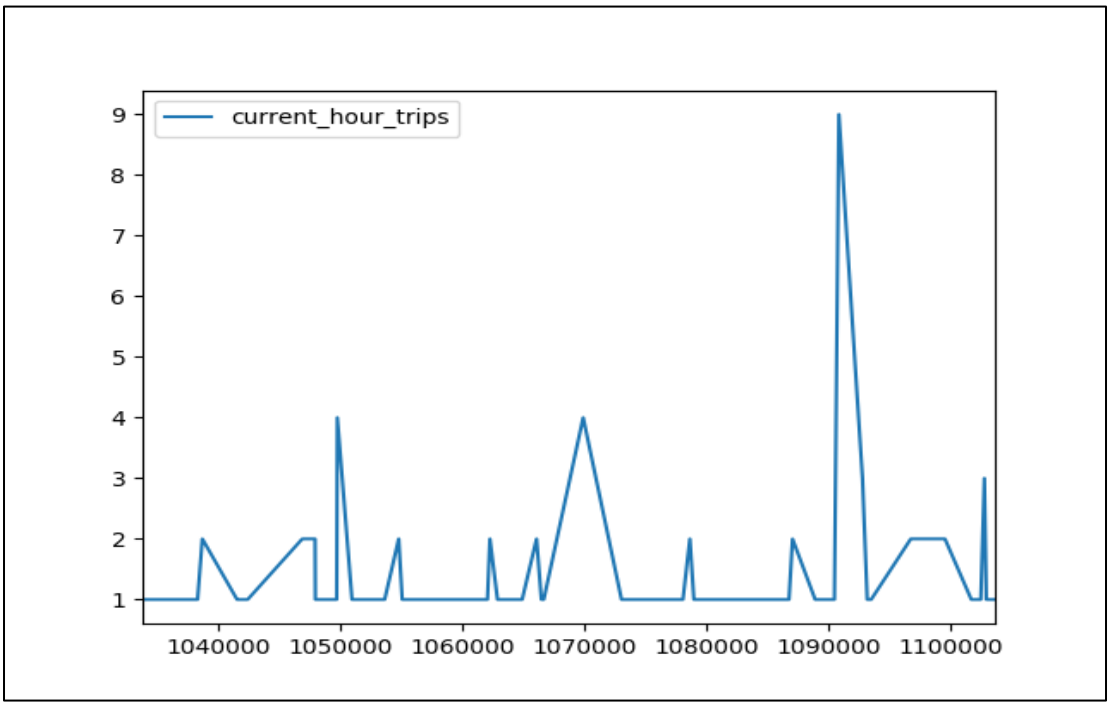

Figure 11: Time Series Trend Visualization

ARIMA is a Box-Jenkins approach and hence we also used auto-correlation plots as per station to check the randomness in the data. Table 1 lists all different values of the lagged observations and moving average that we used for experiments.

Table 1: Experimental values for ARIMA model

Number of Lagged Obs.(p)	Differencing Value	Size of Moving Average(q)
5	0	0
3	0	0
3	1	0
12	0	0
24	0	0
48	0	0

5.3.2 SARIMA Model:

SARIMA stands for Seasonal Auto-Regressive Integrated Moving Average model. The additional part that distinguishes SARIMA from ARIMA is the Seasonality trend analysis. Since the bike rides demand pattern exhibits a strong seasonal pattern due to effect of daily office hours which is repeating more or less on the same time every day, it is said that, seasonal ARIMA (SARIMA) models are particularly relevant to model traffic flow behavior. In many studies, the SARIMA model is found to out-perform simple ARIMA.

SARIMA model accepts three parameters for seasonality check (P, D, Q, m) which stands for seasonal auto-regression, seasonal difference order and seasonal moving average and time steps to be considered for a single seasonal period. Table 2 lists all the different values of P, D, Q and m for which experiments were conducted.

Table 2: Experimental values for SARIMA model

Seasonal Lagged Obs.(p)	Seasonal Differencing	Seasonal Moving Average(q)	Time Steps(m)
1	1	1	24
2	1	1	24
5	1	1	24
1	1	1	168
2	1	1	168

5.3.3 LSTM Model:

The previously implemented ARIMA, SARIMA models are mainly linear models and hence they are inefficient in predicting the non-linear and stochastic nature of the bike flow [10]. Hence, we implemented the deep neural network architecture of LSTM to forecast the bike demand at a particular hour using just the time series data.

We stacked three layers of LSTM blocks to better capture the temporal representations. Using adaptive learning rates and optimizers Adam, AdaGrad and SGD, we trained our network on 200 epochs with batch size of 24. As our prediction was a single step prediction of future, we avoided considering Bi-directional LSTM for this problem.

5.4. Graph Methods

Experimenting with both linear time-series and deep learning methods for temporal bike demand forecasting, we were able to observe that the effect of one station over another is not captured in the process of prediction. We then implemented the Station-Graph architecture as discussed in Chapter 2. We created a distance graph and adjacency matrix as inputs along with the temporal data. The complete architecture of Station-graph is

convolutional and hence the kernel parameter holds a great importance. We divided the whole dataset into sequences with history window for 24 hours, 72 hours and 168 hours with the future window of 1 hour. Each of the observation represents the bike rides of that particular hour. For both spatial and temporal convolutions, we used the kernel size values of 3, 5, 6, 9 and 12. The channels of the three-layers in each ST-Conv Block are 64, 16, 64 respectively. We stacked three ST-Conv Blocks with the final layer as dense layer with a single step prediction output. We trained our models by using Adam, AdaGrad, RMSProp optimizers with RMSProp optimizers performing the best. The model was trained over 200 epochs with batch size of 100 with total 219 batches with the learning rate of 0.001. To avoid using a static learning rate, we applied time-decay rate of 0.8 after every 10 epochs [8].

The models were trained and tested on a cloud Google Cloud Platform (GCP) with Tesla K80 GPU. The models were evaluated in three different settings, Station-level, Cluster-level and Average station-level using the Root Mean Square Error as the metric. As not all stations are as frequent as compared to the stations near downtown or busy-streets, the RMSE score for those stations was comparatively high. Considering this scenario, we selected the top 10 frequent stations from where the in-flow and out-flow of bikes was maximum. The average station-level considered all the stations for evaluation of the model's prediction.

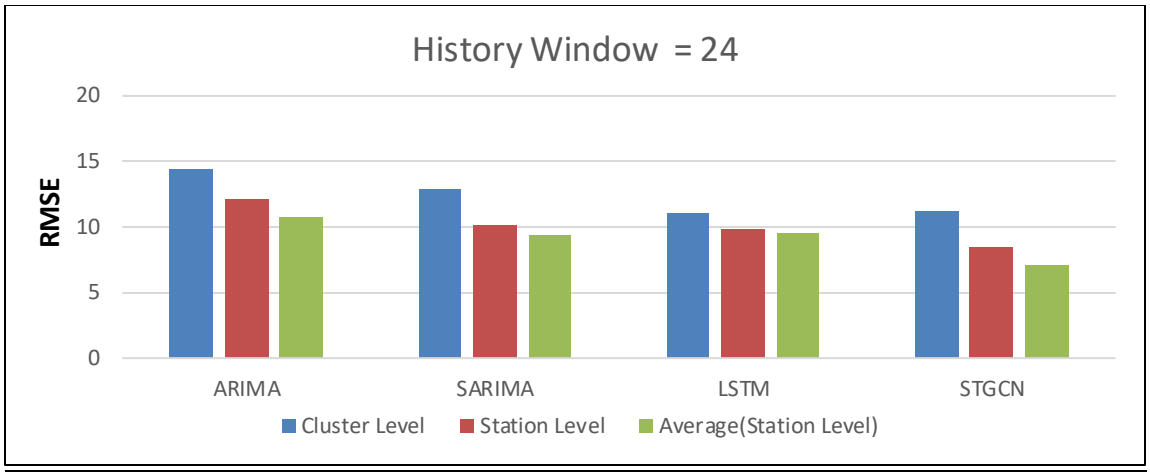

Figure 12: Comparative results for history window of 24 hours

In Figure 12, the results obtained for a history window of 24 hours are demonstrated. Aggregated ride information for the last 24 hours across all stations was used to predict the ride demand for next hour. ST-GCN outperformed ARIMA, SARIMA and LSTM with station-level RMSE score of 8.421.

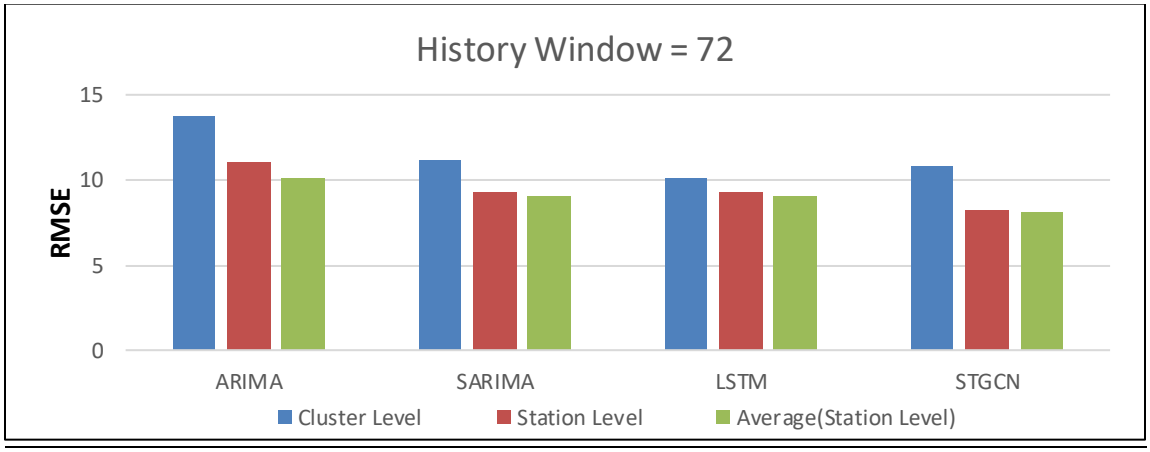

Figure 13: Comparative results for history window of 72 hours

We repeated the experiments with the history window of 72 hours as shown in Figure 13 and it gave better results as compared to 24 hour history window with RMSE of 8.207 due to the repeated patterns in the demand value.

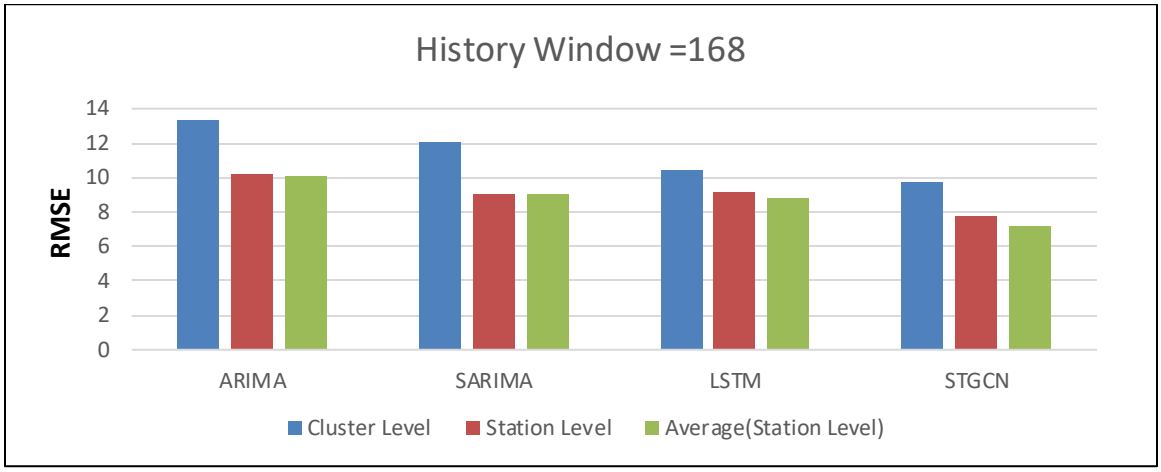

Figure 14: Comparative results for history window of 168 hours

As the bike rides at stations observe the same pattern every week during specific hours of the day like early commute hours to work and evening hours while returning from work, using the history window of 168 hours fetched the best results at station-level with RMSE score of 7.794. Figure 14 shows the results for the history window of 168 hours for prediction. The best of the RMSE score value for ARIMA was obtained with 24 hour lagged observations input as the rides at a certain hour of the day at a particular station had almost the same pattern on daily basis. But, for experimental purposes, we used the complete week data and not only the weekdays data. To match the week pattern, we implemented the seasonal version of the ARIMA model called SARIMA.

In comparison to normal ARIMA model, SARIMA performed comparatively better in terms of prediction for seasonality of one week i.e. 168 hours.

Chapter 6

RELATED WORK

6.1 Flow and Demand Prediction

Zeng et al. did the early work in the direction of traffic flow and demand prediction where the problem of tourism flow in a short-term forecast was addressed using the linear time-series forecasting box-jenkins approaches, ARIMA and ARMA [6]. An approach of using time-series for solving problem using the hybrid model combining ARIMA and Artificial Neural Network (ANN) using the Hangzhou city traffic flow dataset sampled over every 8 mins recorded from peak hours of 7 am – 7 pm was implemented by Zeng et al. [6]. In order to enhance the previous work on traffic flow prediction, Li and Cao discovered the seasonal trend in the traffic flow and proposed the Seasonal Auto-Regressive Integrated Moving Average (SARIMA) model by considering the whole day traffic and aggregated the 10 minutes traffic into a single one to better analyze the trend and giving a better performance overall [10]. New models using the linear time-series models that were implemented were insufficient in non-linear patterns in the data. Kumar and Vanajakshi came up with an architecture of the recurrent neural network architecture called Long Short-Term Memory (LSTM) with both simplex and stacked LSTM architectures considering the travel data of a museum spanned over 2 years and got a better prediction result. In general, all used the approach of linear time-series models and recurrent neural networks to solve the traffic flow problem. Fattah et al. developed an LSTM architecture to predict the bike flow across 800 stations for the city of New York

along with addressing the issue of data scarcity in the bike sharing dataset due to the great variance in the accessibility and traffic at different stations across multiple regions [2]. All the work proposed for traffic flow prediction or bike sharing demand prediction considered the prediction process as solely dependent upon the time-series which somehow proved incapable of working with factors of connectivity, accessibility and locality of the streets or the bike stations.

6.2 Graph Convolutional Neural Networks

Sperduti and Starita originally designed a graph neural network by applying directed acyclic graphs on neural networks to learn about the node representation by using neighborhood information propagation rule [11]. On the other side, a graph convolutional neural networks for semi-supervised classification was proposed by Kipf and Welling where they demonstrated the creation of graph data from the grid data using the concepts of adjacency matrix, feature matrix and weight matrix and explained the use of it over a toy-dataset of karate club [12]. As the graph neural networks started to revolutionize the classification problems, Li et al. developed a graph convolutional neural network for scene graph generation, where the scene understanding was performed over an image and the pairwise relationship between the detected objects was predicted using the graphs [13]. A new architecture of Graph R-CNN was proposed by Yang et al. where a Relation Proposal Network (RePN) was highlighted that deals with all possible relationship between the detected objects which, both, efficiently and effectively seizes the contextual information

among the objects and the relationship matrix between them [14]. While the application of graph neural network in image processing increased due to graph's efficiency in identifying the patterns, graph methods gained popularity for natural language processing tasks too. Yao et al. came up with a new graph convolutional neural network architecture for text classification using the 20Newsgroup dataset and others where they built a single text graph for whole text using the co-occurrence of the word and the word relationship across documents' corpus [15]. Scalability had been a major concern as the graph size grew exponentially with increase in the number of entities in the dataset. Gao et al. originally developed a large scale graph convolutional neural network to address the concern of scalability [16]. All the works proposed where somehow based on a single graph convolution considering which, Chai et al. developed a multi-graph convolutional architecture for prediction of bike-flow at station level using the Citi bike dataset for New York City and Chicago [7]. Chai et al. created multiple graphs namely a distance weight graph, a feature correlation graph and a connectivity graph and fused them to serve as an input to the graph convolutional network which followed by a LSTM layer predicted the number of rides at station-level [7]. On the other side, Zhao et al. proposed a temporal graph convolutional network for traffic forecasting considering the urban topology of road network by handling the dynamic changes in the traffic using the gated recurrent network [18]. Training a graph neural network holds equal importance to creating graph and a graph neural network. To address this concern, Bui et al. came up with a detailed study of neural graph learning [19]. Monti et al. performed a detailed study of geometric deep learning for

convolutional neural network which highlighted the reason behind the power of graphs for prediction [20].

Our approach differs from all aforementioned in the following aspects. (1) We considered a single graph i.e. distance graph for spatial relationship analysis. (2) We applied spatial-temporal graph convolutional network to predict the bike flow across all stations using multiple history windows of 24 hours, 72 hours and 168 hours. (3) We reduced the need to apply recurrent neural network layer separately to make the approach more effective and efficient.

Chapter 7

CONCLUSION AND FUTURE WORK

In this report, we introduced a new type of neural network architecture, Station-graph, based on Spatial-temporal graph convolutional network to predict the flow of bikes at station-level and at cluster-level. Station-Graph predicts the number of bikes for the upcoming hour based on the number of bike rides in the previous hours. Station-graph combines the bike ride data for temporal convolution with the distance graph and connectivity graph for spatial convolutions. Extensive experiments on the real-world Ford Go bike dataset show that Station-graph outperforms the state-of-the-art models for demand prediction in three different settings, station-level, cluster-level and average station-level in terms of RMSE.

During the complete course of the project, we faced a couple of challenges which we overcame to successfully complete this project. Data preparation was a daunting task which included segregating the data into time series data and spatial graphs. Additionally, analyzing the spatial data demanded much efforts for developing an efficient relationship between stations. Developing a non-conventional neural network architecture required a lot of research and study. We faced substantial amount of challenges for training the network on cloud along with trying multiple set of hyperparameters to decrease the error rate.

In future, we plan to extend the work by trying weekday and weekend data separately along with anomaly detection. We also plan to apply attention networks to try getting a better accuracy in prediction. We also plan to implement multi-graph Spatial-

temporal graph convolutional network by creating multiple feature graphs, frequency graph and easy-of-connectivity graph.

References

1. BikeMunk, "Biking statistics articles that won't surprise you at all if you love cycling," [Online]. Available: https://www.bikemunk.com/biking-statistics/ [Accessed: February 2019].

2. J. Fattah, L. Ezzine, Z. Aman, Haj El Moussami and A. Lachhab, "Forecasting of demand using ARIMA model," *International Journal of Engineering Business Management,* vol. 10, no. 1, pp. 311-320, October 2018.

3. S.V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal ARIMA model with limited input data," *European Transport Research Review,* vol. 7, no. 3, pp. 1-9, September 2015.

4. Y. Pan, R. C. Zheng, J. Zhang and X, Yao, "Predicting bike sharing demand using recurrent neural networks," in *Proceedings of the 2018 International Conference on Identification, Information and Knowledge in the Internet of Things,* Beijing, China, pp. 562-566, October 2018.

5. S. Zhang, H. Tong, J. Xu and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks Journal,* vol. 6, no. 1, pp. 244-267, September 2019.

6. D. Zeng, J. Xu, J. Gu, L. Liu and G. Xu, "Short Term Traffic Flow prediction using hybrid ARIMA and ANN models," in *Proceedings of the 2008 Workshop on Power Electronics and Intelligent Transportation System*, Guangzhou, China, pp. 621-625, August 2008.

7. Di Chai, L. Wang and Q. Yang," Bike flow prediction using multi-graph convolutional networks," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Seattle, WA, pp. 397-400, November 2018.

8. B. Yu, H. Yin and Z. Zhu, "Spatio-Temporal graph convolutional neural networks: A deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, pp. 3634-3640, July 2018.

9. E. Weisstein, "Adjacency Matrix," [Online]. Available: http://mathworld.wolfram.com/AdjacencyMatrix.html/ [Accessed: April 2019].

10. Y. Li and H. Cao, "Prediction for tourism flow on lstm neural network," in *Proceedings of the 2017 International Conference on Identification, Information and Knowledge in the Internet of Things*, Qufu, China, pp. 277-283, October 2017.

11. A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, May 1997.

12. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, pp. 1412- 1425, April 2017.

13. Y. Li, W. Ouyang, B. Zhou, K. Wang and X. Wang "Scene graph generation from objects, phrases and region captions," in *Proceedings of the 2017 IEEE International Conference on Computer Vision*, Venice, Italy, pp. 261-270, October 2017.

14. J. Yang, J. Lu, S. Lee, D. Batra and D. Parikh, "Graph R-CNN for Scene Graph Generation," in *Proceedings of the 15th European Conference on Computer Vision*, Munich, Germany, pp. 690-706, September 2018.

15. L. Yao, C. Mao and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, HI, pp. 7370-7374, January 2019.

16. H. Gao, Z. Wang and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, London, United Kingdom, pp. 1416-1424, August 2018.

17. W. Shalaby, B. AlAila, M. Korayem, L. Pournajaf, K. AlJadda, S. Quinn and W. Zadrozny, "Help Me Find a Job: A Graph-based Approach for Job Recommendation at Scale," in *Proceedings of the 2017 IEEE International Conference on Big Data*, Boston, MA, pp. 1544-1553, December 2017.

18. L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng and H. Li, "T-GCN: A temporal graph convolutional neural network for Traffic Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1-11, August 2019.

19. T. D. Bui, S. Ravi and V. Ramavajjala, "Neural graph learning: Training neural networks with graphs," in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, Los Angeles, CA, pp. 64-71, February 2018.

20. F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda and M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, pp. 5425-5434, July 2017.