

Spring 2023

BASE STATION LOAD PREDICTION IN 5G-V2X HANDOVER

Madhujita Ranjit Ambaskar
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects



Part of the [Artificial Intelligence and Robotics Commons](#), and the [OS and Networks Commons](#)

Recommended Citation

Ambaskar, Madhujita Ranjit, "BASE STATION LOAD PREDICTION IN 5G-V2X HANDOVER" (2023). *Master's Projects*. 1236.

DOI: <https://doi.org/10.31979/etd.p6m7-444n>

https://scholarworks.sjsu.edu/etd_projects/1236

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

BASE STATION LOAD PREDICTION IN 5G-V2X HANDOVER

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Madhujita Ranjit Ambaskar

May 2023

© 2023

Madhujita Ranjit Ambaskar

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

BASE STATION LOAD PREDICTION IN 5G-V2X HANDOVER

by

Madhujita Ranjit Ambaskar

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2023

Dr. Navrati Saxena Department of Computer Science

Dr. Abhishek Roy MediaTek Inc.

Dr. Genya Ishigaki Department of Computer Science

ABSTRACT

BASE STATION LOAD PREDICTION IN 5G-V2X HANDOVER

by Madhujita Ranjit Ambaskar

5G V2X networks transmit large amounts of data with low latency, allowing for real-time communication between vehicles and other infrastructure. In 5G V2X networks, handover is a process that allows a connected vehicle to transfer its connection from one base station to another as it moves through the network coverage area. Handover is critical to maintaining the quality of service (QoS) and ensuring uninterrupted communication. The base station load is a critical factor in ensuring reliable and efficient 5G V2X connectivity. Prediction of traffic load on base stations ensure resource optimization and smooth connectivity during handovers. This research predicts the load of a base station using recurrent neural networks on a dataset of traffic loads of base stations spanning over a week. Recurrent neural networks can be used for time series data as they can capture complex patterns in the data, including seasonality, trends, and cyclical patterns. The predicted dynamic load value is then used in a handover algorithm and its effect on the handover performance is measured.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my thesis advisor, Dr. Navrati Saxena, for her unwavering support and guidance throughout the entire process. Her insights, feedback, and encouragement were invaluable in helping me to shape and refine my ideas. I would also like to thank the members of my thesis committee, Dr. Abhishek Roy and Dr. Genya Ishigaki, for their time and expertise. Their thoughtful feedback and suggestions helped me to improve the quality of my research and writing.

I am grateful to my colleagues in the Computer Science department for their camaraderie and support. Their feedback and encouragement were instrumental in helping me to persevere through the challenges of graduate school. Finally, I would like to express my deepest appreciation to my family and friends for their support. Without their encouragement and understanding, this thesis would not have been possible. Thank you all for your contributions to this work.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
2	Related Work	5
2.1	Dynamic load prediction	7
2.2	Using dynamic load during handover	8
3	Dynamic Load Prediction on Base Station	9
3.1	Data Preprocessing	9
3.1.1	Feature Engineering	11
3.1.2	Data Cleaning	12
3.2	Clustering	14
3.2.1	TimeSeriesKMeans with Euclidean distance for 5 clusters	15
3.2.2	Euclidean distance over Soft Dynamic Time Wrapping (DTW)	15
3.3	Recurrent Neural Network - LSTM (Long Short Term Memory) for Load Prediction	16
3.3.1	LSTM Model architecture	17
3.3.2	Cluster specific modelling	19
3.4	Results	19
3.4.1	Root Mean Square Error	20
3.5	Discussions	21
4	Handover Using Dynamic Load	26
4.1	Simulation Framework	26
4.2	Handover using NS3	29

4.3 Results	30
5 Conclusion	32
6 Future Work	33
LIST OF REFERENCES	35
APPENDIX	
A Appendix A	37
A.1 Clustered base stations provided by Shanghai Telecom	37
B Appendix B	38
B.1 Code snippets	38

CHAPTER 1

Introduction

5G V2X networks refer to the fifth generation (5G) of wireless communication technology that enables vehicle-to-everything (V2X) communication. It is a technology that enables communication between vehicles and other entities. The entities could be pedestrians, infrastructure, and other vehicles. [1] Various combinations of cellular network technologies, such as millimeter-wave (mmWave), sub-6GHz, and massive multiple-input and multiple-output (MIMO) antennas are used by these networks to enable faster and more reliable communication. To serve numerous V2X use cases, such as improved traffic efficiency, increased road safety, and vehicle automation, these networks are anticipated to offer low latency, high bandwidth, and high reliability communication services. Various modes of V2X networks enable different types of communication scenarios, such as collision avoidance, traffic management, and remote vehicle diagnostics. 5G V2X networks to play an important role in enabling the deployment of connected and automated vehicles. This can help in creating safer, more efficient, and more sustainable transportation systems.

Handover is a critical feature of 5G V2X networks, because seamless connectivity and uninterrupted communication depends on it. When a vehicle moves between different coverage areas, handover is required to be done between base stations. A base station is a communication device that connects wireless devices to a central network and acts as a hub for wireless communication. It provides coverage over a specific geographic area, known as a cell. Handover in 5G V2X networks involves transferring the ongoing communication session from one base station or access point to another while maintaining the quality of service and minimizing the disruption of the communication. To perform a handover in 5G V2X networks, many factors are considered. These include signal strength of the serving cell and neighbouring

cells, network capacity, network congestion and interference, traffic loads and network topology, and so on.

The load of a base station refers to the amount of traffic or data that the base station is handling at any given time. The load can vary depending on various factors, such as the number of connected devices, the type of communication, the quality of service (QoS) requirements, and the network capacity. This load can be measured using various metrics, such as the number of active connections, the data throughput, and the resource utilization. If the load of a base station exceeds its capacity, the network may experience congestion, leading to degradation in the quality of service and communication latency. In order to develop efficient handover algorithms, load prediction of base stations can be immensely useful and can help in optimal resource allocation.

Wireless network traffic has significantly increased recently and has been expanding exponentially. In the past year, 5G subscriptions reached 1 billion, with the average data consumption per smartphone to be more than 19 GB according to [2]. The main causes of this growth are the growing use of smartphones and the introduction of data use in many smart gadgets. As a result of this increased data usage, more base stations are needed to support more devices, which raises energy consumption significantly. The majority of the energy used by the network infrastructure is consumed by base stations, hence energy conservation is essential. Predicting base station traffic can have several important uses, including:

- Energy conservation: By accurately predicting the traffic on base stations, base stations can be turned off when they are not required during low-load times. This can lead to saving a lot of energy.
- Capacity planning: Predicting base station traffic can help network operators

plan and optimize their network capacity. By understanding how much traffic is expected on each base station, they can allocate resources more efficiently and ensure that the network is capable of handling the expected load.

- **Quality of service:** Predicting base station traffic can also help improve the quality of service for users. By ensuring that base stations are properly sized and located, network operators can reduce congestion and provide faster and more reliable connectivity to users.
- **Cost reduction:** Predicting base station traffic can help network operators reduce costs by avoiding over-provisioning of resources. By accurately predicting traffic, they can avoid investing in unnecessary infrastructure and equipment.

Base stations in a network can have varying traffic loads, with some having low traffic and others experiencing high traffic. The traffic load on a base station can also fluctuate over time. When using time series forecasting, having a larger amount of training data can improve performance. However, if the data exhibits different behaviors, a larger dataset may not lead to better performance and could even harm it. Therefore, it is necessary to train base stations with similar traffic loads together. We employ clustering to group base stations with comparable traffic loads, which helped address this issue.

There are different ways to carry out load forecasting, with the most frequently used approach involving statistical techniques such as simple moving average (SMA), exponential smoothing (ES), and autoregressive integrated moving average (ARIMA). However, in recent years, researchers have shifted towards using machine learning and deep learning methods. Of these, recurrent neural networks have gained significant attention as a means of performing load forecasting.

Recurrent Neural Networks (RNNs) are a type of neural network that is commonly

used for processing time series data. Time series data refers to data that is collected over time, such as stock prices, or traffic loads at base stations. They process sequential data by maintaining an internal memory that allows them to capture the temporal dependencies in the data. They do this by using a hidden state that is updated at each time step based on the input and the previous hidden state. The output at each time step is then determined based on the updated hidden state [3]. LSTM (Long Short-Term Memory) is a type of RNN, which are commonly used in time series analysis due to their ability to handle long-term dependencies and avoid the vanishing gradient problem that can occur in Simple RNNs. The input data is typically fed into the network in sequences or windows, where each sequence or window represents a subset of the time series data. The output of the network can be used to make predictions about future values in the time series.

The use of deep learning and recurrent neural networks in 5G V2X can help improve traffic management and reduce congestion. By predicting traffic load, traffic management systems can dynamically adjust traffic signals and routing to optimize traffic flow and reduce congestion. It allows for better quality of service and better connectivity.

CHAPTER 2

Related Work

As the world rapidly transitions to V2X networks, the infrastructure of these networks becomes more crucial and has significant implications. The need for optimized handover algorithms is more than ever. The quantity, regularity, and duration of handovers, as well as dwell time on a certain base station, are examples of optimization parameters [4]. Significant research has already been done on handovers in 5G-V2X networks, and this literature can help guide suggested handover techniques.

[1] proposes a dwell time estimation (DTE) scheme, a cell selection strategy based on Vehicle-to-Everything (V2X) communications. To reduce handover (HO) rate, it chooses small cells that have the longest dwell duration in the vehicle's direction. Two datasets, the 5G small cell and the vehicles datasets, are used to evaluate the proposed DTE method. The cell selection task can be carried out intelligently and with less computations using machine learning (ML) approaches. ML-modeling methods such as recurrent neural networks (RNN) and convolutional neural networks (CNN) can be used to forecast the ideal base station to be connected with.

[5] discusses techniques for choosing base stations for platoons. There are small base stations and large base stations in the 5G V2X architecture. Small base stations are utilized to reduce dead angles in the areas that large base stations cover because they have shorter ranges. The findings of simulations of communication between base stations and vehicles, signals, vehicle velocity, and platooning are used to create an effective handover algorithm. They build a model that optimizes handoffs for platoons using deep learning. The research in [6] discusses key components of 5G NR V2X. These include physical layer, QoS and resource management, and mobility management for V2N. It discusses coexistence of 5G NR V2X and LTE V2X. The system architecture of 5G NR V2X is also reviewed.

[7] gives an overview of deep learning while discussing deep learning models, frameworks and applications in detail. The paper discuss great ability of unsupervised learning, owing to the high amount of data available in the world today and in future. Fundamentals of recurrent neural networks and long-term short memory networks are talked about in [8]. The paper shows that the canonical formulation in RNN can be done with sampling delay differential equations. The article in [9] proposed a DL model that utilizes unidirectional LSTM for V2X traffic forecasting. This model allows for more accurate predictions by predicting future values based on past values and remembering values over a long time. Simulation results showed that the model had the highest prediction accuracy when 4 packets were transmitted per second. This outperformed other models and demonstrated excellent performance. Conversely, the models with a prediction of 14 packets per second had the lowest prediction accuracy compared to the others. The model that predicted 12 packets per second had the fastest processing time, while the models with a prediction of 14 packets per second had the slowest processing time compared to their competitors.

In [3], research was done to predict network traffic load where the prediction was modeled as a time series forecasting problem. Various clustering techniques were applied on a public dataset [10] to get highest accuracy. Different statistical methods, machine learning and deep learning methods were applied and the results were compared to conclude that recurrent neural networks had the best performance. The study explored various clustering techniques with different distance metrics to determine the most effective approach and the optimal number of clusters with the least error. It was found that Timeseries KMeans using Soft-DTW as the distance metric produced the least error. The base stations were then grouped to train and test each group separately. The dataset was analyzed using statistical methods, machine learning, and recurrent neural networks (RNNs) commonly used in time

series forecasting. The performance of each algorithm was evaluated using root mean square error, and the results indicated that RNNs outperformed both statistical methods and machine learning algorithms. These findings suggest that using clustered base stations is more effective than using all base stations. The authors used Amazon DeepAR, an online tool designed specifically for time series forecasting, as a benchmark to compare the performance of the tested algorithms.

[4] proposes an optimized algorithm that can reduce the frequency of handovers and improve dwell times in 5G-V2X networks, specifically in vehicular networks. The simulation includes building models and path loss considerations that account for real-world obstacles, distances, and other factors. The proposed approach involves two phases. In the first phase, vehicle destination is predicted based on current location, vehicle speed, and time of the day. In the second phase, this information is used to create a trajectory for the vehicle and select the most optimal base stations based on their proximity to the trajectory. The baseline approach uses RSRQ values to make the handover decision. Compared to the baseline approach, the proposed approach shows significant improvement in the number of handovers as well as the average dwell time.

This research aims to calculate the dynamic traffic load of a base station at a given time by performing time series forecasting using recurrent neural networks, and evaluates how dynamic load improves results in 5G V2X handovers. The steps followed are:

2.1 Dynamic load prediction

- Choosing a public dataset that can be applied to the task of predicting base station traffic.
- Dataset analysis and preprocessing.

- Grouping base stations in clusters based on their behavior.
- Training and testing the model using LSTM networks.

2.2 Using dynamic load during handover

- Handover simulation using SUMO and NS3.
- Utilizing dynamic load values of candidate base stations during handover.
- Comparing dynamic load results with static load performance.

CHAPTER 3

Dynamic Load Prediction on Base Station

In order to implement a handover algorithm, it is crucial to have a realistic value of traffic load on base stations. This enables the algorithm to efficiently evaluate candidate base stations that could be used for handovers as a vehicle moves from its starting location to the destination. Various statistical and machine learning methods have been used to calculate the traffic load on a base station. However, the Long Short-Term Memory variety of Recurrent Neural Networks predicts the load with very high accuracy [3]. This is because the data of traffic loads on base stations are of multivariate time-series nature. Dynamic prediction of load on a base station during a user's journey improves the 5G V2X handover performance by efficiently selecting base stations with lighter loads along the route of a vehicle.

3.1 Data Preprocessing

The data obtained from [10] is a public dataset used in this research. This dataset is a city cellular traffic map of China. The data spans over the time of one week. It provides traffic load data per hour on approximately 13,300 base stations. The given data provides the longitude and latitude of the base station location. It is observed that the load on base stations generally increases during the peak time between 5 AM to 3 PM and 6 PM to 11 PM. This is observed because of the rush hour traffic of devices during these times. The load during the day tends to be higher than at night. Figures 1 and 2 depict these spikes in traffic load.

Several data preparation steps were implemented to effectively use this dataset for forecasting, including feature engineering consisting of 2 steps, and data cleaning consisting of 3 steps.

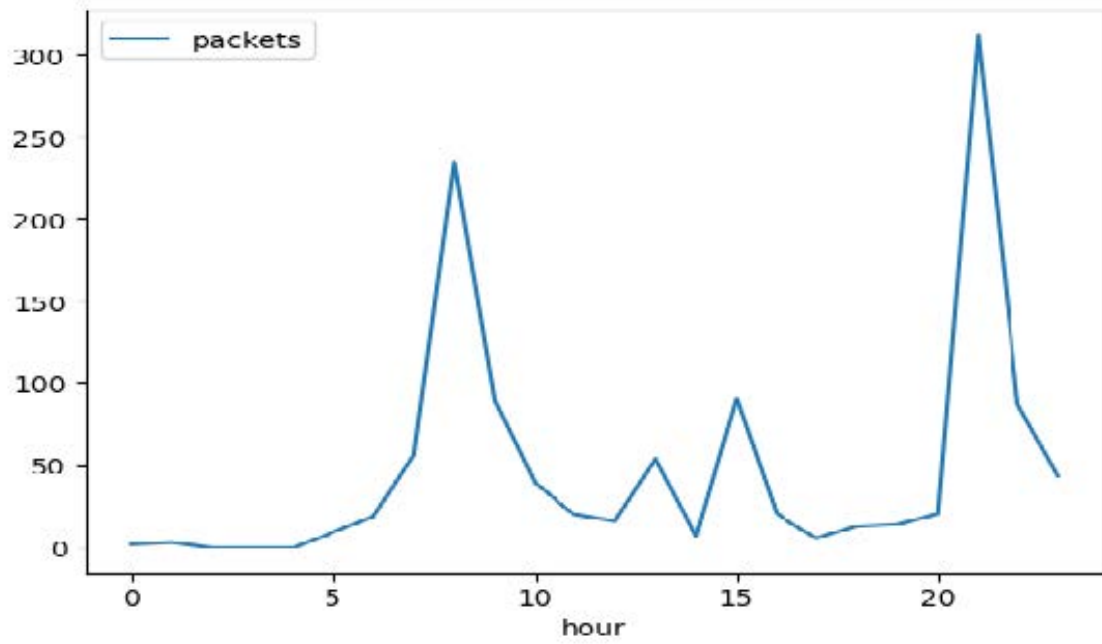


Figure 1: Average load per hour on base station #12

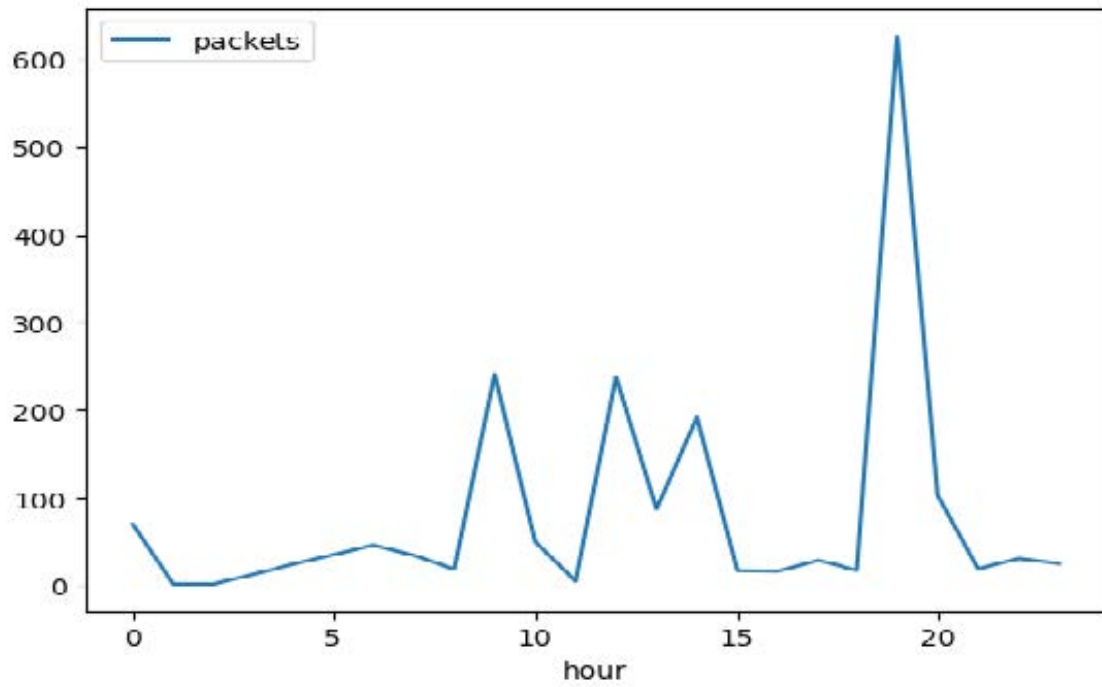


Figure 2: Average load per hour on base station #18

3.1.1 Feature Engineering

Feature engineering is the process of selecting and transforming raw data into useful features that can be used for training. It involves identifying features that can be made more meaningful, extracting them, and transforming them into a format that can be used for model training.

- UNIX epoch time: The dataset contains a field ‘time_hour’ that is a UNIX epoch timestamp. We derive the fields ‘month’, ‘day’, and ‘hour’. Furthermore, using the ‘day’ field, we generate a new binary feature ‘week_day’ that represents 1 (true) when the day is a weekday and 0 (false) when the day is a weekend.
- Haversine Distance: Haversine distance is a formula used to calculate the great-circle distance between two points on a sphere, such as the Earth. It takes into account the curvature of the Earth’s surface and provides a more accurate distance measurement than simple Euclidean distance. The formula uses the latitude and longitude coordinates of two points and calculates the distance between them by taking into account the Earth’s radius. The origin point is taken as (0,0). We measure the Haversine distance between the latitude and longitude of a base station and this origin point and add the value as a new field to our data, ‘haversine_dist’. The Haversine formula is given by:

$$h = \sin^2(a) + \cos(lat_origin)\cos(lat_bs)(\sin^2(b)) \quad (1)$$

where, a is difference between lat_origin and lat_bs and b is the difference between lon_bs and lon_origin and, lat_origin and lon_origin are the latitude and longitude of the origin point; lat_bs and lon_bs are the latitude and longitude of the base station.

- Preserving the cyclic nature of the feature ‘hour’: We added ‘hour_sin’ and ‘hour_cos’ as additional fields in our data to encode the cyclical nature of time,

which is important in time series analysis. When we work with time series data, we often encounter cyclical patterns, such as daily or weekly cycles, where the pattern repeats over a fixed time period. Representing cyclical patterns using traditional numerical encoding techniques can lead to problems because these methods assume that the data is linearly related. However, time is cyclical, so a non-linear representation would be more appropriate. In order to represent time in a cyclical way, we can use sine and cosine functions. By encoding time using sin and cos functions, we can represent the cyclical nature of time and avoid issues that may arise from treating time as a linear variable. For example, in the case of hourly data, we can represent each hour of the day as a point on a unit circle, where the angle of the point corresponds to the hour of the day. We can then use the sine and cosine functions of the angle to encode the hour. This approach ensures that the hour is represented in a cyclical and continuous manner, where the difference between 23:00 and 00:00 is small, unlike a numerical representation where these two times are far apart.

$$hour_sin = \sin(hour) \tag{2}$$

$$hour_cos = \cos(hour) \tag{3}$$

3.1.2 Data Cleaning

Cleaning the time series data involves identifying and addressing missing values, dealing with outliers, handling duplicates or errors, and removing irrelevant data that may interfere with the forecasting.

- Removing low-frequency base stations from the dataset: In the given dataset, the number of records provided by each base station varies greatly. Some base stations were found to have less number of records. A threshold value was considered to filter these base stations. This value was considered as the minimum number of base station records that had to be present in order for the

data record to be taken into consideration. After taking into account the total number of base stations and their average records, a value of 67 was chosen. This value is 40% of the maximum number of records any base station has. The base stations with less than the threshold value were filtered out and removed.

- Handling outliers: Sometimes outliers are introduced in a dataset due to errors or noise during data collection. These are data points that are very different from all the other data points in the dataset. Because of these outliers, the forecasting model performance can be negatively affected. Figure 3 shows a scatter plot of all data points including the outliers. It is crucial to make sure we remove these outliers after identifying them so that the model is accurate. This is done by first calculating the Z-score of all the data points on the ‘packets’ columns, and then removing a data point if its score exceeded 3. The threshold of 3 was chosen because 99.7% of the data had a Z-score lower than that value. We replace the value of the outlier with NaN and proceed to the next step. The NaNs in our dataset were values that were either originally missing or were converted due to being outliers. The next step was to replace the NaNs with a value from the dataset of the time instance prior to the current.
- Forward Fill null values: The method of replacing missing values with the previous record’s value is called "forward fill" or "last observation carried forward" (LOCF). It is a common approach in time series data imputation, where missing data is replaced with the most recently observed data. For all the NaN values in the dataset that originated from the missing values or the outliers, outliers, we replaced this value with the value of the record at the previous. The percentage of data points affected by forward filling was 1.27%.

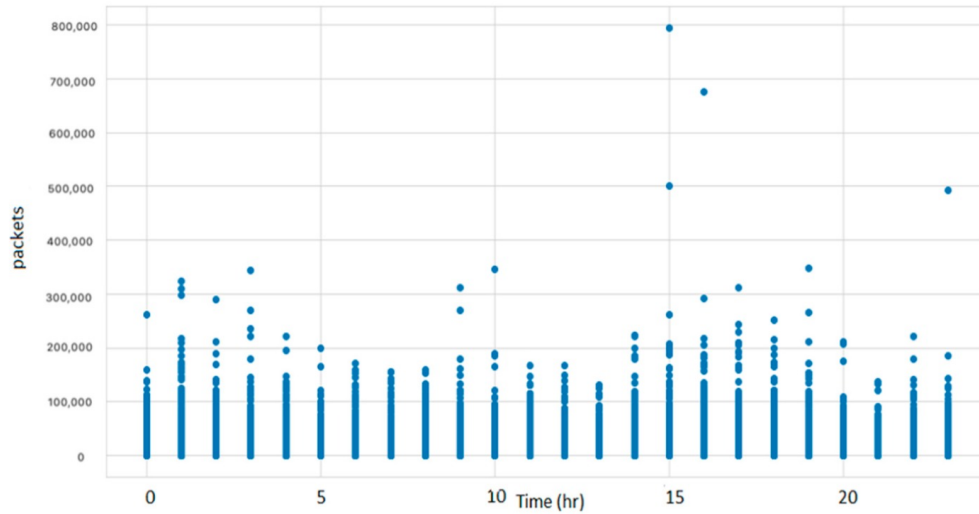


Figure 3: Scatter plot to show outliers

3.2 Clustering

Clustering is a useful technique in time series data analysis because it can help identify patterns and group similar time series together. When clustering is performed on time series data, the approach is different than clustering static data. The reason behind this is that the features are dynamic and continuously changing. Traditional clustering techniques are not always effective for time series data because they are designed to identify similarities in static features. Clustering was used in the research to group base stations with similar traffic loads together for the purpose of improving the accuracy of time series forecasting of traffic load on base stations. The reason for clustering is that different base stations process different amounts of traffic load, and some base stations within a network may have a low load, whereas others may have an extremely high load. By clustering base stations with similar traffic loads, the training data becomes more focused and specific, which can lead to better forecasting performance. An example of a problem statement where clustering helps is while developing a grid based energy saving scheme where the base stations with low loads

Unclustered	Cluster-0	Cluster-1	Cluster-2	Cluster-3	Cluster-4
1059.9387	137.6398	873.9473	1426.5502	17.2557	415.8907

Table 1: MSE of unclustered data vs individual clusters

can be switched to sleep mode during hours of low or negligible traffic load, without compromising on the quality of service for users. Clustering also mitigates the model variance. The variance is measured in terms of the Mean Squared Error (MSE) value. Table 1 gives the MSE values for unclustered data, as well as all clusters which shows that the MSE values are lesser for 4 out of 5 clusters than for unclustered data.

3.2.1 TimeSeriesKMeans with Euclidean distance for 5 clusters

Spatial clustering clusters the base stations according to the base station locations and assumes that base stations near each other have the same behavior. However, time series clustering based on base station behaviors is more accurate. We perform clustering using TimeSeriesKMeans with Euclidean distance as the distance metric. The number of clusters was chosen as 5.

3.2.2 Euclidean distance over Soft Dynamic Time Wrapping (DTW)

Soft Dynamic Time Warping (Soft DTW) is slower than Euclidean distance in TimeSeriesKMeans because it involves more computations. Soft DTW is a flexible variant of Dynamic Time Warping (DTW) that computes a differentiable loss function by incorporating a soft-minimum operation into the DTW cost matrix. This operation involves computing the exponential of the negative distances between each pair of time series points and summing them up along all possible alignments. The soft-minimum operation allows the optimization algorithm to backpropagate the gradients through the DTW algorithm, enabling end-to-end training of models that use DTW as a component. On the other hand, Euclidean distance is a simpler metric that involves only the computation of the distance between the corresponding points of

two-time series, making it computationally faster. However, Euclidean distance does not account for the temporal misalignment between the two-time series and may not capture the similarity between them as well as DTW-based methods. However, Soft DTW is a computationally expensive algorithm, and clustering a million time series with 10 features using Soft DTW distance can take a considerable amount of time, even on a high-end system. The time required can range from several days to a few weeks, depending on the factors mentioned above. Hence, we used Euclidean distance with lower accuracy in tradeoff with the time and cost of computing Soft DTW.

3.3 Recurrent Neural Network - LSTM (Long Short Term Memory) for Load Prediction

ARIMA/SARIMA are popular statistical methods for time series forecasting. However, flexibility in modeling different types of time series data is provided by Recurrent neural network (RNN). RNNs can be used to model different types of time series data, such as multivariate time series data, irregularly sampled time series data, and time series data with missing values. Overall, RNNs offer greater flexibility and adaptability than ARIMA/SARIMA models, which makes them more suitable for a wider range of time series forecasting tasks, including ours. Recurrent neural network (RNN) can be used to predict the network load on base stations in time series forecasting. RNNs are able to capture time dependencies by utilizing past patterns to predict future values. However, traditional RNNs face the vanishing gradient problem when considering a long range of previous time steps. Two popular solutions to this problem are LSTMs and GRUs, which both use gates to determine what information to keep and forget. LSTMs have four gates and are used to determine what to keep from the new cell state, what to forget from the existing memory, and what the next hidden state should be. GRUs have two gates and determine which past information to reset or forget and what to use to update the new cell. Both LSTMs and GRUs

are used to solve the vanishing gradient problem, but they differ in their specific processes.

Hyperparameter	Definition	Values
Input dimensions	<p>In general, the input dimensions for an RNN can be represented as a 3D tensor of shape (batch_size, time_steps, input_dim)</p> <p>*batch_size: # of samples in each batch *time_steps: # of time steps in input sequence *input_dim: # of features in each time step of input sequence</p>	Number of time series in training data for each cluster
Output dimensions	The output dimension refers to the number of neurons in the output layer. It determines the shape and size of the output generated by the RNN model after processing the input sequence. Since we're	1

Figure 4: RNN LSTM Hyperparameters

3.3.1 LSTM Model architecture

The model architecture consists of a Sequential model from Keras, which is a linear stack of layers.

- The first layer is an LSTM (Long Short-Term Memory) layer with 128 memory units and return_sequences=True to return the full sequence of output values instead of only the last one. The input shape is set to (1, 8) since we are processing 8 features as a sequence of length 1.
- After the LSTM layer, a dropout layer is added with a rate of 0.2, which helps to prevent overfitting by randomly setting some of the input units to 0 during

	predicting, the output dimension should be 1.	
Batch size	Batch size refers to the number of samples that are processed before the model's weights are updated during training	32
Optimizer	The optimizer is an algorithm that updates the weights of the network during the training process to minimize the loss function.	Adam
Epochs	Epochs refer to the number of times the entire dataset is passed forward and backward through the neural network during training. Each epoch consists of two phases: forward propagation and backpropagation.	10
Metrics	Mean Absolute Error is a commonly used metric for regression models that measures the average absolute difference between the predicted and actual values.	mae

Figure 5: RNN LSTM Hyperparameters

training. The same LSTM layer is added again with the same parameters as before, followed by another dropout layer with the same rate.

- Finally, a Dense layer is added with a single output unit, which is the predicted value for the time series. The loss function used for training the model is specified by the `loss_function` variable, and the optimizer used for minimizing the loss is specified by the `optimizer` variable. The metrics used for evaluating the model during training are 'mae' (mean absolute error).

The input data scaled to a minimum and maximum value of 0 and 1 using a `MinMaxScaler()` function. The training and testing data is reshaped using a `reshape()` function to match the input shape of the LSTM layer. During training, the model is fit to the training data with hyperparameters defined in Figures 4 and 5. Early stopping is used to prevent overfitting, and the trained model is saved as a pickle file. Finally, the trained model is used to make predictions on the test data, and the root mean squared error (RMSE) between the predicted values and the actual values is calculated. During training, the model is fit to the training data with hyperparameters defined above. Early stopping is used to prevent overfitting, and the trained model is saved as a pickle file. Finally, the trained model is used to make predictions on the test data, and the root mean squared error (RMSE) between the predicted values and the actual values is calculated.

3.3.2 Cluster specific modelling

For each cluster, a separate model was trained, and a sample of base stations belonging to each cluster was considered for the results. The traffic loads of these base stations were shown against the corresponding forecasted values.

3.4 Results

There are several evaluation metrics that can be used for evaluating LSTM RNN models for time series data, some of which are:

- Mean Absolute Error (MAE): MAE measures the average absolute difference between predicted and actual values. It is a common evaluation metric used in LSTM RNN models.
- Mean Squared Error (MSE): MSE measures the average of the squared differences between predicted and actual values. It is a common loss function used in training LSTM RNN models.

- Root Mean Squared Error (RMSE): RMSE is the square root of the MSE, and it measures the average difference between predicted and actual values, taking into account the scale of the values.
- R-squared (R2): R2 measures the proportion of variance in the target variable that is explained by the model. It ranges from 0 to 1, with higher values indicating a better fit of the model to the data.
- Mean Absolute Percentage Error (MAPE): MAPE measures the percentage difference between predicted and actual values. It is commonly used in forecasting models to assess the accuracy of the model's predictions.

3.4.1 Root Mean Square Error

We calculated the RMSE for each cluster, as given in Table 2, and then the average RMSE was calculated for all base stations belonging to the same cluster. This approach provided an overall performance measure for the LSTM RNN model for each cluster. Lower RMSE values indicated better performance of the model in accurately forecasting the traffic loads of the base stations. Both MSE and RMSE measure the performance of a model by calculating the difference between the predicted values and the actual values. However, RMSE is preferred in many cases, including time series forecasting, because it penalizes larger errors more heavily than smaller ones, and it is presented in the same units as the actual observations, making it easier to interpret. Hence, RMSE was used as the evaluation metric for each cluster's model because it provides a more comprehensive measure of the performance of the models across different base stations and clusters. Figures 6, 7, 8, 9, 10 show graph plots of model loss per epoch for each cluster.

Cluster-0	Cluster-1	Cluster-2	Cluster-3	Cluster-4
11.7320	29.5626	37.7697	4.154	20.3934

Table 2: RMSE for RNN LSTM for each cluster

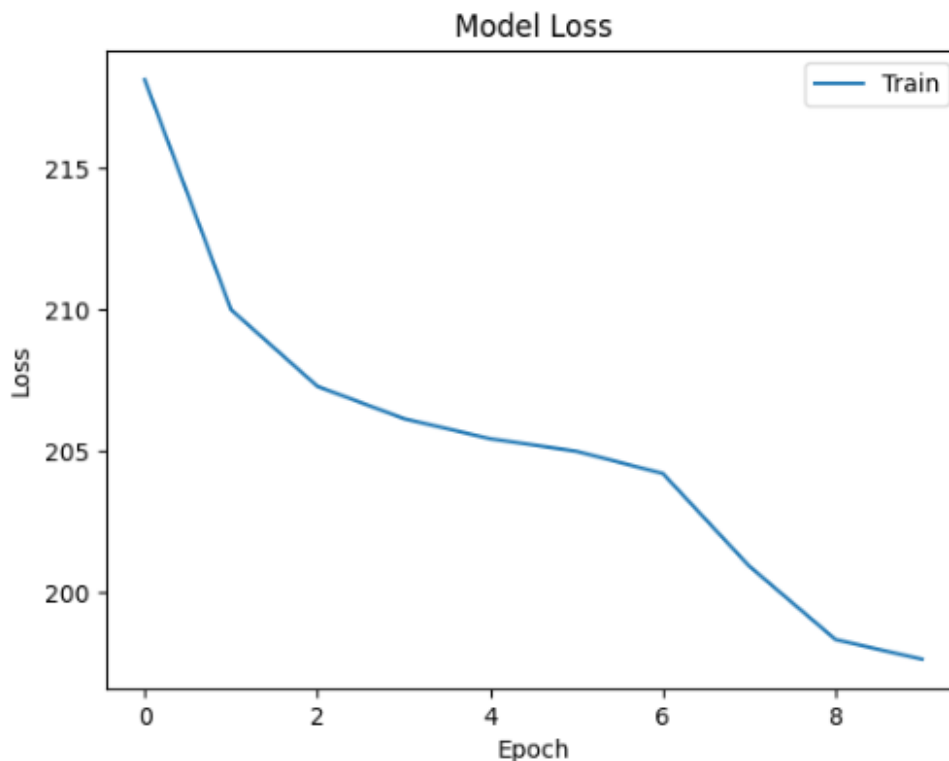


Figure 6: Model loss per epoch for Cluster-0

3.5 Discussions

- If there are enough data in each cluster, clustering base stations according to how they behave before applying time series forecasting techniques can increase the precision of their load forecasts.
- The TimeSeriesKMeans clustering algorithm often yields the lowest error for time series data when compared to other clustering algorithms.
- For time series forecasting, Recurrent Neural Networks, including LSTMs and GRUs, outperform statistical techniques and machine learning.

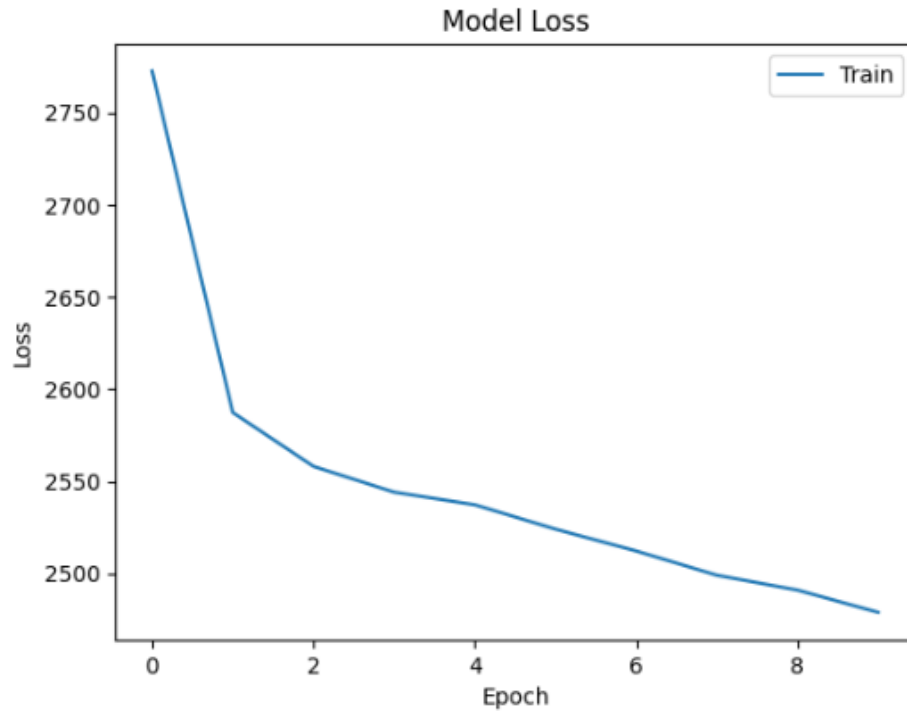


Figure 7: Model loss per epoch for Cluster-1

Future research should test these strategies on additional datasets that may circumvent the constraints of the dataset utilized in this study.

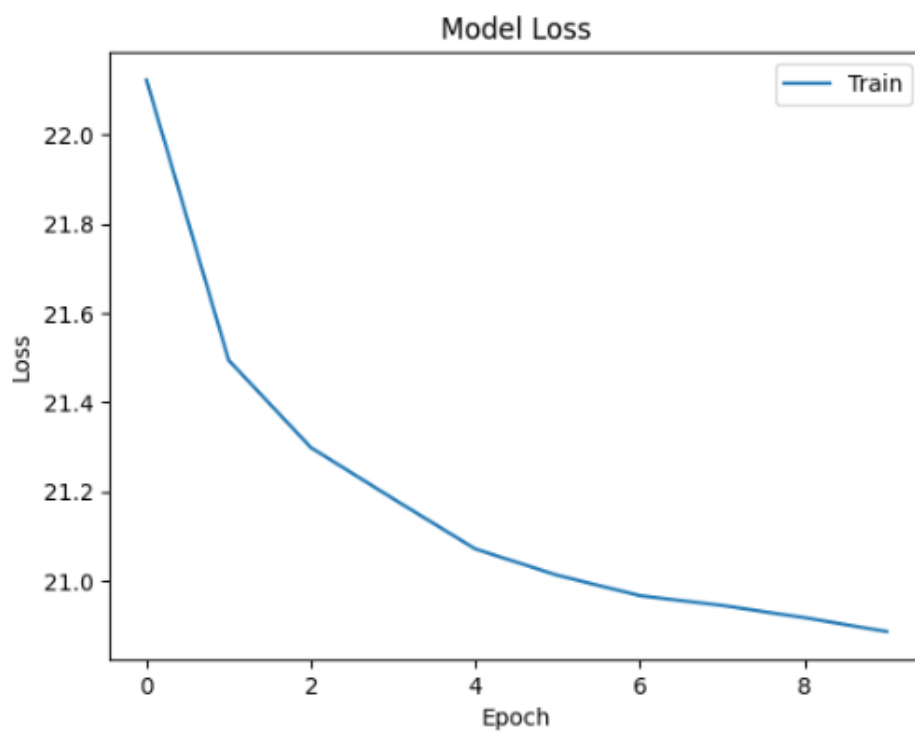


Figure 8: Model loss per epoch for Cluster-2

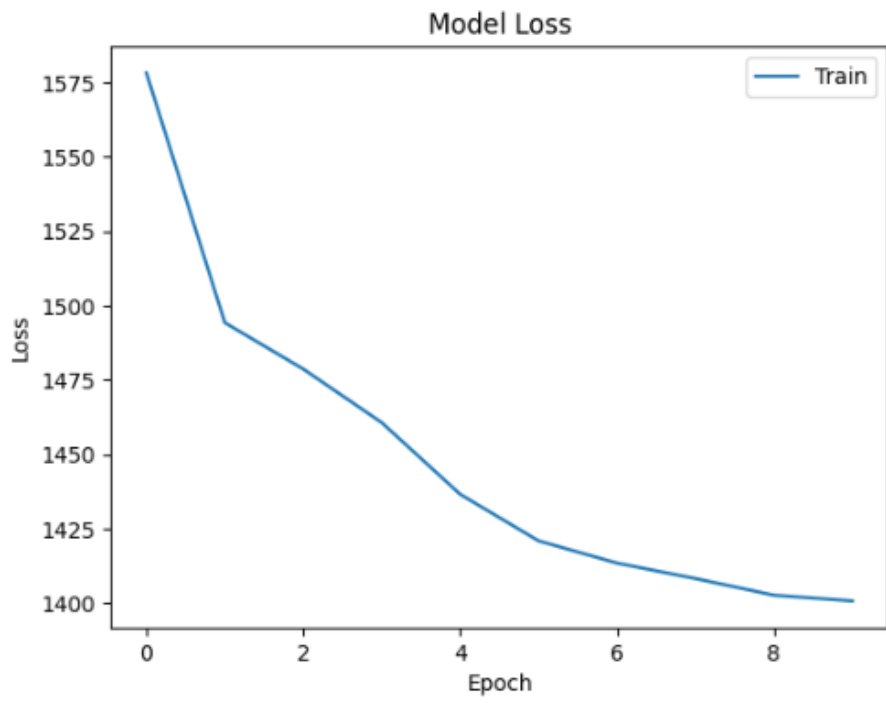


Figure 9: Model loss per epoch for Cluster-3

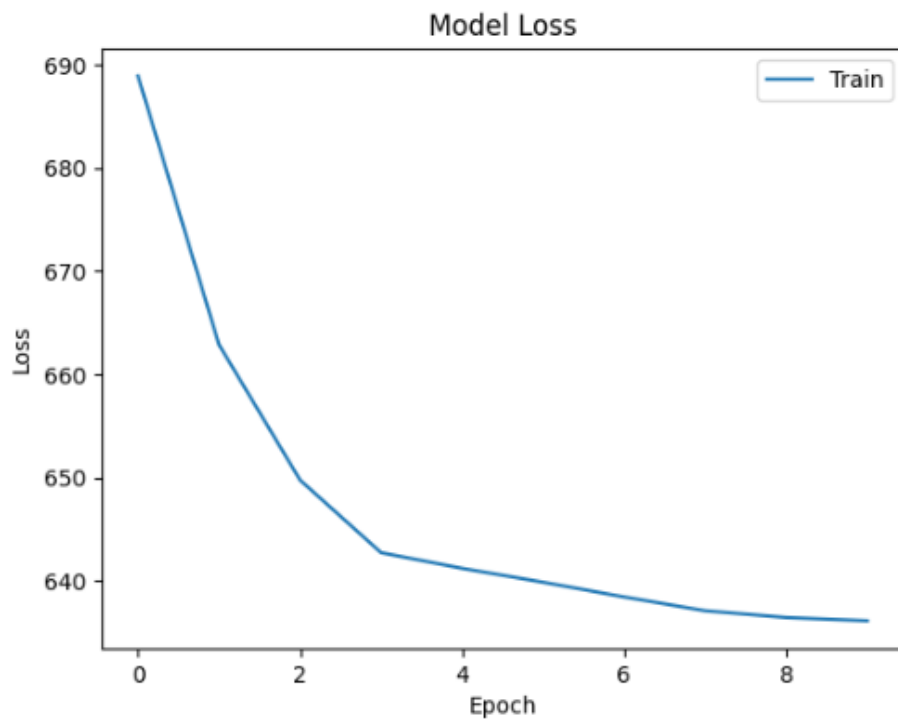


Figure 10: Model loss per epoch for Cluster-4

CHAPTER 4

Handover Using Dynamic Load

4.1 Simulation Framework

An open-source traffic simulation software known as SUMO (Simulation of Urban Mobility) allows modeling and simulating different transportation systems, including road networks, public transportation, and pedestrians. It is often used for research and development purposes in fields related to transportation, such as traffic engineering, urban planning, and autonomous vehicles. SUMO simulates the behavior of different traffic participants, such as vehicles, bicycles, pedestrians, and public transportation vehicles, based on predefined traffic rules and customizable parameters. It can be used to analyze traffic flow, test traffic management strategies, evaluate the impact of new infrastructure projects, and optimize transportation systems.

OpenStreetMap (OSM) is a collaborative project that aims to create a free and editable map of the world. SUMO (Simulation of Urban Mobility) can use OSM data to create realistic simulations of urban traffic networks. OSM provides a vast amount of geographical data, including roads, buildings, and points of interest, which can be used to create detailed and accurate traffic network models, as shown in Figure 13. SUMO can import OSM data and convert it into a network representation that can be used for traffic simulation. By using OSM data in SUMO simulations, researchers and planners can accurately model traffic patterns and test different traffic management strategies in a realistic and detailed environment. OSM data can also be used to validate and calibrate traffic simulation models and to assess the impact of new infrastructure projects on traffic flow and transportation accessibility. The combination of OSM and SUMO provides a powerful tool for transportation research and planning, enabling users to create detailed and accurate traffic simulations that can help inform decision-making and improve transportation systems.

We leverage a dataset [11], [12], [13] provided by Shanghai Telecom comprising over 7.2 million records of mobile internet access, collected over a six-month period from more than 9,000 mobile phones and over 3,000 base stations. The dataset includes the location of each base station in Shanghai, as well as six parameters such as month, data, start and end times, base station location, and mobile phone ID. This information can be used to track user trajectories and aid researchers in evaluating solutions related to mobile edge computing, such as edge server placement, service migration, and service recommendation. Visualizations in Figures 11 and 12 show the distribution of these base stations by plotting their locations using Google Maps library.

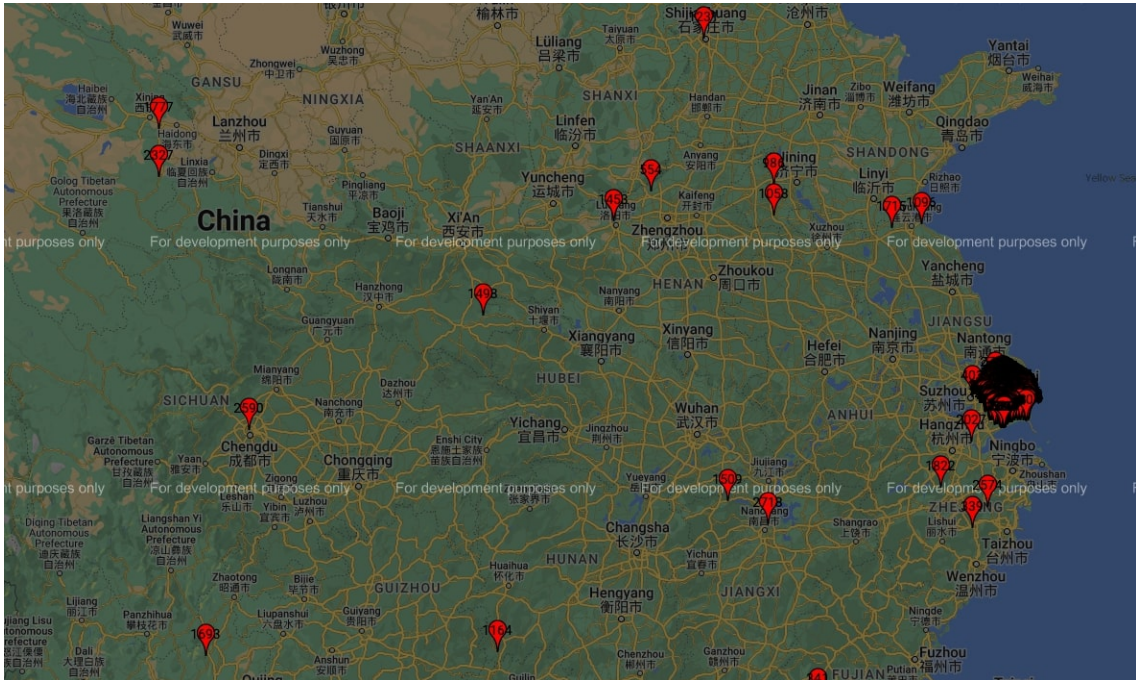


Figure 11: Base stations plotted on Google maps

We select a region in Shanghai in the OSM and start the simulation with one vehicle. Figure 3 shows area selection in the OSM. At the end of the simulation, we obtain logs that we use to convert into a trace file. To use the simulation results in a

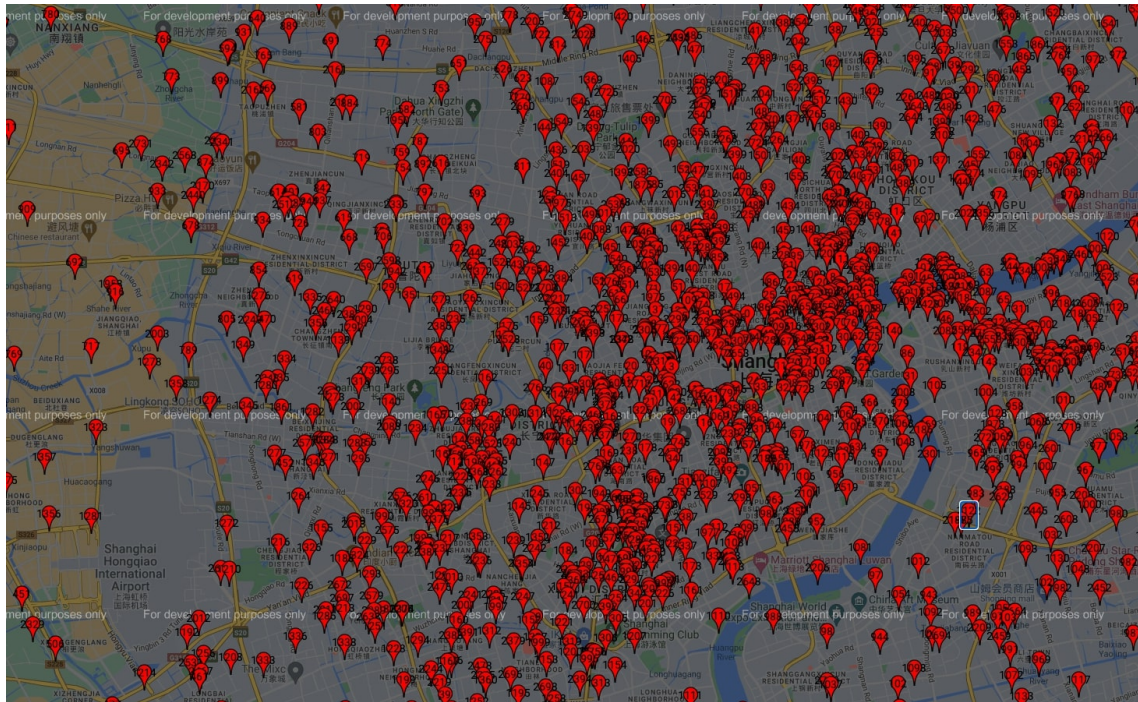


Figure 12: Cluster of base stations in Shanghai region

handover, we require NS3.

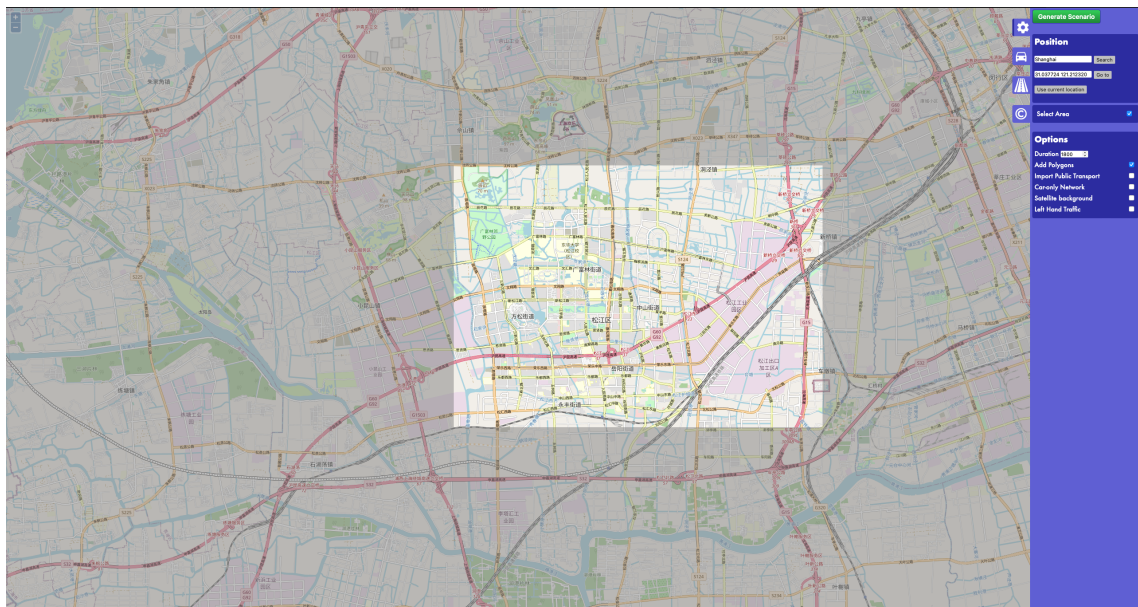


Figure 13: Region selection in OSM

NS3 (Network Simulator 3) is a discrete-event network simulator that is widely used for research in computer networks. It is an open-source software that provides a platform for designing and simulating network protocols, applications, and architectures. NS3 is written in C++ and offers a flexible and modular architecture that allows for customization and extension of its functionalities. NS3 supports a range of networking technologies, including wired, wireless, and cellular networks, and can be used to simulate different layers of the network stack, such as the physical, data link, network, and transport layers. It also supports a variety of routing protocols, transport protocols, and application protocols, including TCP, UDP, HTTP, and FTP. NS3 provides a comprehensive set of features for network simulation, including support for visualization, tracing, and debugging of simulation results. It also includes various modules for network emulation, traffic generation, and energy modeling, which allow for more realistic and accurate simulations.

4.2 Handover using NS3

SUMO and NS3 are used together in to simulate vehicular communication. In this case, SUMO can be used to simulate the vehicular traffic and movement, while NS3 can be used to simulate the wireless communication and network behavior. The path of vehicle generated by SUMO can be seen in Figure 14. This integrated simulation approach can provide more realistic and accurate results, as it considers both the mobility and communication aspects of the scenario. To simulate a handoff and measure performance, we use a combination of mmWave module with NS3 [14]. The algorithm used maintains a list of potential base stations to handover to, with the target base station being determined based on proximity to the user equipment and higher Reference Signals Received Quality (RSRQ) values. This is the basic principle of the algorithm. The algorithm can be written as a function of distance and RSRQ

values, where the likelihood of a handover is directly proportional to both these factors. The algorithm, however, does not take into consideration the load at the base station at that particular time of day. In real life scenarios, predicting the load at a base station will help the algorithm select a better target base station. The load value is inversely proportional to the likelihood of a handover. Lesser the load value at a base station, higher is its chance of being chosen as a target base station during handover.

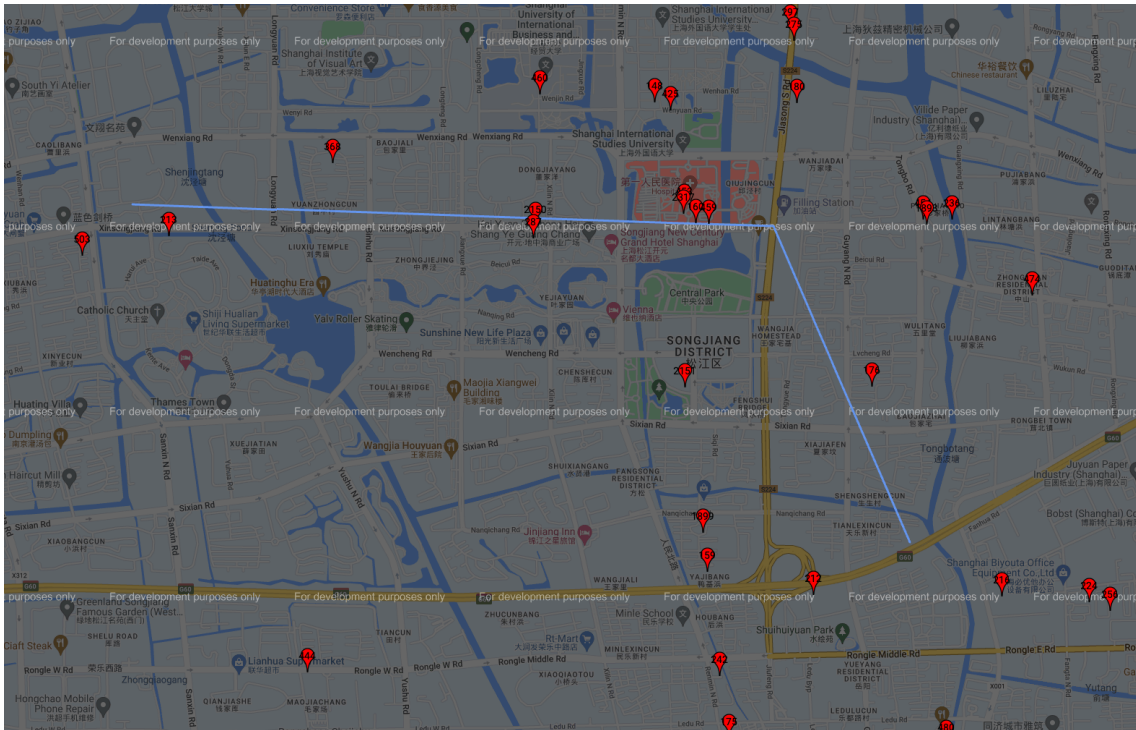


Figure 14: Vehicle path generated by SUMO

4.3 Results

Once the candidate base stations were identified, their dynamic load values were predicted using the deep learning model. Table 3 shows the predicted load values for each base stations. In the case of time series prediction, LSTM is used as a regression model to predict continuous values, hence it returns a float value in order to get an estimate of the number of users. Applying the floor function and rounding off the

BS1	BS2	BS3	BS4	BS5	BS6	BS7	BS8	BS9
24.6	14.7	19.3	24.1	23.7	28.7	25.5	36.7	22.9

Table 3: Predicted traffic load on each base station

number to the nearest integer will give the exact number of users. The algorithm used for handover simulation was NS3's A2A4RsrqHandoverAlgorithm. The experimental setup consisted of the list of candidate base stations and one user equipment (UE). The initial and the final position of the vehicle was specified as obtained from SUMO results. At the end of the simulation, the total number of handovers were observed. The number of handovers without considering the load value came out to be 29, whereas after considering the predicted load it was reduced to 23. Overall, a 20% decrease was observed. Figure 15 shows this result in a tabular format.

No. of handovers with static load	29
No. of handovers with dynamic load	23
% decrease in no. of handovers	20%

Figure 15: Handover observations

CHAPTER 5

Conclusion

The research presented in this report aimed to predict network traffic load on base stations using time series forecasting techniques. Clustering of base stations was done based on their behavior as a solution for the nonsimilar behavior of different base stations. Good results were obtained by implementing TimeSeriesKMeans with Euclidean Distance. The base stations were grouped into five groups and tested each group separately using recurrent neural networks. RNNs like LSTMs provided good performance for time series forecasting. The main conclusions of the research are that clustering base stations based on their behavior can significantly improve the accuracy of their load forecasting, TimeSeriesKMeans usually results in the lowest error for clustering time series data, and RNNs usually provide better performance than statistical methods and machine learning for time series forecasting. The dataset used in this work has several limitations and future work must consider applying and testing the techniques used in this work with other datasets that overcome these limitations.

CHAPTER 6

Future Work

The future of dynamic load prediction of base stations looks promising, as there are several developments that are likely to enhance its capabilities. One of the major trends is the adoption of more machine learning and artificial intelligence (AI) techniques in load prediction models. These techniques have the potential to improve the accuracy of load prediction by analyzing large volumes of data and identifying patterns and trends that would be difficult for human operators to detect. Another possible development is the use of real-time data analytics and edge computing. By analyzing data at the edge of the network, close to the base stations, it is possible to improve the accuracy and speed of load prediction, and to respond more quickly to changes in network conditions. The dynamic prediction model implemented in this research can be used as a 'plug-n-play' model in various handover algorithms and network optimization techniques.

To check for the stationarity of a time series, future work can use statistical tests like the Augmented Dickey-Fuller (ADF) test or the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test. These tests check whether the time series has a unit root (i.e., a root of 1) or not. A time series is considered stationary if it has no unit root. If the time series is not stationary, we can use techniques like differencing, seasonal differencing, or log transformation to make it stationary. Differencing eliminates the level fluctuations in a time series, leading to a decrease in trend and seasonality. Log transformation is a common transformation technique that works by applying the log function on the time series to transfer each value to its logged one.

We used Euclidean distance over the slower Soft Dynamic Time Wrapping in TimeSeriesKMeans because it involves more computations to incorporate the soft-minimum operation into the DTW cost matrix, but it offers a more flexible and

powerful approach to time series clustering that accounts for temporal misalignments between time series. Future work can use clustering using the Soft DTW distance to achieve much better results. Since the dataset used only provides hourly traffic data, we cannot use the feature 'minute'. Using an alternate dataset with the minute information will increase the number of features in the model, and consequently, the model's complexity.

Future work can be done using the DeepAR algorithm to examine the effect of clustering on forecasting. DeepAR is a supervised learning algorithm designed for time series forecasting using autoregressive recurrent neural networks (RNNs). It does not require clustering. The algorithm utilizes the similarity found between related time series data to improve forecasting performance by learning a global model from the past data of the existing related time series. The advantage of the DeepAR algorithm is that it can provide future predictions for new time series with no previous records because it learns from related time series. Therefore, it can be used to evaluate the effect of clustering on forecasting by training and testing the DeepAR model on clustered and unclustered time series data separately and comparing their performance using evaluation metrics such as RMSE.

LIST OF REFERENCES

- [1] I. A. Alablani and M. A. Arafah, "Applying a dwell Time-Based 5G V2X cell selection strategy in the city of los angeles, california," *IEEE Access*, vol. 9, pp. 153 909--153 925, 2021.
- [2] Ericsson, "Ericsson Mobility Report," 2022.
- [3] B. M. et al., "A Clustering-Driven approach to predict the traffic load of mobile networks for the analysis of base stations deployment," *Journal of Sensor and Actuator Networks; Basel*, vol. 9, no. 4, p. 53, 2020.
- [4] P. Shyamsundar, "A novel handover method using destination prediction in 5G-V2X networks," Master's thesis, San Jose State University, 2022.
- [5] C.-C. Ho, B.-H. Huang, M.-T. Wu, and T.-Y. Wu, "Optimized base station allocation for platooning vehicles underway by using deep learning algorithm based on 5g-v2x," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, 2019, pp. 1--2.
- [6] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5G NR V2X communications," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 3, pp. 1972--2026, 2021.
- [7] X. Du, Y. Cai, S. Wang, and L. Zhang, "Overview of deep learning," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2016, pp. 159--164.
- [8] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long Short-Term memory (LSTM) network," *Physica D*, vol. 404, p. 132306, 2020.
- [9] A. R. Abdellah, A. Muthanna, M. H. Essai, and A. Koucheryavy, "Deep learning for predicting traffic in V2X networks," *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, vol. 12, no. 19, 2022.
- [10] X. C. et al., "Analyzing and modeling spatio-temporal dependence of cellular traffic at city scale," in *Communications (ICC), 2015 IEEE International Conference on*, 2015.
- [11] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 55--67, 2022.

- [12] Y. G. et al., “User allocation-aware edge cloud placement in mobile edge computing,” *IEEE Internet of Things Journal*, vol. 50, no. 5, pp. 489--502, 2020.
- [13] S. W. et al., “Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939--951, 2021.
- [14] M. M. et al., “End-to-end simulation of 5g mmwave networks,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 2237--2263, 2018.

APPENDIX A

Appendix A

A.1 Clustered base stations provided by Shanghai Telecom

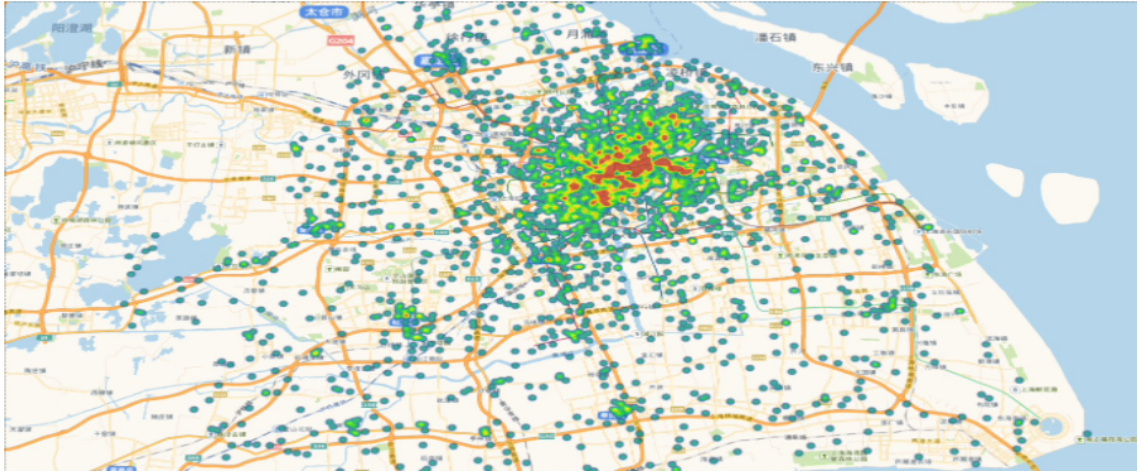
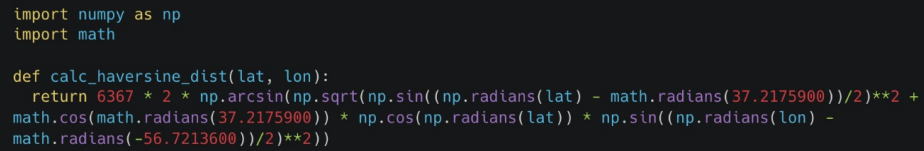


Figure A.16

APPENDIX B

Appendix B

B.1 Code snippets

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays Python code for a function named 'calc_haversine_dist'. The code imports 'numpy as np' and 'math'. The function 'calc_haversine_dist(lat, lon):' returns a calculation involving '6367 * 2 * np.arcsin(np.sqrt(np.sin((np.radians(lat) - math.radians(37.2175900))/2)**2 + math.cos(math.radians(37.2175900)) * np.cos(np.radians(lat)) * np.sin((np.radians(lon) - math.radians(-56.7213600))/2)**2))'.

```
import numpy as np
import math

def calc_haversine_dist(lat, lon):
    return 6367 * 2 * np.arcsin(np.sqrt(np.sin((np.radians(lat) - math.radians(37.2175900))/2)**2 +
    math.cos(math.radians(37.2175900)) * np.cos(np.radians(lat)) * np.sin((np.radians(lon) -
    math.radians(-56.7213600))/2)**2))
```

Figure B.17: Python function to calculate Haversine distance

```

from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.optimizers import Adam

modell = Sequential()
modell.add(LSTM(128, return_sequences=True, input_shape=(1, 8,)))

'''
Dropout regularization can help prevent overfitting in LSTM RNN models by randomly dropping out some
units during training.
Add a Dropout layer after each LSTM layer with a dropout rate of 0.2-0.5.
'''
modell.add(Dropout(0.2))
modell.add(LSTM(128, return_sequences=True, input_shape=(1, 8,)))
modell.add(Dropout(0.2))
modell.add(Dense(1))

modell.compile(loss=loss_function, optimizer=optimizer, metrics=['mae', 'accuracy'])

X_train, X_test, y_train, y_test = normalize(cluster_1)

scaleMinMax(X_train)
scaleMinMax(X_test)

X_train, X_test, y_train, y_test = reshape(X_train, X_test, y_train, y_test)

early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)
modell.fit(X_train, y_train, batch_size=batch_size, epochs=num_epochs)
with open(f'cluster-1-model.pkl', 'wb') as f:
    pickle.dump(modell, f)

X_test_res = modell.predict(X_test)
X_test_res = X_test_res.squeeze(axis=1)
print(f"Cluster 1 - RMSE: {getRms(y_test, X_test_res)}")

```

Figure B.18: LSTM model architecture