CHALMERS
UNIVERSITY OF TECHNOLOGY

Graph Neural Network for Traffic Demand Prediction

Master's thesis in Infrastructure and Environmental Civil engineering

Richard Chamoun
Alexander Wallenbring

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGIEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2022
www.chalmers.se

MASTER'S THESIS ACEX30

Graph Neural Network for Traffic Demand Prediction

RICHARD CHAMOUN
ALEXANDER WALLENBRING

Department of Architecture and Civil Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2022

Graph Neural Network for Traffic Demand Prediction
RICHARD CHAMOUN
ALEXANDER WALLENBRING

Master's Thesis 2022
Department of Architecture and Civil Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Abstract

The urban transportation systems are under more pressure than ever before, with more people living in urban areas, higher travel demand, congestion and increased emissions. It is an urgent and indispensable trend to develop sustainable transportation to reduce traffic-related issues. One of the emerging sustainable transportation systems is shared mobility such as ride hailing, shared e-scooter and bike/e-bikes. Shared mobility enables the sharing of different types of vehicles and increase the usage efficiency of vehicles. Meanwhile, active mobility sharing such as bike, e-bike and e-scooter sharing are environmentally alternatives as compared to private cars and are very prevalent in current European countries. One of the fundamental components of operating shared mobility is accurate demand prediction in spatial and temporal dimensions, which are key inputs for vehicle dispatching and rebalancing. Thanks to the digitalization of transportation systems, big data regarding the usage demand in different periods and areas in a city are collected. Meanwhile, advancements in artificial intelligence and machine learning have paved the way for new techniques to leverage the great power of data-driven methods and big data for precise travel demand prediction of shared mobility system.

The aim of this study is to design a graph neural network algorithm to quantify short-term demand of shared mobility systems. The algorithm specially takes both spatial- and temporal aspects into consideration when predicting the demand in a data-driven way. We implement the proposed algorithm for a case study in bike-sharing demand prediction in the city of Nanjing, China. The data in the case study cover three months including October, November and December 2016 where each month has more than 3 million transaction data. The given data contains among other things the ID for each station, rental- and return time and longitudinal- and latitudinal coordinates. Before initiating the coding process, a literature review was conducted, where artificial intelligence and most importantly deep learning and neural networks are analyzed and explained in detail in order to lay the theoretical foundation and select the best deep learning method for our research purpose. The algorithm is structured by first defining the structures and layers, and then proceeding with building the model and training- and testing the model. There are over 800 bike stations in the datasets. However, not all stations are used for training and testing the graph neural network on account of computation efficiency and time. The analysis uses 50-, 100-, and 200 stations separately on account of the potential impacts of data amount on results. In order to measure the accuracy of the proposed algorithm, two commonly used error metrics in deep learning will be employed, mean absolute error (MAE) and root mean square error (RMSE). It should be noted that We utilize the available bike sharing data for case study, but the algorithm is general and can also be used for other mobility systems such as regular car traffic or train systems as well.

The results show that the MAE for the validation set and the testing set are around 6.2 and 6.27, respectively, while the RMSE for the validation set and the testing set are approximately 8.6 and 9.1, respectively. These numbers were for the base architectural layout of the graph neural network and for the hourly demand. This study also conducts additional experiments for hyperparameter tuning and also changing the block layout for the model to find the best model, for instance the most notable improvements were for the 100-station analysis which showed approximately 12-19 % decrease in the error metrics. Most notably, the lowest MAE and RMSE values are obtained for the 200-station analysis. In additional to the two error metrics, there will be graphs added that highlights the demand errors for every station analysis. This showed, from a monthly perspective, that the graph neural network actually predicted the demand relatively close to the real demand.

The proposed graph neural network has the ability to predict short-term bike demand with relatively high accuracy. However, there is much room for improvement where meteorological condition could have also been implemented in the model, which would with a high probability increase the accuracy of the predictions.

Keywords: Artificial intelligence, deep learning, neural network, graph neural network

Sammanfattning

De urbana transportsystemen är under högt tryck, med fler människor som bor i stadsområden, högre resebehov, trängsel och ökade emissioner. Det är en vitalt att utveckla hållbara transporter för att minska trafikrelaterade problem. Ett av de framväxande hållbara transportsystemen är delad mobilitet som e-scooter och cykel/e-cyklar. Delad mobilitet möjliggör delning av olika typer av fordon och ökar användningseffektiviteten hos fordon. Samtidigt är aktiv mobilitetsdelning som cykel, e-cykel och escooter-delning ett miljömässigt alternativ jämfört med privata bilar, och är mycket utbredd i nuvarande europeiska länder. En av de grundläggande komponenterna för att driva delad mobilitet är noggrann efterfrågeförutsägelse i rumsliga och tidsmässiga dimensioner, vilket är nyckelinputs för fordonssutryckning och ombalansering. Tack vare digitaliseringen av transportsystem samlas extrema mängder data under olika perioder och områden i staden. Samtidigt har framsteg inom artificiell intelligens och maskininlärning banat väg för nya tekniker för att dra nytta av den stora kraften hos datadrivna metoder och big data för exakt förutsägelse av resebehov av delat mobilitetssystem.

Syftet med denna studie är att implementera grafstrukturerad data, som ofta förekommer i trafiknätverk, för att kvantifiera kortsiktig cykelefterfrågan i staden Nanjing, Kina. Detta kommer att göras genom att använda programmeringsspråket python i jupyter lab för att designa en neural nätverksalgoritm. Algoritmen kommer att ta hänsyn till både rumsliga och tidsmässiga aspekter vid beräkning av efterfrågan. Data som kommer att implementeras i denna studie kommer att vara för oktober, november och december 2016 där varje månad hade en separat excel-fil som innehöll mer än 3 miljoner rader med data. Innan kodningsprocessen påbörjades genomfördes en litteraturgenomgång, där artificiell intelligens och viktigast av allt djupinlärning och neurala nätverk analyserades och förklarades i detalj för att lägga grunden för avhandlingen. Koden strukturerades genom att först definiera skikt, och sedan fortsätta med att bygga modellen och träna- och testa modellen. Efter dessa steg implementerades data enligt vår fallstudie. De givna uppgifterna innehåller bland annat ID för varje station, hyres- och returtid samt longitudinella- och latitudinella koordinater. Eftersom trafiknätverk fungerar väldigt likt cykelnät när det gäller grafstruktur, kan algoritmen även användas för andra mobilitetssystem som biltrafik eller tågsystem. Det finns över 800 stationer i staden, på grund av tidsbrist kommer inte alla stationer att användas, och analysen kommer att begränsas till 50-, 100- och 200 stationer. För att mäta noggrannheten i det neurala nätverket kommer två felmått som vanligtvis ses inom djupinlärning att användas, mean average error (MAE) och root mean square error (RMSE).

Resultatet visade att medelvärdet för MAE för valideringsuppsättningen var cirka 6,2 och 6,27 för testsetet medan RMSE för valideringsuppsättningen var ungefär 8,6 och 9,1 för testsetet. Dessa siffror gällde den grundläggande arkitektoniska layouten för det neurala nätverket, där det även finns ytterligare experiment för hyperparameterinställning och även att ändra blocklayouten för modellen, till exempel var de mest anmärkningsvärda förbättringarna för 100-stationanalysen som visade ungefär 12-19% minskning av felmätningarna. Framför allt erhölls de lägsta MAE- och RMSE-värdena för 200-stationsanalysen. Utöver dessa två felmått implementerades ytterligare ett mått mer anpassat till fallstudien för att analysera modellen mer intuitivt. Detta visade, ur ett månadsperspektiv, att det neurala nätverket faktiskt beräknade efterfrågan relativt nära den verkliga efterfrågan. Sammanfattningsvis har det neurala nätverket förmågan att förutsäga kortsiktig cykelefterfrågan med relativt hög noggrannhet. Det finns dock mycket utrymme för förbättringar där meteorologiska

förhållanden också hade kunnat implementeras i modellen, vilket med stor sannolikhet skulle öka noggrannheten i förutsägelserna.

Nyckelord: Artificiell intelligens, djup lärning, neurala nätverk, grafneurala nätverk

Acknowledgements

First and foremost, we would like to express our sincerest gratitude towards Kun Gao and Yang Liu. Both have been an integral part of this thesis where Kun Gao has provided us with vital support throughout the project and Yang Liu has provided a helping hand in terms of coding and special expertise in AI and deep learning, even outside regular teaching hours in order to finish the thesis in time. Not only have they been our supervisors, but also our friends.

Table of Contents

1. Introduction

In today's society, the field of transportation engineering has many different challenges that are continuously becoming more complex. As the global population increases combined with economic development, the need for transportation will significantly increase. In the coming years, transportation engineers will face advanced problems that will be difficult to solve with conventional methods where the transportation systems will have more components and aspects that need to be taken into consideration. This, combined with the continuous increase in urbanization, will exhibit immense pressure on todays and future urban transportation systems and will cause traffic-related issues such as increased congestion and lower traffic efficiency (Rico et al., 2021).

Finding efficient ways of managing travel demand has in recent years become increasingly important and is a vital component regarding the efficiency of shared mobility systems. This means finding new and more innovative ways of predicting travel demand in both temporal and spatial dimensions. There has been a substantial increase in the amount of data available due to the ongoing digitalization of many systems, including transportation. The increase in available data means that the capability of computation also significantly improves, which provides a vital opportunity to solve challenges by utilizing innovative solutions (Rico et al., 2021). These innovative solutions can be intelligent transportation systems, abbreviated ITS, which have started to gain increasingly more attention. With the amount of traffic sensors and innovative traffic sensor technologies, big data and artificial intelligence provides great potential for the accurate demand predictions of spatiotemporal demands in transportation systems (Lv et al., 2014).

Today, there are several different travel demand prediction models based on old approaches, it has been a long time since a new innovative approach has been presented within this field of transportation. Deep learning algorithms can utilize massive amounts of data efficiently and accurately to predict traffic demand. When it comes to deep learning in traffic prediction, artificial neural networks are one of the leading methodologies (David et.al., 2022). This can be attributed to the computing power that can successfully model traffic patterns that are non-linear and complex (David et.al., 2022). The most common neural network architectures that are utilized in traffic prediction are Convolutional-, Feedforward-, and Recurrent Neural Networks (David et.al., 2022).

Over the last few years, in order to accurately model the natural occurring graph structures in transportation systems, graph neural networks have been successfully utilized and have shown excellent performance in forecasting traffic demand (Jiang et.al., 2022). As previously mentioned, predicting traffic demand can be challenging due to complex spatial- and temporal dependencies. There have been advanced models designed to deal with temporal dependency, such as temporal convolutional- and recurrent neural networks (Jiang et.al., 2022). However, there are not many existing models that deal with the complex spatial aspects of traffic demand forecasting (Jiang et.al., 2022). Spatial elements refer to the nonlinear relationship when it comes to traffic status in separate locations (Jiang et.al., 2022). These locations could for example be bike rental stations, road intersections, and subway stations.

1.1 Aim and research questions

The aim of the study is to create a deep learning algorithm using graph neural network to predict the short-term travel demand of shared mobility system. We firstly develop the architecture and model

formulations of graph neural network. To test the performance of the model, we utilize the dataset regarding bike sharing systems in Nanjing, China for case study. We implement the deep learning method in Python. The following research questions have been set up for the study to answer.

- How accurately can the graph neural network predict traffic demand?
- How can the model take spatial- and temporal correlations into consideration?
- How will prediction accuracy be affected when hyperparameters are modified?

2. Literature review

In this section, a brief overview will be provided for the reader to understand the aspects and fundamental pillars of this project. In terms of artificial intelligence, machine learning, deep learning, neural network, graph, graph neural network, and graph convolutional network. An introduction to how these subjects corresponds to one other and their development throughout the last decade. The relationship between each subarea is illustrated in Figure 1. However, the relationship between each subject could have been made more complex and divided into even smaller subareas to get an even deeper dive into AI. For simplicity and subject-related terms, Figure 1 has only been restricted to the areas of interest.

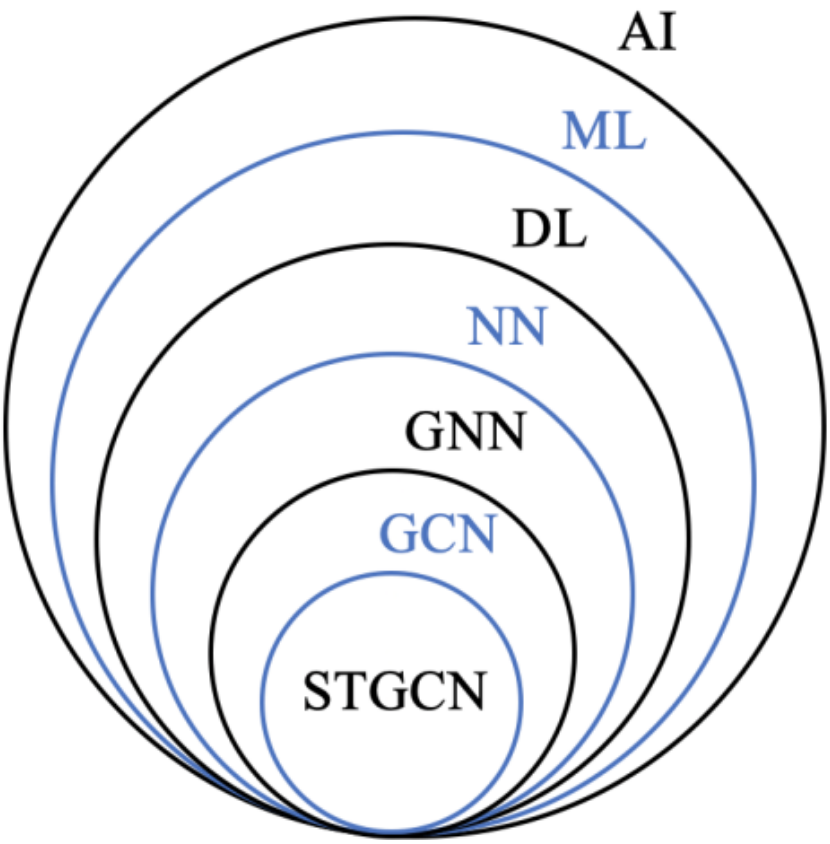


Figure 1. Illustration of the relationship between each sector.

To further understand the concept, a list is provided below to understand Figure 1. Each sector will be described in-depth in order for the reader to understand the concept. However, the study will mainly focus on graph neural network, but it is important to understand all areas to fully understand the result and technical aspects of this study.

AI = Artificial Intelligence
ML = Machine Learning
DL = Deep learning
NN = Neural Network
GNN = Graph Neural Network
GCN = Graph Convolutional Network
STGCN = Spatial-Temporal Graph Convolutional Network

2.1 Artificial intelligence

Artificial intelligence, also known as AI, is the first topic to be covered, as shown in Figure 1. AI contains more subareas than what is visualized in Figure 1 and more complex intelligence methods. However, for the study purpose, it has been narrowed down. According to Russell and Norvig (2010), AI was first mentioned at Dartmouth college in 1956 during a conference (Dunjko and Briegel, 2018) whose focus was to discuss the possibility of learning a machine to think like a human brain and take actions accordingly. This was a very new subject that had not been heard of before. Even in the late 90s, the development of AI was still an underdeveloped subject with a lack of interest, but the interest has been catching up during the last decade.

However, after the growth in AI (Oke 2008), artificial intelligence could be divided into sixteen smaller categories where it was useful: reasoning, programming, artificial life, belief revision, data mining, distributed AI, expert systems, genetic algorithms, systems, knowledge representation, machine learning, natural language understanding, neural networks, theorem proving, constraint satisfaction, and theory of computation. In these sixteen categories, this study will focus on machine learning and neural networks. According to Razavian (2020), AI is nowadays used a lot in computer programs in different forms such as board games, recognizing speech, identifying objects in images, understanding relations between entities in text, or translating text between two languages. Besides this, AI will be an important subject in more developed areas such as medical care, transportation, construction, manufacturing, and more (Chan, 2019). Artificial intelligence is highly complex but essential in today's society.

Artificial intelligence can be coded with several different languages and computer programs. For example, using Python for coding is more appropriate compared to HTML, which is suitable for creating websites. According to Wirth (1971), a computer program consists of two essential parts: Actions and Data. Depending on the type of description of the data, action will be taken accordingly. This is a very brief overview of the language and can be divided into smaller areas.

2.2 Machine learning

Machine learning (ML) is a more technical area compared to AI in general. Machine learning is a wide topic. According to Razavian (2020), there are three main subareas, unsupervised learning, supervised learning, and reinforcement learning. However, Haldorai et al. (2020) stated that there are four different subareas of machine learning which are supervised, semi-supervised, unsupervised, and reinforcement learning. In this chapter, the four main subjects are covered both in general terms and in a technical way. The input variable is the given data set. The output is what the model is trying to predict. The last parameter is training, which is the result of how good the model is at predicting the outcome and the level of errors in the model (Razavian, 2020).

2.2.1 Supervised

According to Berry et al. (2019) and Dunjiko and Briegel (2018), what makes supervised learning unique compared to the other methods is that supervised learning only uses labeled data in the input variable. Otherwise, the program cannot calculate an outcome. This means all columns and rows have their own labels for the program to be able to calculate a prediction. This is done by first labeling the data and then training the data. A good visual illustration of how supervised learning works is shown in Figure 2. Supervised learning is not programmed to recognize patterns and is simpler and more straightforward. However, there is two types of supervised models, regression and classification (Nasteski, 2017). Nasteski (2017) stated that the most used and most relevant model is the

classification model, hence why this chapter have focused on classification. The supervised classification model only considers the labeled data and draws a linear classifier that is shown in a black dotted line in Figure 2. This line is an average of all the data, and dependent on the X and Y values.

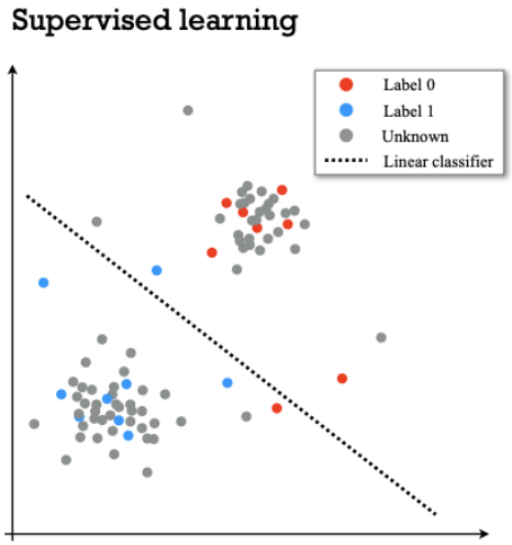


Figure 2. Supervised learning. Picture is taken from (Dunjko and Briegel, 2018)

Cunningham et al. (2008) stated that supervised learning is the most important method of machine learning. In a supervised learning method, X is the input and Y is the output. The purpose is to infer a function f from a data set, An , which includes input and output variables (Cunningham et al., 2008). The training set An is calculated by Eq. (1).

$$An = \left((X_1, Y_1), \dots, (X_n, Y_n)\right) \in (X \cdot Y)^n \quad [1]$$

One of the fundamental pillars in the algorithm for supervised learning is that the training set or data set needs to be an unknown variable and independent but a fixed joint for the probability function $P(X, Y)$ (Cunningham et al., 2008). The algorithm will try and solve the problem until it finds a function f that satisfies the conditions of $f \in H$ where H is the boundary condition that needs to be fulfilled in order to find the correct function f .

Cunningham et al. (2008) also said that the second important fundamental pillar of the supervised method is the error or loss level after the function has calculated a prediction. The importance of this is to identify the loss and to examine if the prediction is at an acceptance level (Cunningham et al., 2008). The equation to calculate the error or loss is shown in Eq. (2).

$$L = Y\, x\, Y \to IR^+ \quad [2]$$

2.2.2 Unsupervised

On the other hand, unsupervised machine learning is primarily using pattern recognition to find an outcome. This means the data does not need to be labeled for the method. A visual illustration of how this work is shown in Figure 3. There are no labels on the data, and the program is using artificial intelligence to recognize a pattern in the data set which created these two clusters of data. The two clusters can be seen in Figure 3 where one data set is marked with a red dotted line, and the other cluster is marked with a blue dotted line. Data outside these clusters are called outliers and are therefore not considered in this method since it will affect the result in a bad way.

Unsupervised learning has three important parameters, input, output, and training. In unsupervised learning, the input is referred as X and the output as Y (Ghahramani, 2003). According to Ghahramani (2003), the goal of the unsupervised learning method is to create a prediction, based on pure data and not take the environment into consideration, meaning not to use labels from the dataset. The unsupervised method is good for processing pure data, as shown in Figure 3. The method is based on creating probability, P, as a prediction.

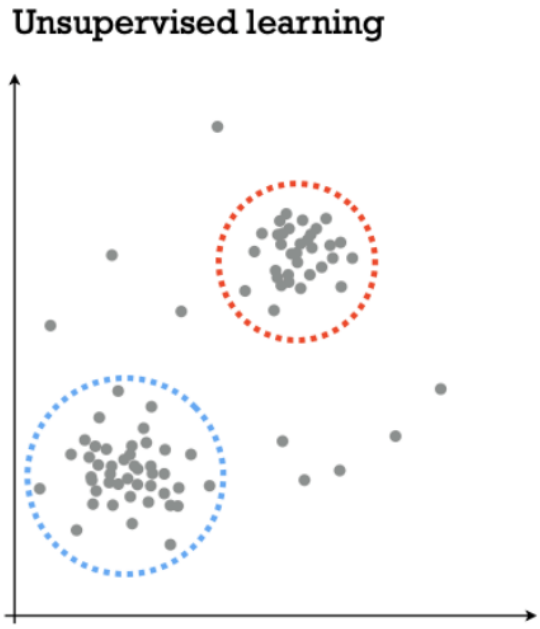


Figure 3. Unsupervised learning in cluster form. Picture is taken from (Dunjko and Briegel, 2018)

2.2.3 Semi-supervised

Semi-supervised learning is a method combining supervised and unsupervised learning. Semi-supervised learning is used when only a small amount of labeled data is available (Berry et al., 2019). Similar to supervised learning as in Eq. (1), semi-supervised learning uses unknown data $An \rightarrow An, u$, meaning, u is an unknown labeling. According to Van Engelen et al. (2020), semi-supervised learning trains and utilizes data that is unlabeled and creates a new learner which gives better result than only using labeled data. By doing this, the data will be both supervised and unsupervised to extend.

According to Van Engelen et al. (2020), there are four assumptions that need to be made for the semi-supervised learning method to work. The first one is, if the underlying marginal data $p(x)$ contains information about the posterior $p(y|x|)$, the method can use the unlabeled data to find information. But if $p(x)$ does not contain any information about $p(y|x|)$, the semi-supervised learning cannot improve the prediction accuracy (Zhu, 2005).

The second assumption is called the smoothness assumption. It assumes that if two different input values x and x' are close to each other in a dataset, the output y and $y'$are also close. Using unlabeled data together with label data, this theorem or assumption can be used. In other words, if $X_1 \in X_L$ and $X_{2,}X_3 \in X_U$ ($X_L$is the labeled data and X_U is the unknown data), and X_1 is close to X_2 and $X_2$is close to X_3, it can be assumed that X_1 is close to X_3 and have the same label (Van Engelen et al., 2020).

The third assumption is called the low-density assumption, in a dataset there is both a low-density zone and a high-density zone. A low-density zone means a low number of points in that area compared to the whole dataset. A high-density zone is similar to a cluster in a dataset where there are a lot of points in the same area. If the smoothness assumption holds, this means if an input value X_1 is inside a low-density zone, and it has a value close to X_2. In a high-density zone, it is already assumed to have similar values. However, if this is also true in a low-density zone then a semi-supervised learning method can be used (Van Engelen et al., 2020).

The last assumption is called the manifold assumption. According to Van Engelen et al. (2020), a dataset in machine learning is called Euclidean space. For example, a value X_n in a 3D dimensional can still be placed in a 2D dimension and belong to a 3D dimension. The 2D space is called a manifold and this can be implemented in semi-supervised learning. The manifold assumption follows two conditions. The first condition is as follows; the input space consists of multiple different manifolds. The second condition is; that if two values are in the same manifold, it will have the same label. If the program fulfills the four-assumption mentioned before semi-supervised learning method is a great method to use.

2.2.4 Reinforcement learning

Reinforcement learning is a more complicated and different method compared to supervised, unsupervised, and semi-supervised learning. An overview of how a reinforcement learning process operates in more technical terms is shown in Figure 4. The agent's task is to interact with the environment and take the best action to maximize the reward. A reward can either be +1, -1, or 0, depending on if the agent did a respectable job or not. Where +1 is a good job, and negative is a bad job. A job can be referred as testing the function to see if a new parameter changed the outcome in a good way. However, A state s is a summary of all the episodes and all the actions and rewards combined to present an overview of the method (Garnier et al., 2021). The agent experiences a large sequence of states, s, and actions, a. This can also be described as a trajectory shown in Eq. (3). Is equation is later used in order to calculate the reward shown below.

$$\tau = (s_0, a_0, s_1, a_1, \ \ldots, s_n, a_n) \quad [3]$$

After the decision process, whether the work was good or bad, a reward needs to be calculated. The equation to calculate the reward is shown in Eq. (4), this equation is called a discontinued reward (Garnier et al., 2021).

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t \quad [4]$$

Another author called Glorennec, (2000) said there can also be another reward system called the average reward model which is presented in Eq. (5) (Glorennec, 2000). Both reward systems are functional and dependent on the function which has the best fit.

$$R(t) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n} r_t + k \quad [5]$$

Glorennec, (2000) also said that reinforcement learning can be described as “learning with critic” which is true due to reward system. As said earlier, reinforcement learning is a bit different the other mentioned methods. But it is also an important method worth bringing up even if it does not contain a large share of the total machine learning methods.

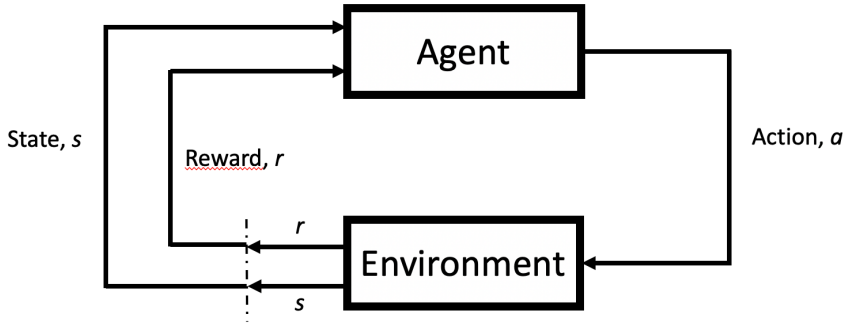


Figure 4. Process of reinforcement learning. (Garnier et al., 2021)

2.3 Deep learning

Deep learning is a more advanced method compared to machine learning where deep learning uses more parameters and larger datasets which means more complex problems can be solved. The network structure of deep learning is similar to machine learning, what differs the two methods apart is the hidden layer in between the input and output layer. Regarding the layers in a deep learning method, it contains of an input layer, an output layer and hidden layers. To get a better understanding of the structure of deep learning, a visual illustration shown in Figure 5 is presented. The green dots represent, the input layer, the blue the hidden layers and the red the outcome. The most complex calculation is processed here in the hidden layer. Below Figure 5 a short description about each parameter is done in order for the reader to understand better. Deep learning can also be divided into more sections. However, this section will only focus on deep learning from a broader view.

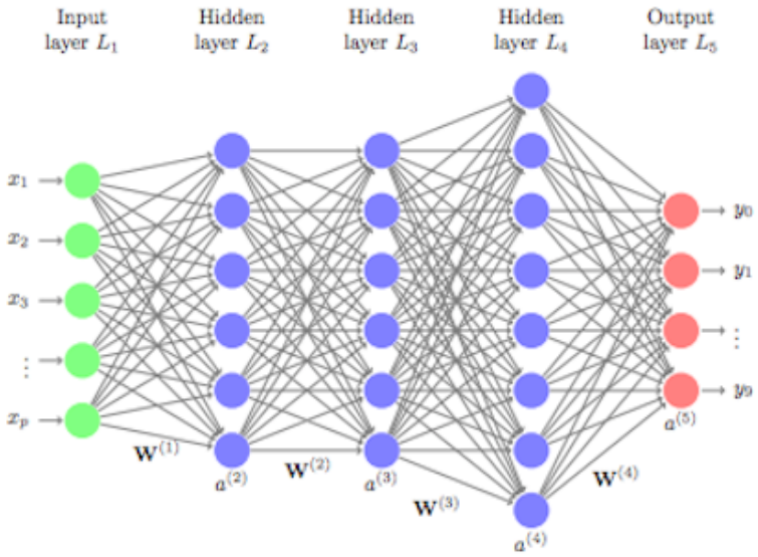


Figure 5. Layers for deep learning. (University of Cincinnati, n.d)

X_n = are the input values or neurons as they also can be called.

W_n = is the weight of the units between the hidden layers.

a_n = is the node after the weight (W_n) and the previous interaction (a_{n-1}).

L_n = is the hidden layers which include a and W .

Y_n = are the output value.

The process of deep learning as mentioned previously are mostly used for large datasets due to the time consuming and advanced calculation process. According to Li (2017), the calculation process for deep learning is as follows. First the input value interacts with each node of the first hidden layer. So, in Figure 5 the input value, X , interacts with the node a in the hidden layer. During this interaction there is also a weight process for the program to determine how much it will affect the outcome. Meaning several different conditions have been set up and it is the program's job to weigh if the node is more important than the other or not. After the interactions, the program uses nonlinear transformation in order to find a new representation of the input variable. This means after the first interaction has been calculated, in node, a_1 using the weight, W_1 and input value X_1. The value in a_1 now represents the new "input" value for the next hidden layer. In short, using nonlinear transformation creates a new representation of the input variable for the previous layer (Li, 2017).

There are mainly three methods used to analyze the result of deep learning. The methods are mean absolute error, mean absolute percentage error and root mean square error, also known as MAE, MAPE and RMSE. MAE is a measurement to find the error or loss from the predicted value and to find the true value from the observations. A low MAE score represents a low level of error in the predicted result which is essentially good because it proves that the model is fairly accurate. Training with a larger dataset will also result in a lower error compared to a thinner dataset, the equation to calculate the MAE score is shown in Eq. (40) (Wang et al., 2018). MAPE is the second method used to evaluate the result. MAPE is also a measurement of how accurate the model actually is, in a more technical view MAPE is a loss function that calculates the error of the deep learning model. MAPE and MAE is more similar evaluation methods compared to RMSE. RMSE generates a value of how far off the predicted value is from the measure true value using Euclidean distance, the equation used for this method is shown in Eq. (41) (Wang et al., 2018). An important word that will be used in this study for the coding part is "epoch" and epoch mean the number of times a training set needs to be re-calculated in order for the set to get as accurate results as possible. A higher number of epochs will result in a more accurate result compared to a lower number of epochs. However, using too many epochs will result in an overfit of the data and give a higher error value, therefore, it is important to find a balance.

2.4 Neural networks

Artificial neural networks, or more commonly referred to as neural networks, are mathematical models that aims to reproduce the function and structure of a biological neural network that is present in the human brain (Suzuki, 2011). The foundation of these neural networks consists of artificial neurons, which all have the purpose to simulate the function of a biological neuron. An artificial neuron is a mathematical function that utilizes multiplication, summation and finally activation (Suzuki, 2011). In the first layer the neurons receive input that are later multiplied with weights, w_1 to w_n, that are specifically designed for each unique neuron (Suzuki, 2011). These weights signal how significant each neuron is to the neural network. In the middle layer these weighted inputs are all summed, and a numerical bias, b_i, is added to the total input sum. In the final output layer these weighted sums are fed through an activation function (Suzuki, 2011).

The learning process of a neural network can be broken into two main processes, forward- and backward propagation. Forward propagation is the propagation of information from the input layer to the output layer. We can define our input layer as several neurons x_1 to x_n. Backward propagation is where information is transmitted from the output layer to the input layer. This kind of propagation is the main reason as to why neural networks are highly intelligent and can learn by themselves. In the final step of forward propagation, the neural network quantifies a prediction. In backwards propagation the neural network can assess its own performance and evaluate if the prediction is right or wrong. If the computation is wrong, the neural network utilizes a loss function, which calculates the deviation from the expected output. This information is then back propagated to the hidden layers so the weights and biases can be modified, which will increase the precision of the artificial neural network. The process is also lightly addressed under the Deep learning chapter too.

Figure 6 displays a simple artificial neural network with inputs being fed into the input layer, which is later fed forward to the hidden layer and finally passed through to the output layer.

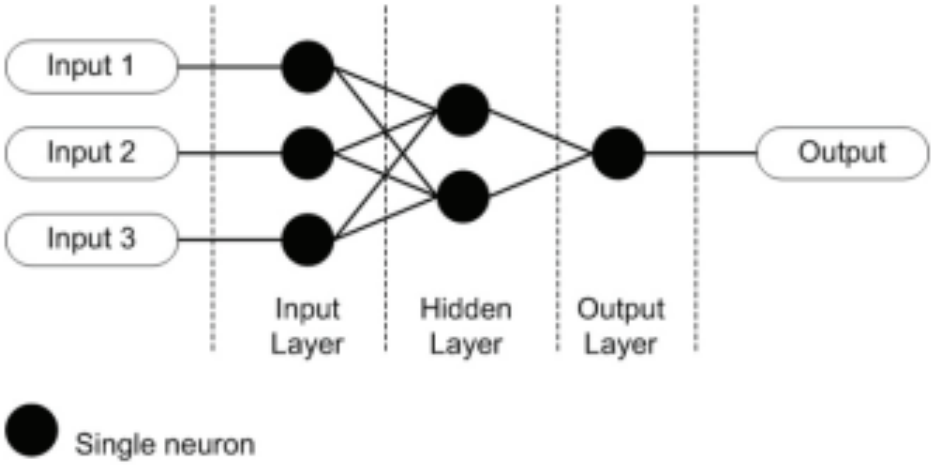


Figure 6. A simple neural network. (Suzuki, 2011)

Neural networks can have immense mathematical complexity due to each connection between each individual artificial neuron (Suzuki, 2011). Connecting each neuron randomly would make the network inconvenient and less efficient, there are therefore several different types of architecture for an artificial neural network that can solve advanced problems more efficient and easier (Suzuki, 2011). Each type of topography is often tailored for different problems. After successfully establishing what kind of problem that needs to be solved, the architectural layout of the network can be determined

and appropriately modified (Suzuki, 2011). An artificial neural network needs to be trained and taught how it could solve the given problem before it is utilized (Suzuki, 2011). Similar to biological neural networks adapting and learning on the inputs from the surrounding environment, artificial neural networks can simulate the same abilities (Suzuki, 2011). As previously mentioned, there are four different paradigms of learning: unsupervised-, supervised-, semi-supervised learning, and reinforced learning.

2.4.1 Artificial neurons

Artificial neurons are the building blocks of artificial neural networks. Figure 7 shows how a biological neuron is structured compared to an artificial neuron. In a biological neuron, the information (input) is fed through the dendrite, where the information is then processed in the soma and later passed on to the axon (Suzuki, 2011).

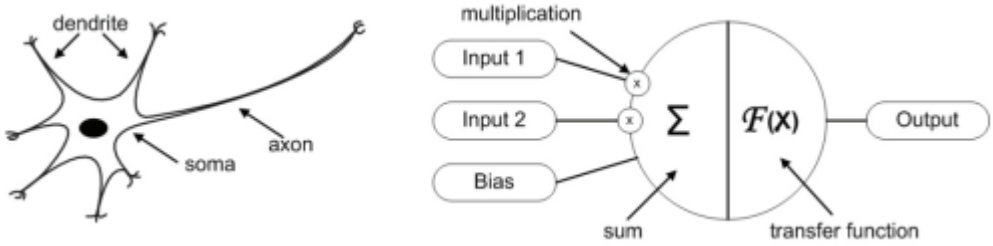


*Figure 7. Biological neuron and an artificial neuron side-by-side (*Suzuki, 2011*)*

As previously mentioned, artificial neurons function by feeding inputs that are weighted where the bias is later added and summed in order to receive an output that is obtained with an activation function. Eq. (6) showcases how the mathematical process is executed:

$$y = F\left(\sum_{i=0}^{m} w_i * x_i + b\right) \quad [6]$$

Where:

- w_i = weights where the value range of *i* is from 0 to m
- x_i is the input value where the value range of *i* is from 0 to m
- b is the bias
- F is the activation function (or also called transfer function)
- y is the output value

Eq. (6) highlights that the essential part of our model that is unknown, which is the activation function. The activation function is vital for the precision of the neural network, along with the number of hidden layers. It determines if the neuron will be activated and ultimately decides if the input the neuron provides is relevant to the neural network, so the role of this function is to obtain an output from a set of given values (Pragati Baheti, 2022). There are multiple different types of activation functions and each has its advantages for different types of problems that need to be solved. There are three main types of activation functions: Linear functions, Non-linear functions and Step functions.

2.4.2 Step functions

Step functions are binary functions that rely on two values that decide if the neuron will be activated or not, these are called threshold values (Pragati Baheti, 2022). The input that is transferred to the activation function is differentiated from the threshold values; if the input value is higher than the threshold value, the neuron is activated, or else it will be deactivated. If it is deactivated, then the output will not be transferred to the next layer in the neural network (Pragati Baheti, 2022). This can be explained mathematically with Eq. (7).

$$y = \begin{cases} 0 \; if \; w_i x_i < threshold \\ 1 \; if \; w_i x_i \geq threshold \end{cases} \quad [7]$$

However, utilizing a binary step function in a neural network comes with some limitations. The process of backward propagation could be troublesome due to the gradient of the step function being zero (Pragati Baheti, 2022). The step function would also be problematic to use for multi-class classification problems because of not having the ability to produce multi-value outputs (Pragati Baheti, 2022).

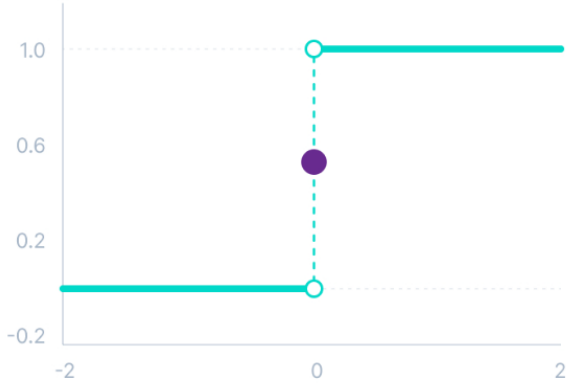


Figure 8. Binary step function (Pragati Baheti, 2022).

2.4.3 Linear function

A linear function is an activation function where the activation value is proportional to the input value, as seen in Figure 9. Mathematically it can simply be presented as:

$$f(x) = x \quad [8]$$

The major problem with having a linear function as an activation function in an artificial neural network is that the derivative of a linear function is always constant and subsequently has no correlation to the x-value, which is the input (Pragati Baheti, 2022). Therefore, the backward propagation process would not be possible as learning through gradient descent would not be possible. The major reason why neural networks can be so intelligent is the possibility of having multiple hidden layers. However, if a linear function is used as the activation function, all the layers would be linear in nature, which would lead to the final output value only being a linear function of the first layer in the neural network (Pragati Baheti, 2022). This means that the major advantage of building an advanced neural network with multiple layers will be lost.

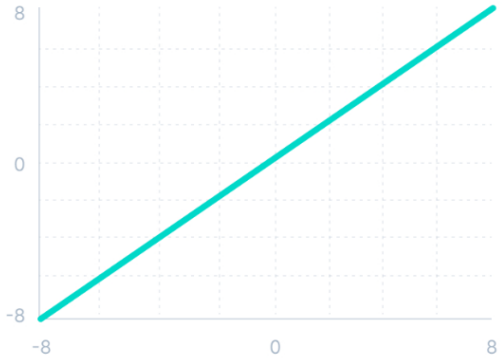


Figure 9. Linear function (Pragati Baheti, 2022).

2.4.4 Non-linear functions

The major problem with step- and linear functions is clearly the inability to utilize backward propagation, which is due to the derivative and gradient. However, with non-linear activation functions this problem is solved. Now that the activation function is non-linear, backpropagation is possible, and the derivative is now interconnected to the input value (Pragati Baheti, 2022). This allows the neural network to combine and stack multiple layers of neurons where the output is now the result of an input being processed in several layers with a non-linear combination (Pragati Baheti, 2022). There are multiple different non-linear functions that can be used, each beneficial for different types of complex problems.

The sigmoid function can use any random given real value as input where the following result will be an output value in the range of $0 - 1$. When the input value increases, it will approach the value 1, whereas if the input value is smaller, the output value will approach 0, as demonstrated in Figure 10.

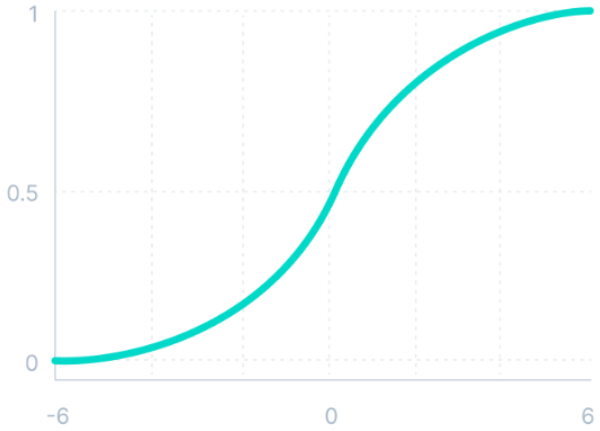


Figure 10. Sigmoid function (Pragati Baheti, 2022).

This function is written mathematically according to Eq. (9):

$$f(x) = \frac{1}{1 + e^{-x}} \quad [9]$$

Then there is the tanh function, which is very similar to the sigmoid function. The larger the value, the close to 1 the output value will be. Similarly, the smaller the value, the closer the output value will be to -1 (Pragati Baheti, 2022).

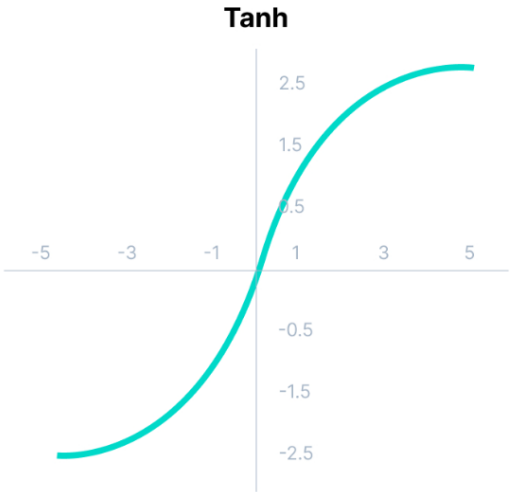


Figure 11. Tanh function (Pragati Baheti, 2022).

The sigmoid- and tanh function both have similar problems when it comes to the derivative. As can be seen from Figure 12, the value of the gradient is only relevant in the interval of -3 to 3 for the sigmoid function. This means that when the input values are greater than three or less than -3, the gradients will become extremely small, which will hinder the network from learning (Pragati Baheti, 2022). The exact same problem, but with slightly different values, occurs for the tanh function.

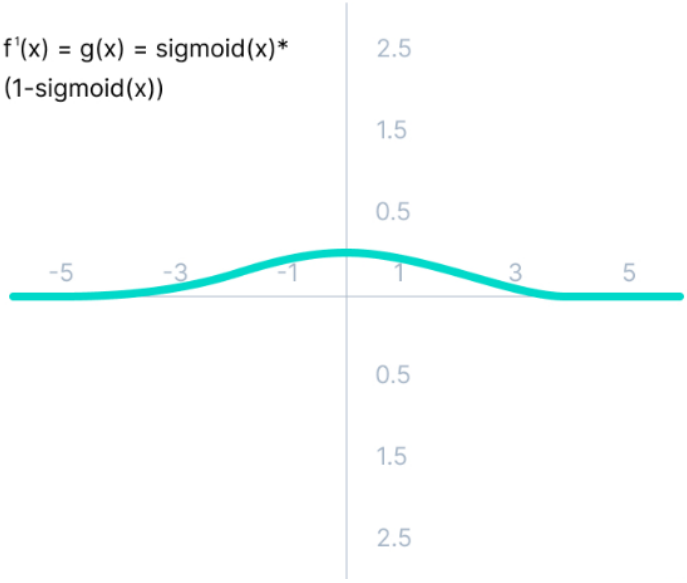


Figure 12. Derivative of sigmoid function (Pragati Baheti, 2022).

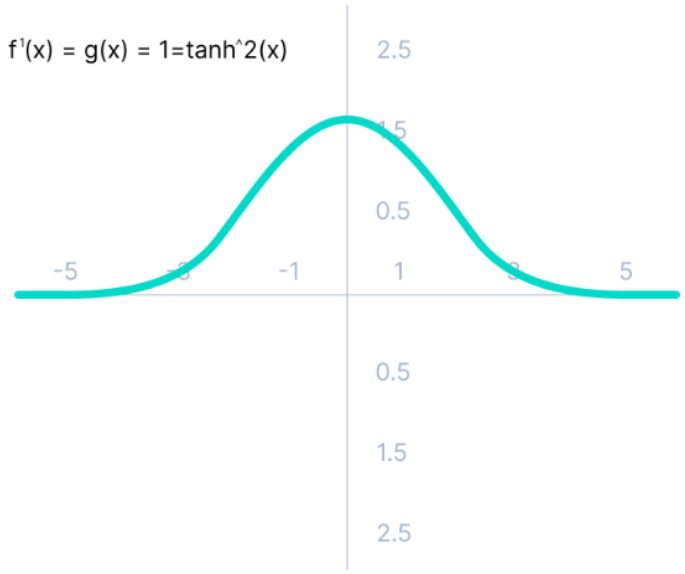


Figure 13. Derivative of tanh function (Pragati Baheti, 2022).

The rectified linear unit (ReLU) function is an activation function that is very commonly used in artificial neural networks. If the input to the function is negative, it will return the value 0. However, if the function receives any value that is positive, denoted x, it will return the same positive value (DanB, 2018). The main advantage of utilizing the ReLU function in a neural network is that this activation function does not activate all neurons during the computing process, which makes it far more efficient compared to other activation functions (DanB, 2018). Unlike a regular linear function, the ReLU function has a derivative and therefore allows for backward propagation (DanB, 2018). Figure 14 gives a graphical representation of the function, and Eq. (10) gives a mathematical representation of the function:

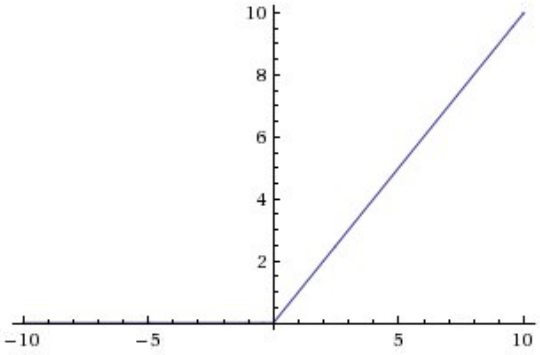


Figure 14. ReLU function (DanB, 2018).

$$f(x) = \max(0, x)\ (x) \qquad [10]$$

2.5 Graph neural networks

Before analyzing what graph neural networks are and how they are utilized, it is necessary to understand what graphs are in computer science. Graphs can be seen everywhere, where objects in the real world are often described with regards to the connection between each other. A collection of objects and their connections can be conveyed as a graph (Benjamin et al., 2021). Graphs are simply a network consisting of nodes that are connected with edges.

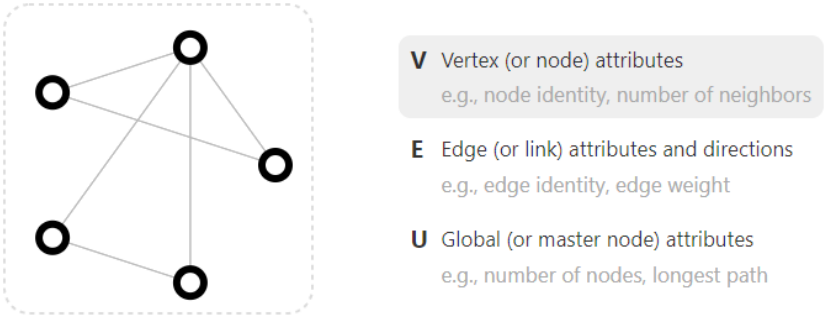


Figure 15. Nodes in a simple graph network (Benjamin et al., 2021).

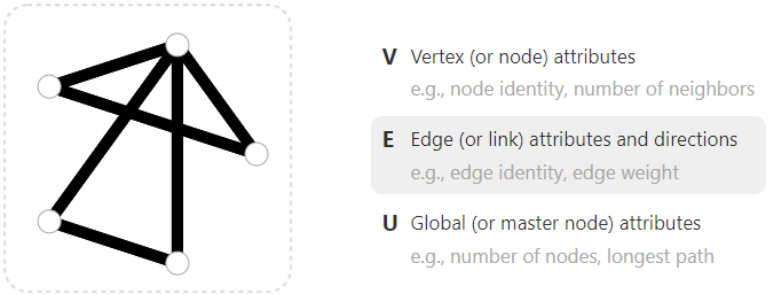


Figure 16. Edges (links) in a simple graph network (Benjamin et al., 2021).

Two nodes connected with an edge are called neighbors, and a sequence of nodes connected by edges in the graph is called a path (Isaac Computer Science, N.D). For many years, researchers have developed artificial neural networks that can operate on data expressed as graphs, otherwise known as graph neural networks (Benjamin et al., 2021). Recently, the advancements and continued development in GNNs have shown a significant increase in their computing capacity and demonstrative power (Benjamin et al., 2021). This progression has led to applying GNNs in several different areas

such as chemistry, physics simulation, natural language processing and more recently, traffic prediction (Benjamin et al., 2021). There are mainly three different variants of graph neural networks in deep learning:

- Recurrent Graph Neural Network
- Spatial-temporal Graph Neural Network
- Graph Convolutional Neural Network

As previously mentioned, this project will focus on a spatial-temporal implementation of a graph convolutional neural network in order to solve the complex dynamic difficulties with traffic networks.

2.5.1 Convolutional neural network

The architecture of a convolutional neural network consists of either one or several blocks of convolutional- and pooling layers followed by a single or multiple fully connected layers and, ultimately an output layer (Ghosh et al., 2020). The convolutional layers are highly central to the convolutional neural network (Ghosh et al., 2020). These layers are made up of multiple learnable convolution kernels that are utilized in order to compute 2D matrices of neurons, otherwise known as feature maps (Ghosh et al., 2020). Every unit of the 2D matrix is connected to a receptive field in the preceding layer (Ghosh et al., 2020). New feature maps are created by convolving the input with kernels and implementing activation functions elementwise that are non-linear in nature (Ghosh et al., 2020).

Convolutional neural networks are a type of artificial neural network that have feed-forward architecture, meaning that the flow of information in the neural network only motions forward (Ghosh et al., 2020). Compared to other types of neural networks, a convolutional neural network has the ability to learn various features of objects, particularly spatial data (Ghosh et al., 2020). The initial layers of the neural network learn and abstract high-level features (such as objects that can be images) while layers located deeper in the network abstract low-level features (such as edges) (Ghosh et al., 2020).

2.5.1.1 Convolutional layer

A convolutional layer is the most vital element of the overall architecture of a convolutional neural network. The layer contains convolutional kernels, which are otherwise known as filters (Ghosh et al., 2020). These get convolved with the image used as input (matrix with N-dimensions) in order to produce a feature map as output (Ghosh et al., 2020).

2.5.1.2 Kernel

A kernel can be represented as a matrix of numbers or discrete values, where every value represents the weight of the kernel (Ghosh et al., 2020). During the training procedure for a convolutional neural network, each weight is assigned a random number in the kernel (Ghosh et al., 2020). The weights are later adjusted for every epoch and the kernel adapts accordingly and finally extracts features that are important (Ghosh et al., 2020).

0	1
-1	2

Table 1. Simple 2x2 kernel

2.5.1.3 Pooling layer

The pooling layers in the network operate by sub-sampling the 2D matrices of neurons (feature maps) (Ghosh et al., 2020). After sub-sampling the feature maps, the layer then conserves the most vital features in every pool step (Ghosh et al., 2020). The pooling process is implemented by identifying the size of the pooled region and the steps of the process subsequently (Ghosh et al., 2020). There are many different pooling techniques, such as min pooling, max pooling, gated pooling and average

pooling (Ghosh et al., 2020). The major disadvantage of pooling layers in convolutional neural networks is that they tend to reduce the computing capacity (Ghosh et al., 2020). The size of the feature map in the output layer can be quantified according to Eq. (11-12):

$$h' = \left[\frac{h - f}{s}\right] \quad [11]$$

$$w' = \left[\frac{w - f}{s}\right] \quad [12]$$

Where:

- h' = feature map height (output)
- w' = feature map width (output)
- h = feature map height (input)
- w = feature map width (input)
- f = region size of pooling
- s = stride of pooling process

2.5.1.4 Fully connected layer

The fully connected layers are located towards the final parts of the neural network and consists of neurons that are connected to other neurons in the foregoing layer (Ghosh et al., 2020). The fully connected layer located towards the end of the network is the classifier and is responsible for predicting the output (Ghosh et al., 2020). Fully connected layers belong to the category of feed-forward artificial neural networks, they operate similarly to conventional multi-layer perceptron neural networks (Ghosh et al., 2020). These layers receive input from the last pooling or convolutional layer in the form of feature maps which are later flattened in order to produce a vector which is then transmitted to the final fully connected layer (Ghosh et al., 2020).

2.5.2 Loss functions

In the output layer of a neural network, it is crucial to quantify the prediction error. This is done by utilizing a loss function. The prediction error communicates with the network and informs it regarding the difference between the prediction and the confirmed output (Ghosh et al., 2020). This information is then used in the learning operation which will improve the accuracy of the artificial neural network (Ghosh et al., 2020). A loss function utilizes two different parameters to quantify the error, the label and the prediction (Ghosh et al., 2020).

2.5.2.1 Cross-entropy (Soft-max)

A cross-entropy loss function is commonly used in deep learning when it comes to convolutional neural networks (Ghosh et al., 2020). The output of this loss function can be described as:

$$p \in \{0, 1\} \quad [13]$$

Where p stands for the probability for every category of output (Ghosh et al., 2020). This kind of loss function takes advantage of softmax activations to produce an output within a probability distribution, such as:

$$p, y \in \mathbb{R}^N \quad [14]$$

Where y is described as the output, the probability for every class of output can then be computed as:

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \quad [15]$$

In Eq. (15), N is described as the quantity of neurons in the output layer whereas e^{a_i} is the unnormalized output from the foregoing layer (Ghosh et al., 2020). Ultimately, the cross-entropy loss can be computed according to Eq. (16-17):

$$H(p, y) = -\sum_{i} y_i \log(p_i) \quad [16]$$

$$i \in [1, N] \quad [17]$$

2.5.2.2 Euclidean

Euclidean loss function, also known as mean squared error, is commonly used in regression analysis (Ghosh et al., 2020). The predicted output can be described mathematically as:

$$p \in \mathbb{R}^N \quad [18]$$

While the actual output in every neuron can be described as:

$$H(p, y) = (p - y)^2 \quad [19]$$

$$where\ y \in \mathbb{R}^N \quad [20]$$

For every neuron, N, the Euclidean loss function in the output layer can finally be written as:

$$H(p, y) = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2 \quad [21]$$

2.5.2.3 Hinge

The hinge loss function is often utilized in binary classification problems (Ghosh et al., 2020). The purpose of this loss function is to amplify the margins between two target classes and can be denoted mathematically as (Ghosh et al., 2020):

$$H(p, y) = \sum_{i=1}^{N} \boldsymbol{max}(0, m - (2y_i - 1)p_i) \quad [22]$$

Where:

- m = margin (usually = 1)
- p_i = predicted output
- y_i = desired output

2.5.3 Training process CNN

The training process of a convolutional neural network consists of four different steps:

- Pre-processing of data
- Initialization of parameters
- Regularization of the CNN
- Selecting optimizer

2.5.3.1 Pre-processing of data

Pre-processing of data consists of artificial modification to the dataset, which includes the dataset for both training and testing (Ghosh et al., 2020). The purpose of data pre-processing is to make it easier

for the neural network to learn and present it in a more consistent format (Ghosh et al., 2020). This step is done before using the data as input for the model. There is a direct connection between the amount of data used and the quality of data with the performance of a neural network. If the network has more data to learn from, the accuracy will significantly increase (Ghosh et al., 2020). However, if the pre-processing data is poor, then the performance of the network will decrease (Ghosh et al., 2020). There are mainly two main ways of doing data-pre-processing; mean-subtraction and normalization. Mean-subtraction consists of subtracting the mean from each feature (data point) in order to make it zero-centered.

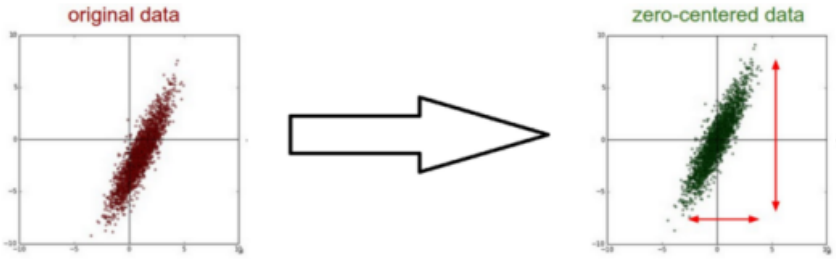


Figure 17. Data after implementing mean-subtraction (Ghosh et al., 2020)

Mathematically this can be described as:

$$X' = x_i - x^* \quad [23]$$

x_i is each individual data point and x^* is the mean

$$x^* = \frac{1}{N}\sum_{i=1}^{N} x_i \ \ and\ here\ N\ is\ the\ size\ of\ the\ training\ dataset \quad [24]$$

Normalization of data consists of normalizing the dimension of the sample data by dividing every dimension with the respective standard deviation (Ghosh et al., 2020). This operation can be described in the following way:

$$X'' = \frac{X'}{\sqrt{\frac{\sum_{i=1}^{N}(x_i - x^*)^2}{N-1}}} \quad [25]$$

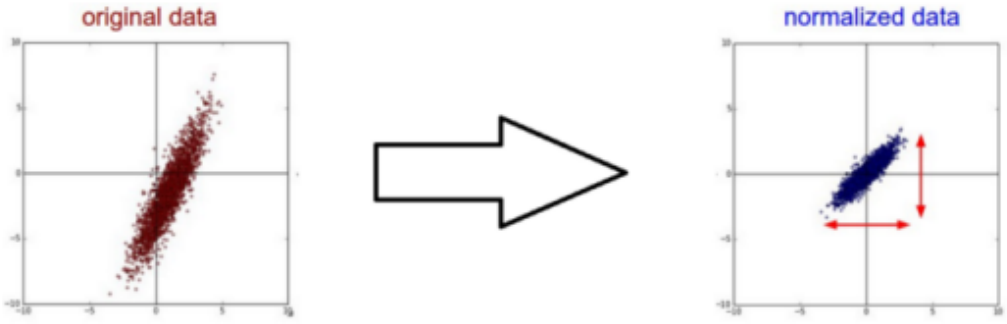


Figure 18. Data after normalizing data (Ghosh et al., 2020)

2.5.3.2 Initialization of parameters

A deep convolutional neural network model can be comprised of several million or even billions of parameters. Therefore, it is vital that the model is initialized properly at the beginning of the training process (Ghosh et al., 2020). The initialization of weights has a major impact on the accuracy of the

network, and there are many different approaches one can take: random initialization, Xavier initialization and unsupervised pre-training initialization.

2.5.3.3 Regularization of the CNN

When it comes to algorithms in deep learning, the main challenge is to correctly adapt to new input data. This ability is called generalization (Ghosh et al., 2020). A high level of accuracy for generalization can be difficult to achieve, a model can be over-, under-, or well-fitted (Ghosh et al., 2020). When a model is overfitted, it performs extraordinarily well on training data but has very low performance on test data (Ghosh et al., 2020). An under-fitted model is when the model has poor performance on the training data and fails to generalize new data (Ghosh et al., 2020). However, a well-fitted model performs well on both the training- and the testing data, which is the goal for all deep learning algorithms.

Figure 19. Different types of fitted models (Hoffman, 2021)

There are many different techniques for regularization of a neural network: dropout, i^2 regularization, i regularization, batch normalization and layer normalization *(Ghosh et al., 2020).*

2.5.3.4 Optimizer selection

The fourth and final step of the training process consists of selecting the optimizer for the neural network. The main purpose of optimizers is to minimize the error (loss function), meaning the difference between the predicted output and the correct output (Ghosh et al., 2020). There are multiple different optimizers that one can choose to implement:

- Gradient descent
    - Batch gradient descent
    - Mini batch gradient descent
    - Stochastic gradient descent
- Momentum
- AdaGrad
- AdaDelta
- RMSProp
- Adaptive Moment Estimation (Adam)

Gradient descent works by updating the parameters of the model continuously during every training epoch as a means to minimize the training error (Ghosh et al., 2020). The algorithm calculates the slope of the objective function with the first-order derivative relative to the parameters of the model and in order to decrease the predictive error of the algorithm, and updates the parameters in the opposite direction of the gradient (slope). The parameters are updated during back-propagation by back-propagating the gradient next to each neuron backward for every layer (Ghosh et al., 2020). Gradient descent can be described with the following equations:

$$w_{ij}^{t} = w_{ij}^{t-1} - \Delta w_{ij}^{t} \quad [26]$$

$$\Delta w_{ij}^{t} = \eta * \frac{\partial E}{\partial w_{ij}} \quad [27]$$

Where:

- w_{ij}^{t} = final weight in the t^{th} training epoch
- w_{ij}^{t-1} = weight in the previous training epoch
- η = learning rate
- E = prediction error

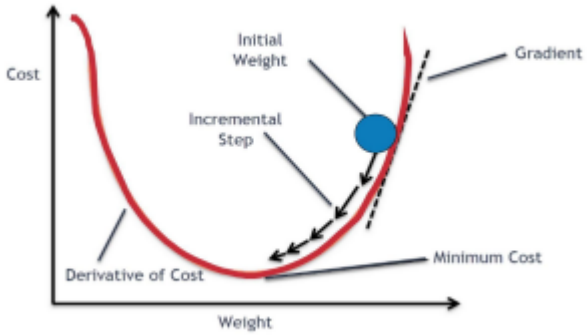


Figure 20. How gradient descent algorithm operates (Ghosh et al., 2020)

RMSProp stands for Root Mean Square Propagation and builds on gradient descent (Sanghvi, 2021). This algorithm is developed in order to accelerate the optimization process and functions by utilizing a moving average of squared gradients which subsequently normalizes the gradients (Sanghvi, 2021). The mathematical operation of the RMSProp can be described with the following equations:

$$w_{ij}^{t} = w_{ij}^{t-1} - \frac{\eta}{\sqrt{E[\delta^2]^t}} * \delta_{ij}^{t} \quad [28]$$

$$E[\delta^2]^t = \gamma E[\delta^2]^{t-1} + (1-\gamma)(\delta^t)^2 \quad [29]$$

Where:

- δ_{ij}^{t} = local gradient of parameter w_{ij} in the t^{th} epoch
- γ = suggested 0.9

3. Methodology

3.1 Methodological framework

The framework of the thesis was divided into four different steps in Figure 21. The first step is to analyze and organize the data given for the case study. The following step is to design the graph neural network, which consists of multiple different levels of layers. There are many different programming languages to choose from when it comes to algorithms and building neural networks. Python is an advanced programming language, straightforward and relatively easy to learn. The programming language offers code that is readable and concise (Beklemysheva, n.d.). Complex algorithms combined with workflows that are versatile are the cornerstone of artificial intelligence and machine learning (Beklemysheva, n.d.). The simplicity of python allows us to design a reliable system with its comprehensive selection of available libraries. The fourth and final step of the methodology was to define the structure of the code.

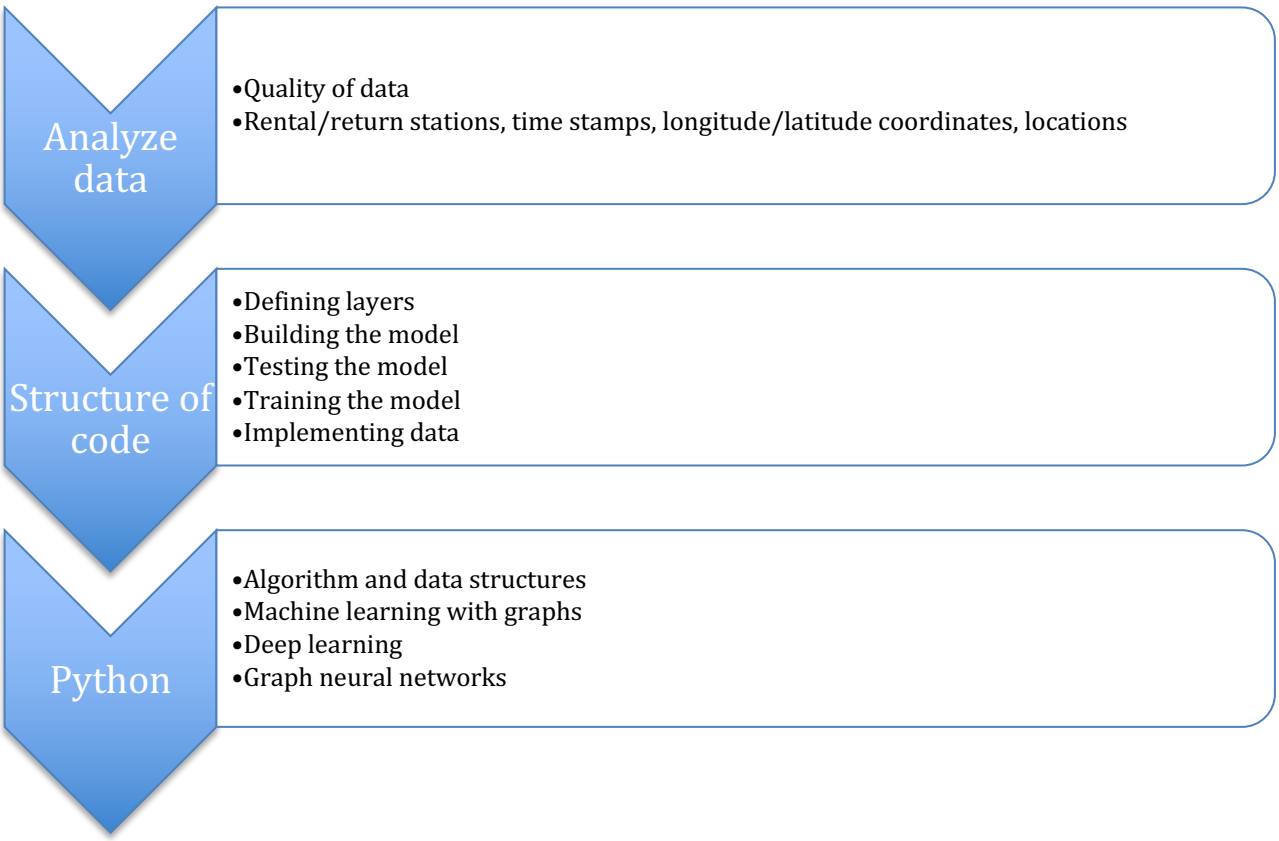


Figure 21. Methodological framework.

3.1.1 Analyzing data

The data provided is public bike data operated by a governmental department. It contains several different types of data:

- Rental- and return station ID
- Rental- and return time
- Longitudinal/Latitudinal coordinates of rental stations
- Maximum capacity of each rental station

The data includes the final three months of 2016, October, November, and December. The month of October had a total of 3 051 105 rental- and return station events (trips), November had 3 681 918 and December had 3 781 112 trips if all stations are taken into consideration. The total amount of rental stations exceed 800. Analyzing every trip to all stations would take a substantial amount of time and would also require immense computing power. Therefore, it is appropriate to perform the analysis with a set number of stations. All data was originally in China and had to be preprocessed in order to make it applicable to implement in python using the library pandas.

3.1.2 Network structure

The architectural layout of the neural network is quite complex and will be elaborated in this section step-by-step. The spatial-temporal graph convolutional neural network is comprised of spatial-temporal convolutional blocks, where every block has two succeeding gated convolutional layers with a spatial graph convolutional layer between every sequence of gated layers (Yu et al., 2018).

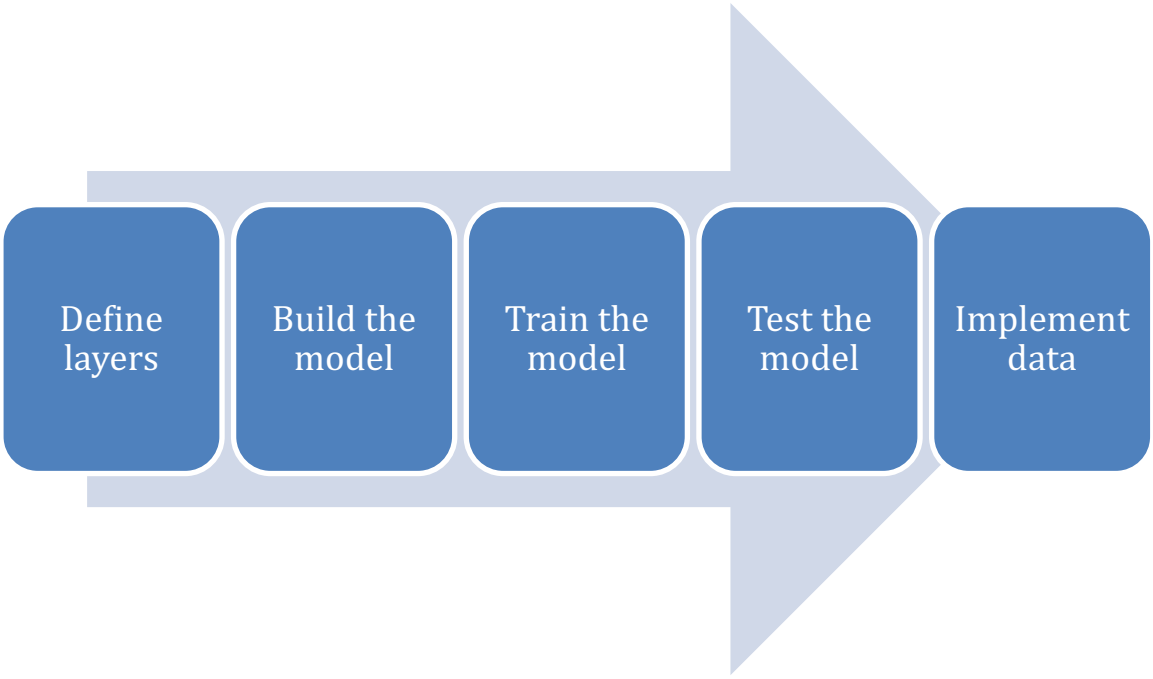


Figure 22. Structure of code

As previously mentioned, traffic networks can naturally be represented with graph structures. Therefore, it is appropriate to design networks mathematically as graphs. Many models have ignored the spatial attributes in road networks where the connectivity is often disregarded (Yu et al., 2018). This can mainly be because of the separation of traffic networks into several different grids. This model will implement graph convolution directly on the graph-organized data in order to accurately analyze important patterns and features in the space realm (Yu et al., 2018). First, the concept of a graph convolution operator needs to be instituted. This mathematical operator is denoted $*_G$ and is established on the idea of spectral graph convolution (Yu et al., 2018).

$$\Theta *_G x = \Theta(\mathrm{L})\mathrm{x} = \Theta\left(\mathrm{U}\Lambda\mathrm{U}^{\mathrm{T}}\right)x = U\Theta(\Lambda)\mathrm{U}^{\mathrm{T}}x \qquad [30]$$

Where:

- $x \in \mathbb{R}^n$ is a signal with the kernel Θ
- $U \in \mathbb{R}^n$ denotes the matrix of eigenvectors of the normalized graph Laplacian, L
  - $L = I_n - D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = U\Lambda U^T \in \mathbb{R}^{n\,x\,n}$ [31]
    - I_n is the identity matrix
    - $D \in \mathbb{R}^{n\,x\,n}$ is the diagonal degree matrix and $D_{ii} = \sum_j W_{ij}$, where W_{ij} is the weighted adjacency matrix with for the i^{th} row and j^{th} column
- $\Lambda \in \mathbb{R}^{n\,x\,n}$ is the diagonal matrix of eigenvalues of L
- $\Theta(\Lambda)$ is the filter with a diagonal matrix (Yu et al., 2018).

There are two different approaches that can be selected in terms of approximation procedure, the Chebyshev Polynomials Approximation and the 1^{st} order Approximation. This approach will utilize the former. Here, the kernel, denoted Θ, will be confined to a polynomial of Λ according to the following expression

$$\Theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^{\mathrm{k}} \qquad [32]$$

Where $\theta_k \in \mathbb{R}^k$ is a vector comprised of coefficients that are polynomial in nature (Yu et al., 2018). K denotes the size of the kernel of the graph convolution, which subsequently decides the max radius of the convolution measured from the nodes located centrally (Yu et al., 2018). The Chebyshev Polynomial, denoted $T_k(x)$, is usually utilized in order to estimate kernels and can be described accordingly (Yu et al., 2018):

$$\Theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\Lambda}) \quad [33]$$

Where Eq. (34) is the rescaled diagonal matrix.

$$\widetilde{\Lambda} = \frac{2\Lambda}{\lambda_{max} - I_n} \quad [34]$$

And λ_{max} is simply the maximum eigenvalue of L. Ultimately, the graph convolution can be simplified according to the following equation:

$$\Theta *_G x = \Theta(\mathrm{L})\mathrm{x} \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x \quad [35]$$

Here, $T_k(\tilde{L})$ denotes the Chebyshev polynomial of order k and is computed through

$$\tilde{L} = \frac{2L}{\lambda_{max} - I_n} \quad [36]$$

By repeatedly calculating the K-confined convolutions through the polynomial estimation, the computing cost of Eq. (35) can be lowered (Yu et al., 2018). The previously mentioned mathematical operator, $*_G$, for graph convolution for $x \in \mathbb{R}^n$ can be expanded to tensors of multiple dimensions (Yu et al., 2018). For a signal x with C_i channels where $x \in \mathbb{R}^{n\ x\ C_i}$, we can generalize the graph convolution with Eq. (37) (Yu et al., 2018)

$$y_j = \sum_{i=1}^{C_i} \Theta_{i,j}(L)x_i \in \mathbb{R}^n, 1 \leq j \leq C_o \quad [37]$$

Where

- C_i is the size of the input feature maps
- C_o is the size of the output feature maps

For two-dimensional variables, the graph convolution can be expressed as $\Theta *_G X$ for $\Theta \in \mathbb{R}^{K\ x\ C_i\ x\ C_o}$ (Yu et al., 2018). When it comes to traffic prediction, the input is comprised of frames denoted v_t, where each frame is a road graph (Yu et al., 2018). These frames can be viewed as matrices where column i outlines the C_i-dimensional value of v_t in the graph G_t at the i^{th} node (Yu et al., 2018). The graph G_t for time step t can be denoted as $G_t = (V_t, \mathcal{E}, W)$, where V_t stands for a fixed amount of vertices and are related to the number of observations coming from n monitor stations and $W \in \mathbb{R}^n$ symbolize the weighted adjacency matrix of G_t (Yu et al., 2018). Figure 23. visualizes how this process works, where H is the time steps, M is the amount of traffic observations and $v_t \in \mathbb{R}^n$ is the vector for n road elements at time t (Yu et al., 2018).

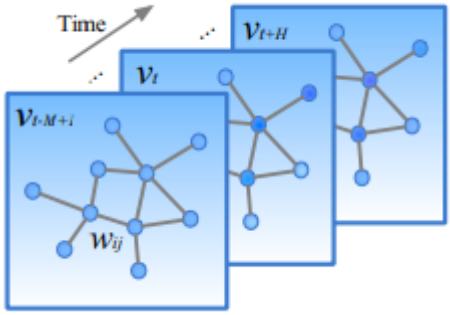


Figure 23. Visualization of graphs (Yu et al., 2018)

While graphs are employed in order to analyze spatial features, gates will be used in order to analyze temporal features. By taking inspiration from (Yu et al., 2018), this model will apply convolutional formations on time elements in order to analyze dynamic temporal patterns in traffic demand (Yu et al., 2018). This architectural layout has a training procedure that can be controlled with several layers of convolutional structures that are hierarchically organized (Yu et al., 2018). The temporal convolutional layer includes a one-dimensional convolution where its width is denoted K_t (temporal kernel) and is succeeded by gated linear units (Yu et al., 2018).

The temporal convolution analyzes each input for neighbors K_t for every node in graph G (Yu et al., 2018). Therefore, the input of temporal convolution for every node in the graphs can be described as the length M with C_i channels with input Y where $Y \in \mathbb{R}^{M \times C_i}$ (Yu et al., 2018). After designing a convolution kernel, $\Gamma \in \mathbb{R}^{K_t \times C_i \times 2C_0}$, the input Y can be transformed to a unique output $[P\ Q] \in \mathbb{R}^{(M-K_t+1)*(2C_0)}$ and the temporal gated convolution can be expressed mathematically with the following equation:

$$\Gamma *_\tau Y = P \odot \sigma(Q) \in \mathbb{R}^{(M-K_t+1) \times C_0} \quad [38]$$

Where both P and Q are the input for the gates, $\odot$ is the Hadamard product performed element-wise (Yu et al., 2018). $\sigma(Q)$ denotes the sigmoid gate and the purpose of this gate is to manage which input P is of importance in order to discover dynamic variances in the time series (Yu et al., 2018).

In order to combine the features of both domains (spatial and temporal) we design spatial-temporal convolutional blocks that have the ability to operate graph-structured time series. There are two temporal layers with a spatial layer in between where layer normalization will be implemented as a precaution for overfitting the neural network. Regarding the data preprocessing, the adjacency matrix is calculated by taking the distance between the bike stations into consideration. The matrix, denoted W, can then be formulated as follows:

$$w_{ij} = \begin{cases} e^{\left(-\frac{d_{ij}^2}{\sigma^2}\right)}, i \neq j \text{ and } e^{\left(-\frac{d_{ij}^2}{\sigma^2}\right)} \geq \epsilon \\ 0 \end{cases} \quad [39]$$

w_{ij} represents the weight of every edge, and this is computed by d_{ij}, which is the distance between station i and station j (Yu et al., 2018). ϵ and σ^2 are simply values that limit the sparsity and distribution of the matrix W

3.1.3 Python

Coding in python consisted of defining the layers, building the base model, training the model and finally testing the model. Defining the layers included designing the graph convolution, layer normalization, temporal convolutional layer, spatial-temporal convolutional layer, spatial-temporal

convolutional block, fully connected layer and an output layer at the end. Afterward, the base model is built containing the kernel size of both the spatial- and temporal convolution and channel configurations of the spatial-temporal convolutional block. The trainer consists of training the base model where the model loss is defined, along with learning rate settings, learning rate decay and optimizers. Finally, the tester involves defining a multi-prediction function which includes the number of frames and routes, the size of batches, and the length of the predictions

3.2 Study Area

For this study, a dataset from the city of Nanjing, China will be analyzed. Wu (2005)mentioned that there is not possible to develop necessary urban structures due to the high density in urban areas. Due to this, urban transportation systems that worked well a decade ago are not functional today. Therefore, this study will focus on bike-sharing transportation. Bike-sharing has become more popular in big cities where bikes or public transport are often faster and more efficient compared to cars for shorter distances. Looking at year 2016, Nanjing's bike-sharing network contains more than 800 stations, and each station has its own unique station ID, and each ID belongs to one of the six zones in Nanjing. Figure 24 shows a color-based map where each color represents one of the six zones. All the zones are explained in Table 1. The first column is the zone, the second is the color, and the third is the first three numbers in the station ID. A station ID contains a total of 6 numbers, but only the first three determine the zone. However, the analysis will not include all 800+ stations since the calculation time will be too long and the computer requires a lot of computer power. Therefore, the study has aimed to evaluate randomly 50, 100 and 200 stations across 800 stations.

Table 2. Zone descriptions.

Zone	Color	First three numbers
1	Blue	011
2	Red	012
3	Green	013
4	Yellow	014
5	Pink	015
6	Black	016

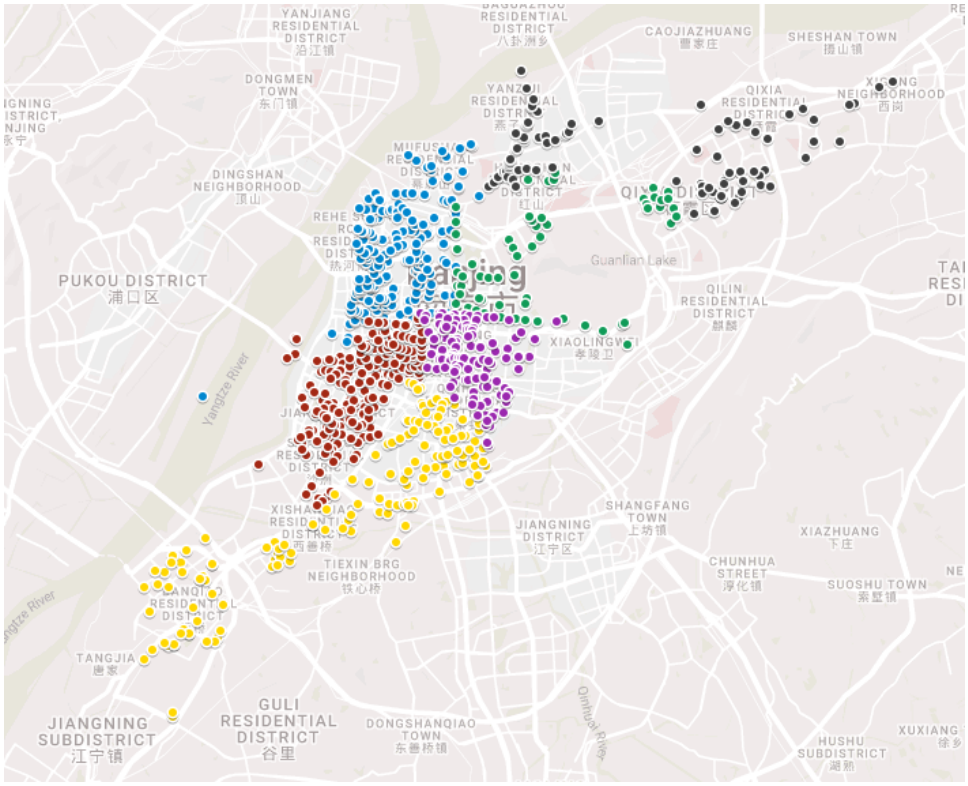


Figure 24. Bike-sharing stations in Nanjing for all 800+ from Google my maps.

4. Experiments and Results

Nanjing has over 800 bike-sharing stations. However, this analysis will only evaluate 50, 100, and 200 stations in order to see how the results vary with the number of stations and also due to time constraints. The neural network will use the data for October and November to train the model and then test the model in order to compute the travel demand for the month of December. The analysis will present the actual demand for 10 stations each (25 stations for the 200-station analysis). More importantly, a comparison between the actual demand and the predicted demand by the proposed graph neural network is conducted to highlight the difference. To improve the accuracy of the neural network, two different hyperparameters will be modified. The first parameter is the spatial-temporal convolutional blocks. There are two blocks in the graph neural network model. The base case for both blocks will be the following:

$$[1, 32, 64] \; and \; [64, 32, 128]$$

This model is relatively small and having too many channels in the base layout would cost too much in terms of computation and time. Therefore, it is appropriate to have the initial layout with 32 channels and increase the number of channels with experimental layouts in order to investigate how the efficiency fluctuates. Figure 25 highlights the structure of a spatial-temporal convolutional block. As previously mentioned, every spatial-temporal convolutional block contains two gated temporal convolutions with a spatial graph convolution layer in between. The first block has one channel as input, 32 channels in the middle and 64 channels as output. The second block will subsequently have 64 channels as input, 32 channels in the middle, and 128 channels as output. These will be modified in order to see how the metrics MAE and RMSE can be decreased.

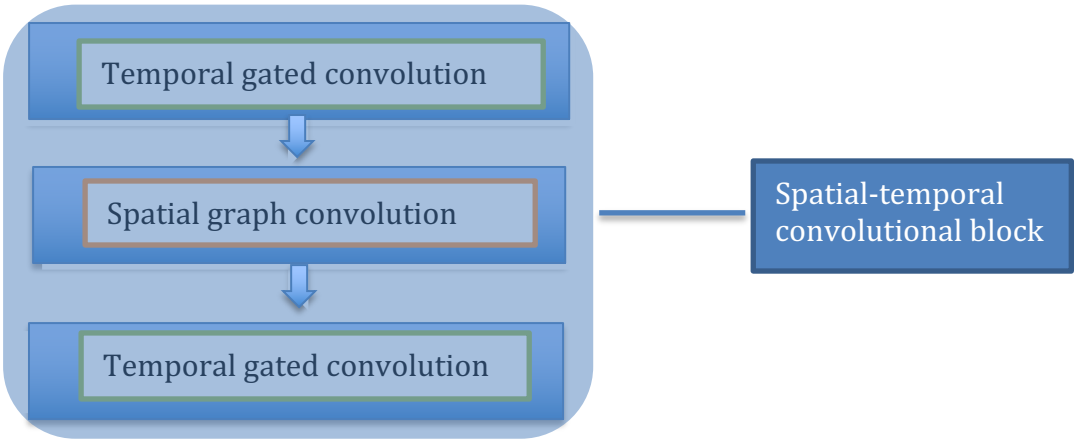


Figure 25. Structure of spatial-temporal convolutional block

The first experimental block layout will be:

$$[1, 64, 128] \; and \; [128, 64, 256]$$

and the second experimental block layout will be:

$$[1, 128, 256] \; and \; [256, 128, 512]$$

The second hyperparameter is the learning rate during training. This hyperparameter is responsible for updating the weights in the model in the backpropagation during the training procedure and is typically defined in the range of $0 - 1$ (Suzuki, 2011). Instead of constantly changing weights, the learning rate is multiplied by the weights in order to reduce error estimates (Suzuki, 2011). This initial learning rate that is used for the base cases is 0.001. The experimental learning rates that will be tested are 0.005 and 0.01. The number of epochs will be set to 30 and the number of batches will be set to 10, both remaining constant throughout the analysis. The results showing the MAE and RMSE for the validation- and testing set highlight that for most cases 15-20 epochs would be more than enough,

however 30 epochs will be implemented to see if the model overfits and if there are noticeable changes in the error metrics. The algorithm will use 12 hours of traffic to compute the next 3 hours of traffic, according to the following schematic shown in Figure 26:

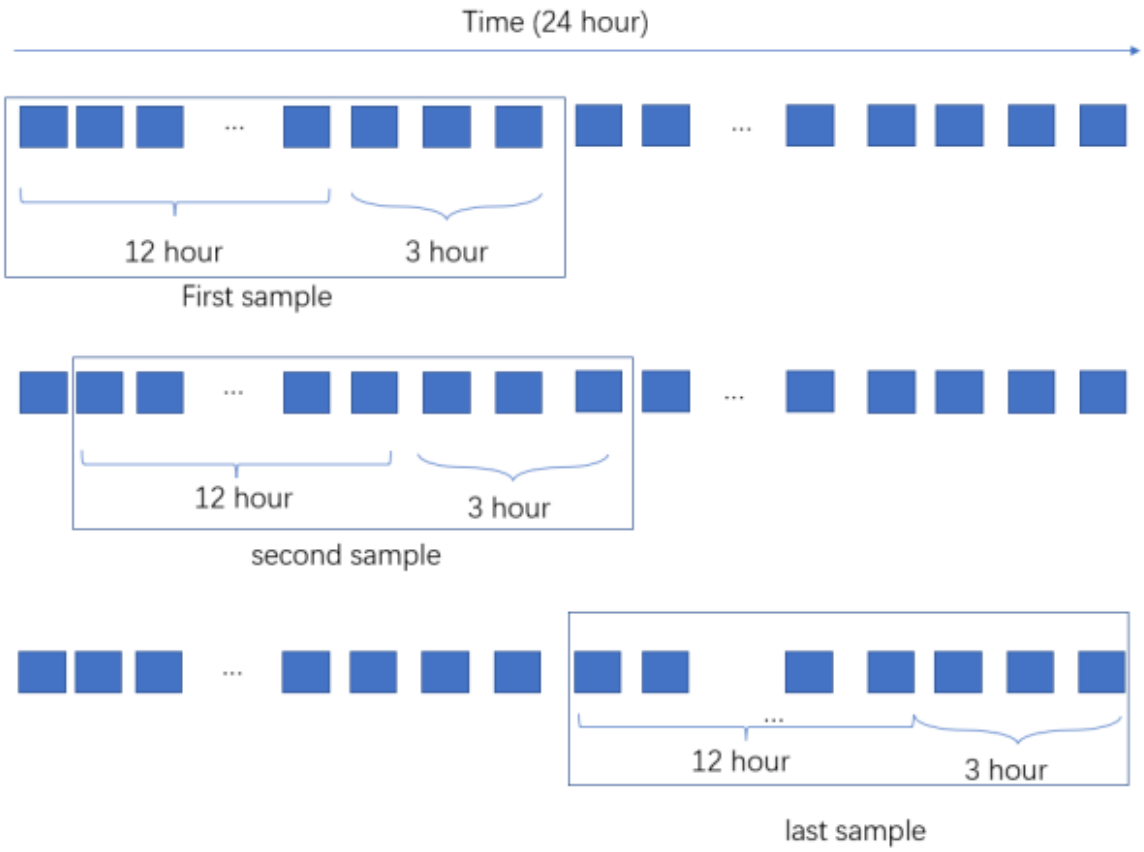


Figure 26. Schematic of demand calculation

Figure 26 shows the actual demand will be the hourly demand, while the comparison for the actual- and predicted demand will be for the monthly demand in order to better highlight the magnitude of difference in a more visible way. The MAE and RMSE are computed according to the following equations:

$$MAE = \frac{\sum_{n=1}^{N} |\hat{r_n} - r_n|}{N} \quad [40]$$

$$RMSE = \sqrt{\sum_{n=1}^{N} (\hat{r_n} - r_n)^2} \quad [41]$$

Where:

- $\hat{r_n}$= prediction rating.
- r_n = true rating in test data.
- N = Sample size

4.1 Analysis results- 50 stations

4.1.1 Station 1-10

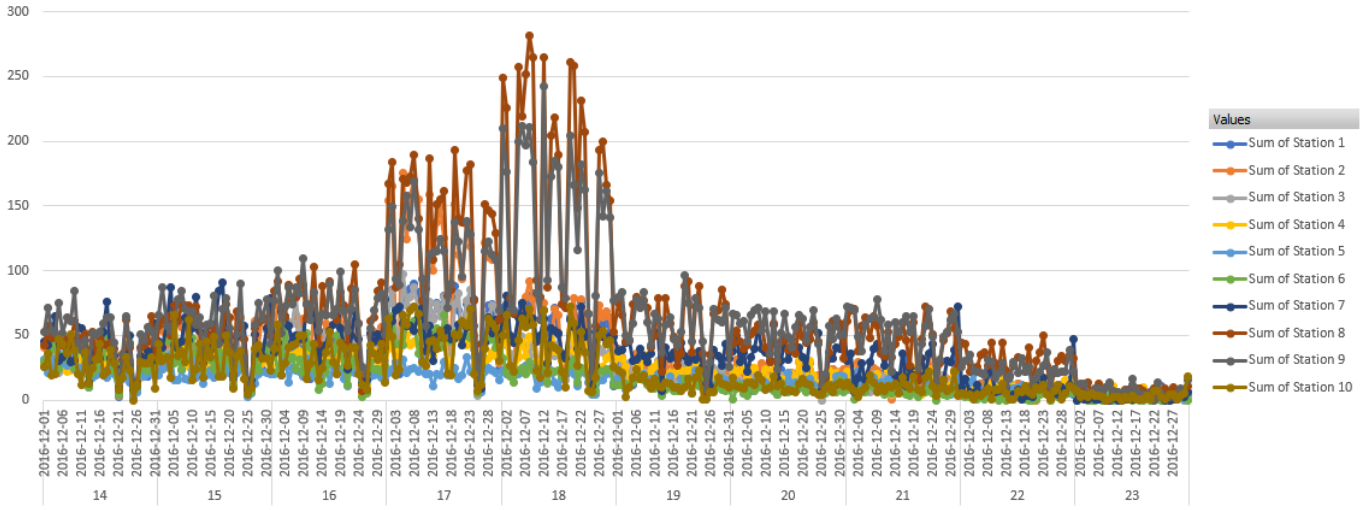


Figure 27. Actual hourly demand for stations 1-10

Figure 27 shows the actual hourly demand for stations 1-10 starting from 14:00. The graph shows that the bike demand is mostly between 25-100 between 14:00 – 16:00 and then significantly increases during the end of the working hours, which are normally between 17:00-19:00. Towards the end of the day after 19:00, the graph highlights that the traffic demand for bikes steadily decreases and later shows a significant decrease after 21:00.

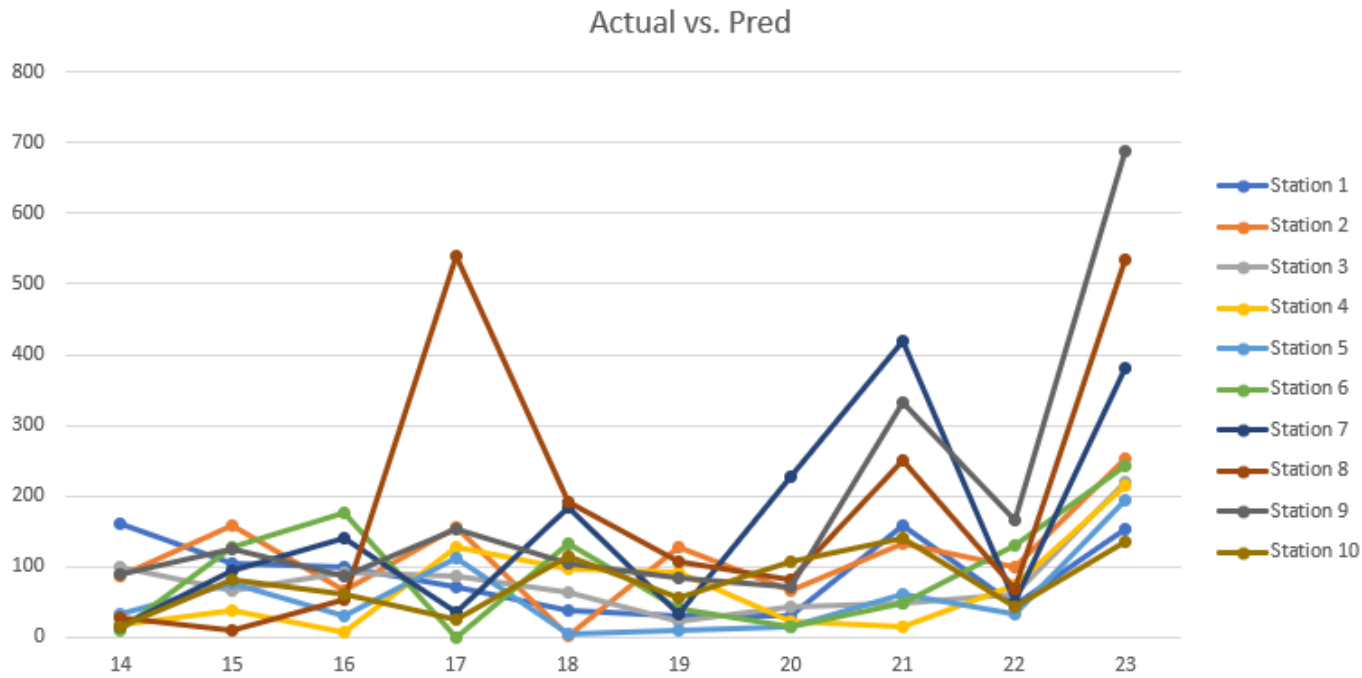


Figure 28. Difference between actual monthly demand and predicted demand

The plot in Figure 28 showcases the difference between the actual demand and the predicted demand computed by the developed GCN algorithm. The aim is to investigate how accurately the developed algorithm can predict the actual travel demand of using bike-sharing in each station at the resolution of one hour. For every station analysis, there will be graphs displaying the predictive errors for the hourly demand.

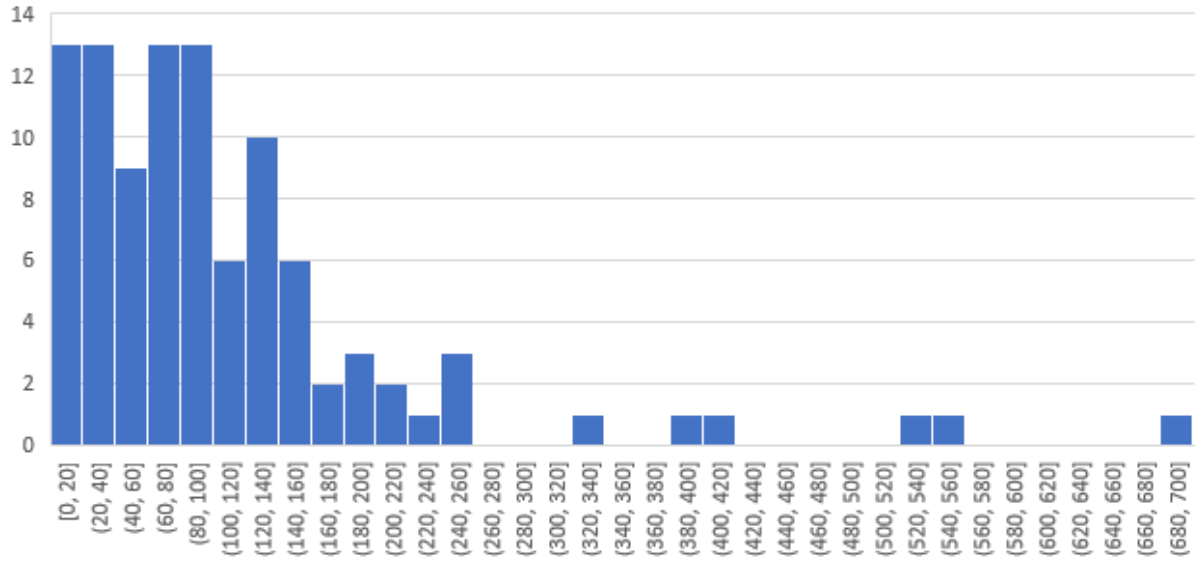


Figure 29. Predicted demand errors for stations 1-10

Figure 29. shows the accuracy of the hourly prediction with a 20-bike interval for stations 1-10. It can be seen that the majority of the predictions have less than 100 bike difference compared to the real demand. For example, 13 predictions have between 0-20 differences when comparing the prediction computed by the neural network and the real demand.

4.1.2 Station 11-20

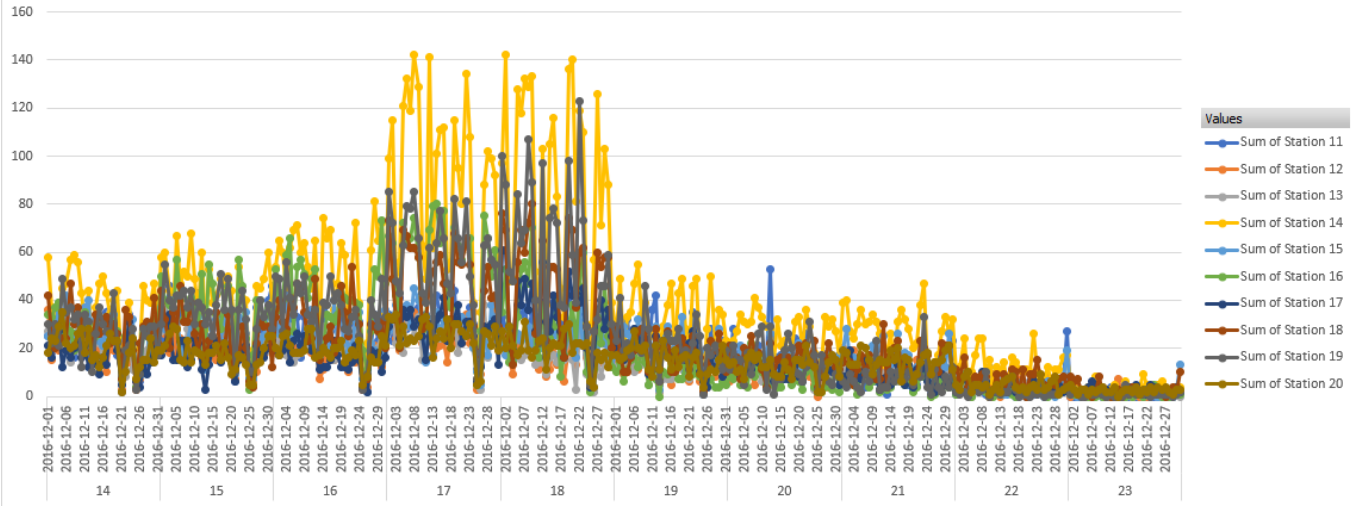


Figure 30. Actual hourly demand for station 11-20

The actual bike demand for station 11-20 exhibits similar patterns compared to station 1-10. The demand is between 20-60 for the majority of stations between 14:00-16:00 and then remarkably increases for hour 17:00-18:00 with a peak of around 140 bikes for station 14. Towards the end of the day the demand of bikes steadily decreases between 19:00-21:00 and continues to decrease after 21:00 to the end of the day.

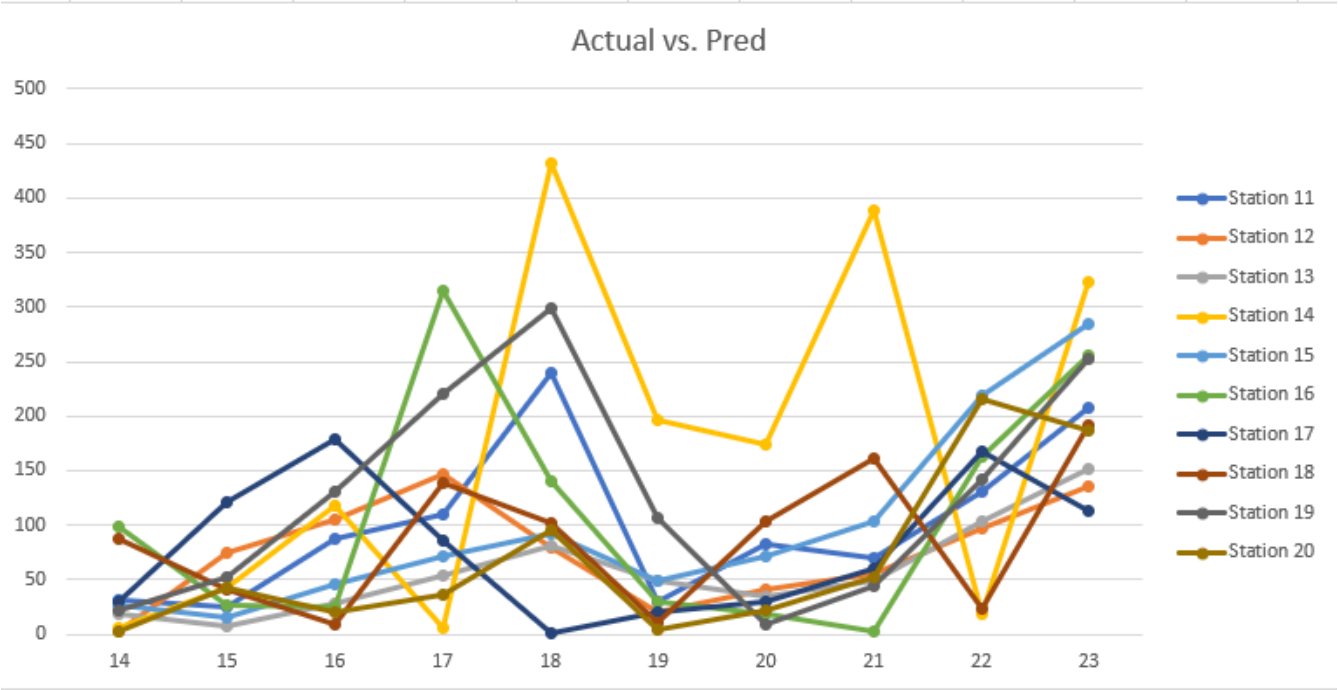


Figure 31. Difference between actual monthly demand and predicted demand

Figure 31 shows the difference between actual bike demand and predicted bike demand computed by the neural network. It can be seen that the majority of data points are below 150 where many of the data points over seem to be for the final hour of the day. It is clear that the neural network struggles to provide an accurate prediction for that final hour (23:00). Figure 32 shows that the majority of the predictions by the algorithm have a difference between 1-21 and 21-41 compared to the real demand.

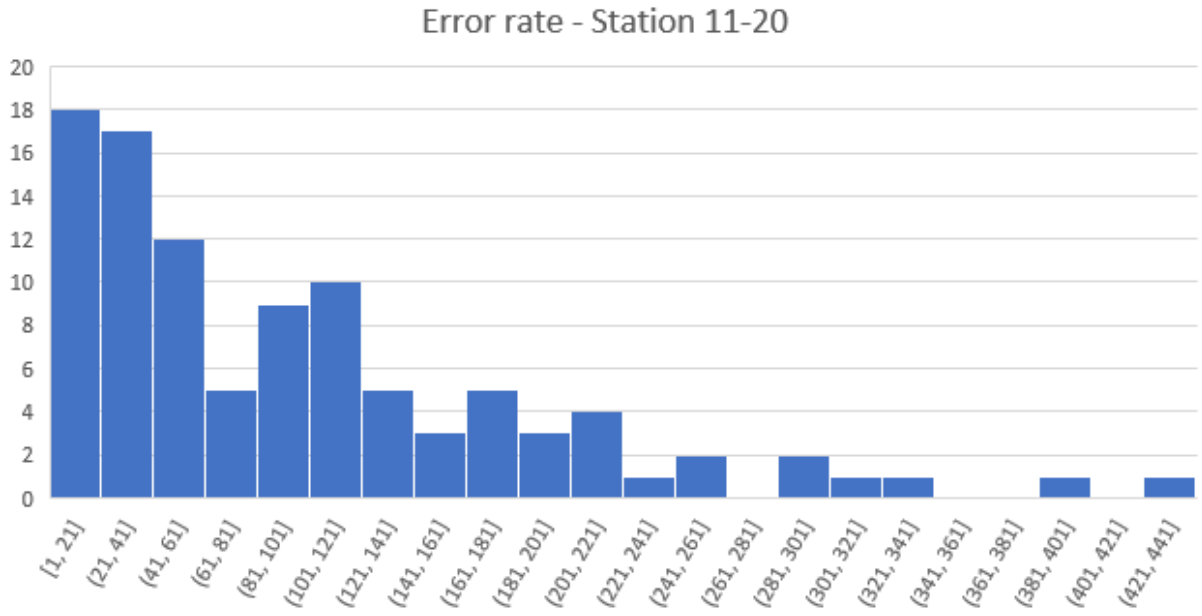


Figure 32. Predicted demand errors for stations 11-20

4.1.3 Station 21-30

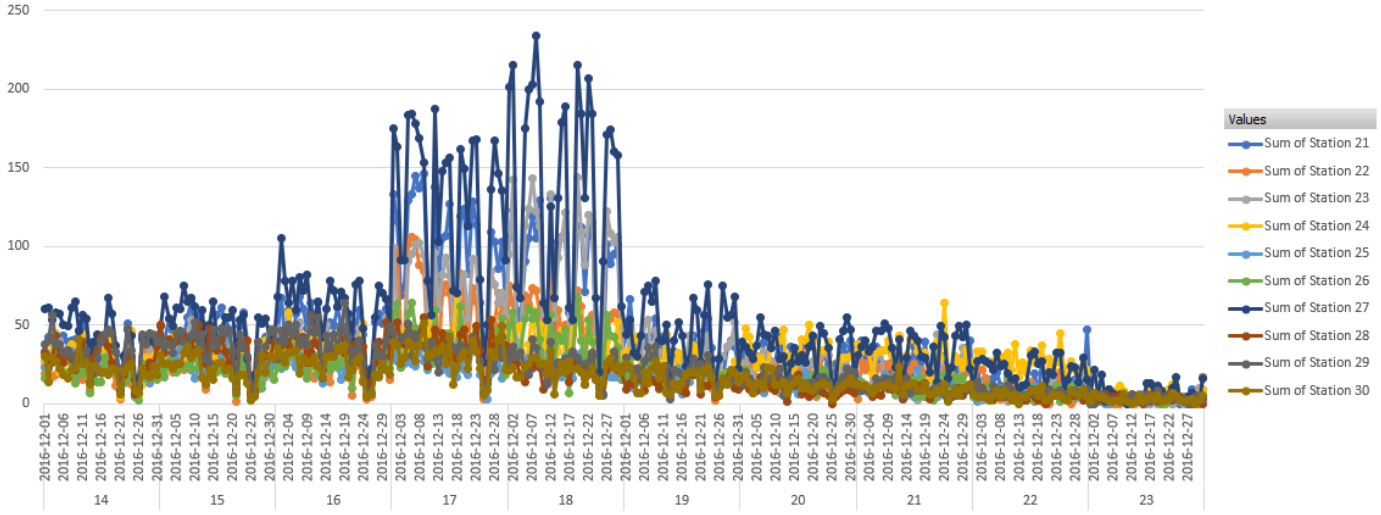


Figure 33. Actual hourly demand for station 21-30

The bike demand for station 21-30 shows almost identical patterns to station 11-20, where the main difference here is a higher peak of bike demand between 17:00-18:00 reaching between 200-240 bikes. After 19:00 the bike demand falls under 100 bikes/hour and continues to decrease after 21:00.

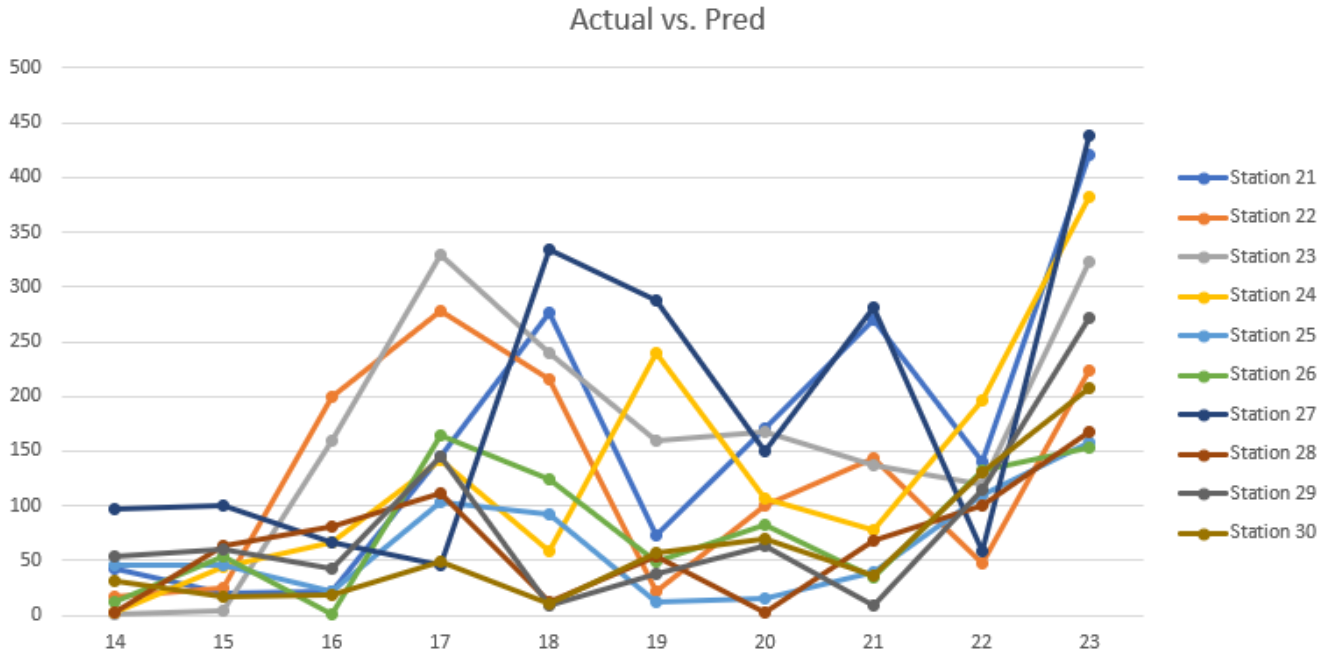


Figure 34. Difference between actual monthly demand and predicted demand

For the actual vs. predicted demand for station 21-30 the data points are very spread out, where the most accurate predictions seem to be around 14:00-15:00 and the lowest accuracy seem to be around 17:00-19:00 and again for the final hour around 23:00. Figure 35. highlights that a notable decrease in the amount of predictions ranging from 21-41 compared to previous station analysis could be seen but overall follows the same pattern.

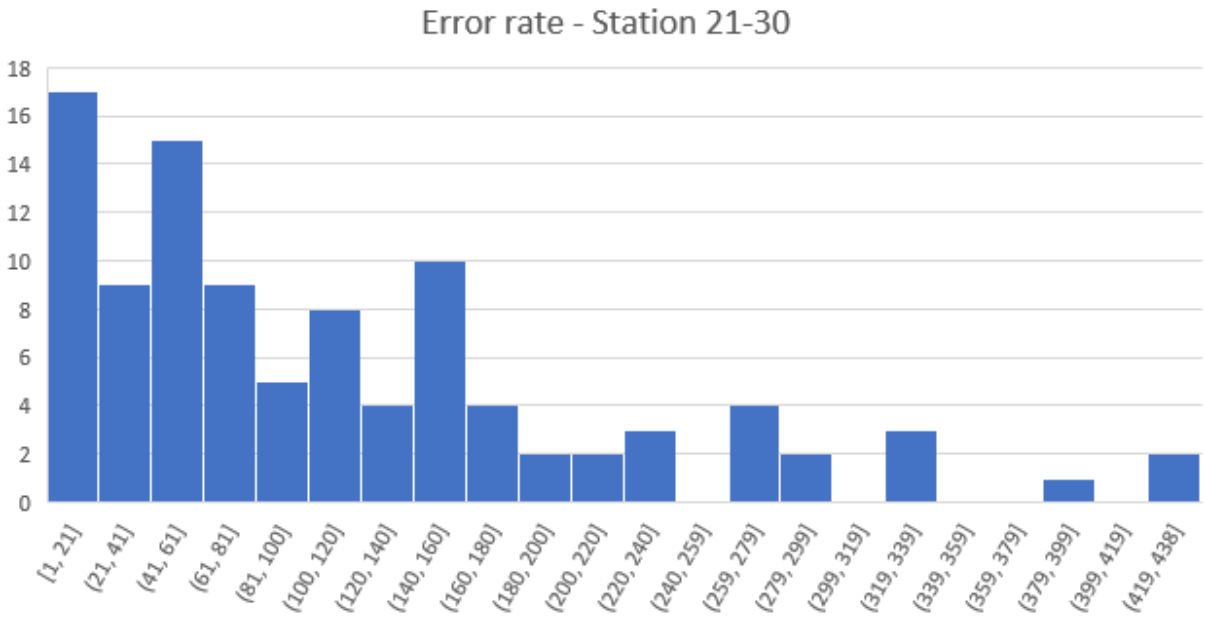


Figure 35. Predicted demand errors for stations 21-30

4.1.4 Station 31-40

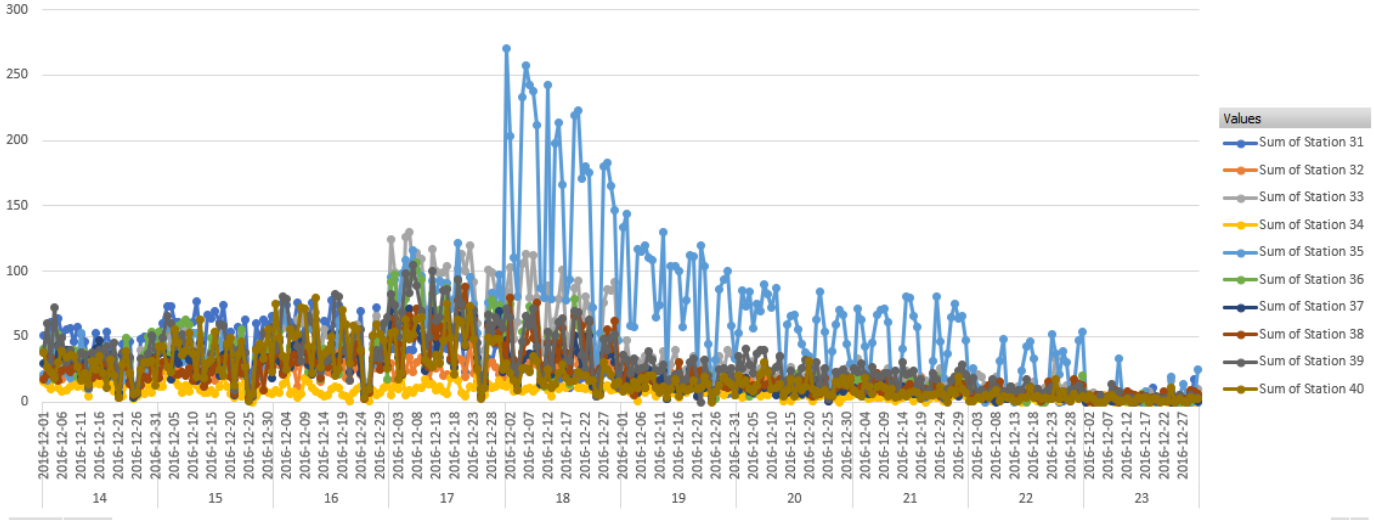


Figure 36. Actual hourly demand for station 31-40

The bike demand for station 31-40 shows approximately 0-70 bikes for 14:00-16:00 with a small increase to 17:00 to almost 100 bikes. The major difference between station 31-40 and 21-30 is a higher peak around 18:00 and a slightly lower peak around 17:00. Station 35 seems has a noticeable higher demand from 18:00 to 20:00 compared to other stations.

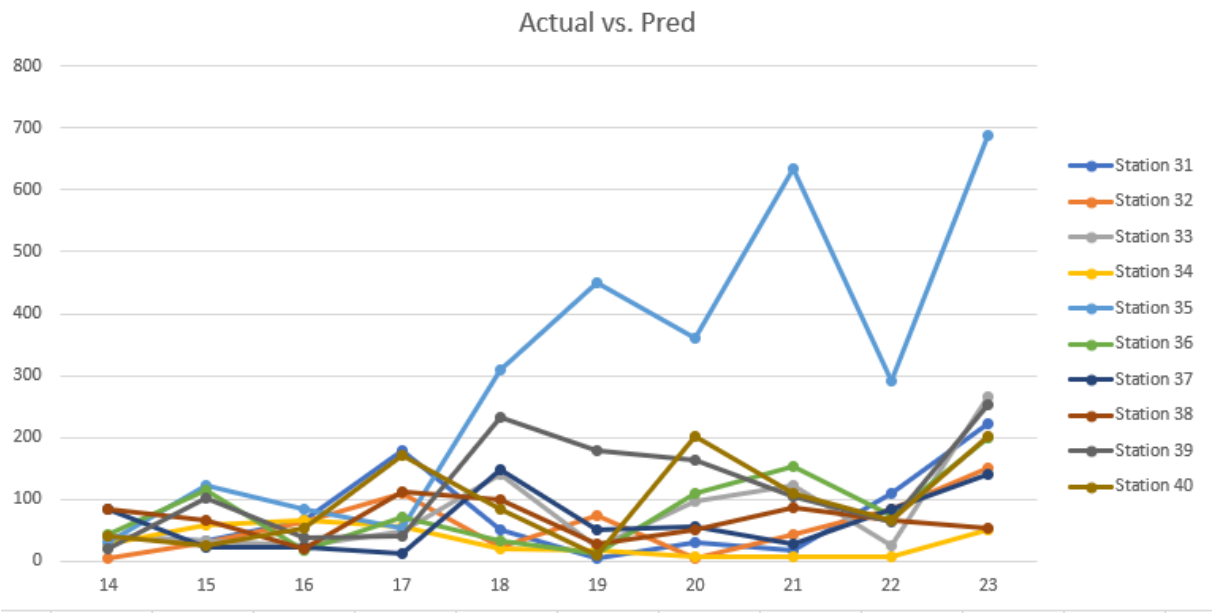


Figure 37. Difference between actual monthly demand and predicted demand

In terms of predicted demand computed by the neural network it can be seen that many of the predictions have a difference between, 3-23, 23-43 and 43-63 where few predictions have a difference of over 120 bikes, showed in Figure 38.

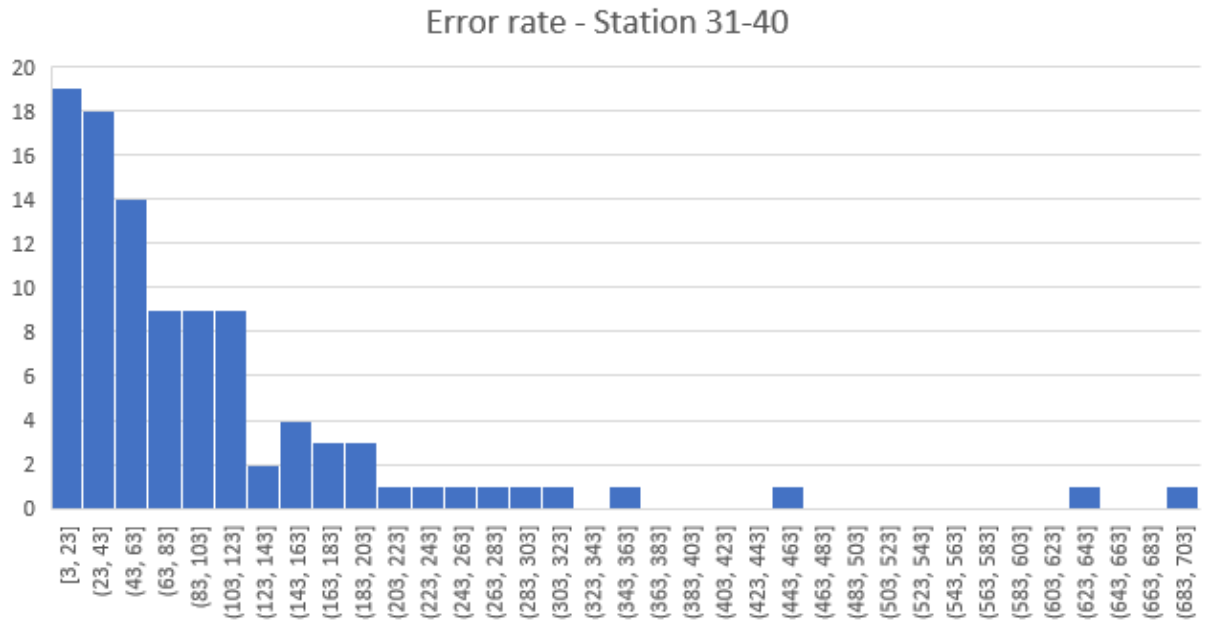


Figure 38. Predicted demand errors for stations 31-40

4.1.5 Station 41-50

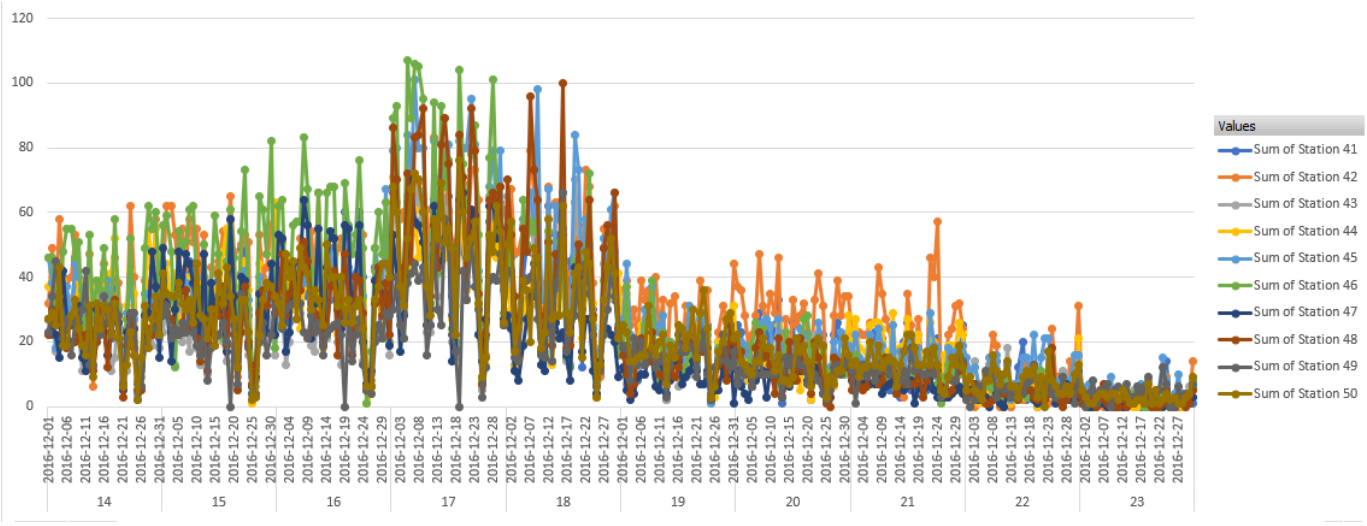


Figure 39. Actual hourly demand for station 41-50

For station 41-50 Figure 39 shows that similar patterns are established where the demand at hours 14:00-16:00 is around 10-60 bikes and peaks at 100 around 17:00. However, unlike previous station analysis which usually peaked around 18:00 the demand instead decreased.

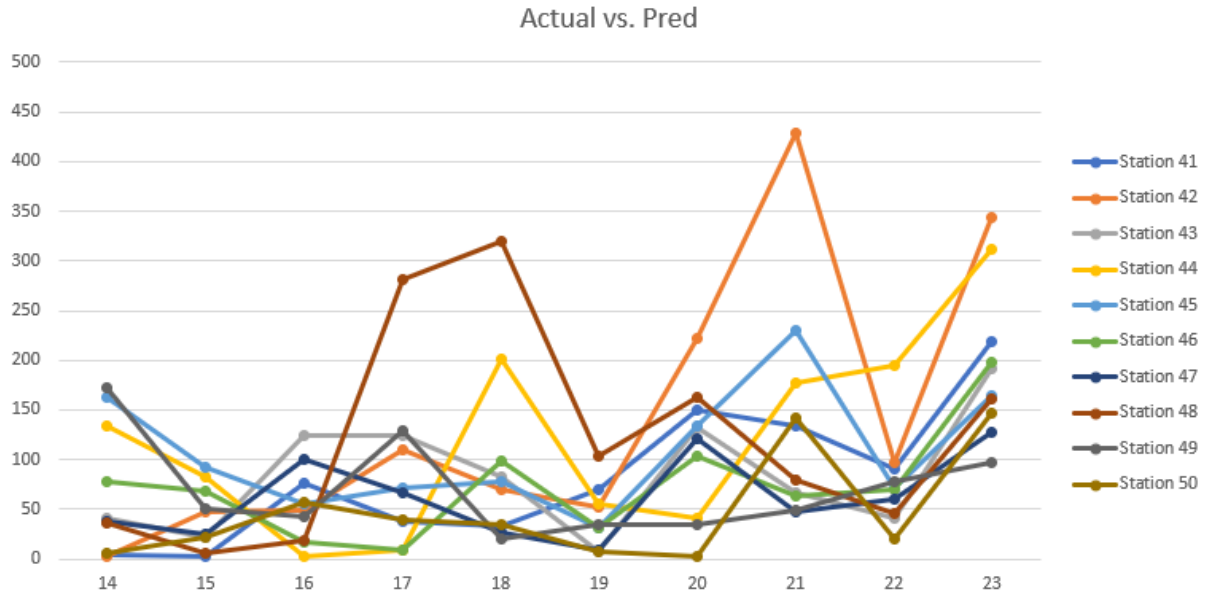


Figure 40. Difference between actual monthly demand and predicted demand

The predicted demand for stations 41-50 shows that 18 predictions had a low difference between 2-22 bikes compared to the real demand, where 15 predictions where between 22-42 and 62-82. There are some predictions, similar to previous error rates, which have higher differences compared to the real demand (such as over 200), however these are quite few.

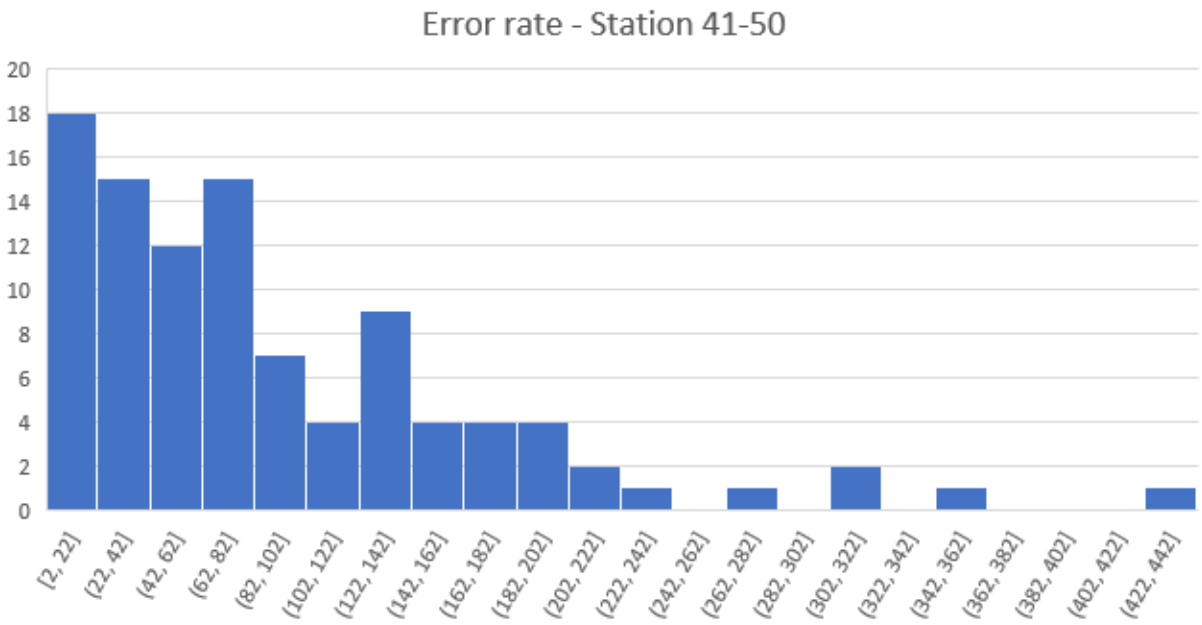


Figure 41. Predicted demand errors for stations 41-50

4.1.6 Model performance

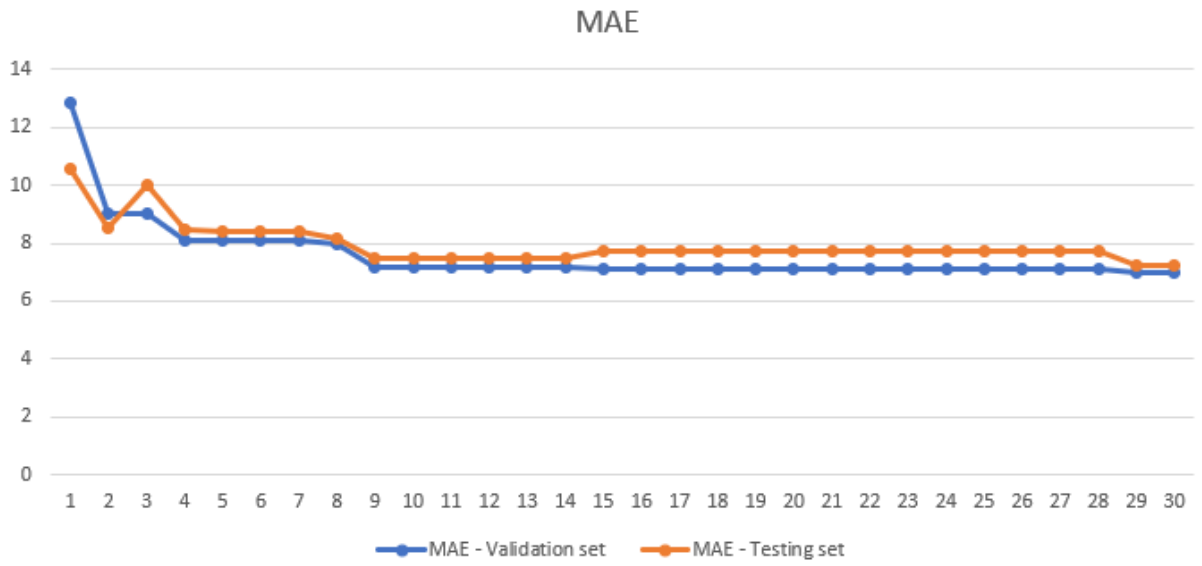


Figure 42. MAE plotted as a function of epochs

Figure 42 shows the mean absolute error for the base analysis plotted against the number of epochs. There are two different categories for the metric. There's one line that shows how the MAE for the validation set (blue) and another one that shows how the MAE is on the testing set (orange). Initially, both sets have high values but significantly decrease after four completed epochs. From epochs 4-7, the MAE for both sets practically remains unchanged whereas a slight decrease occurs from epochs 7-9. From epoch 10 to epoch 14, the MAE remains constant where the MAE for the testing set slightly increases while the MAE for the validation set goes the opposite way and decreases. The final value for the validation set was 6.985 and 7.241 for the testing set.

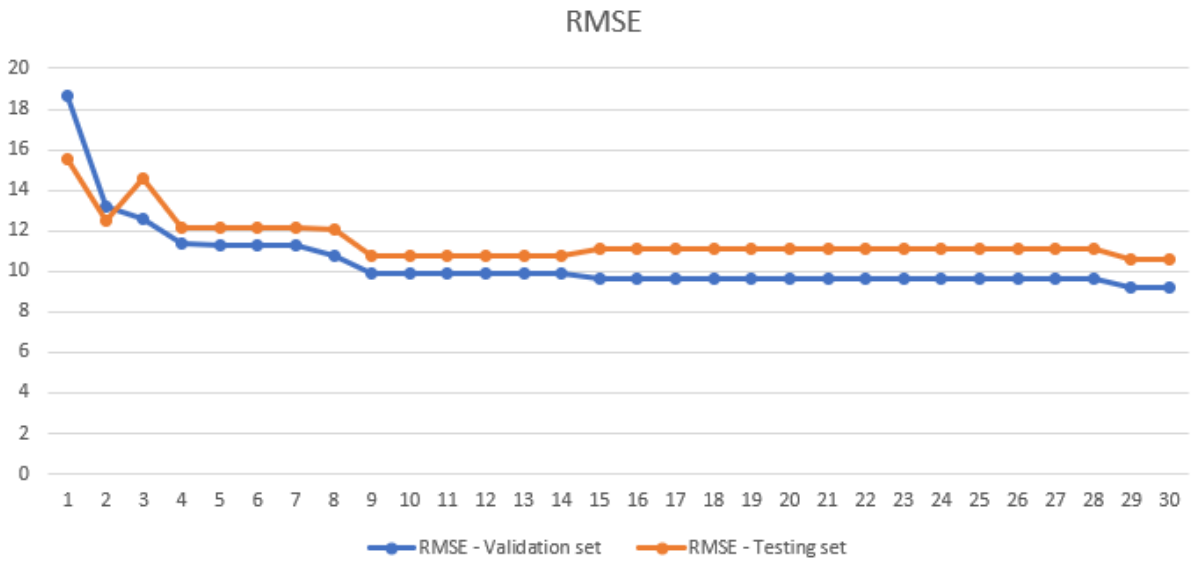
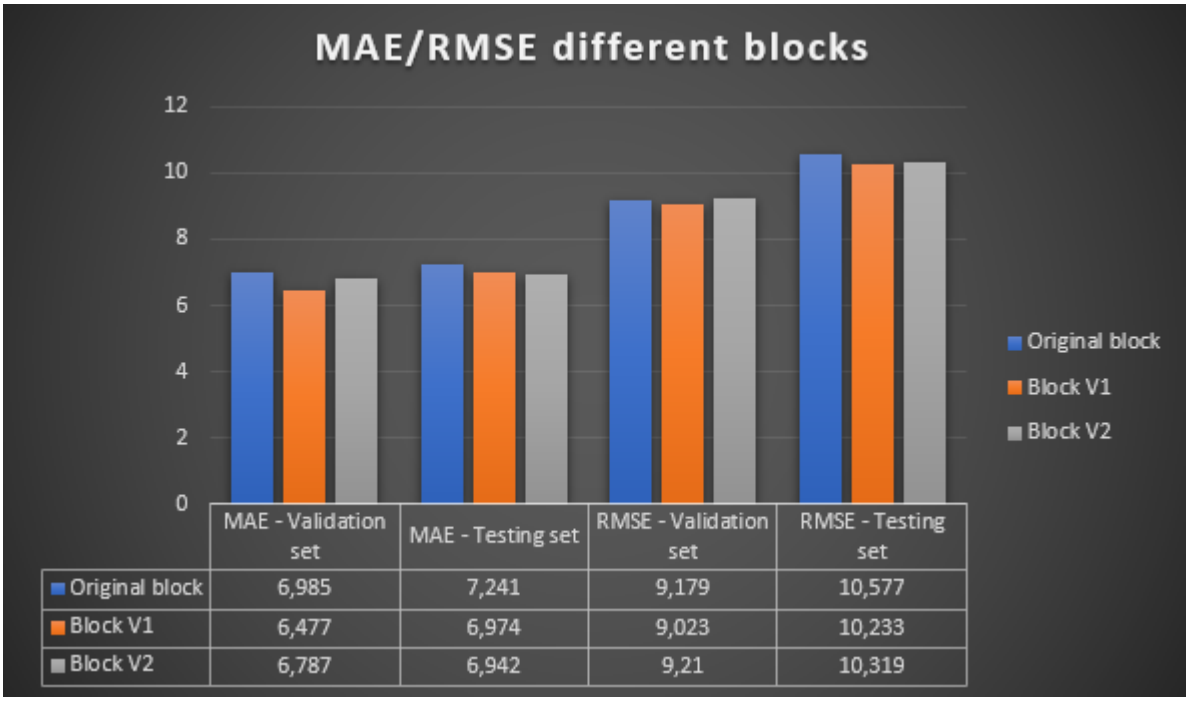


Figure 43. RMSE plotted as a function of epochs

Figure 43 shows the root-mean-square error for the validation- and testing set. The RMSE exhibits similar patterns compared to the MAE for the base analysis. Initially, both sets have high values but decrease to epoch 4 and is constant from epochs 4-7. The RMSE for both sets slightly decreases towards epoch 10 and again stays constant for 4 epochs, where the same pattern is repeated for epochs 15-28. The RMSE reaches its lowest value during the final two epochs, where RMSE for the validation set is approximately 9.2 and 10.6 for the testing set.

4.1.6.1 Block sizes

As previously mentioned, there are two sets of experimental block settings that were analyzed for each station analysis. Figure 44 shows the result for implementing these architectural layouts for the blocks, and the results show a small difference. Comparing the final values, the new MAE is 6.47 for the validation set and 6.9 for the testing set, which represents a 0.5 and 0.3 decrease, respectively.



	MAE - Validation set	MAE - Testing set	RMSE - Validation set	RMSE - Testing set
Original block	6,985	7,241	9,179	10,577
Block V1	6,477	6,974	9,023	10,233
Block V2	6,787	6,942	9,21	10,319

Figure 44. MAE/RMSE for different block layouts

Figure 45. shows the percentage improvements for the error metric compared to the original block layout. The biggest improvement was the MAE for the validation set, which showed a 7 % decrease (improvement). A negative percentage improvement means that the MAE or RMSE instead increased.

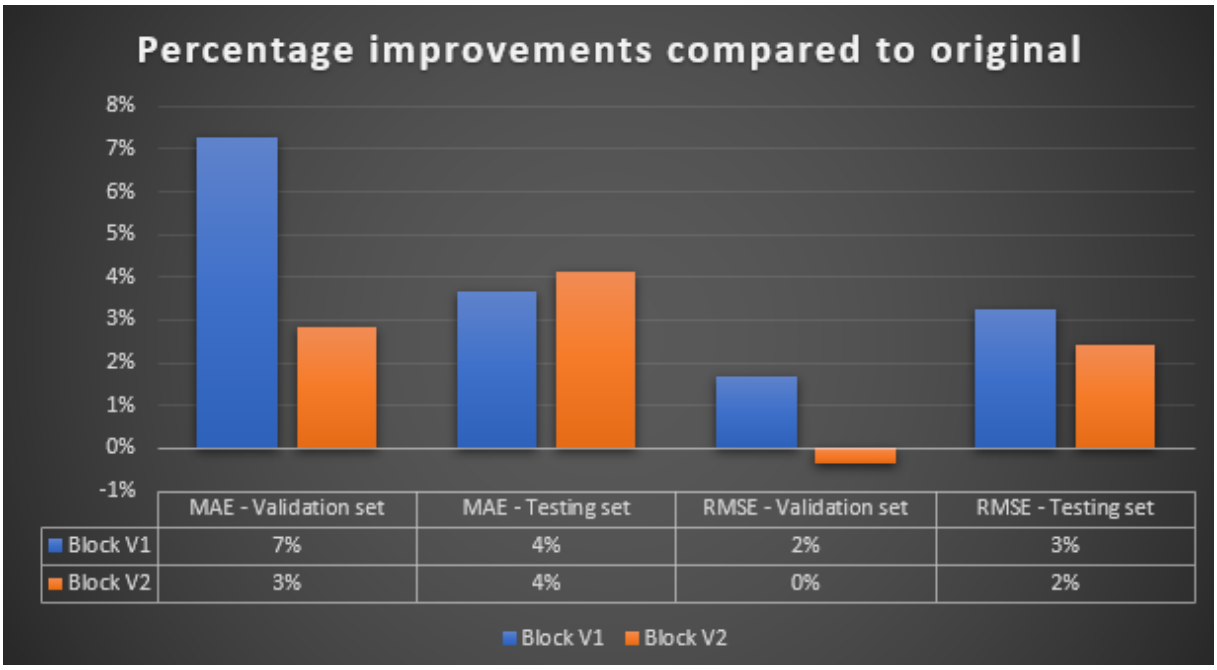


Figure 45. Percentage improvement compared to original block layout

4.1.6.2 Learning rates

For the learning rates, Figure 46 shows the values for both MAE and RMSE at the final epoch for the base case with a learning rate of 0.001 and the two experimental learning rates of 0.005 and 0.01. The neural network model with a learning rate of 0.005 performed best for both metrics and for both the validation- and testing set.

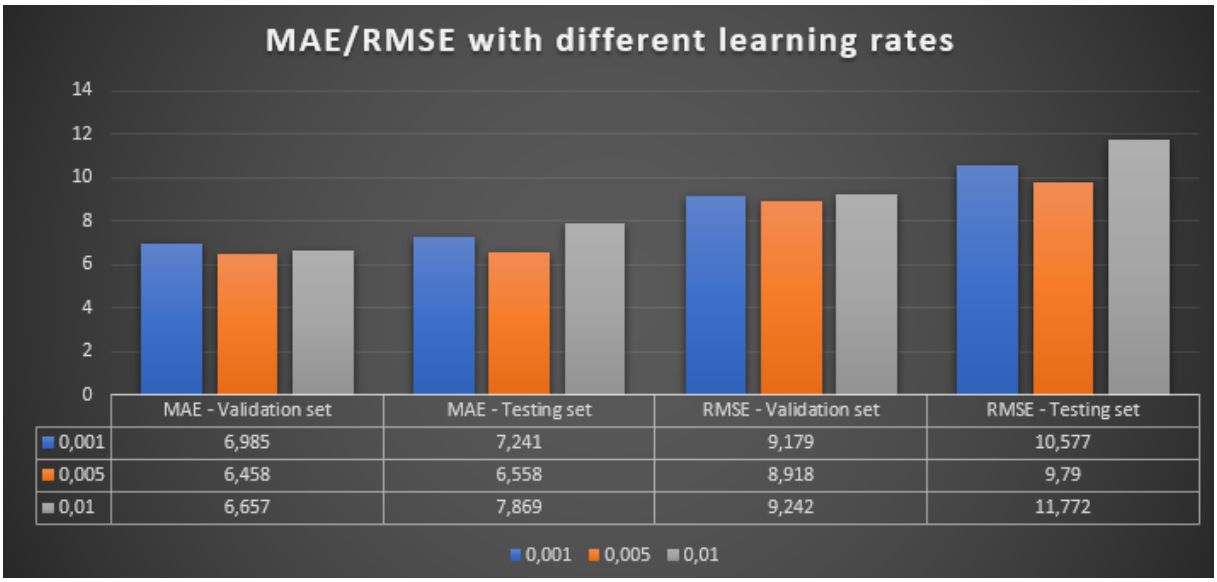
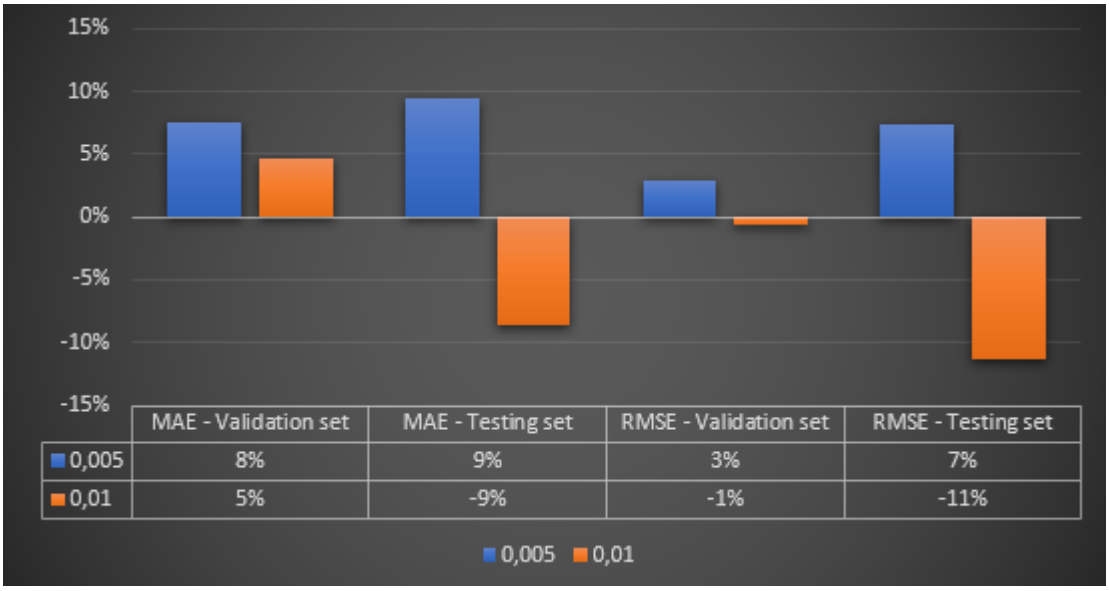


Figure 46. MAE/RMSE with different learning rates



	MAE - Validation set	MAE - Testing set	RMSE - Validation set	RMSE - Testing set
0,005	8%	9%	3%	7%
0,01	5%	-9%	-1%	-11%

Figure 47. Percentage Improvement compared to original learning rate

4.2 Analysis – 100 stations

4.2.1 Station 1-10

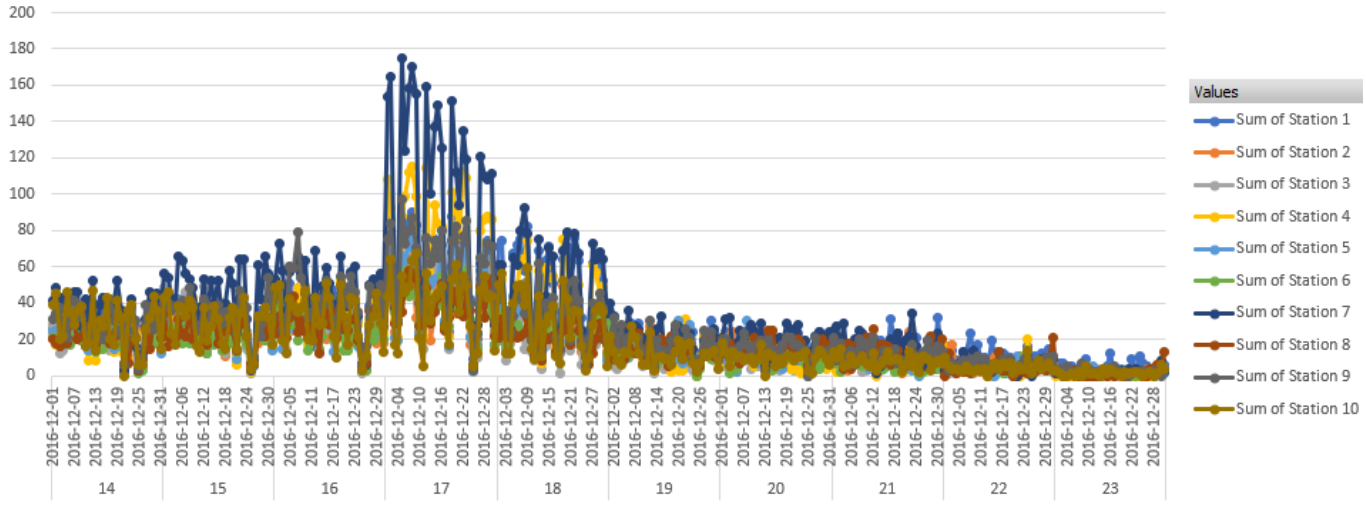


Figure 48. Actual hourly demand for stations 1-10

In Figure 48, it can be seen that for the majority of the station between 14:00-16:00 the station demand is around 20-80 bikes where there is a significant increase 17:00, especially for station 4 and station 7. After 17:00 the demand decreases remarkably to 19:00 where it later gradually decreases towards 23:00 where there is almost no bike demand for all stations.

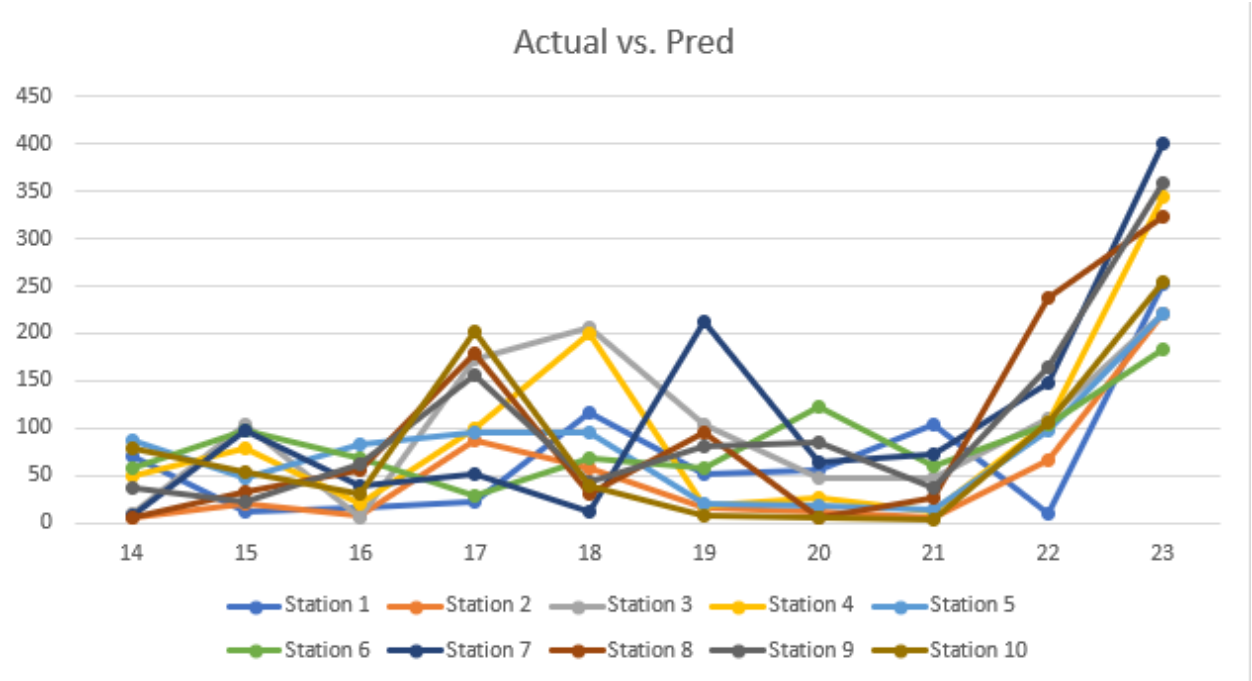


Figure 49. Difference between actual monthly demand and predicted demand

The actual vs. predicted demand for station 1-10 for the 100-station analysis shows that the majority of the predictions have a difference between 3-23 bikes compared to the real demand and many other predictions range between 23-43 and 43-63. It can be seen that there are very few predictions that have a difference larger than approximately 140 bikes compared to the real demand.

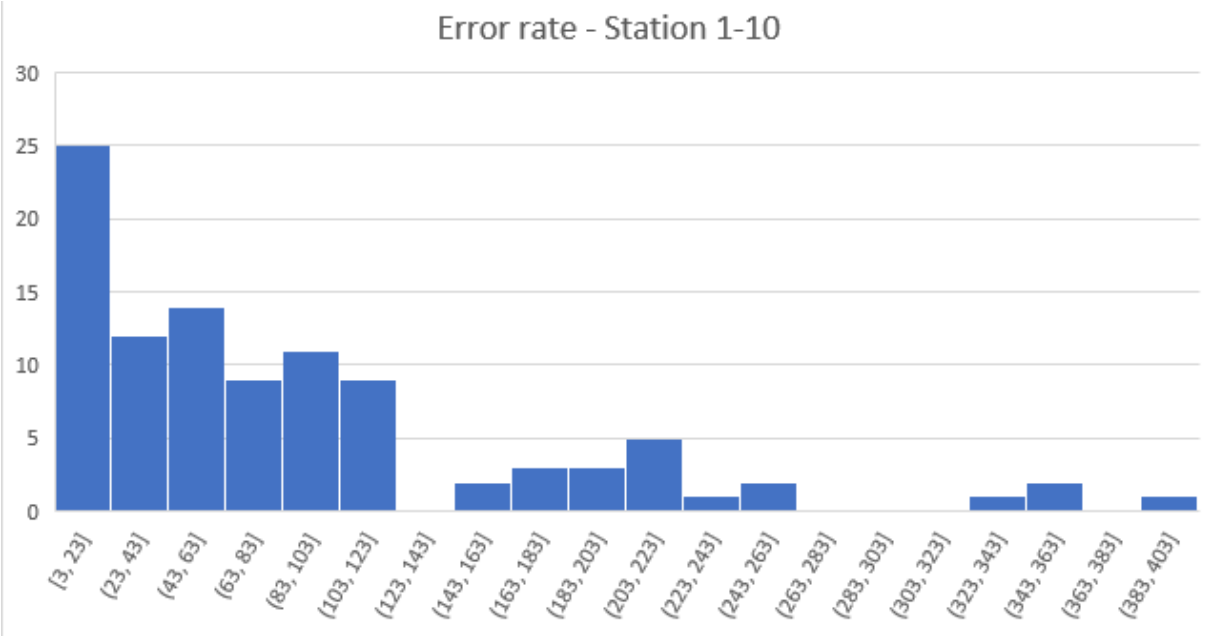


Figure 50. Predicted demand errors for stations 1-10

4.2.2 Station 11-20

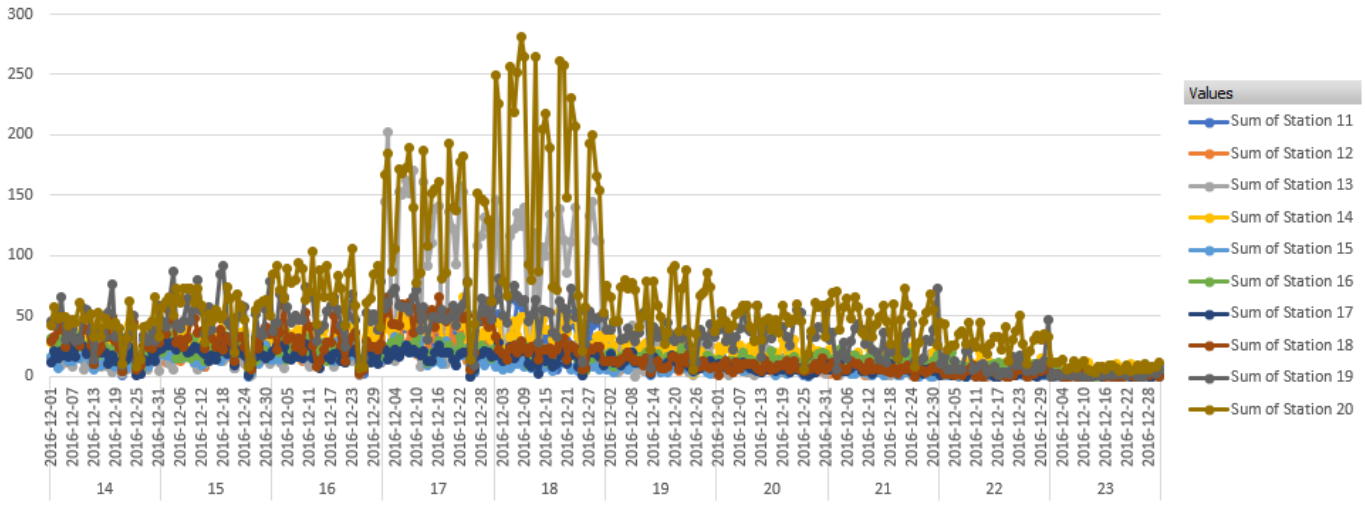


Figure 51. Actual hourly demand for station 11-20

Similar to station 1-10, the station demand for stations 11-20 show a bike demand of around 0-100 at 14:00-16:00 with a significant increase towards 17:00 which shows a bike demand of around 200. The difference here is that the peak is significantly higher at 18:00 (around 250) compared to the previous demand at the same time which was around 80. From 19:00-23:00 the demand exhibits similar patterns and gradually decreases.

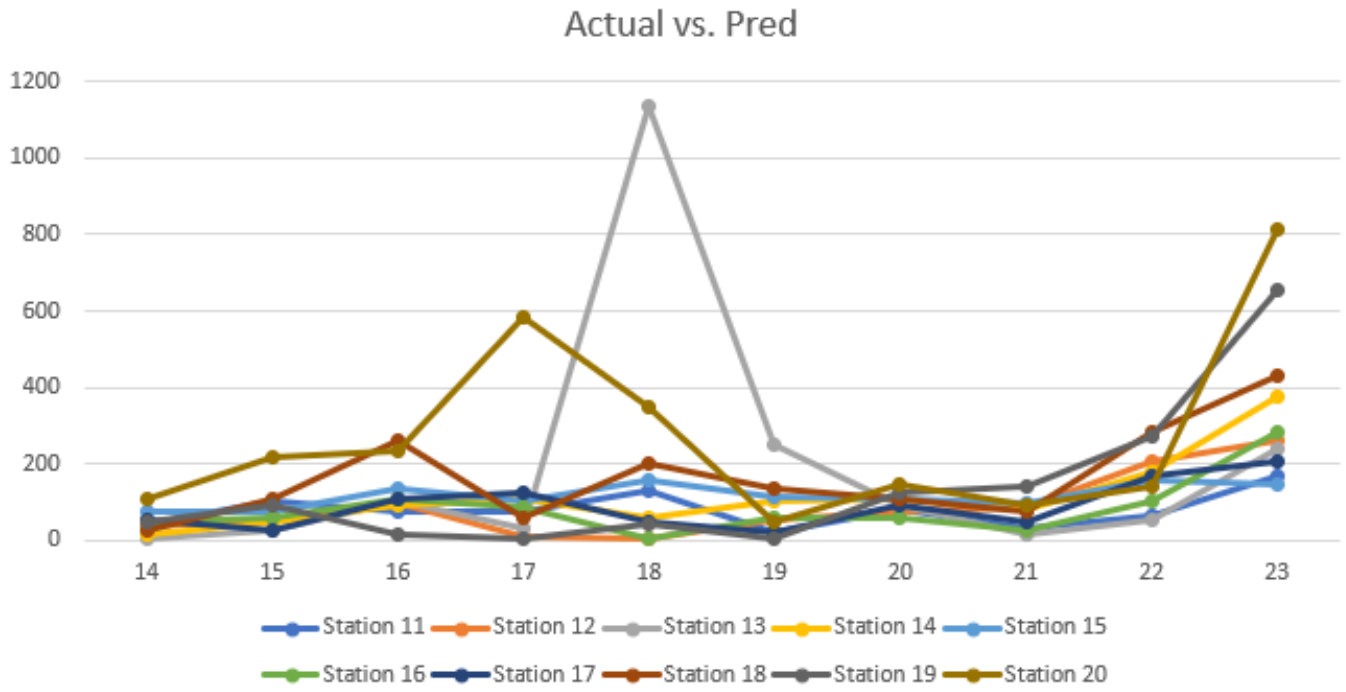


Figure 52. Difference between actual monthly demand and predicted demand

The computed demand by the neural network for stations 11-20 shows that many predictions have a difference of less than 100 bikes compared to the real demand, where the majority of predictions for the monthly demand had between 41-61 bike difference. Most of the prediction that had a difference above 160 bikes were for the final hour of the day, which can be seen in Figure 52.

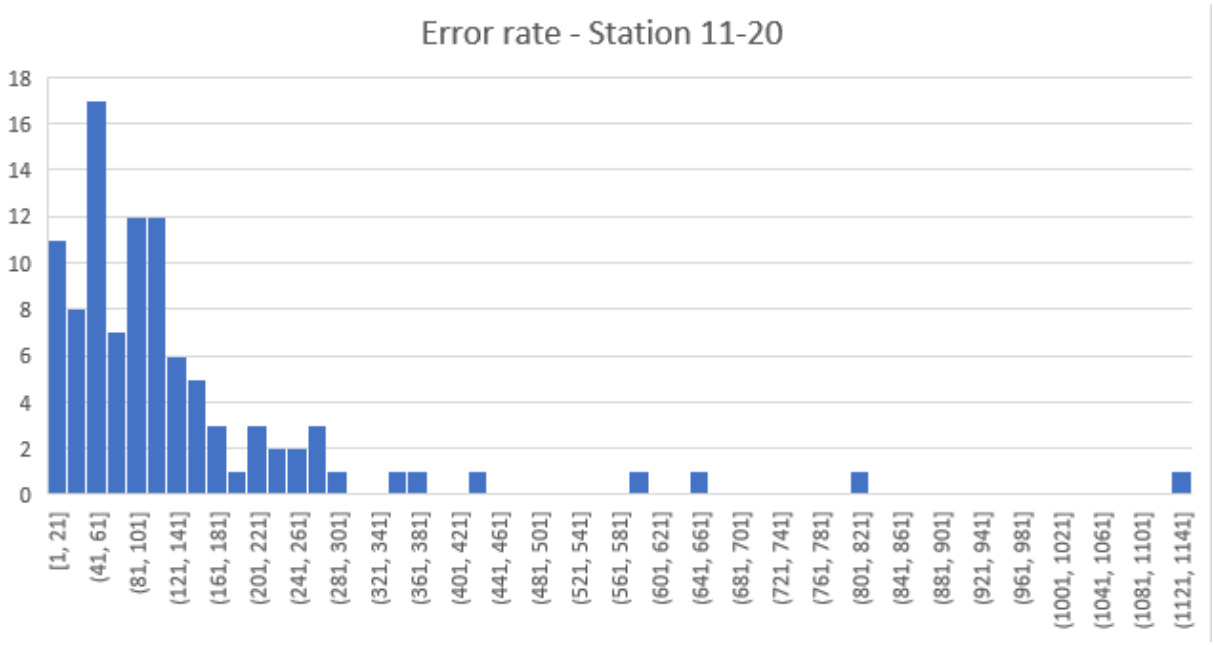


Figure 53. Predicted demand errors for stations 11-20

4.2.3 Station 21-30

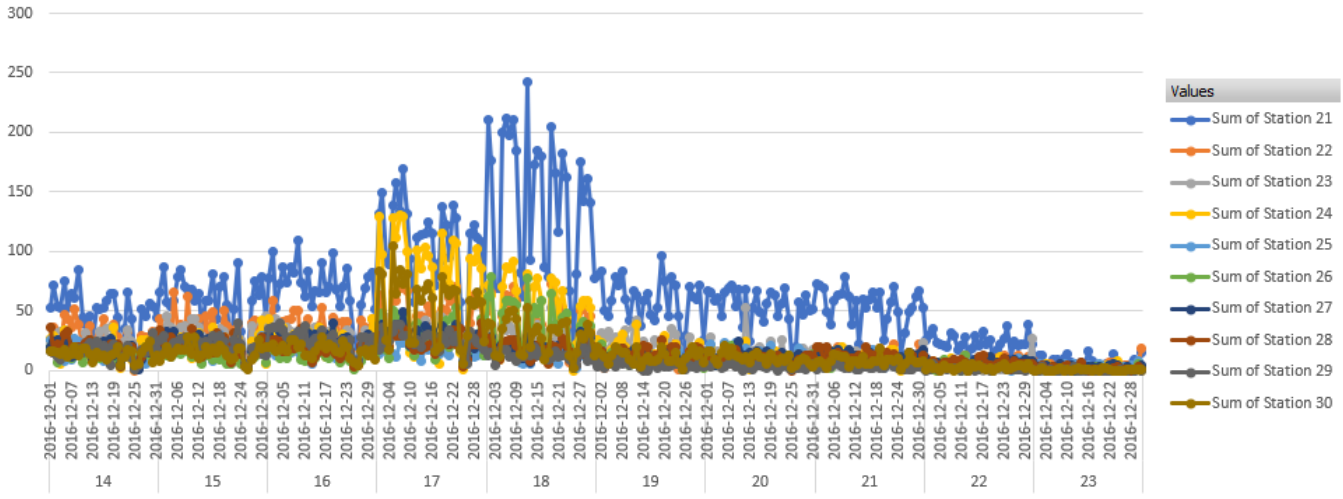


Figure 54. Actual hourly demand for station 21-30

For station 21-30 almost identical patterns and station demand can be seen compared to the station analysis for 11-20. Figure 54. shows that station 21 seem to have higher demand compared to other stations in all hours, most notably during 18:00 but also during continuing hours.

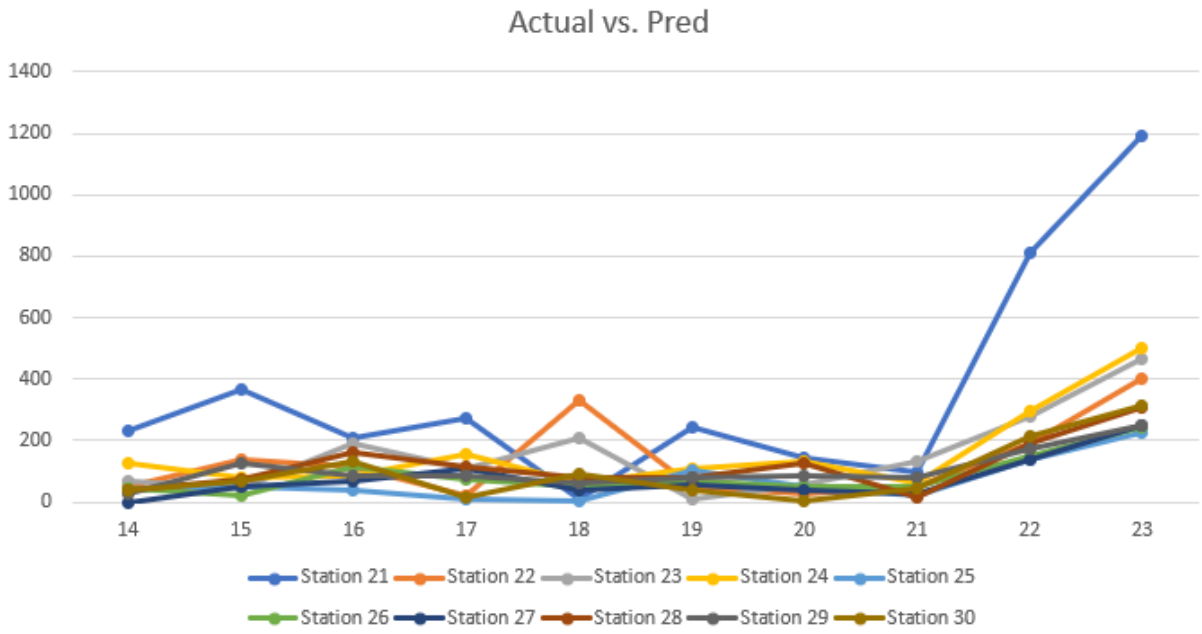


Figure 55. Difference between actual monthly demand and predicted demand

The actual vs. predicted demand shows that the majority of the predictions computed by the neural network had less than 100 bike difference compared to real demand, where 17 predictions had a difference between 40-60 bikes. In Figure 55 also here it can be seen that many of the data points that have more than 200 bikes difference are for hour 23:00.

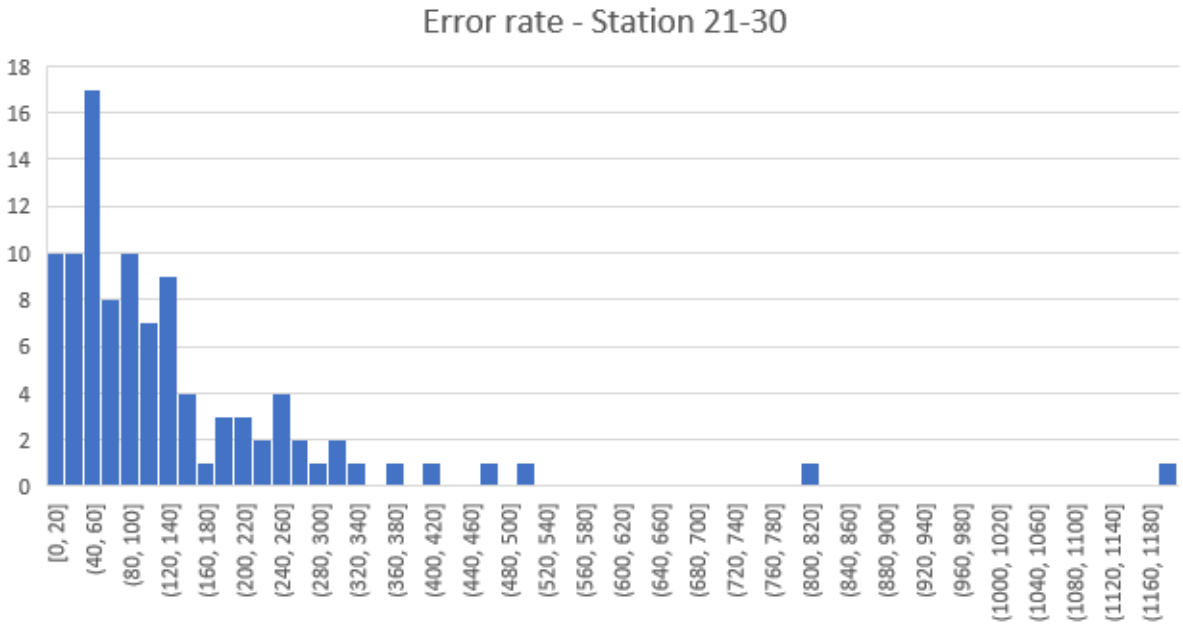


Figure 56. Predicted demand errors for stations 21-30

4.2.4 Station 31-40

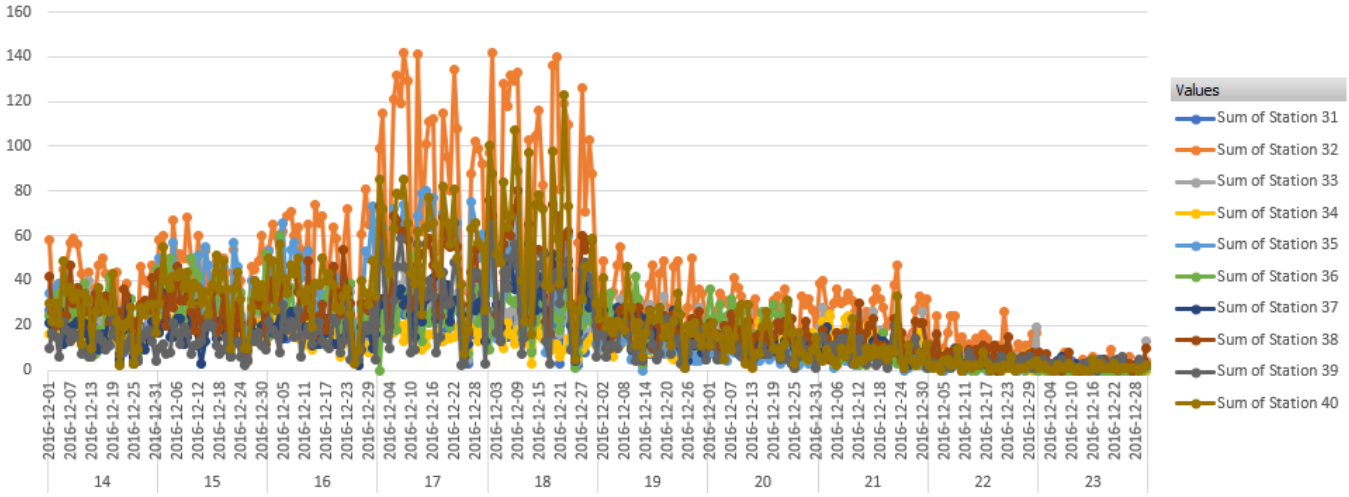


Figure 57. Actual hourly demand for station 31-40

The actual demand plotted for station 31-40 in Figure 57. show that station 32 has the highest demand throughout the day whereas the rest of stations follow similar patterns of demand. The highest peak for station 32 is approximately 140 bikes and occurs between 17:00-18:00. The demand later falls below 60 for all stations and continues to decrease towards 23:00.

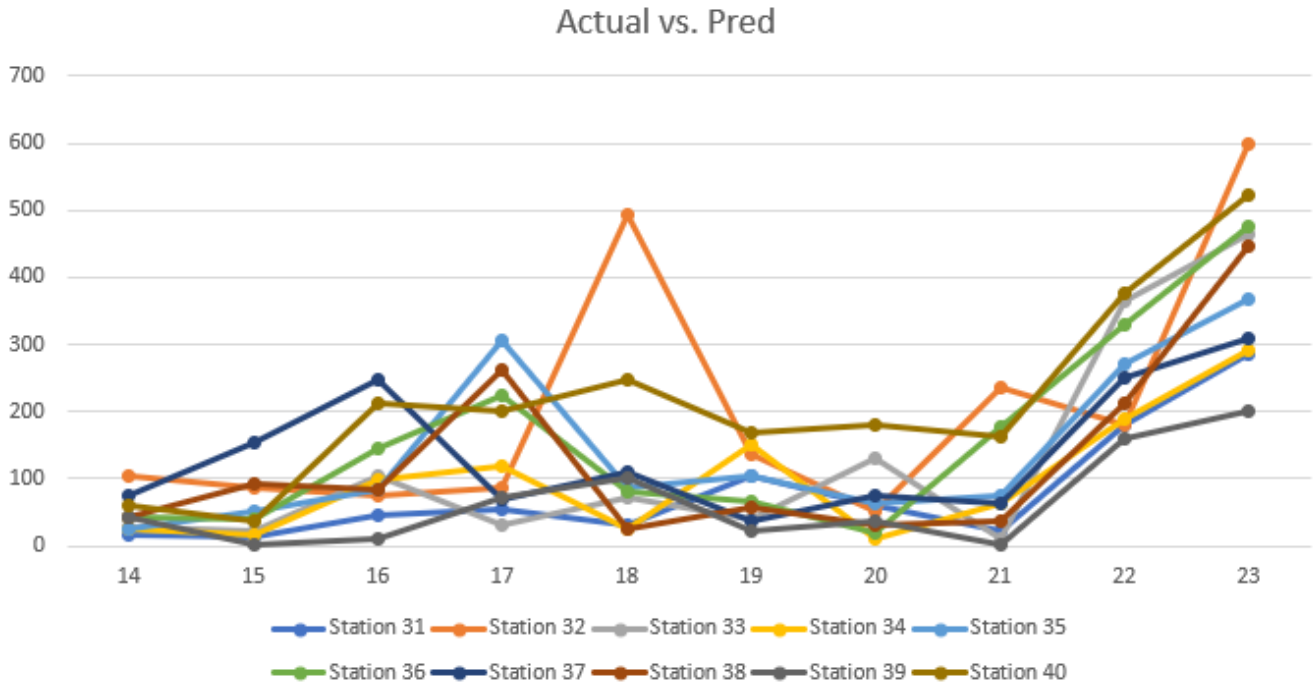


Figure 58. Difference between actual monthly demand and predicted demand

Comparing the actual demand vs. the predicted demand for the stations showcase that many datapoints have a difference of less than 100 bikes compared to real demand. However, Figure 58. shows that the neural network is not very accurate when it comes to 22:00-23:00, where a large portion of the non-accurate predicted demand is from the last two hours of the day.

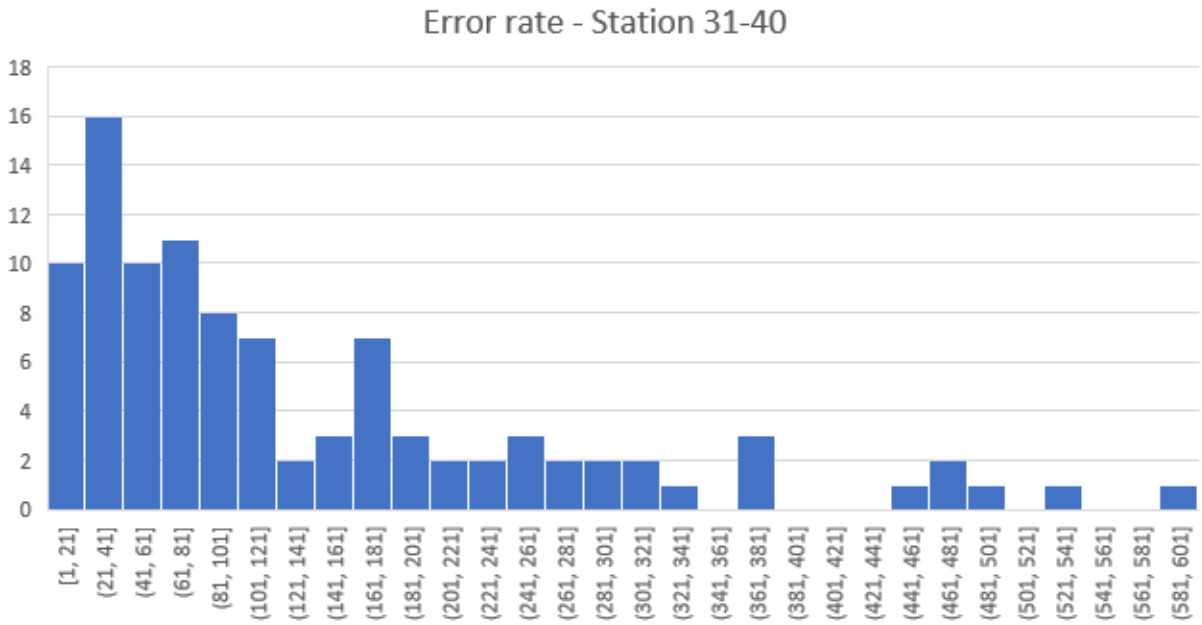


Figure 59. Predicted demand errors for stations 31-40

4.2.5 Station 41-50

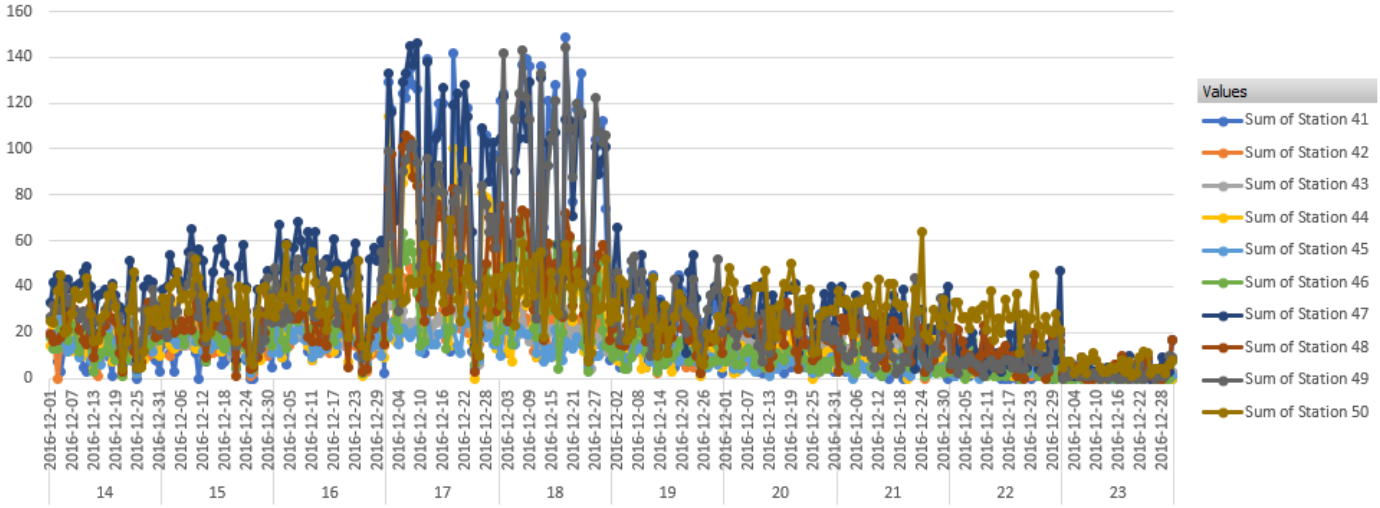


Figure 60. Actual hourly demand for station 41-50

The demand for station 41-50 shows that follow the same pattern where some stations have a higher peak during 17:00-18:00, most notably stations 45, 47 and 49. After 19:00 all stations fall below 60 and continue to decrease every hour to 23:00.

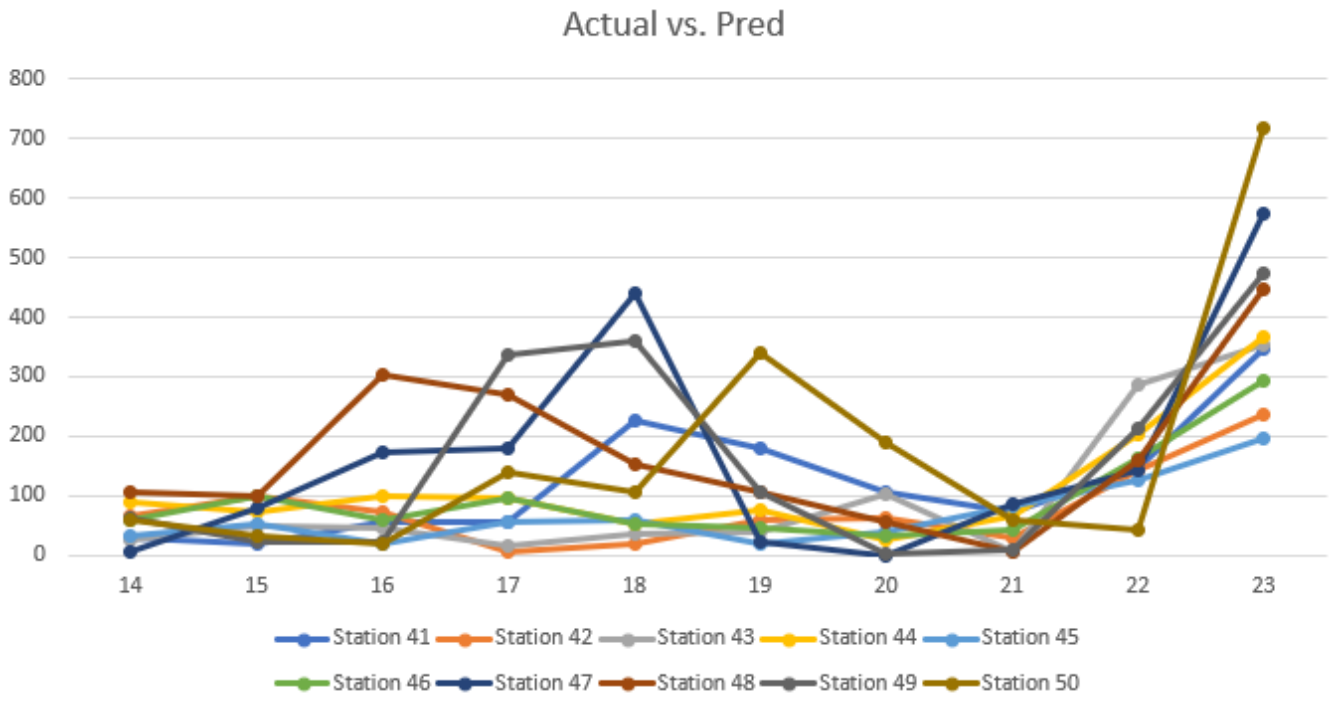


Figure 61. Difference between actual monthly demand and predicted demand

Comparing the predicted demand with actual demand for stations 41-50 shows that most predictions had a difference ranging between 40-60 bikes, where 12 predictions had between 0-20 bikes difference and 13 predictions had between 20-40 difference. Figure 61 shows that most of the prediction with lower accuracy came at 23:00 with some occurring during 17:00-18:00.

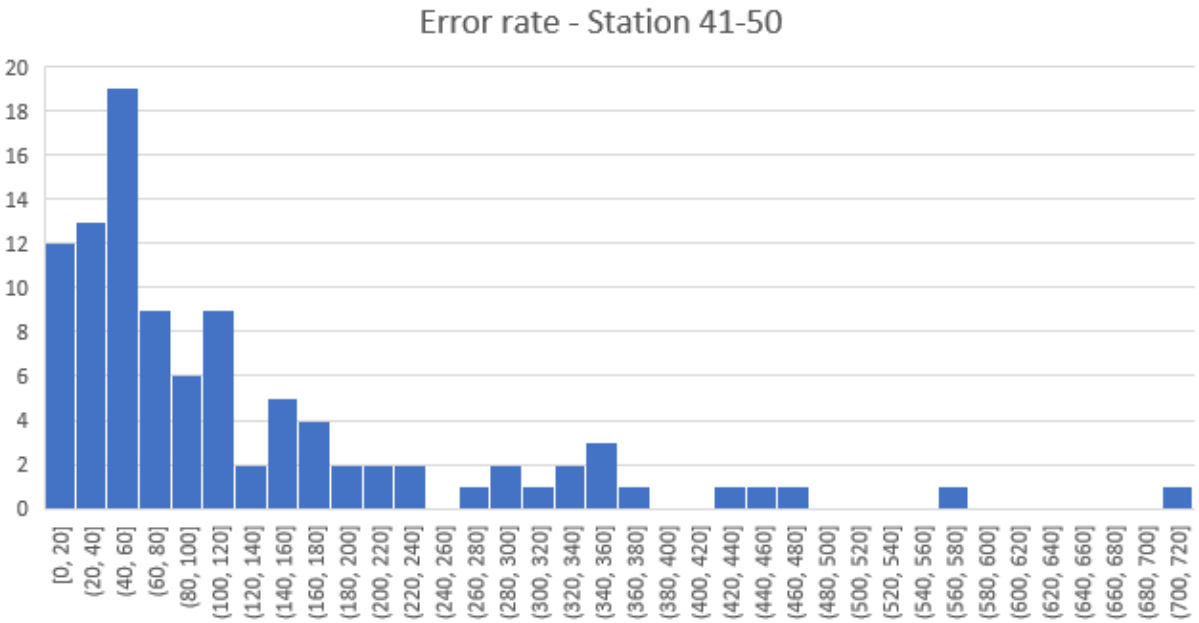


Figure 62. Predicted demand errors for stations 41-50

4.2.6 Model performance

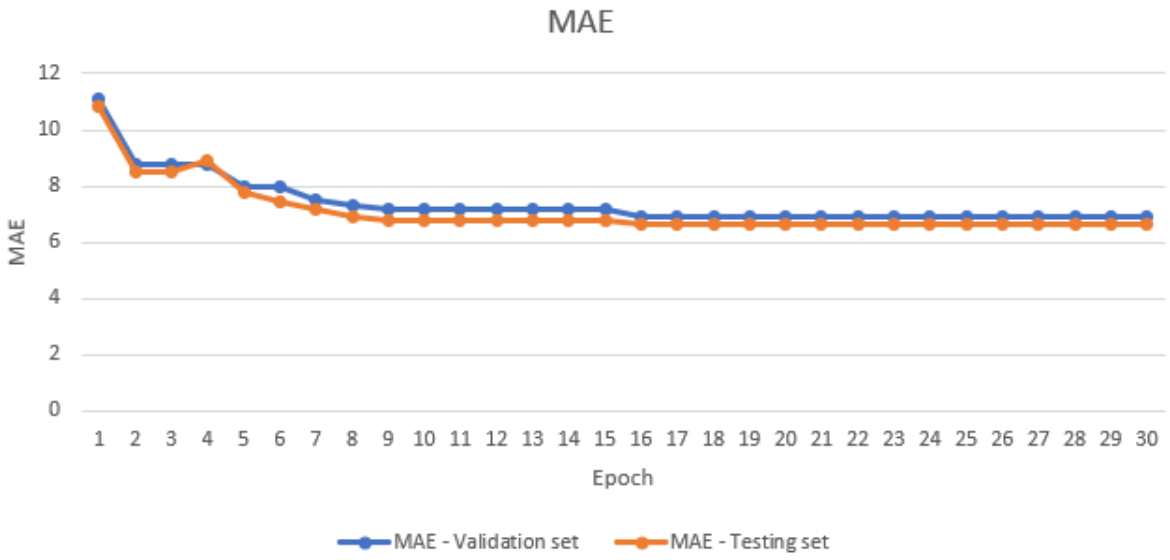


Figure 63. MAE plotted as a function of epochs

The mean absolute error for the validation- and testing set show that the initial values for MAE are both around 11. After completing two epochs it can be seen that the error rate decreases to approximately 8.8 for both sets. From epoch 6 to epoch 15 the mean average error for both sets decrease approximately 0.9 and then decrease another 0.2 from epoch 15 to 16 and stabilizing from epoch 16 and onwards.

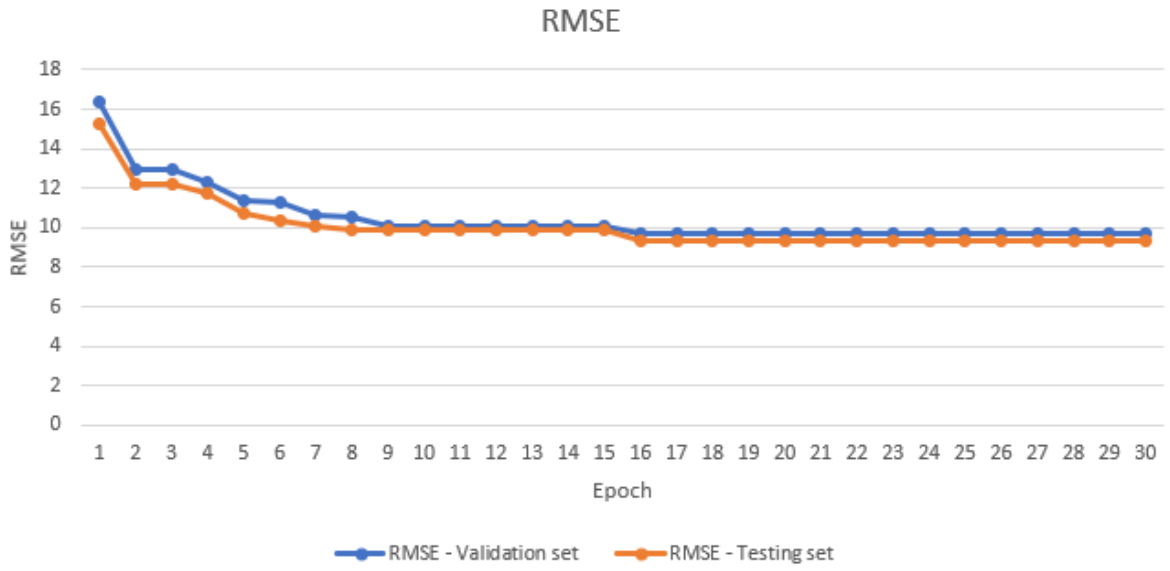


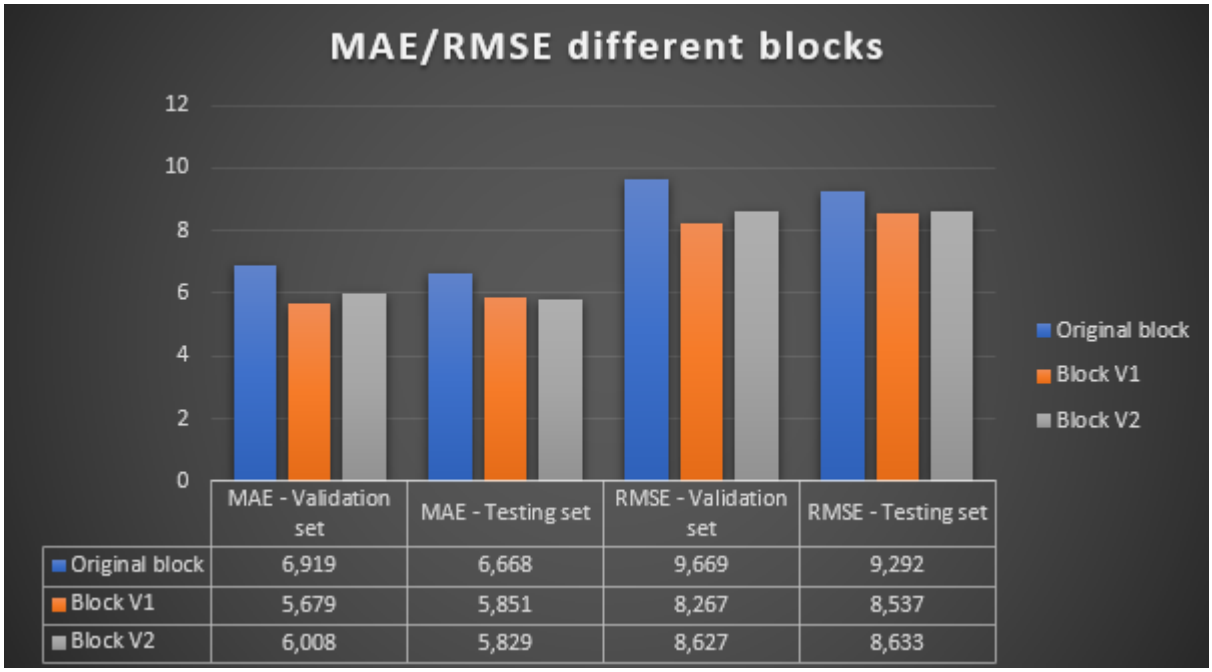
Figure 64. RMSE as a function of epochs

The root mean square error for the validation- and testing sets show that the error value is initially quite high around 15-16 for both sets but gradually decreases towards the ninth epoch for the model. Figure 64. show that from epoch 9 to epoch 15 it can be seen that the RMSE value remains unchanged and slight decreases from epoch 15 to epoch 16 where it remains constant to epoch 30. The final value was 9.7 for the validation set and 9.3 for the testing set.

4.2.6.1 Block sizes

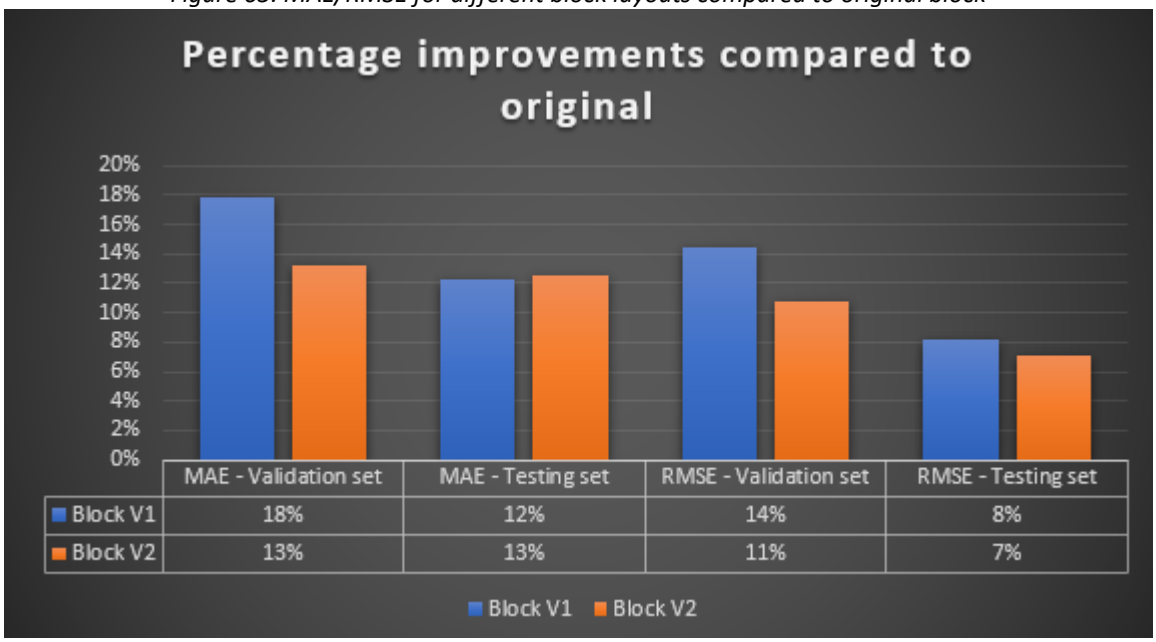
The experimental block layouts gave the following error rates according to Figure 65. The figure shows that the first version of block layout, which is $[1, 64, 128]$ and $[128, 64, 256]$, has the best results for

both metrics and sets. For the validation set the MAE decreases from 6.919 to 5.679, which is almost a 20 % decrease, shown in Figure 66, compared to a 13 % decrease with the second version of the block layout. Similarly, the RMSE for the validation set decreased by 14 % for the first version of the block layout compared to a 10 % decrease for the second version of the block layout.



	MAE - Validation set	MAE - Testing set	RMSE - Validation set	RMSE - Testing set
Original block	6,919	6,668	9,669	9,292
Block V1	5,679	5,851	8,267	8,537
Block V2	6,008	5,829	8,627	8,633

Figure 65. MAE/RMSE for different block layouts compared to original block



	MAE - Validation set	MAE - Testing set	RMSE - Validation set	RMSE - Testing set
Block V1	18%	12%	14%	8%
Block V2	13%	13%	11%	7%

Figure 66. Percentage improvement compared to original layout

4.2.6.2 Learning rates

The experimental learning rates of 0,005 and 0,01 were implemented for the 100-station analysis and gave the following results according to Figure 67. This figure shows the final value for both metrics for the validation- and testing set with three different blocks each representing a learning rate. For the mean absolute error, the model with learning rate 0,01 performed best by slim margins for the validation set while the model with learning rate 0,005 performed best for the testing set. When it comes to the root-mean-square error Figure 68 shows that the model with learning rate 0,005 performed best for both sets.

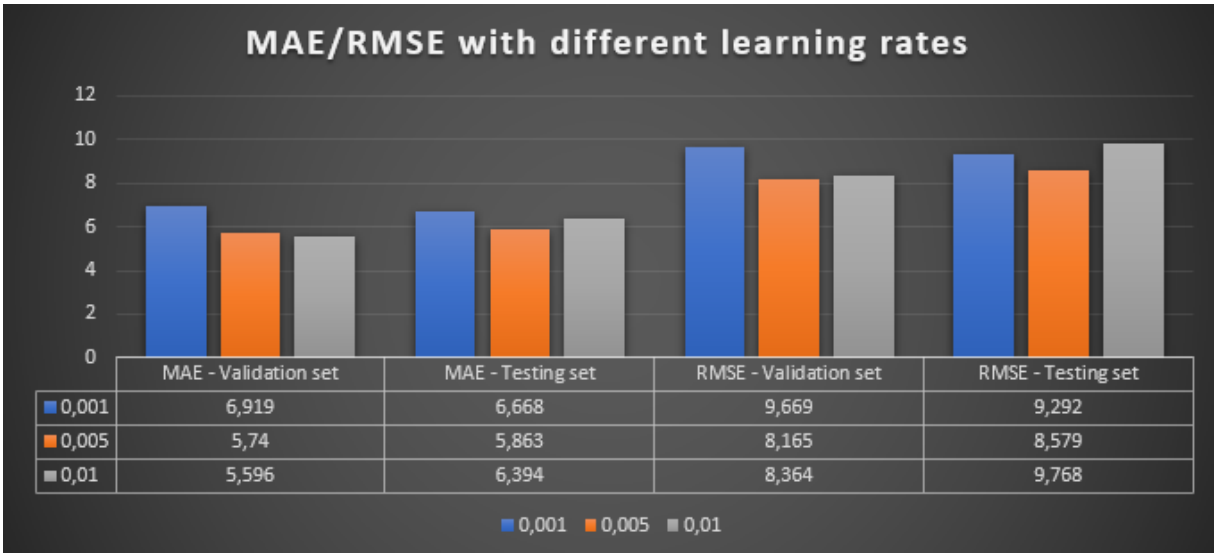


Figure 67.MAE/RMSE for different learning rates for the neural network

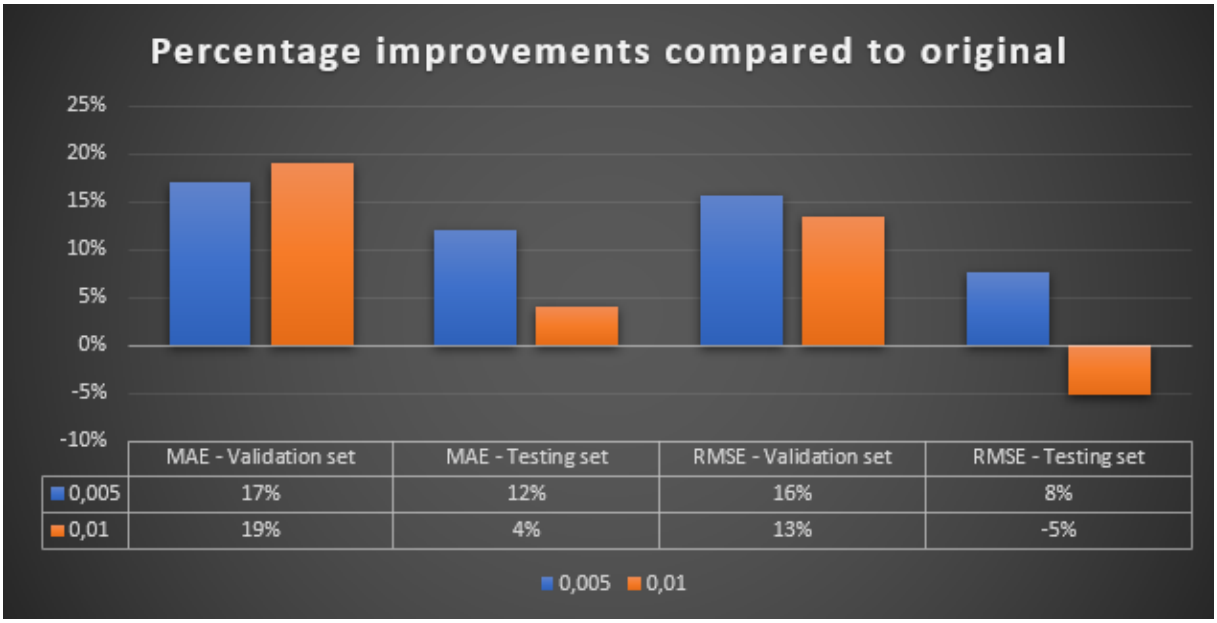


Figure 68. Percentage Improvement compared to original learning rate

4.3 Analysis – 200 stations

4.3.1 Station 1-25

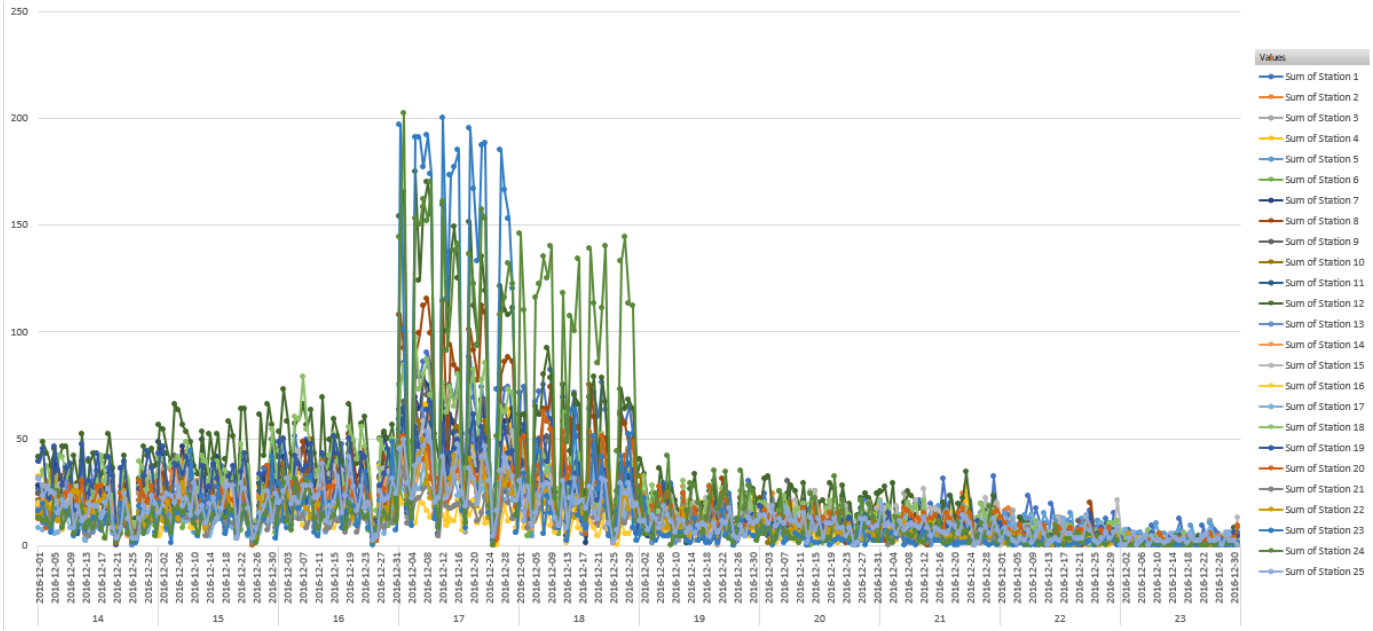


Figure 69. Actual hourly demand for station 1-25

The demand for stations 1-25 in the 200-station analysis show that the majority of stations from 14:00-16:00 have a demand between 0-50 bikes. Looking at 17:00 it can be seen that many stations have their peak demand, where the highest is for station 5 which is approximately 200 bikes. Many stations fall below 50 bike demand after 19:00-20:00 and reach below 25 for 23:00.

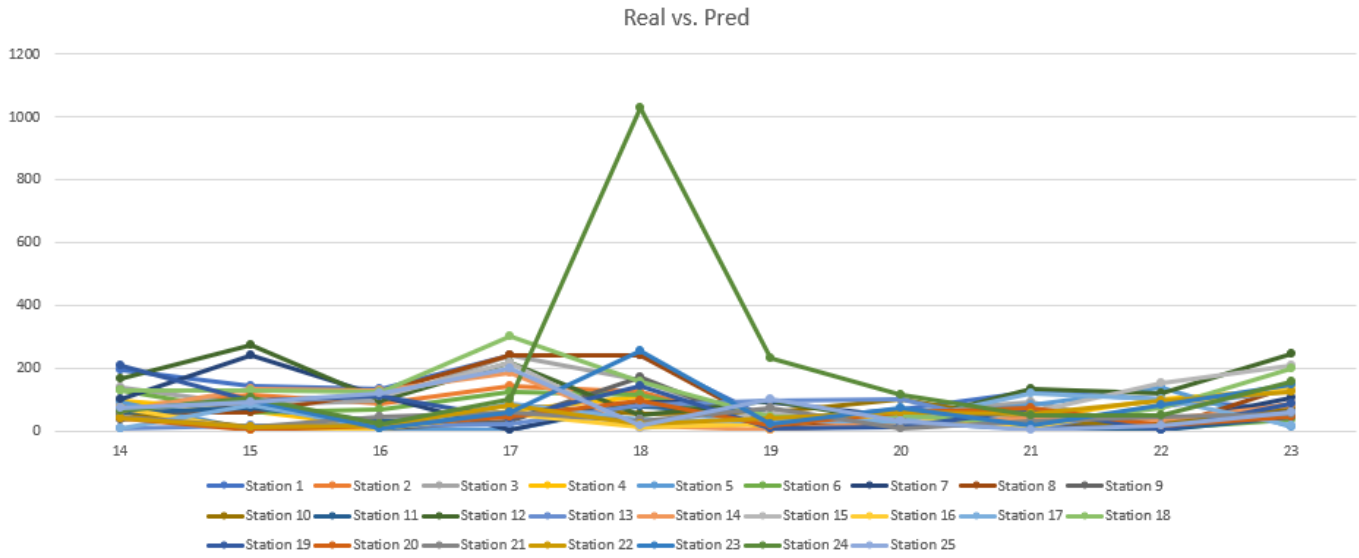


Figure 70. Difference between actual monthly demand and predicted demand

For the comparison of the predicted demand by the neural network and the actual demand, Figure 70. shows that the majority of the predictions have a difference of less than 200 bikes compared to the real demand, with some exceptions. In Figure 71. it can be seen that almost 60 predictions had a difference ranging between 2-22 compared to the real monthly demand, which is a significant increase compared to previous station analysis.

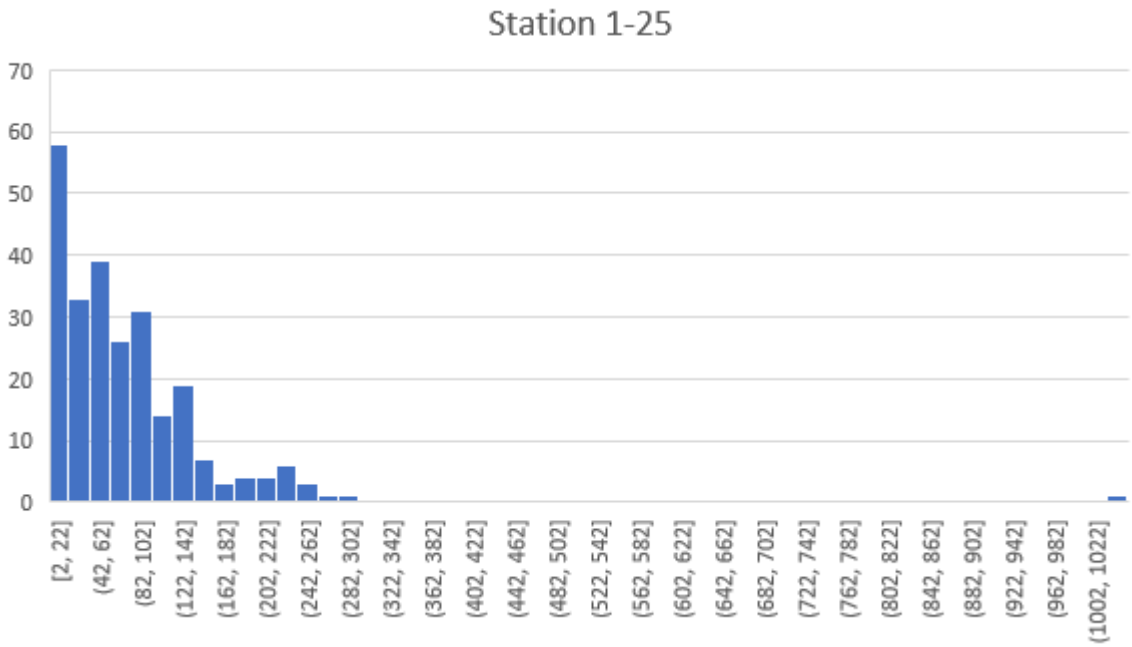


Figure 71. Predicted demand errors for stations 1-25

4.3.2 Station 26-50

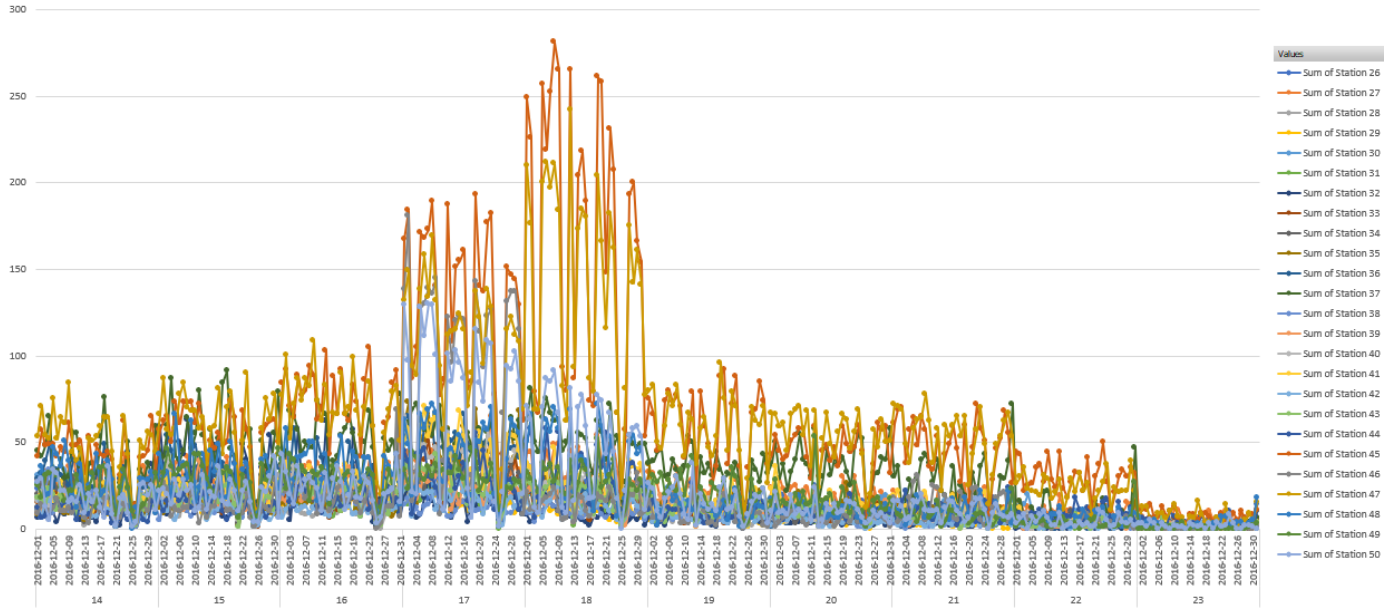


Figure 72. Actual hourly demand for station 26-50

For station 26-50 in Figure 72 shows that similar to station 1-25, the demand for the majority of the stations are between 0-100 with the exception of three stations. Many stations reach peak demand at 17:00 except for stations 45 and 47, which have their peak at 18:00.

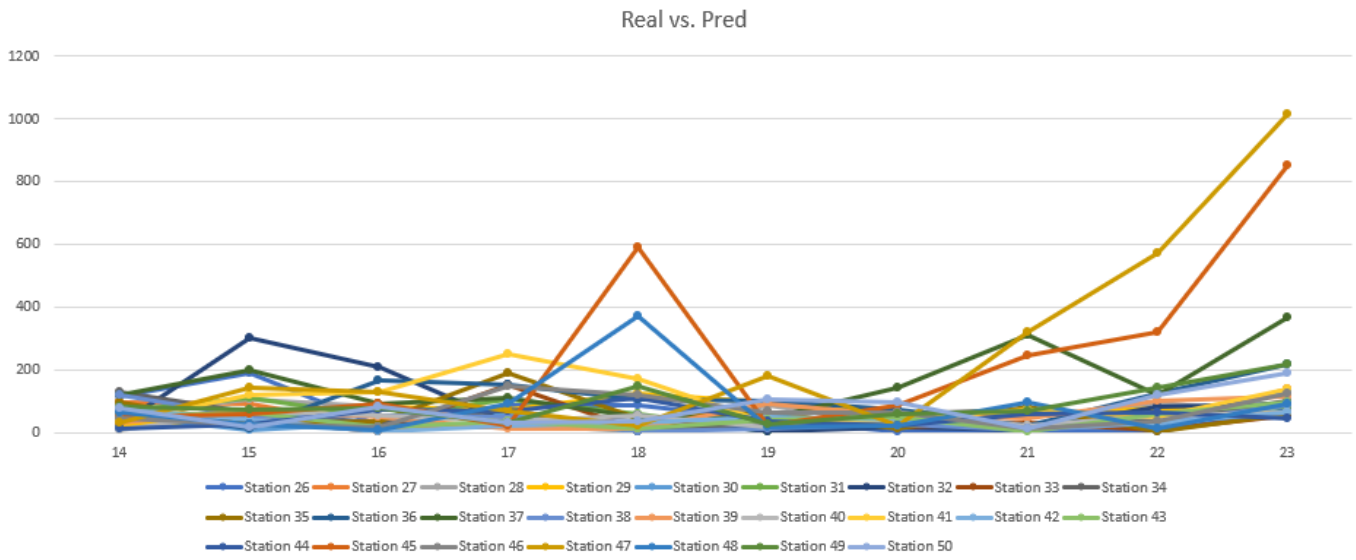


Figure 73. Difference between actual monthly demand and predicted demand

Comparing the real demand with the predicted demand showed that the majority of the predictions have a difference ranging between 1-21 and 21-41, shown in Figure 74. Figure 73 shows that many of the predictions had a difference of less than 200 bikes with a few anomalies, most notably for 23:00.

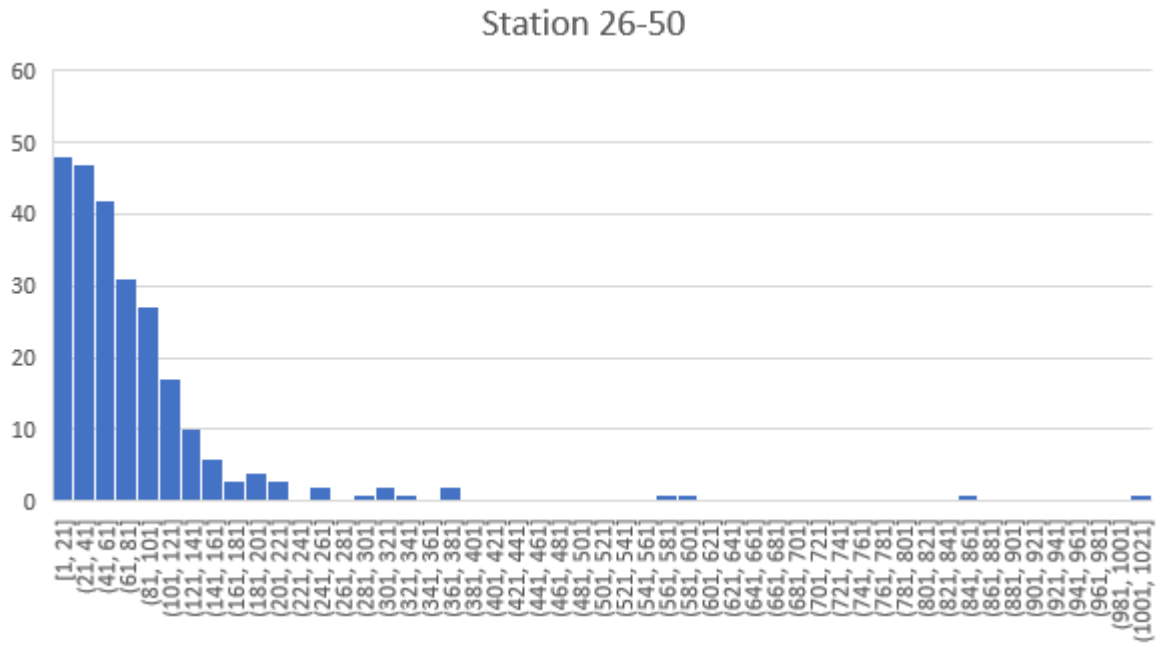


Figure 74. Predicted demand errors for stations 26-50

4.3.3 Model performance

The mean absolute error for the 200-station analysis has an initial value of approximately 8.9 for the validation set and 8.1 for the testing set. The MAE for both the validation- and testing set converge after two epochs and decrease at almost the exact same rate. After five epochs it can be seen from Figure 75 that the MAE is approximately 5.9 for both sets and decreases in two steps after 13 epochs where it remains constant towards the final epoch.

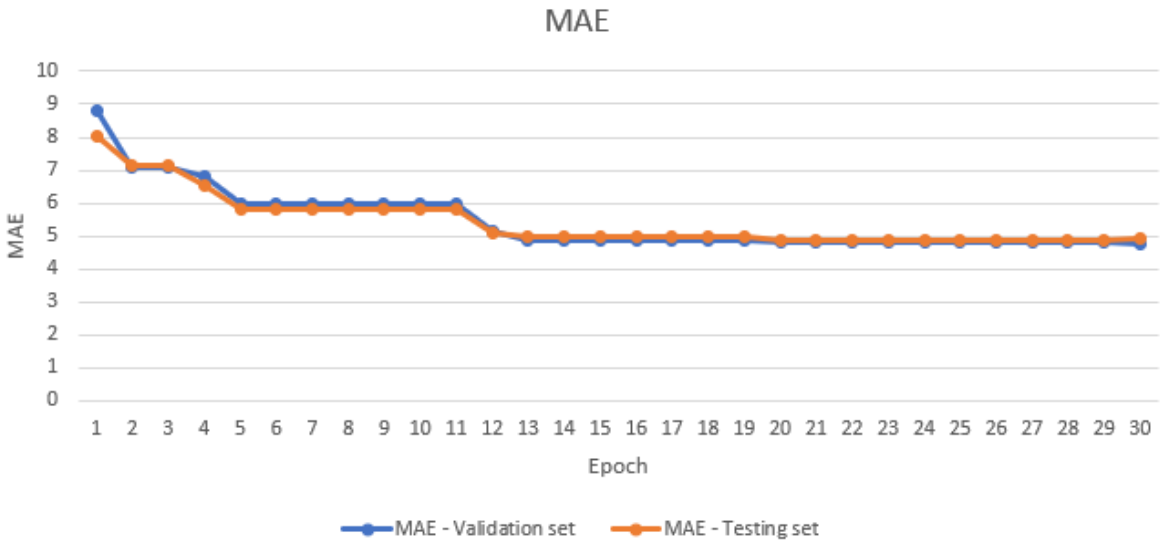


Figure 75. MAE plotted as a function of epochs

For the root-mean-square error value it can be seen from Figure 76. that the error rate mainly decreases in three separate stages, from epoch 1-2, epoch 3-5 and epoch 11-13. The RMSE converges for both sets around epoch 13 and almost remains entirely constant towards epoch 30.

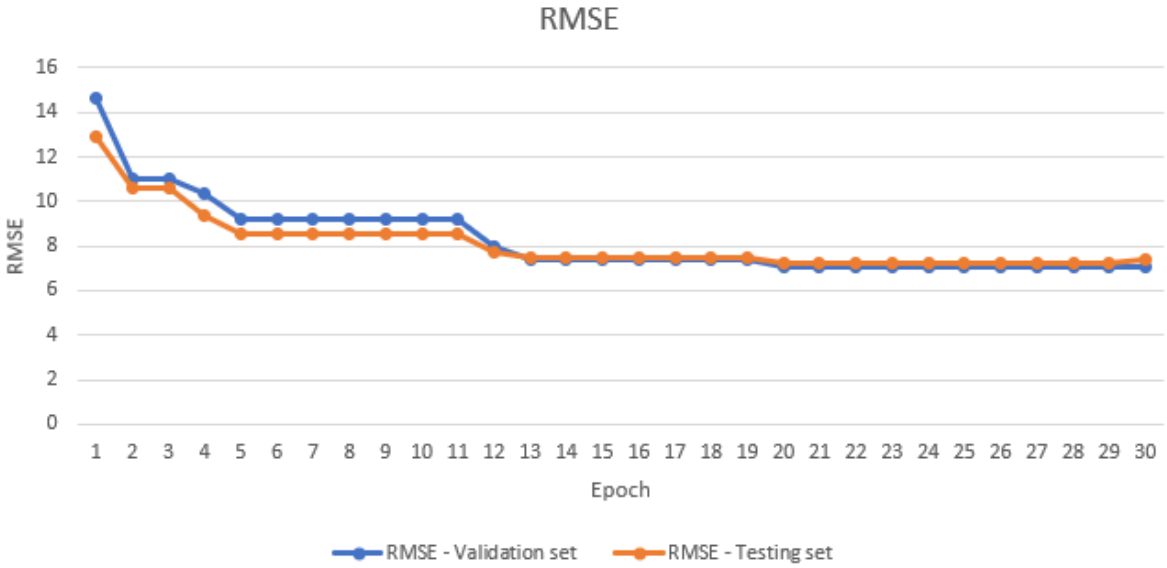


Figure 76. RMSE plotted as a function of epochs

4.3.3.1 Blocks

The MAE and RMSE values with regards to the experimental architectural layout of the blocks can be seen in Figure 77. For the mean absolute error, the results show that there are almost no improvements when looking at the validation set, and for the testing set instead an increase can be noticed. The RMSE value for the validation set increased slightly for the first version of the experimental block layout and decreased marginally for the second version of the block layout, similarly for the testing set.

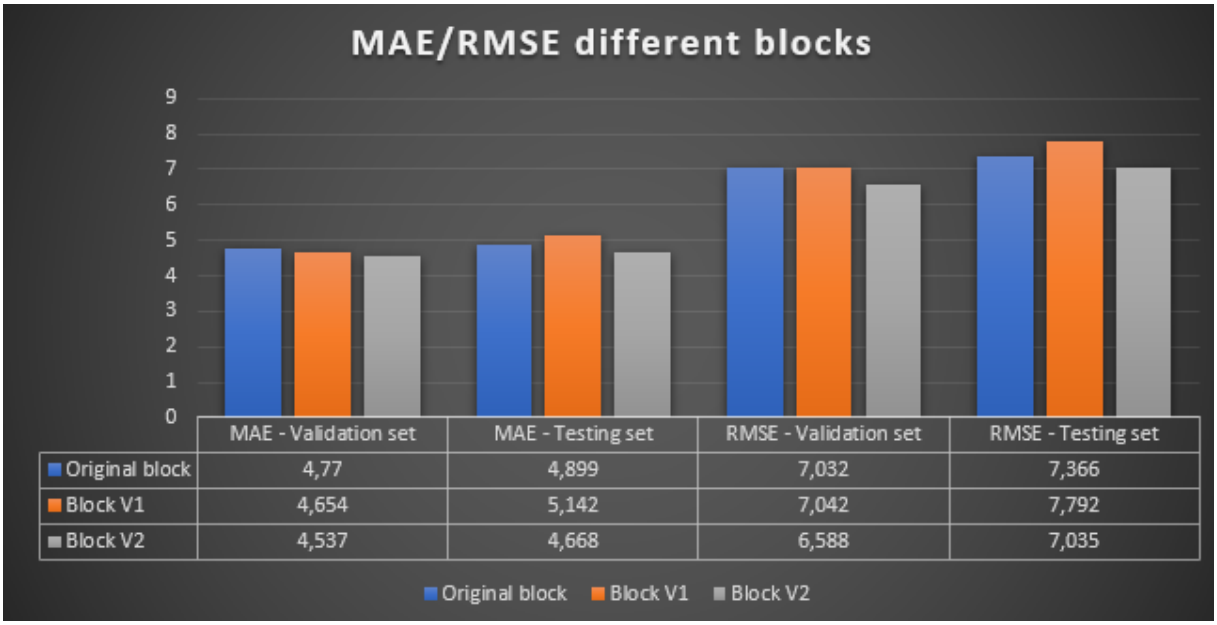


Figure 77. MAE/RMSE for block sizes

Figure 78 shows the percentage improvements for both the first- and second version of the block layouts. A negative percentage means that the metric has instead increased compared to the error rate for the original block layout. The best performing block layout was the second version which showed 5 % improvements in terms of MAE for both sets and a 6 % improvement for the validation set and 4 % improvement for the testing set when it comes to the RMSE value.

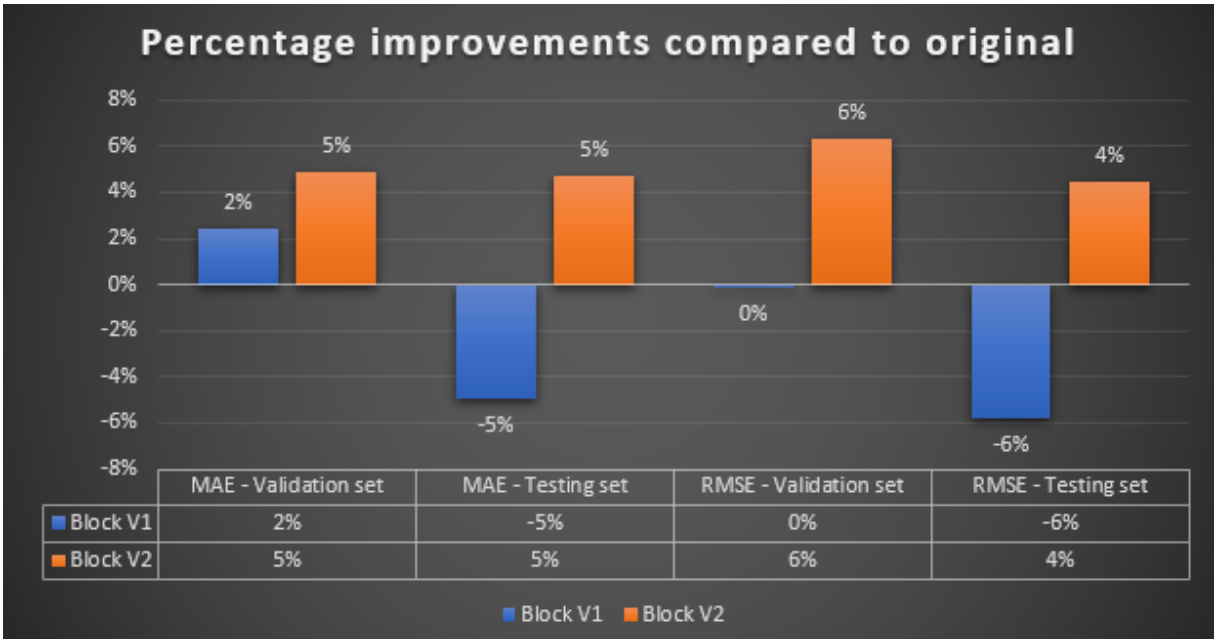
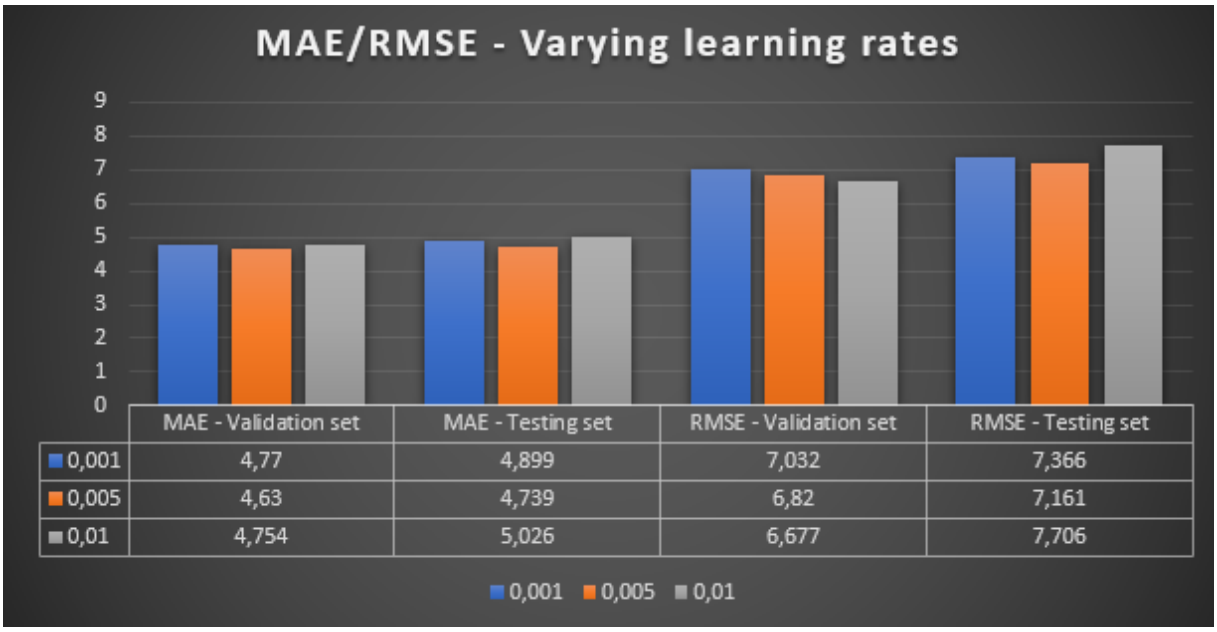


Figure 78. Percentage Improvement compared to original block layout

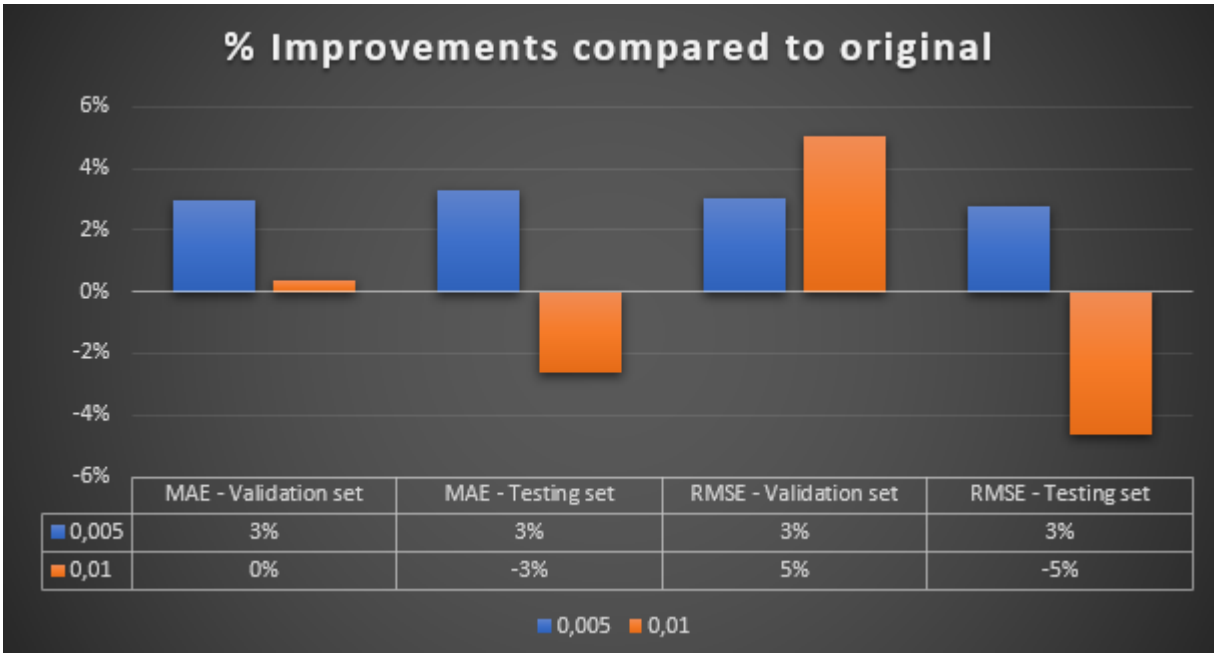
4.3.3.2 Learning rate

Figure 79 highlights how the MAE and RMSE varies with the experimental learning rates and Figure 80. shows the percentage of improvement compared to the original learning rate for the neural network. It can be seen that the improvements are almost negligible for both validation- and the testing set for both metrics, where there are even some increases in the error metrics instead.



	MAE - Validation set	MAE - Testing set	RMSE - Validation set	RMSE - Testing set
0,001	4,77	4,899	7,032	7,366
0,005	4,63	4,739	6,82	7,161
0,01	4,754	5,026	6,677	7,706

Figure 79. MAE/RMSE for varying learning rates



	MAE - Validation set	MAE - Testing set	RMSE - Validation set	RMSE - Testing set
0,005	3%	3%	3%	3%
0,01	0%	-3%	5%	-5%

Figure 80. Percentage Improvement compared to original learning rate

5. Discussion

In the results, it can be seen that the spatial-temporal graph neural network algorithm shows promising results with relatively high accuracy for the traffic demand. There is a myriad of different factors and variables that influence the accuracy of the neural network: architectural layout, complexity, quality of data and hyperparameters such as batches, epochs, learning rate, optimizers and loss functions. These elements are vital to take into consideration when assessing the performance of an artificial neural network. As previously seen, the analysis was conducted for 50, 100 and 200 stations and was done so in order to see how the accuracy of the neural network would fluctuate with larger quantity of data used. The result was structured in a way that simplified the comparison process and made it clear how the demand was for every station at specific times during the day compared to the predicted demand computed by the neural network. The graphs were able to highlight how the demand for the bike stations increased/decreased throughout the day and the difference between the actual demand and predicted demand.

5.1 Model error analysis

5.1.1 Error graphs

The purpose of introducing these graphs was to highlight the difference between actual demand and predicted demand and how close the predictions actually are from the neural network. While the actual hourly demand was showed for daily demand, the actual vs. predicted demand comparison will be for the hourly demand. This will give a better overview of the difference between the computed bike demand by the neural network and the real demand. For the 50-station analysis between 55-68 % of the predictions were within 100 bikes or closer to the actual demand, for the 100-station analysis approximately 53-70% were within the same range and for the 200-station analysis between 74-84% of the predictions had a difference of less than 100 bikes. As previously mentioned, this is just a simple metric that allows for simplified comparison between actual- and predicted demand and showcases how close the predictions computed by the algorithm is.

5.1.2 Patterns

When it comes to patterns in the given data and the predicted demand by the algorithm, some flaws can be seen. Some of the data, presented in the graphs for the actual demand show that for some days the bike demand for some stations suddenly drop to zero, which was a reoccurring pattern for some stations most likely influenced the accuracy of the predictions. One more pattern that could be seen when looking at the comparison between the actual- and predicted demand is that the neural network was very inaccurate for the final hour of the day for many stations where the model very often predicted significantly higher demand then what was.

5.1.3 Evaluation metrics

As previously mentioned, the purpose of using the error metrics MAE and RMSE is to measure the accuracy of the neural network by analyzing the magnitude of the difference between actual- and predicted demand. For the 50-station analysis with the base architectural layout and settings, the neural network exhibited MAE values that were quite low and close relative to each other (6.985 for validation set and 7.24 for testing set). For the 100-station analysis the analysis showed a MAE value for the validation set equal to 6.9 and approximately 6.7 for the testing set. Finally, for the 200-station analysis a noticeable increase in the MAE values can be seen where the MAE for the validation set decreased to 4.77 and 4.9 for the testing set. Determining a good MAE value can be difficult and is highly relative to the dataset used for the analysis. The mean absolute error simply highlights the absolute value of the difference between the actual demand and the predicted demand by the neural

network on average. For example, for the 200-station analysis, the predictions computed by the neural network are on average 4.77-4.9 bikes from the actual demand, which can be seen as very accurate depending on the demand for every specific hour. If the demand is in the 100s and the model is able to compute predictions that are that close to the real demand then the networks is indeed highly accurate, however if the demand is for example in the 20s then the model is not that accurate.

When it comes to the RMSE values, similar conclusions can be drawn. As previously stated, the RMSE value is simply an error metric that quantifies the predictions errors by using standard deviation. Similar to the reasoning behind finding a "good" MAE value, establishing a good RMSE value is problematic and is relative to the dataset and dependent on the variables.

Experimenting with the block layout and increasing the channels in the neural network only showed small improvements in the error metrics and most notably resulted in an increased computing time. The most significant improvement in the error metric when implementing both versions of the block layout came for the 50-station analysis. Similarly, when experimenting with the learning rates the most improvement also came for the 50-station analysis.

Finally, determining if the algorithm is accurate can be done by analyzing if the model is overfitting or underfitting. Discovering whether or not the model is indeed overfitting or underfitting can be established when analyzing the error metrics, which are in this case MAE and RMSE. If the model is performing well on the validation set but has a large error on the testing set then the model is overfitting. If the model has poor performance on both validation- and testing set then the model is instead underfitting. So, finding a balance and subsequently finding a good fit for the neural network is vital. Looking at the Figures showcasing the MAE and RMSE values for the all the station analysis shows that the error metric for both the validation- and testing set converge and are very close relative to each other, meaning that the model has a robust fit and is well-fitted.

6. Conclusion

The aim of this study was to design a graph neural network that has the ability to use graph structured data and take both spatial- and temporal aspects into consideration when predicting the high-resolution demand of shared mobility systems. A case study about bike-sharing systems in Nanjing, a large city in China is conducted based on the proposed method. The results show that the algorithm is able to predict the short-term bike demand with relatively high accuracy and low computing time. The results show that the graph neural network is able to predict traffic demand in an accurate way with MAE/RMSE values that are quite good. The predicted errors for the hourly usage demand prediction of a station are often within 20 bikes. By utilizing station-to-station data combined with time data, the algorithm could accurately predict short-term bike demand. The hyperparameters are modified in order to see how the accuracy fluctuates, and the efficiency of the model is affected. This included changing the learning rate and layout for the blocks. The results vary with hyperparameters, the learning rate that showed the best results was 0.005 compared to the original 0.01. For the block layout, only small improvements were made for both versions of experimental blocks in the 50-station analysis, while the most improvements were made in the 100-station analysis.

This study focuses on the short-term prediction for bike-sharing, which is a base layer for future studies. This base layer can be used to evaluate the transportation accessible in terms of congestion and roads. Based on this study, the major flow patterns can be evaluated, and further development of the transportation routes to make it easier to use a bike in the city of Nanjing. Due to the high populated urban area, it is very hard to build new infrastructure on a bigger scale. Therefore, the bike-sharing transportation system is a good choice in these terms. The study can also be used for a comparison of different transportation systems in Nanjing to evaluate what transportation system is the most beneficial and effective to use. Using flow patterns and data that are easy to analyze. The study and method can also be applied to other transportation sectors, such as road transportation in Nanjing and other cities. The method is the same for all cases, and only the data is needed to be changed. Using the results of this study, a demand prediction map for all the stations in Nanjing can be created. This will result in a good understanding of the bike usage patterns, which will be very useful for the city in terms of finding an acceptable and good balance in the stations. Unfortunately, due to time constraints, the analysis was limited to 200 stations, where it would have been interesting to see how the accuracy of the algorithm would be affected with more stations analyzed. Something that would also be interesting to see would be to implement meteorological conditions into the algorithm, which is a large impacting factor when it comes to biking. This would make the algorithm become one level higher in terms of complexity but would significantly increase the accuracy of the neural network. Human behavior is not considered in this study in terms of weather, temperature, location, and year. The study does not consider how this could have affected the result and changed the outcome. The travel path for the bikes is not considered, and only the station-to-station data are used in this study. These are all potential places that can be improved

7. References

United Nations. (2018, May). *World Urbanization Prospects: the 2018 Revision*. https://www.un.org/en/development/desa/population/theme/urbanization/index.asp

Wu, F., & He, S. (2005). *Changes in traditional urban areas and impacts of urban redevelopment: A case study of three neighbourhoods in Nanjing, China*. Tijdschrift voor economische en sociale geografie, 96(1), 75-95. https://d1wqtxts1xzle7.cloudfront.net/45747532/Changes_in_Traditional_Urban_Areas_and_I20160518-6255-hgyptn-with-cover-page-v2.pdf?Expires=1658148349&Signature=TDh-iGiFB9uCPQchWtLxk8EI4k-T1Au6WLJ1xaY30S2Bqx3z6y75R~XQZpzhr4gULb3xXmuf1goS7c9V4TGA2NMqie7XyEbIe4koa5H1CvS8sb9wWcXqC5b4cgXPszAa5I24PPdfV3Y0pACtY2VpEqWlYDnXwjPthnUN0cX7XTzu6iOu0SuUdlmhXIUNS0tPTHF~Dfyv1cwQYJgs-qY8X3JClLzPt74JRSWNx3rjNzcW6~v2jKaQ0OuxcaUcVJcCHsmuO2RDLMhXdqaVRO2R1NlweKawKiu~cz8XHRdXgFr2is4nCqri9DwFaZLxWDK4bHx2H0MzEH59VPKOcyqSbg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Darren, F. (2010, January). National Household Travel Survey—short trips analysis 2010. *The League of American Bicyclists.* https://www.bikeleague.org/content/national-household-travel-survey-short-trips-analysis

Fuller, D., Gauvin, L., Kestens, Y., Daniel, M., Fournier, M., Morency, P., & Drouin, L. (2011, July). *Use of a New Public Bicycle Share Program in Montreal, Canada*. American Journal of Preventive Medicine, 41(1), 80-83. https://walkabilly.ca/home/wp-content/uploads/2015/01/2011_Fuller_PM_BIXIBaseline.pdf

Razavian, N., Knoll, F., & Geras, K. J. (2020, February). *Artificial intelligence explained for nonexperts*. Thieme Medical Publishers, *(Vol. 24, No. 01, pp. 003-011).* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7393604/

Chan, H. P., Samala, R. K., & Hadjiiski, L. M. (2019, December). *CAD and AI for breast cancer—recent development and challenges*. The British journal of radiology, 93(1108). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7393604/

Dunjko, V., & Briegel, H. J. (2018). *Machine learning & artificial intelligence in the quantum domain*. Reports on Progress in Physics. https://scholarlypublications.universiteitleiden.nl/access/item%3A2970880/view

Russell, S.J., & Norvig, P. (2010). *Artificial Intelligence, A Modern Approach (third edition).* https://zoo.cs.yale.edu/classes/cs470/materials/aima2010.pdf

Wirth, N. (1971). *The programming language Pascal*. Acta informatica. http://www.standardpascaline.org/The_Programming_Language_Pascal_1970.pdf

Berry, M. W., Mohamed, A., & Yap, B. W. (Eds.). (2019). *Supervised and unsupervised learning for data science*. Springer Nature. https://link.springer.com/chapter/10.1007/978-3-030-22475-2_1

Haldorai, A., Ramu, A., & Suriya, M. (2020). *Organization internet of things (IoTs): supervised, unsupervised, and reinforcement learning. In Business Intelligence for Enterprise Internet of Things.* Springer.
https://www.researchgate.net/profile/Madhumala-R-B/publication/342183126_Analysis_of_virtual_Machine_placement_and_optimization_using_Swarm_intelligence/links/5f180b2145851515ef3e48ec/Analysis-of-virtual-Machine-placement-and-optimization-using-Swarm-intelligence.pdf#page=38

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). *Reinforcement learning: A survey*. Journal of artificial intelligence research. https://www.jair.org/index.php/jair/article/view/10166

Li, Y. (2017). *Deep reinforcement learning: An overview*.
https://arxiv.org/pdf/1701.07274.pdf?source=post_page----------

Ghahramani, Z. (2003, February). *Unsupervised learning*. Springer.
https://link.springer.com/chapter/10.1007/978-3-540-28650-9_5

Glorennec, P. Y. (2000). *Reinforcement learning: An overview*.
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.4135&rep=rep1&type=pdf

Garnier, P., Viquerat, J., Rabault, J., Larcher, A., Kuhnle, A., & Hachem, E. (2021). *A review on deep reinforcement learning for fluid mechanics.* https://arxiv.org/pdf/1908.04127.pdf

Brunette, E. S., Flemmer, R. C., & Flemmer, C. L. (2009). *A review of artificial intelligence*. Massey University.
https://www.researchgate.net/profile/Claire-Flemmer/publication/220774315_A_review_of_artificial_intelligence/links/00b7d5331df59bcbf4000000/A-review-of-artificial-intelligence.pdf

Oke, S. A. (2008). *A literature review on artificial intelligence*. University of Lagos.
https://www.researchgate.net/profile/Sunday-Oke/publication/228809837_ARTIFICIAL_INTELLIGENCE_A_REVIEW_OF_THE_LITERATURE/links/5eec7244a6fdcc73be8961c6/ARTIFICIAL-INTELLIGENCE-A-REVIEW-OF-THE-LITERATURE.pdf

Korf, R. E. (1999). *Artificial intelligence search algorithms*. University of California.
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.1135&rep=rep1&type=pdf

Caruana, R., & Niculescu-Mizil, A. (2006). *An empirical comparison of supervised learning algorithms*. Cornell University. https://dl.acm.org/doi/abs/10.1145/1143844.1143865

Cunningham, P., Cord, M., & Delany, S. J. (2008). In Machine learning techniques for multimedia. *Supervised learning* (pp. 21-49). Springer.
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.457.4869&rep=rep1&type=pdf

Van Engelen, J. E., & Hoos, H. H. (2020). *A survey on semi-supervised learning*. Springer.
https://link.springer.com/article/10.1007/s10994-019-05855-6

Zhu, X. (2005). *Semi-supervised learning literature survey*. University of Wisconsin-Madison. https://minds.wisconsin.edu/bitstream/handle/1793/60444/TR1530.pdf?sequence=1

Rico, J., Barateiro, J., & Oliveira, A. (2021). *Graph Neural Networks for Traffic Forecasting*. University of Lisbon. https://arxiv.org/pdf/2104.13096

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y. (2014). Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2), 865-873. http://mocom.xmu.edu.cn/home/project/trajectory/2%20Analysis%20&%20Prediction%20Methods/Traffic%20Flow%20Prediction%20With%20Big%20Data%20%20A%20Deep%20Learning%20Approach.pdf

From University of Cincinnati, (n.d). *Feedforward Deep Learning Models*. http://uc-r.github.io/feedforward_DNN

Wang, W., & Lu, Y. (2018, March). Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model. In *IOP conference series: materials science and engineering* (Vol. 324, No. 1, p. 012049). IOP Publishing. https://iopscience.iop.org/article/10.1088/1757-899X/324/1/012049/pdf

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://s3.us-east-2.amazonaws.com/hkg-website-assets/static/pages/files/DeepLearning.pdf

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, *187*, 27-48. https://deeplearning.ir/wp-content/uploads/2016/07/Deep-learning-for-visual-understanding_-A-review.pdf

Zhang, T., Li, Y., Li, Y., Sun, S., & Gao, X. (2020). *A self-adaptive deep learning algorithm for accelerating multi-component flash calculation*. https://www.researchgate.net/profile/Tao-Zhang-165/publication/342120411_A_self-adaptive_deep_learning_algorithm_for_accelerating_multi-component_flash_calculation/links/5ee89cd3458515814a629ca7/A-self-adaptive-deep-learning-algorithm-for-accelerating-multi-component-flash-calculation.pdf

Pragati Baheti, V7labs. (25th May 2022). *12 types of neural network activation functions: how to choose?.* V7labs. https://www.v7labs.com/blog/neural-networks-activation-functions#activation-function

DanB, Kaggle. (2018). *Rectified Linear Units (ReLU) in Deep Learning.* Kaggle. https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook

Suzuki, K. (Ed.). (2011). *Artificial neural networks: methodological advances and biomedical applications*. BoD–Books on Demand. https://books.google.se/books?hl=en&lr=&id=JuaODwAAQBAJ&oi=fnd&pg=PR11&dq=Suzuki,+K.+(Ed.).+(2011).+Artificial+neural+networks:+methodological+advances+and+biomedical+applications.+BoD–Books+on+Demand.&ots=XVkTWR7M_a&sig=kxuZJnLYmlROrt844kf4-fU9O64&redir_esc=y#v=onepage&q=Suzuki%2C%20K.%20(Ed.).%20(2011).%20Artificial%20neural%2

0networks%3A%20methodological%20advances%20and%20biomedical%20applications.%20BoD–Books%20on%20Demand.&f=false

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). In Recent Trends and Advances in Artificial Intelligence and Internet of Things. *Fundamental concepts of convolutional neural network* (pp. 519-567). Springer. https://link.springer.com/chapter/10.1007/978-3-030-32644-9_36

Benjamin, S., Emily, R., Adam, P., Alexander B. (2021). *A Gentle Introduction to Graph Neural Networks*. Distill. https://distill.pub/2021/gnn-intro/

Isaac Computer Science. (N.D). *Graphs*. https://isaaccomputerscience.org/concepts/dsa_datastruct_graph?examBoard=all&stage=all

Sanghvi, R. (20 feb 2021). A Complete Guide to Adam and RMSprop Optimizer. *Medium*. https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be

From *Machine Learning: How to Prevent Overfitting* [Photography], by Hoffman, K., 2021, Medium. (https://medium.com/swlh/machine-learning-how-to-prevent-overfitting-fdf759cc00a9)

Beklemysheva, A. (n.d). Why Use Python for AI and Machine Learning? [Blog post]. Retrieved from https://steelkiwi.com/blog/python-for-ai-and-machine-learning/

Nasteski, V. (2017). *An overview of the supervised machine learning methods*. https://www.researchgate.net/profile/Vladimir-Nasteski/publication/328146111_An_overview_of_the_supervised_machine_learning_methods/links/5c1025194585157ac1bba147/An-overview-of-the-supervised-machine-learning-methods.pdf

8. Appendices

8.1 100-station analysis

8.1.1 Station 51-60

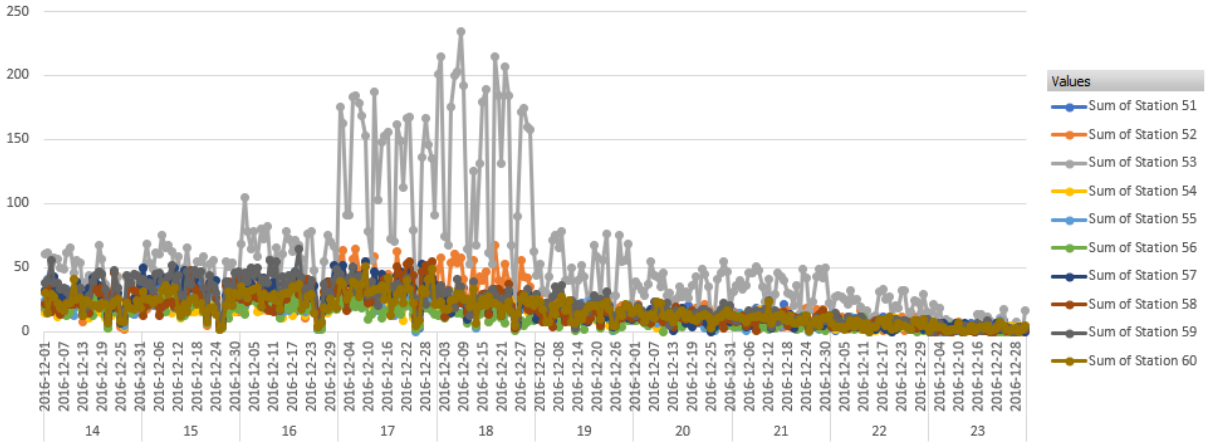


Figure 81. Actual hourly demand for station 51-60

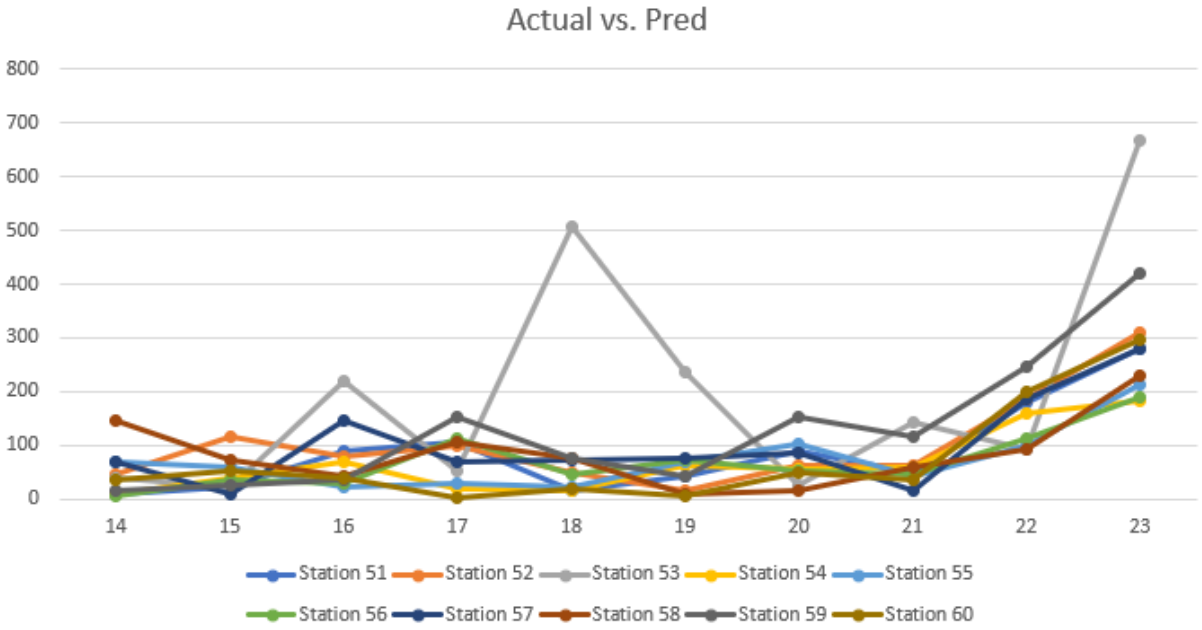


Figure 82. Difference between actual monthly demand and predicted demand

8.1.2 Station 61-70

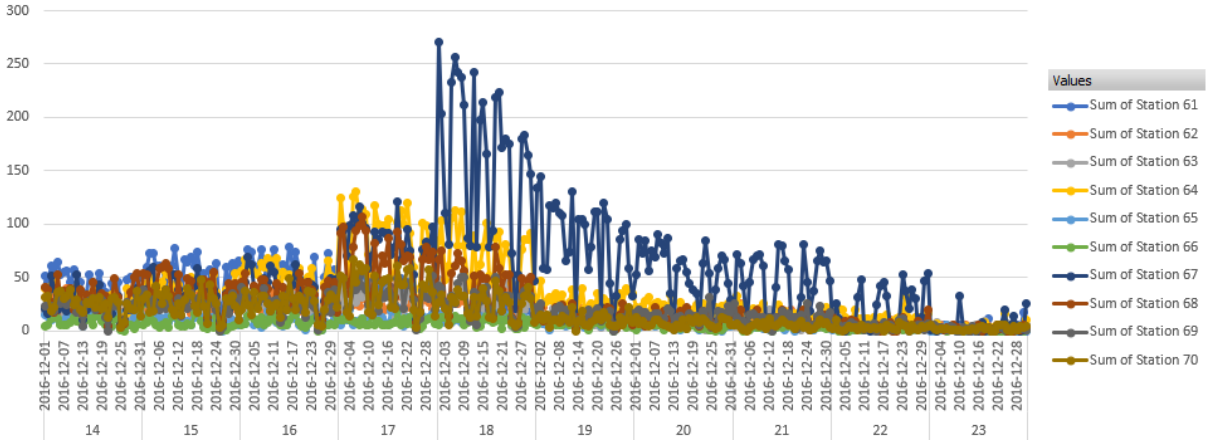


Figure 83. Actual hourly demand for station 61-70

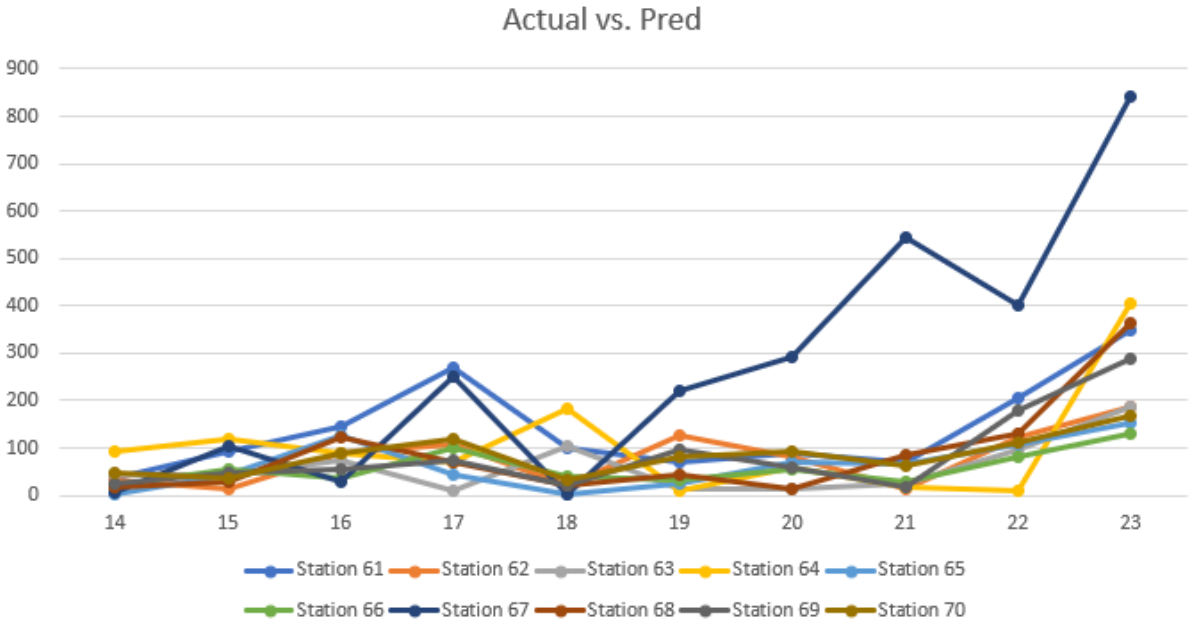


Figure 84. Difference between actual monthly demand and predicted demand

8.1.3 Station 71-80

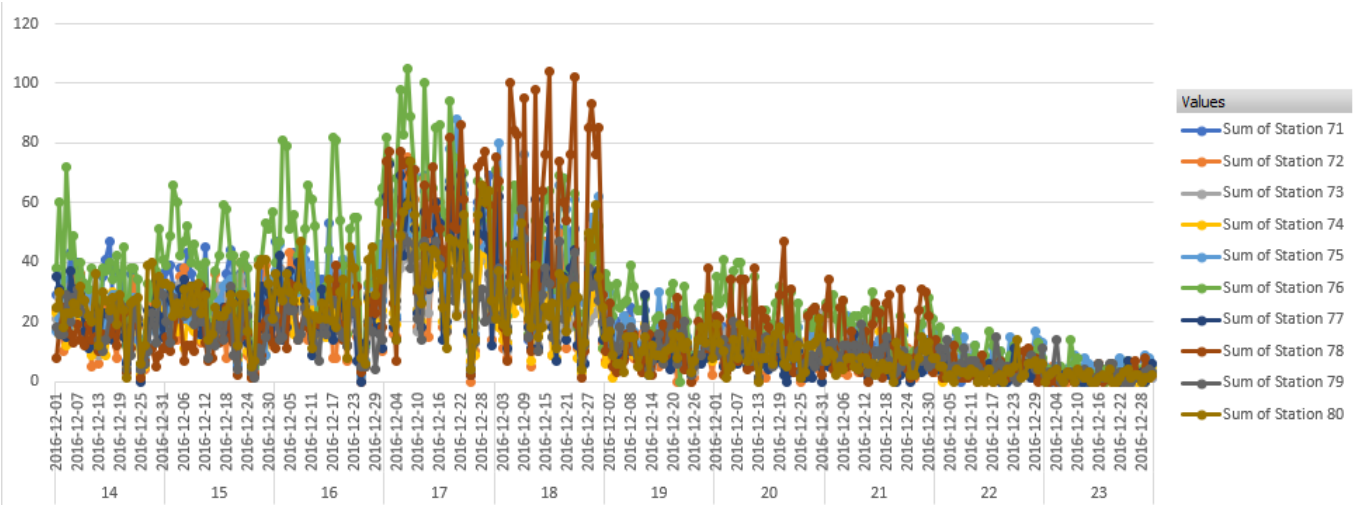


Figure 85. Actual hourly demand for station 71-80

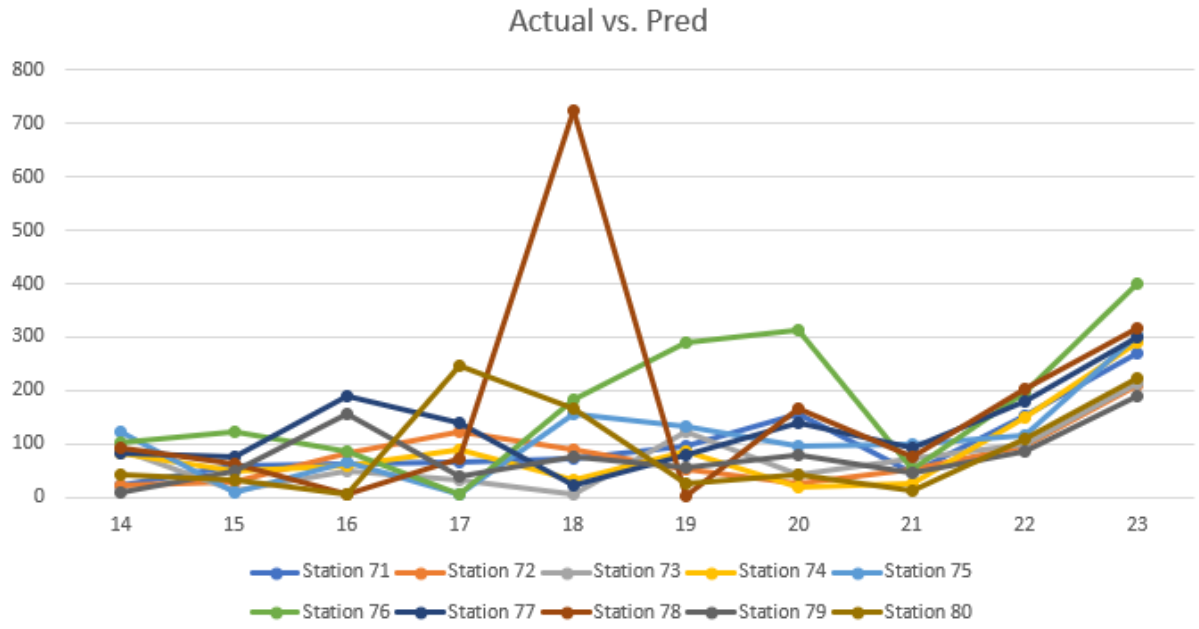


Figure 86. Difference between actual monthly demand and predicted demand

8.1.4 Station 81-90

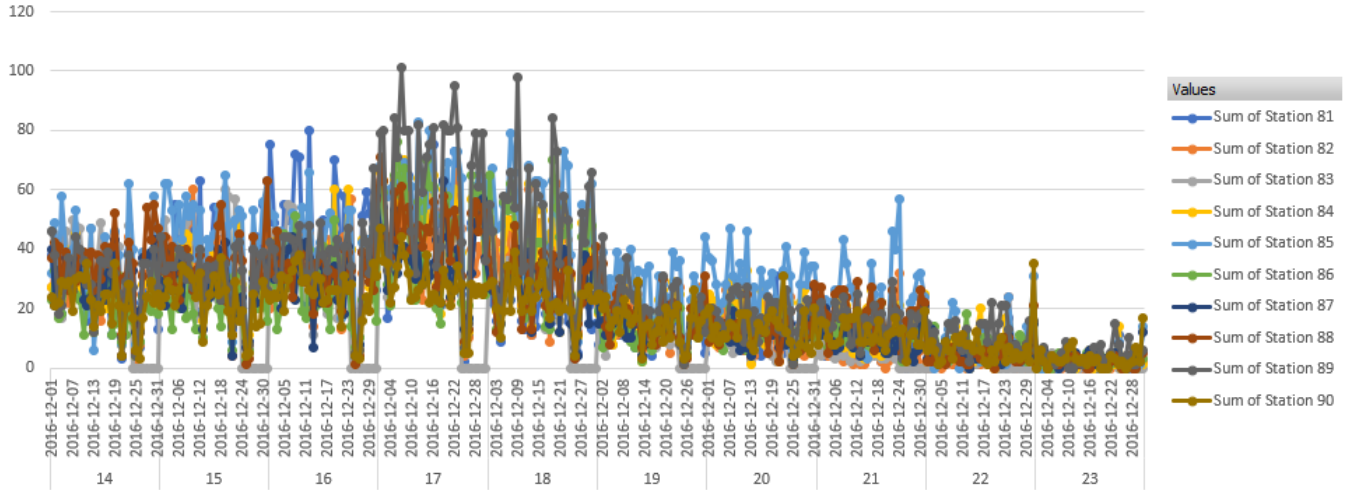


Figure 87. Actual hourly demand for station 81-90

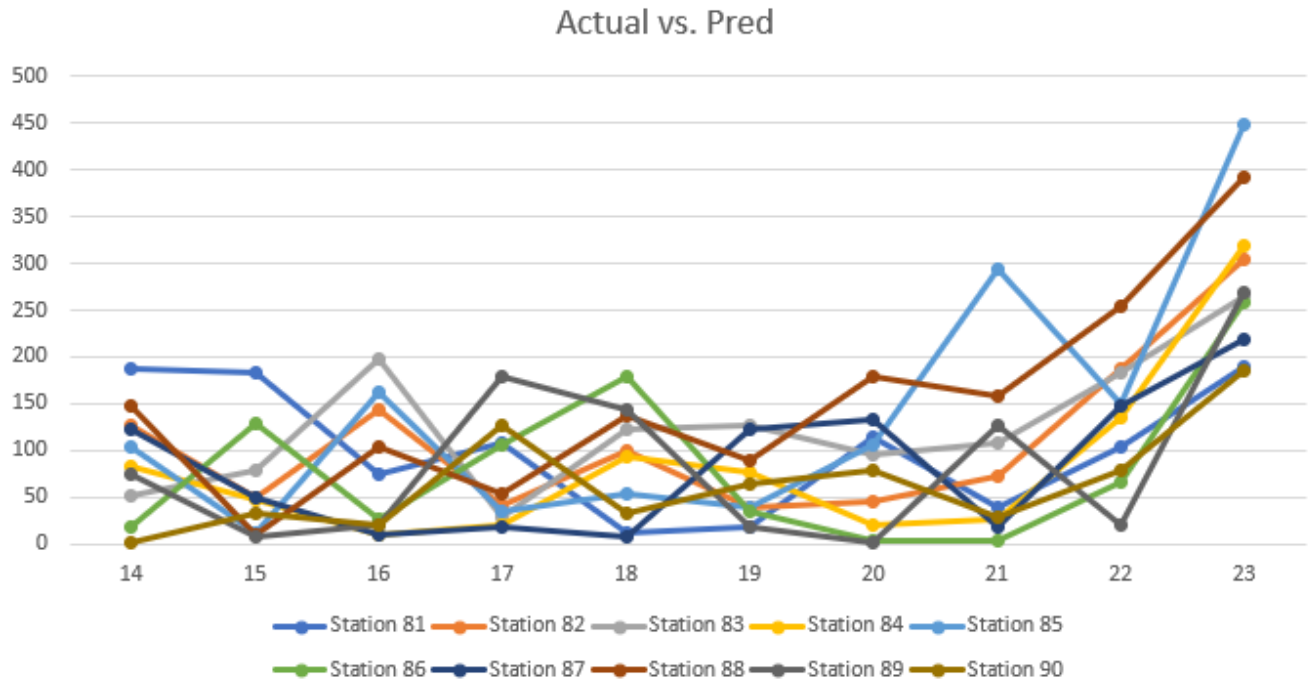


Figure 88. Difference between actual monthly demand and predicted demand

8.1.5 Station 91-100

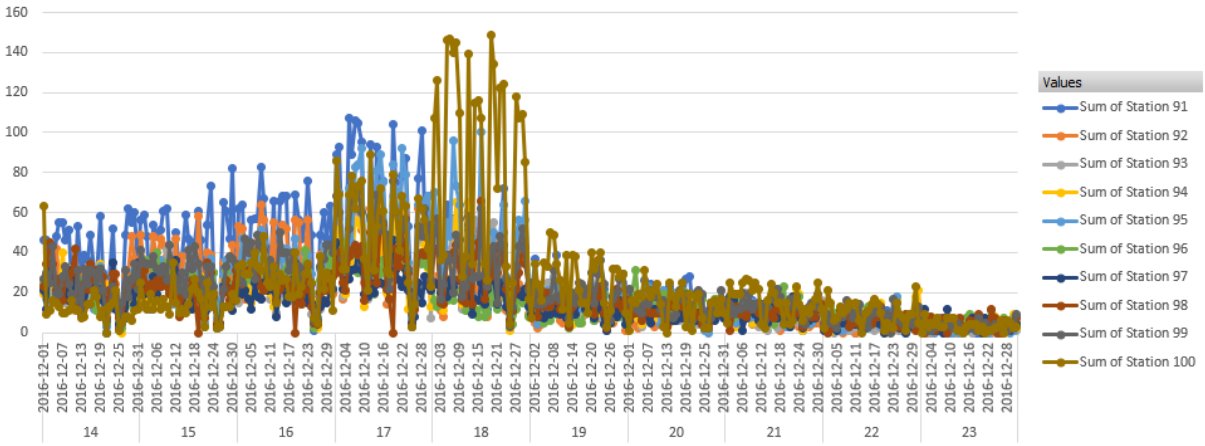


Figure 89. Actual hourly demand for station 91-100

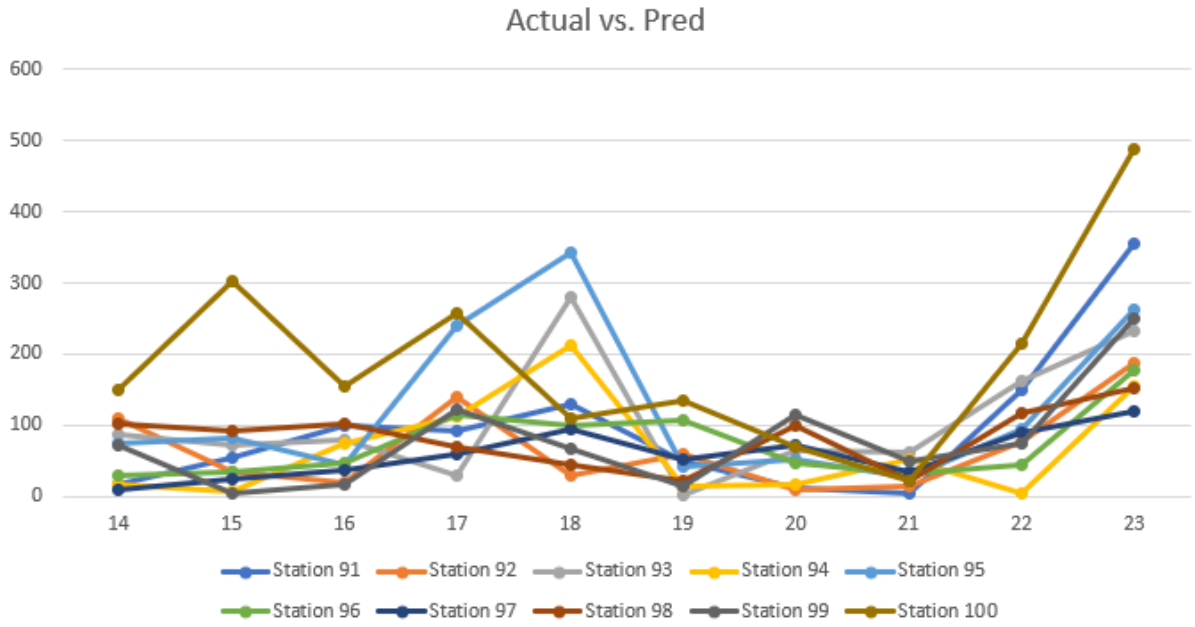


Figure 90. Difference between actual monthly demand and predicted demand

8.2 200-station analysis

8.2.1 Station 51-75

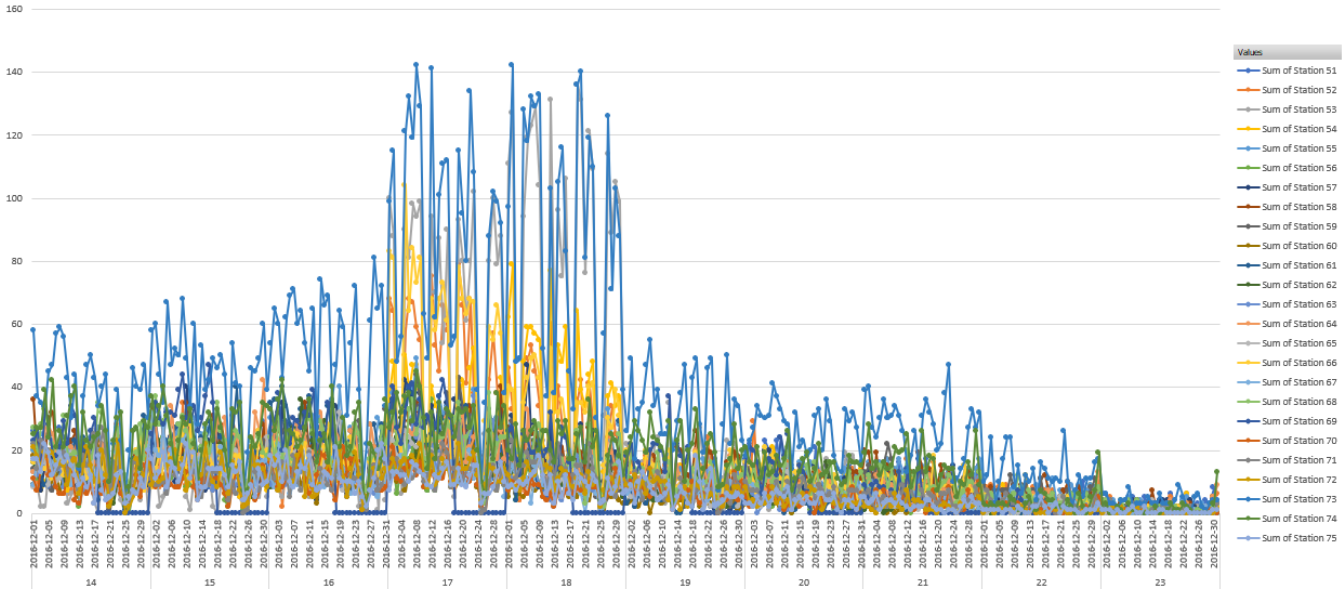


Figure 91. Actual hourly demand for station 51-75

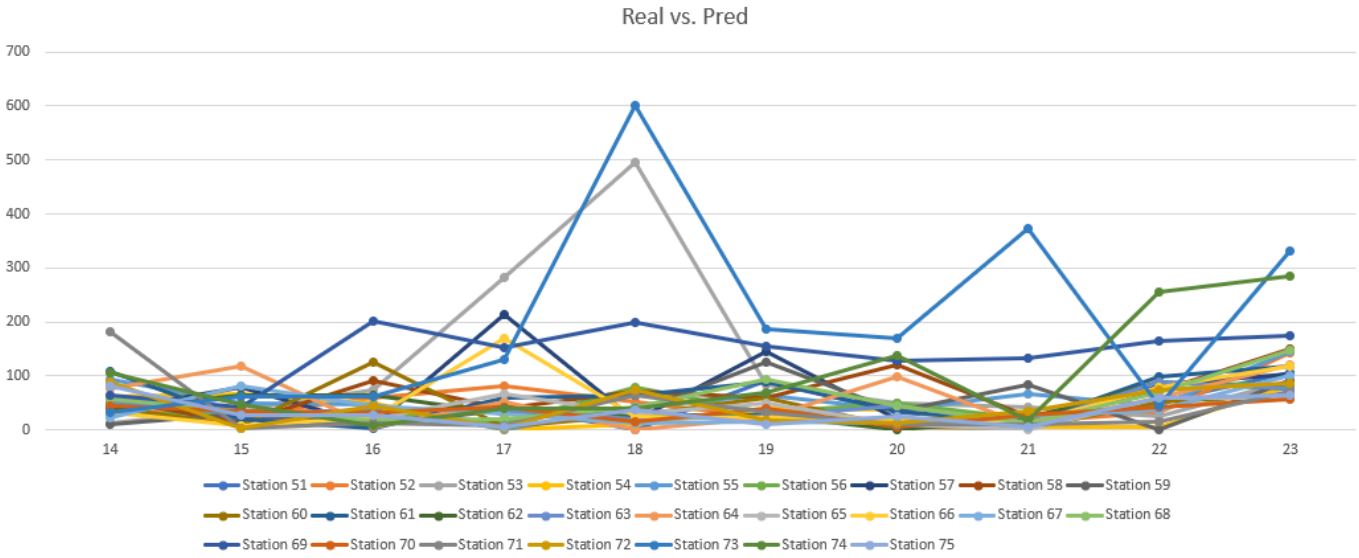


Figure 92. Difference between actual monthly demand and predicted demand

8.2.2 Station 76-100

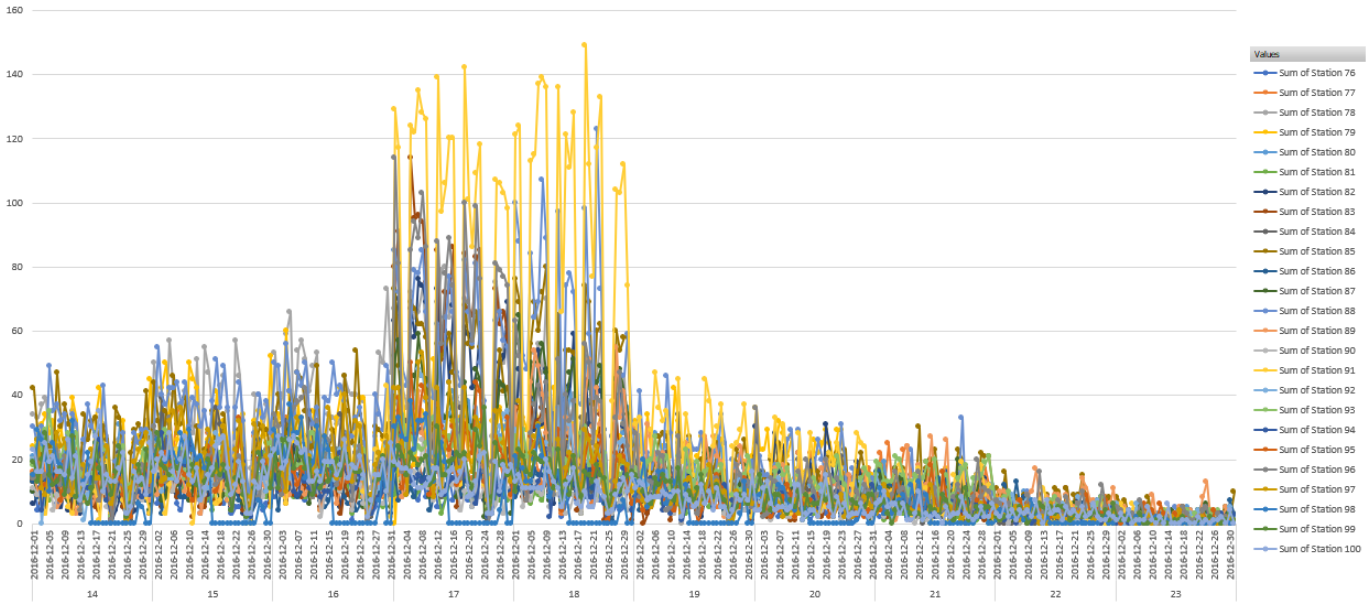


Figure 93. Actual hourly demand for station 76-100

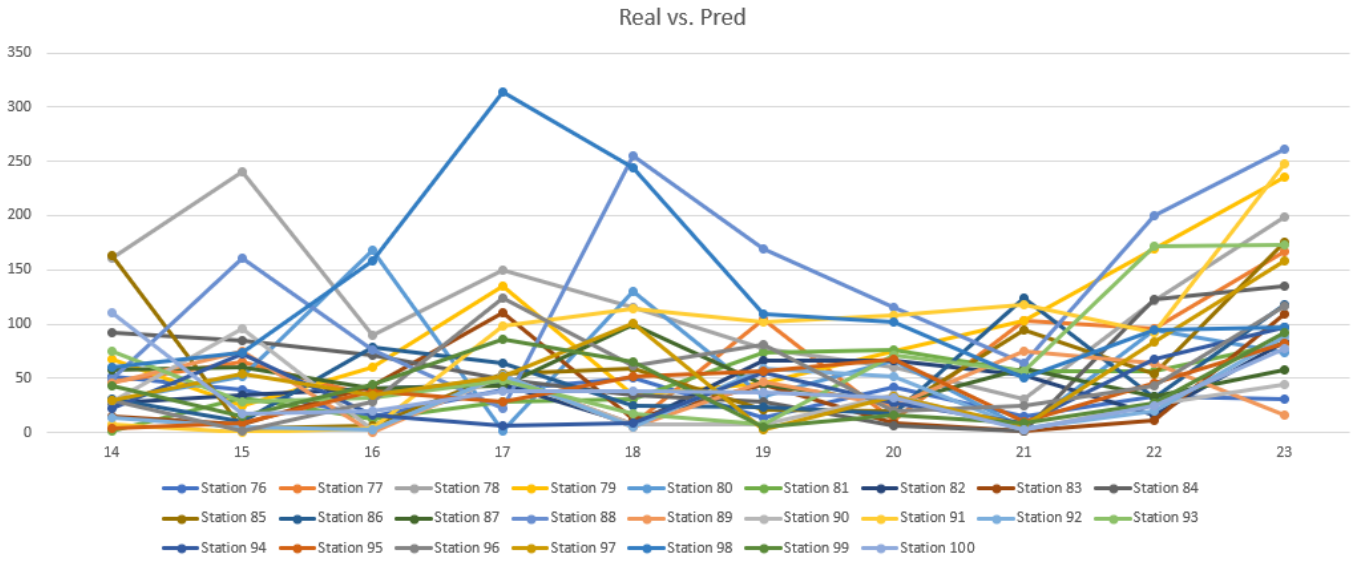


Figure 94. Difference between actual monthly demand and predicted demand

8.2.3 Station 101-125

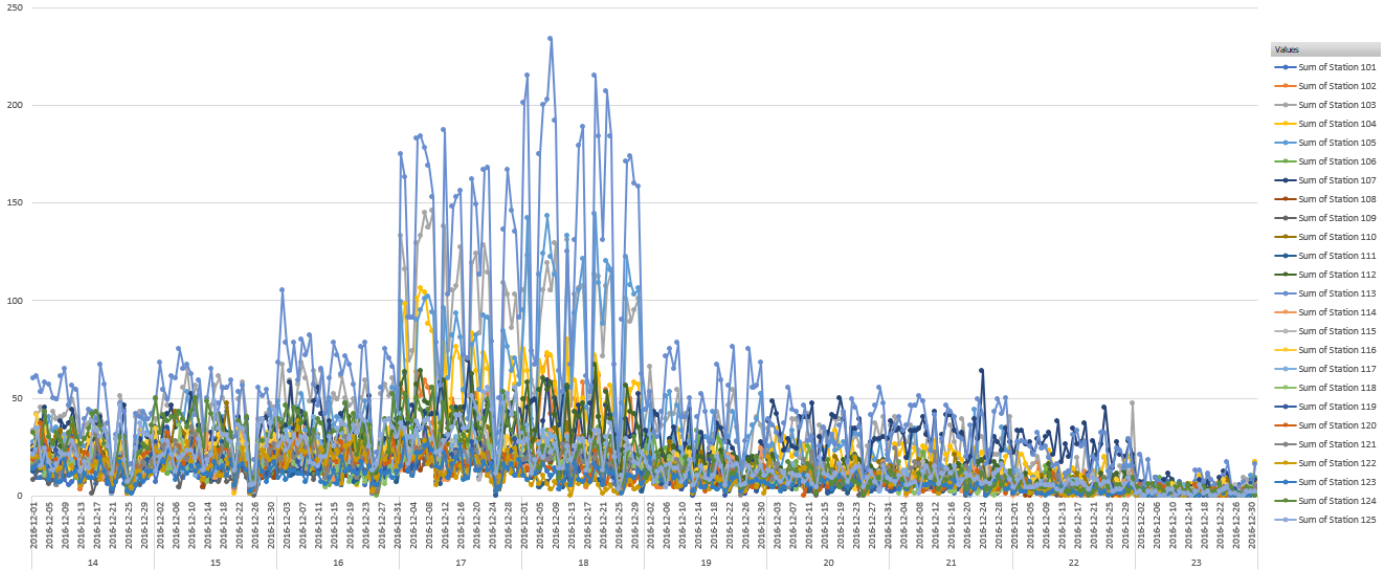


Figure 95. Actual daily hourly demand for station 101-125

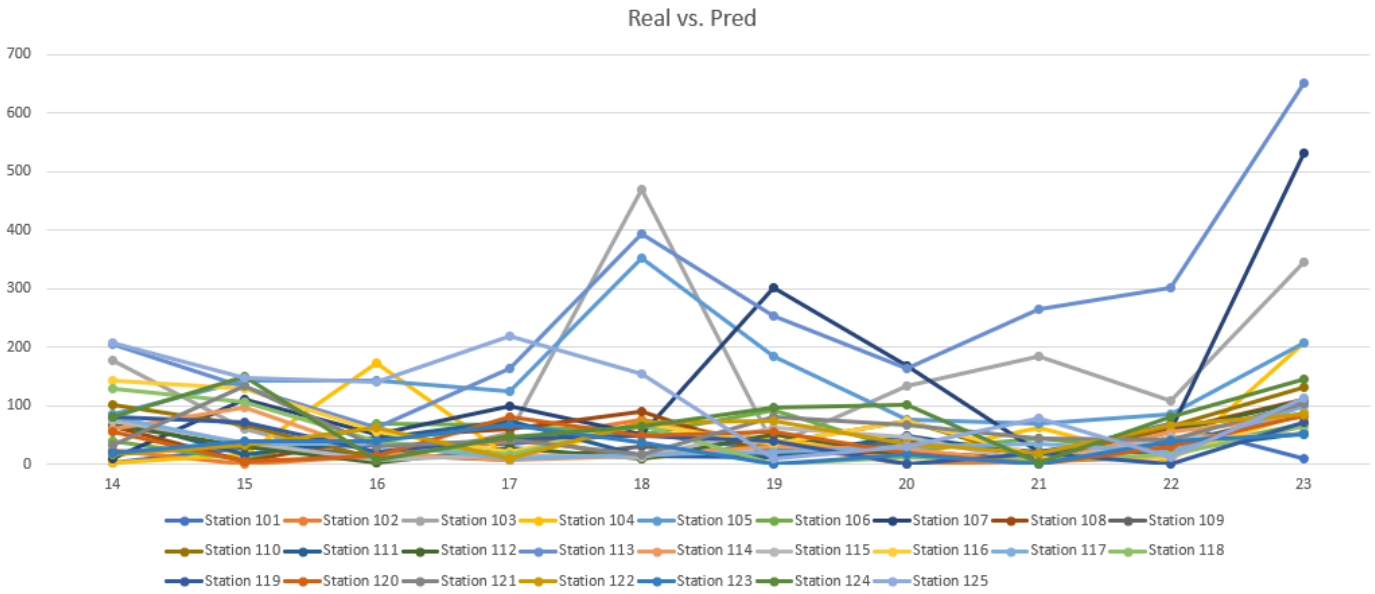


Figure 96. Difference between actual monthly demand and predicted demand

8.2.4 Station 126-150

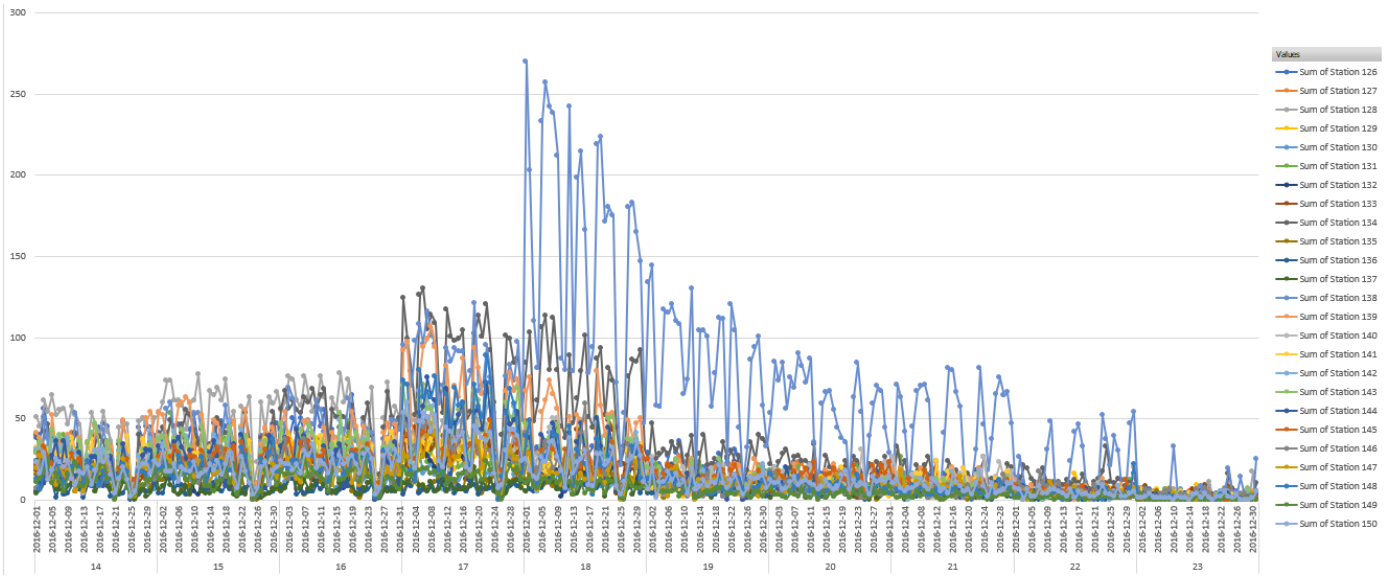


Figure 97. Actual daily hourly demand for station 126-150

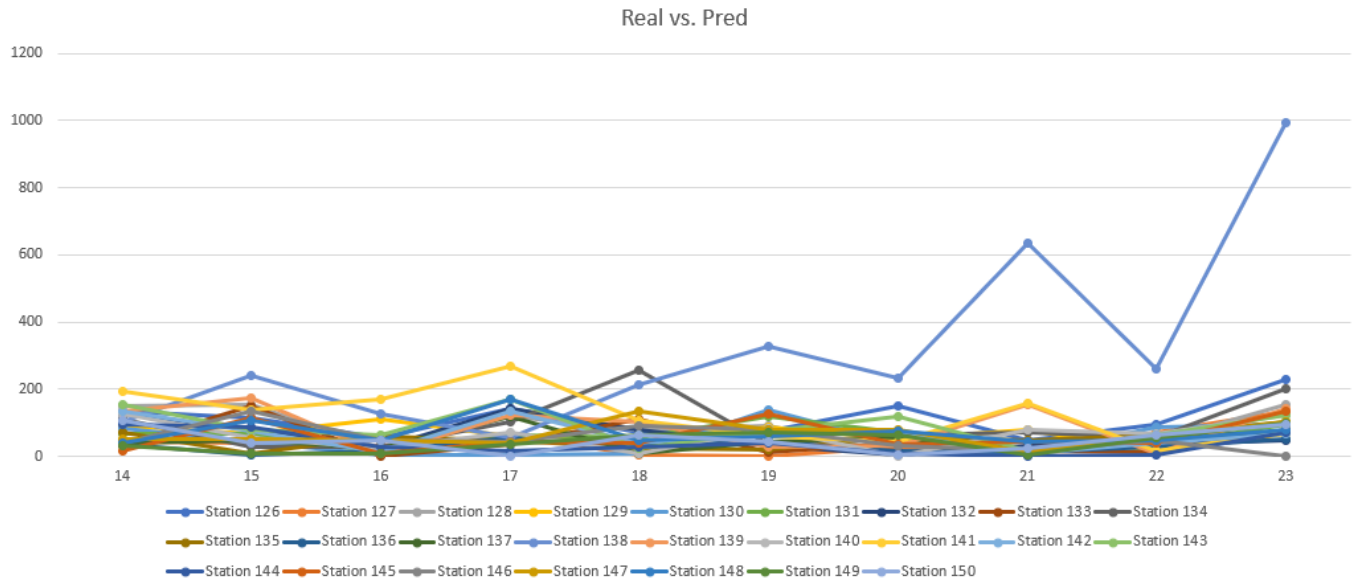


Figure 98. Difference between actual monthly demand and predicted demand

8.2.5 Station 151-175

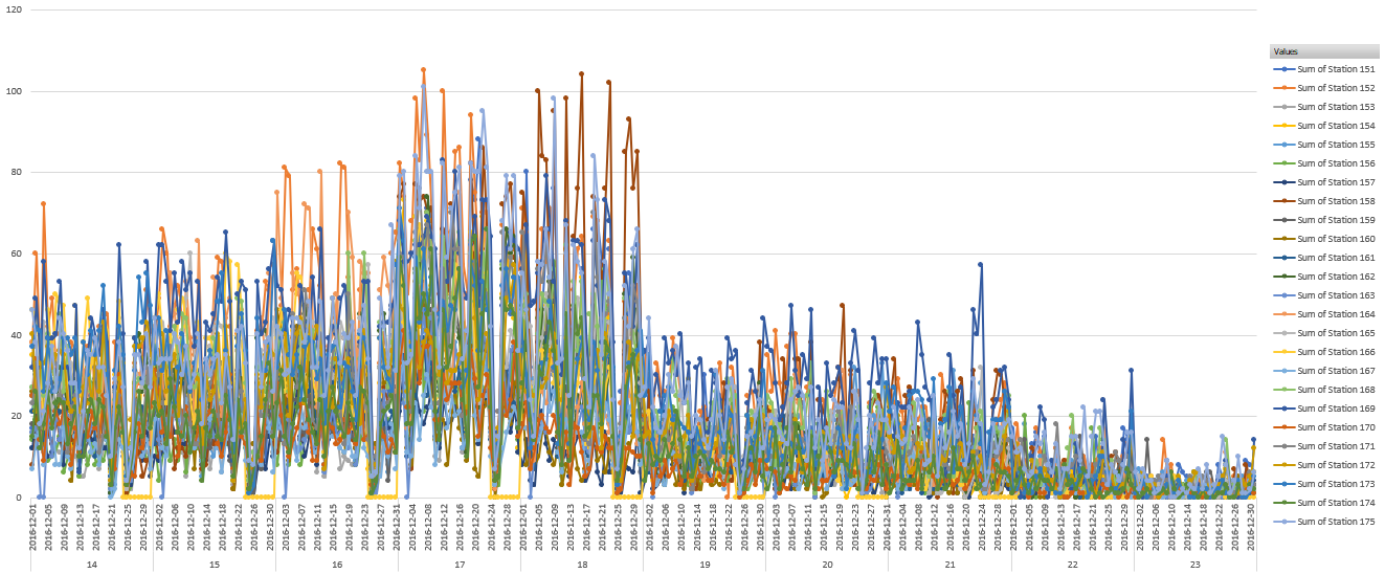


Figure 99. Actual daily hourly demand for station 151-175

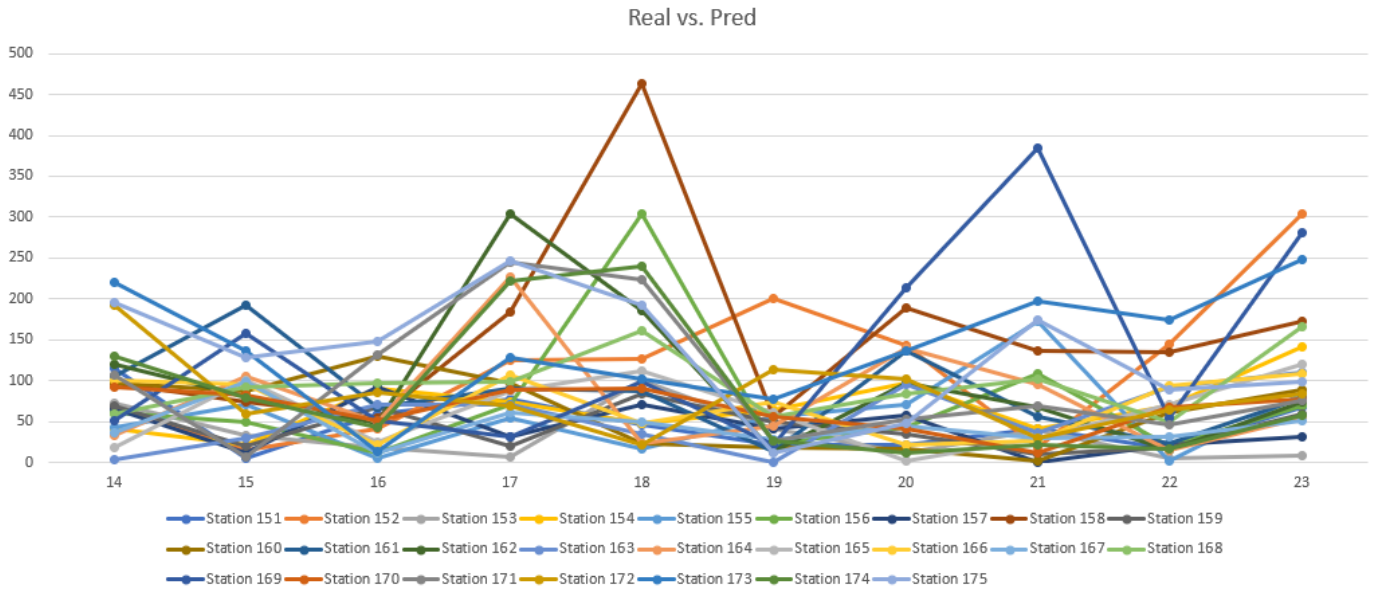


Figure 100. Difference between actual monthly demand and predicted demand

8.2.6 Station 176-200

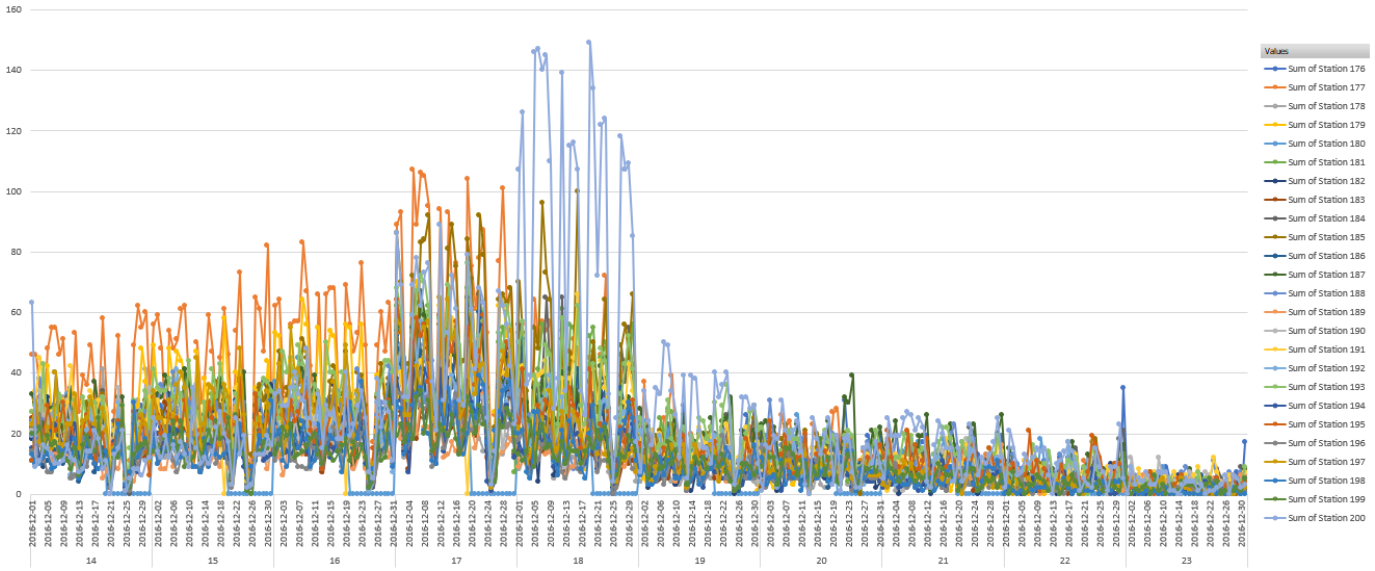


Figure 101. Actual daily hourly demand for station 176-200

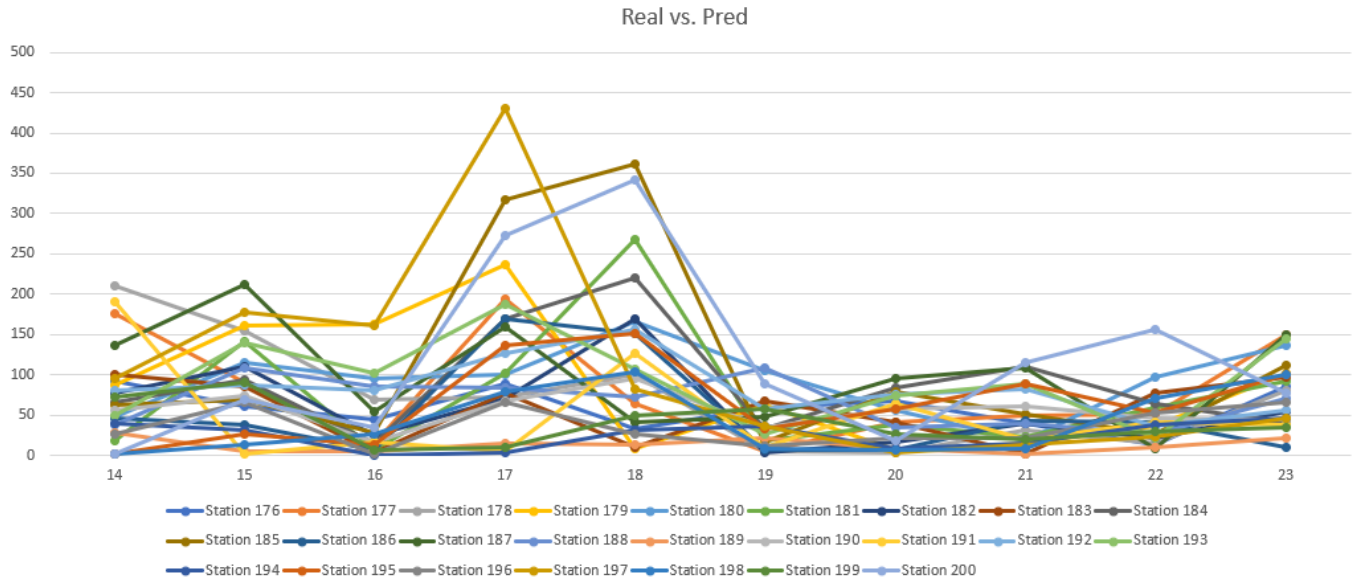


Figure 102. Difference between actual monthly demand and predicted demand