

# Congestion estimation and turning ratio prediction based on machine learning with applications in urban traffic light control

Chen, Qixing

2019

Chen, Q. (2019). Congestion estimation and turning ratio prediction based on machine learning with applications in urban traffic light control. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/143517>

<https://doi.org/10.32657/10356/143517>

---

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

*Downloaded on 09 Aug 2022 15:33:57 SGT*

**Congestion Estimation and Turning Ratio Prediction Based on  
Machine Learning with Applications in  
Urban Traffic Light Control**

**Chen Qixing**

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Master of Engineering

**2019**

**Statement of Originality**

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarized materials, and has not been submitted for a higher degree to any other University or Institution.

31 -12-2019

.....  
Date



.....  
Chen Qixing

### **Supervisor Declaration Statement**

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

31-12-2019

.....

Date



.....

Su Rong

## Authorship Attribution Statement

This thesis contains materials from 1 paper published in the 58th IEEE Conference on Decision and Control, where I am one co-author.

Chapter 2.3 and 2.4 is published in the following paper:

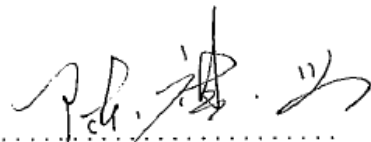
Y. Zhang, R. Su, Q. Chen, Y. Zhang and C. Sun. A hybrid traffic light control strategy based on branching ratio estimation and congestion identification. In Proc. 58th IEEE Conference on Decision and Control, Nice, 2019.

The contributions of the co-authors are as follows:

- A/Prof Su Rong provided the initial project direction and edited the manuscript drafts.
- Dr Zhang Yicheng prepared the manuscript draft. The manuscript was revised by Ms Zhang Yi and me.
- Dr Zhang Yicheng and I co-designed the method and experimental case studies with A/Prof Su Rong and performed all the laboratory work at the School of Electrical and Electronic Engineering
- Mr Sun Chunyang assisted in the traffic network construction in Vissim
- I provide the codes and the result for the congestion identification and prediction mode.

31-12-2019

.....  
Date

  
.....  
Chen Qixing

## Table of Contents

Statement of Originality

Supervisor Declaration Statement

Authorship Attribution Statement

Acknowledgments

Abstract

### 1. Introduction

- 1.1 Introduction to traffic networks and traffic light control
- 1.2 Motivations for congestion region analysis and turning ratio analysis
- 1.3 Literature review
- 1.4 Contributions of this thesis
- 1.5 Organization of the Thesis

### 2. Traffic Congestion Region Identification and Prediction

- 2.1 Basic concepts and problem statement
- 2.2 Congestion level identification
- 2.3 Congestion level prediction
- 2.4 Congestion region clustering
  - 2.4.1 K-means clustering
  - 2.4.2 Hierarchical clustering
  - 2.4.3 Improved the K-means clustering algorithm
- 2.5 Simulation-based experiments
  - 2.5.1 Case study one: Traffic congestion level identification (target: campus traffic network)
  - 2.5.2 Case study two: Traffic congestion level prediction and congestion region clustering (target: Jurong area traffic network)
- 2.6 Summary

### 3. Traffic Network Turning Ratio Prediction

- 3.1 Basic concepts and problem statement
  - 3.2 Machine learning algorithms for turning ratio prediction
    - 3.2.1 An FNN model and parameter identification
    - 3.2.2 An RNN model and parameter identification
  - 3.3 Simulation-based experiments
    - 3.3.1 System setup
    - 3.3.2 Experimental procedure and data collection
    - 3.3.3 Comparisons and discussions
  - 3.4 Summary
- 
4. Closed-loop traffic light control: a realistic case study
    - 4.1 System setup and problem statement
    - 4.2 Data collection and experimental results
    - 4.3 Comparisons and discussions
- 
5. Conclusion and Recommendations
    - 5.1 Conclusion
    - 5.2 Recommendations for further research
- 
6. Appendix
    - 6.1 Tuning result for different combination of hidden layers size
- 
7. Bibliography

# Acknowledgments

First of all, I would like to express my deepest gratitude to my supervisor Professor Su Rong, who always gives me constructive ideas and ardent guidance for my research work. Whenever I got stuck in my research process, he guided me patiently and helped me analyze the problem I faced. I could not complete my MEng project without his effort and help.

I also want to thank the SMEL lab that authorizes me the accessibility to the BSM data for my first project. Also, I want to express my gratitude to my teammate Sun Chunyang who helped me constrict the traffic network in the Vissim and provided the simulation results for both of my projects. He also helped me visualize the result of my simulation on the Google map, which made the data more comprehensible.

Thirdly, I would also like to thank Dr. Zhang Yicheng who helped me debug my codes and integrate my model into his research work on traffic signal control. I feel really lucky to be able to work with him.

Last but not least. I want to thank my parents who give me meticulous care and financial support on my three-year MEng program. I love them.



# Abstract

Increasing transportation efficiency is an interesting and important problem. In the world with convenient means of ICTs, the concept of “smart city” emerged. In the meantime, a lot of data-driven traffic network optimization algorithms have also been developed and applied widely. However, the performance of some optimization algorithms can be improved with some pre-works added. This thesis discusses two such pre-works. The first pre-work is urban traffic network congestion region identification and prediction with two case studies at NTU campus and Jurong area, which utilizes the vehicle data (average speed, GPS-based location, heading direction) via V2X to analyse the traffic condition of each link. Links with similar congestion levels will be clustered together into a region. Our simulation-based case studies show that about 75% of the total queue delay could be reduced with good knowledge of congestion regions in the network. The second pre-work is about traffic network turning ratio prediction, which may be useful in developing more accurate network dynamic models. By constructing a recurrent neural network to predict the vehicle turning ratios at the next time step with prior or online-learned knowledge of network supply functions, traffic light schedules and historical vehicle turning ratios as inputs. This prediction model can be integrated with a real-time traffic signal control algorithm to form an adaptive closed-loop traffic signal control strategy, which in our simulated case studies decreases 24% of the delay time compared to the case without turning ratio prediction.

# List of Figures

Fig.1 RNN unit and the unfold structure display	Pg.13
Fig.2 Different clustering algorithms	Pg.15
Fig.3 Structure of ensemble learning	Pg.21
Fig.4 Example link number labelling and node number labelling	Pg.27
Fig.5 Example of calculating the number of nodes between two links	Pg.27
Fig.6 A Fundamental Diagram for the Congestion Identification	Pg.29
Fig.7 Unfolded general RNN model	Pg.30
Fig.8 Encoder and Decoder part	Pg.33
Fig.9 Installed On-Board Unit	Pg.39
Fig.10 Labelled school campus	Pg.40
Fig.11 Well-tuned Fees-forward Neural Network	Pg.42
Fig.12 Comparison between output and label	Pg.42
Fig.13 Visualizing the identifying result on NTU campus	Pg.43
Fig.14 Overall view of Jurong area transportation network	Pg.44
Fig.15 Single junction layout	Pg.44
Fig.16 Jurong East & West Link Numbers & Junctions (Main area: D4, D5)	Pg.45
Fig.17 J.E. & W. Link No. & Juns. (Main area: E3, E4, partial D3, partial D4)	Pg.45
Fig.18 J.E. & W. Link No. & Juns. (Main area: Main area: D3, E3))	Pg.45
Fig.19 Two RNN models for speed and density prediction	Pg.46
Fig.20 Comparison with actual and predicted congestion level	Pg.47
Fig.21 The largest number of junction vs different number of clusters	Pg.48
Fig.22 Computation time cost for each algorithm	Pg.49
Fig.23 Average congestion level of each cluster	Pg.49
Fig.24 Four traffic light phases with traffic flow allowed	Pg.53
Fig.25 time length and schedule of each traffic light phase	Pg.53
Fig.26 Basic structure of FNN	Pg.56
Fig.27 Basic structure of RNN	Pg.57
Fig.28 Basic structure of LSTM	Pg.58
Fig.29 3*3 traffic network built in VISSIM	Pg.60

Fig.30 Prediction accuracy with different combination of hidden layer size	Pg.61
Fig.31: Comparison with the labeled value and the FNN structure	Pg.62
Fig.32 Prediction accuracy with different combination of hidden layer size	Pg.63
Fig.33 Construction parameters of the tuned FNN	Pg.63
Fig.34 Prediction accuracy with respect to different historical time step used	Pg.64
Fig.35 RNN model with three hidden layers	Pg.65
Fig.36 Tuned RNN with parameter labelled	Pg.70
Fig.37 Ensemble learning model with three independent predictors included	Pg.71
Fig.38 Block diagram for the first experiment process	Pg.76
Fig.39 Block diagram for the second experiment process	Pg.76
Fig.40 Improved Block diagram for the second experiment process	Pg.77
Fig.41 Simulation result for the traffic light control algorithm with different method integrated	Pg.78
Fig.42 Simulation result for the traffic light control algorithm with different turning ratio prediction model integrated	Pg.79

## List of Tables

Table 1 Standard of Level of Service identification	Pg.10
Table 2 Sample data set	Pg.40
Table 3 Sample training data set for training	Pg.41
Table 4 Congestion level identified by MATLAB	Pg.43
Table 5 Sample data collected from VISSIM	Pg.46
Table 6 Congestion level prediction accuracy with different time horizon	Pg.48
Table7 Turning ratio prediction accuracy with different time horizon and sample quantity	Pg.64

# Chapter 1: Introduction

## 1.1 Introduction to traffic networks and traffic light control

With the rapid development of urbanization, the number of vehicles increased exponentially with the improvement of policy effectiveness and income standard. In the meantime, the urban transportation network also expands both in size and complexity. However, the expanding transportation network still cannot adapt this change itself due to many reasons such as the fixed traffic light schedule, delay awareness of the traffic condition. As a result, problems like traffic congestions and traffic accidents occur more frequently, which worsen the traffic status in return. Thus, an efficient solution is needed, which could handle all these problems well. Two decades ago, the concept of “smart city” was first considered in Dubai [1], and many smart methods are suggested which help to build the smart city [2]. **For example, adaptive traffic signals which modifying the traffic signal time by analysing the real-time vehicle data and the traffic flow pattern have been introduced by other researchers [3].** Thus, having a better understanding of current traffic network conditions and the patterns of dynamic traffic flows is the key to increase the efficiency of those optimization solutions. Traffic signals are designed to eliminate many conflicts by assigning right of way. A good signalized traffic control strategy can increase the intersection capacity and reduce the frequency of certain types of crashes such that reducing vehicle travelling delays, balancing traffic flow, and improving operational efficiency of an urban street network.

## 1.2 Motivations for congestion region and turning ratio analyses

According to the survey result shown in Texas A&M Transportation Institute's 2019 Urban Mobility Report, commuters in L.A. spend in average 119 hours a year stuck in traffic. Although the traffic congestion statics in Singapore is better than L.A., commuters in Singapore still spend extra 16 mins per a 30-mins trip in the morning peak and 18 extra mins in the evening peak.[1] Eliminating traffic congestions becomes the most essential work in the improvement process.

Solving the congestion problem for the whole traffic network is time-consuming and requires highly efficient computational capabilities. The effectiveness decreases, when the complexity of the network increases. According to observation, congestion commonly appears in regions. Dividing a large traffic network into small regions and solving signal control problems locally and simultaneously would increase the efficiency greatly, which could be applied in different traffic networks easily.

Traffic light control plays a significant role in adjusting the traffic network performance. A fixed traffic light schedule is widely used due to its simplicity. Nevertheless, this schedule will aggravate the congestion level in some situations, e.g., during peak hours. To overcome this drawback, an adaptive traffic light control strategy is put forward by researchers, which considers the real-time traffic condition and adjusts the green time of each phase accordingly, which could reduce the traffic delay time significantly. A good turning ratio prediction model is critically important for deriving a high-quality network dynamic model, which may significantly improve the performance of a traffic network.

## 1.3 Literature review

### ***A. Congestion Identification***

Upon the literature review, many researchers tried to provide different definitions of traffic congestion. In general, those definitions can be broadly categorized into three groups based on the feature they are referring to. The first group is the method that defines the congestion based on the road capacity and the traffic flows. For example, in the report published in 1999 by ECMT (European Conference of Ministers of Transport), "Congestion is the impedance vehicles impose on each other, due to the speed-flow relationship, in conditions where the use of a transport system approaches its capacity" [4]. The second group is the method that defines the congestion based on the delayed traveling time. For example, in the paper published in 2001 by Weisbrod, Varyand Treyz, "Traffic congestion is a condition of traffic delay (when the flow of traffic is slowed below reasonable speeds) because the number of vehicles trying to use the road exceeds the traffic network capacity to handle them" [5]. The third group is the method that defines the congestion based on the cost occurred on the road. For example, in the paper published in 2005 by VTPI (Victoria Transport Policy Institute)," Traffic congestion refers to the incremental costs resulting from interference among road users" [6]. Although various definitions have been proposed, there is no universally agreed definition of traffic congestion [7]. However, by reading the above definitions, it could be summarized that congestion is the phenomenon when the traffic flow exceeds the designed link capacity.

Furthermore, some researchers also provide a set of criteria that a good congestion measurement should meet. In 1992, Turn suggested that measures to quantify the level of congestion should (i) deliver comparable results for various systems with similar congestion levels, (ii) accurately reflect the quality of service for any type of systems, and (iii) be simple, well-defined and easily understood and interpreted among various users and audiences [8]. In 1996, Levinson and Lomax suggested that a congestion index should (i) be easy to communicate, (ii) measure congestion at a range of analysis levels (a route, subarea or entire urban region), (iii) measure congestion in relation to a standard, (iv) provide a continuous

range of values, (v) be based on travel time data because travel time-based measures can be used for multimodal analysis and for analyses that include different facility types, and (vi) adequately describe various magnitudes of congested traffic conditions.[9] And in 1997, Lomax indicate that an ideal congestion measure would have (i) clarity and simplicity (understandable, unambiguous and credible), (ii) descriptive and predictive ability (ability to describe existing conditions, predict change and be forecast), (iii) statistical analysis capability (ability to apply statistical techniques to provide a reasonable portrayal of congestion and replicability of result with a minimum of data collection requirements), and (iv) general applicability (applicability to various modes, facilities, time periods and scales of application) [10]. Many other similar suggestions are also provided by researchers. Thus, it could be summarized that the measurement of traffic congestion should be:

- Simple and clear for the audience;
- Generally applicable with commonly available traffic data in typical traffic networks;
- Descriptive and predictive to facilitate a congestion prediction model;
- Continuously valued instead of being discrete and range-based.

Based on the criteria listed above, congestion level indicators could be assessed before being decided where to use. The Texas Transportation Institute (TTI) is a leader in developing measurements for determining congestion. There are four most commonly used measures based on mobility developed by them [11] [12]:

1. Volume-Capacity Ratio (V/C Ratio), which is the ratio of the number of vehicles passing through (V) over the number of vehicles that could theoretically pass through when at capacity. The traffic condition is said to be not congested if the ratio is less than one and congested if the ratio is larger than one.

2. The Level of Service (LOS). In this measurement, six levels are used from rank A (free-flow) to rank F (over-saturated) – that indicates how well the roadway or intersection is serving its intended traffic which is based on V/C ratio as well. The specific threshold for each rank is shown below [13]:



Level of Service	Operating Conditions	V/C ratio for arterials
Level-of-service A	Represents a free flow. Individual users are virtually unaffected by others in the traffic stream. Freedom to select desired speeds and to manoeuvre within the traffic stream is extremely high.	0.00 to 0.60
Level-of-service B	Represents the range of stable flow but the presence of other users in the traffic stream begins to be noticeable. Freedom to select desired speeds is relatively unaffected but there is a slight decline in the freedom to manoeuvre within the traffic stream from LOS A.	0.61 to 0.70
Level-of-service C	Represents the range of stable flow but the selection of speed is affected by the presence of others. Manoeuvring within the traffic stream requires substantial vigilance on the part of the user.	0.71 to 0.80
Level-of-service D	Represents high-density but stable flow. Speed and freedom to manoeuvre are severely restricted.	0.81 to 0.90
Level-of-service E	Represents operating conditions at or near capacity level. All speeds are reduced to a low but relatively uniform value. Freedom to manoeuvre within the traffic stream is extremely difficult.	0.91 to 1.00
Level-of-service F	Represents forced or breakdown flow.	Greater than 1.00

Table 1: Standard of Level of Service identification

The main advantage of LOS is comprehensible to the non-technical audiences by using description instead of a certain index. However, LOS is a rank-based method that could result in a sudden change in the operation condition. The use of LOS sometimes generates misleading results, especially when the condition is near a threshold.

3. Travel Rate Index (TRI) /Travel Delay, which calculates the ratio of average peak travel time over an off-peak (free flow) standard. For example, an index of 1.5 indicates that a 20-minute free-flow trip takes 30 minutes in a specific traffic condition. The advantage of TRI is its wide applicability owing to its continuously-ranged outputs. The weakness is that the method does not show the traffic condition explicitly, thus, making it not easily understandable.

4. Percentage of Congestion Travel, which describes the percentage of congested vehicle-miles of travel with respect to the total vehicle-miles of travel is used. The advantage and weakness of this method is the same as those of TRI.

In summary, none of the measurements provided a systematic and comprehensive analytical framework to quantify the relationship between the presence of public transport and the amount of traffic congestion. In fact, a balance between the comprehensive and data-driven results should be made when selecting a proper congestion measurement method. In this thesis, a new congestion measurement method is introduced based on the criteria discussed

above, which aims to find a good balance between comprehensibility (by using three levels to describe the traffic congestion) and predictiveness by considering the trends that include the differentiation when calculating the congestion level index.

## ***B. Predictors***

Traffic congestion prediction plays an important role in intelligent transportation as it has multiple applications in improving traffic network operation efficiency and integration. For example, by predicting the congestion area with high accuracy, some traffic network system management optimization algorithms may significantly improve the quality of their solutions in terms of reducing the congestion. Based on different objectives of a prediction model, three congestion prediction problems could be formulated, i.e., predicting the travel time, predicting the traffic congestion and predicting the traffic volume. This thesis only focusses on traffic congestion prediction.

A general algorithm that could be applied in solving this problem is FFT (Fast Fourier transformation) with the key idea of the decomposition. We know that a reasonably continuous and periodic function can be expressed as the sum of sine terms. The weekly behaviour of the traffic network is observed to be typically periodic. The reason is that most users in the network have highly repetitive weekly schedules, especially during peak hours, resulting in the repetitive traffic patterns in the traffic network. Thus, by analysing the historical periodic traffic congestion curve (e.g. traffic flow, average speed, etc.), a series of sine functions with different amplitudes, frequencies and phases could be calculated and summed up to match the curve. However, the congestion prediction problem discussed is more relevant to the short-term prediction problem. For large discrete time intervals, a situation will eventually be reached where it is no longer possible to theoretically establish a stable correlation model with other detection locations within the traffic network [14], making long-term forecasts practically useless at this point. Thus, FFT is typically not suitable for short-term prediction which requires the predictor to predict the traffic congestion in minutes instead of in weeks. In contrast, many short-term congestion prediction models are

proposed in the past two decades. Some researchers focus on single-site prediction based on one-dimensional traffic time series such as the ARIMA /SARIMA model [14] and the k- nearest neighbour (KNN) method [15].

ARIMA (autoregressive integrated moving average) is well suited for predicting the value of a dependent variable according to time. ARIMA is a generalized model of Autoregressive Moving Average (ARMA) without the requirement on the stationarity of the time series. It combines the Autoregressive (AR) process and Moving Average (MA) process and builds an integrated model of the time series. Autoregressive is a regression model that uses the dependencies between an observation and a number of lagged observations. Moving average is an approach that takes into account the dependency between observations and the residual error terms when a moving average model is applied to the lagged observations

KNN is much easier to understand. The fundamental assumption of KNN algorithms is that future states to be forecasted are similar to a neighbourhood of the past more or less. K is the number of neighbours the algorithm tries to choose from for historical data in accordance with the similarity between them. Those k neighbourhoods will be analysed and summarized to predict future states.

Both methods are well developed and improved over years with a rich family of the parametric algorithms being proposed, and the performances are proven promising. Although the good performance of ARIMA was frequently reported [16], it faces a computational challenge which makes it difficult to be implemented in the real-time transportation systems. While most existing KNN algorithms are single-stepped, which is easy to compute and has higher flexibility to be extended for solving multivariate problems by adding more data, KNN is sensitive to noisy neighbour and may generate overlapping nearest neighbours when it is extended to multi-step forecasting.

## Random forest algorithm

In the meantime, RNN (recurrent neural network) shows its outstanding ability in solving seq2seq (sequence to sequence) prediction problems such as speech recognition, language modelling, translation, image captioning, etc. It benefits from its unique characteristic: persistence of information. Traditional predictors predict the future points by analysing its neighbours or last few cycles, which means a short memory is typically used. Recurrent neural networks, on the other hand, can address this issue effectively. They are networks with loops in them, allowing information to persist.

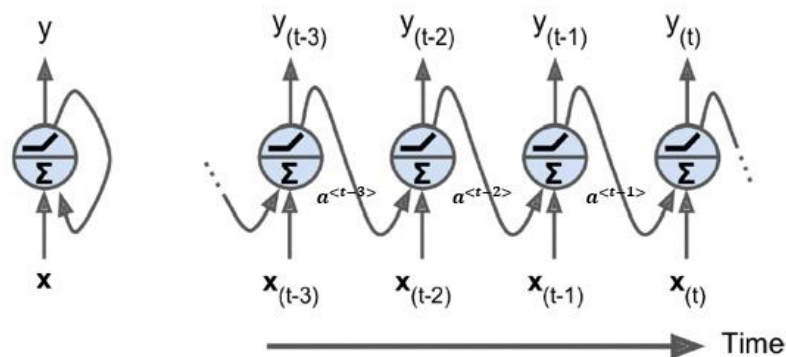


Fig.1 RNN unit and the unfold structure display

The unfolded structure of RNN is like a chain. This nature reveals that they are intimately related to sequences and lists. At each step, the model will calculate the output by analysing both the historical data group and the memory persistent in the well-trained parameters assigned to each node.

Training a recurrent neural network is the process to estimate a series of hypermeters which makes the model most suitable for our problem. Some hypermeters are listed below:

- Number of steps, which determines how many historical time steps need to be used. For example, if the number of steps equals 3, it means that historical data recorded as  $[X^{(t-2)}, X^{(t-1)}, X^{(t)}]$  will be used to determine its output  $Y^{(t)}$

- Batch size, which indicates the number of sample-label sets used for training. For example, if the batch size is equal to 2 with the number of steps equal to 3, it means the data set  $\{[X^{(t-3)}, X^{(t-2)}, X^{(t-1)} \sim Y^{(t-1)}], [X^{(t-2)}, X^{(t-1)}, X^{(t)} \sim Y^{(t)}]\}$  is used to train the RNN at each iteration.
- Structure of the network, which includes two items, i.e., the number of hidden layers and the number of neurons for each hidden layer.
- Activation function, which is used to add the nonlinearity to the network such that the network could be trained to solve much more complex problems. Proper activation could also be used to avoid the gradient vanishing and exploding problem.

In addition, some other parameters or components can be set at default values such as the drop rate, memory unit, optimizer, etc. Generally, while dealing with a simple problem, a rough range of selection for each parameter will be determined first before training the network with a different combination of alternative values. The performance of the trained network will be further examined by the cross-validation data set. However, the chain-like structure and the depth of the loops make RNNs difficult to train because of the vanishing or blowing up gradient problems during the backpropagating process.

There have been a number of attempts to overcome the difficulty of training RNNs over the years. These difficulties were successfully addressed by the Long Short-Term Memory networks (LSTMs) [17], which is a type of RNN with gated structure to learn long-term dependencies of sequence-based tasks. As a representative deep learning method handling sequence data, LSTMs have been proven to be able to process sequence data and applied in many real-world problems, like speech recognition [18], image captioning [19], music composition [20] and human trajectory prediction [21]. In recent years, LSTMs have been gaining popularity in traffic forecasting due to their ability to model long-term dependencies. Several studies [22-30] have been done to examine the applicability of LSTMs in traffic forecasting, and the results demonstrate the advantages of LSTMs. However, it is still a big challenge to predict larger-scale transportation network traffic. Most existing studies utilize

traffic data at a sensor location or along a corridor, and thus, network-wide prediction could not be achieved unless N models were trained for a traffic network with N nodes [23]. Thus, learning complex spatial-temporal features of a large-scale traffic network by using only one model should be explored.

Besides, some alternative prediction algorithms that can also be used for performance comparison: support vector machine (SVM), extreme learning machine (ELM), random forest (RF), gradient boosting decision trees (GBDT).

SVM blends linear modelling within stance-based learning, it selects a small number of critical boundary samples from each category and builds a linear discriminant function that separates them as widely as possible.[51] In the case that no linear separation is possible, the technique of kernel will be used to automatically inject the training samples into a higher dimensional space and to learn a separator in that space. SVM is acknowledged to be among the most reliable and accurate algorithms in most Machine Learning applications.

Extreme Learning Machine is a recently available learning algorithm for single layer feedforward neural network. [52] Compared with classical learning algorithms in neural networks e.g. Backpropagation, ELM can achieve better performance with much shorter learning time. In some of the existing work, it is claimed to yield better performance in comparison with SVM.

Random forest is an ensemble learning method for both classification and prediction problems.[53] It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. RF corrects for decision trees' habit of overfitting to their training set. The gradient boosting method represents an ensemble of single regression trees built in a greedy fashion. It produces a prediction model in the form of an ensemble of weak prediction models, such as decision trees. Stochastic Gradient Boosting Trees (GBDT) [54][55] combines gradient boosting with

bootstrap bagging. At each iteration of the algorithm, a new decision tree model is built based on the residuals of the previous decision trees. GBDT is a simple yet very effective method for learning non-linear functions [56].

### C. Clustering Algorithm

As the networked world continues to expand, the amount of information in the network has grown rapidly. Mining these messages by evaluating those data manually is not realistic. Thus, a lot of data analyzing algorithms are developed, including the data clustering algorithm that is widely used in many fields. The clustering algorithm is an unsupervised learning algorithm which means the data do not need to be labelled before being processed. The objective of the clustering algorithm is to divide disorganized data into different groups based on their features. In other words, data in the same group has a high similarity among each other. Figure 2 shows how the clustering algorithm works:

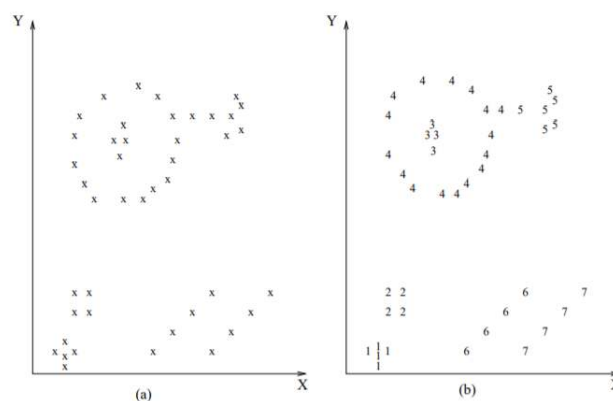


Fig.2 Different clustering algorithms

Generally, clustering algorithms can be categorized into hierarchical clustering methods, partitioning clustering methods, density-based clustering methods, grid-based clustering methods, and model-based clustering methods. More details are provided below:

- Hierarchical clustering methods: a hierarchical clustering algorithm divides the given data set into smaller subsets in a hierarchical manner. The basic steps of hierarchical clustering could be summarized as follows:

- Step one: Calculate the similarity between all the data points and the other points
- Step two: Cluster two points with smallest similarity
- Step three: Calculate the central point of the cluster and the similarity between this central point and the remaining point
- Step four: Go back to step two until one cluster including all the data points is formed.

The strength of this algorithm is easy to understand and straightforward to program. And by using the hierarchical clustering algorithm, the number of clusters no need to decide before applying it. A proper number of clusters could be selected based on the result obtained. However, this algorithm also has a significant weakness. The computational complexity in step one is  $O(n^2)$ , which means the computational cost will increase exponentially towards a large data set.

- Partitioning clustering methods: The partitioning clustering algorithm also classifies the data set into multiple subsets based on the similarity. Representative partitioning clustering algorithms are K-means clustering, K-medoids clustering and CLARA (*Clustering Large Applications*) algorithm. K-means clustering is one of the most commonly used clustering algorithms. Instead of calculating the similarity between each data point, the K-means algorithm only take few reference points into consideration. The basic steps of K-means clustering algorithm could be summarized as follows:

- Step one: Randomly select K reference points initially. Calculate the similarity between the remaining points and each reference point.
- Step two: All the remaining points will be clustering to one of the reference points with the highest similarity (smallest distance/cost)
- Step three: K clusters will be formed after step two. Calculate the imaginary central point for each cluster. K data Points with highest similarity to the imaginary central point will be selected as the new reference point
- Step four: Repeat step one to three until no changes to all the clusters.



The significant advantage of K-means clustering is its low computational complexity, which is  $O(K * n)$ , when compared to the hierarchical algorithm. However, the number of clusters  $K$  is required before running the algorithm which means this algorithm needs to be run multiple times in searching for a suitable cluster number. And K-means algorithm is sensitive to the outliers. With the consideration of reducing the impact of outliers, K-medoids clustering is proposed which, instead of taking the mean value of all the points in a cluster as a reference point, only uses the most centrally located points for calculating the mean value. The basic steps of K- medoids clustering algorithm could be summarized as follows:

- Step one: Randomly select  $K$  reference points initially. Calculate the similarity between the remaining points and each reference point.
- Step two: All the remaining points will be clustering to one of the reference points with the highest similarity (smallest distance/cost)
- Step three: Randomly select a data point (excluding the reference point). Compute the total cost  $S$  of swapping the initial reference point to this new point
- Step four: if the cost is less than 0 which means the quality of new clusters is higher than previous clusters. Replace the reference point with this new data point.
- Step five: Repeat step three and four until the convergence criterion is satisfied.

K-medoids clustering eliminates the impact of outliers by avoiding using the mean value. However, the computational complexity is increased to  $O(K(n - k)^2)$ . Thus, K-medoids clustering algorithm works effectively for small data sets but does not scale well for large data sets. So, CLARA is proposed as an extension to the K-medoids clustering algorithm to deal with data containing a huge number of objects. The basic process is similar to the K-medoids algorithm. Instead of checking all the non-reference point in searching for better clusters quality, CLARA only considers a small number of data with a fixed size and applies the same methodology to generate an optimal set of medoids for the sample while the efficiency depends on the sample size.

- Density-based clustering methods: different from the above two clustering algorithms, density-based clustering works by detecting areas where points are concentrated and where they are separated by areas that are empty or sparse. Especially, not all points will be clustered since some isolated points will be identified as noise. There are three different clustering algorithms belong to this category: 1. Defined distance (DBSCAN) which use a threshold distance to separate dense clusters from sparse noise. 2. Self-adjusting (HDBSCAN) which use multiple threshold distance to separate clusters of varying densities from sparser noise in a hierarchical way. 3. Multi-scale (OPTICS) uses the distance between neighboring features to create a reachability plot for the clustering reference.

Density-based clustering algorithm holds three advantages: 1. No predefined number of clusters is required. 2. Clusters formed by this algorithm can be of any shape including non-spherical ones. 3. High robustness: Able to distinguish the noise point which immune to the impact of outliers. However, this algorithm will fail if there are no density drops between clusters and sensitive to parameters that define density.

- Grid-based clustering methods: Grid-based clustering algorithm is similar to the density-based method which both of them cluster the data with high density. Grid-based algorithm first divides the whole data space into small rectangular cells. The data density of each cell will be calculated after then. Subspace with data density lower than a threshold will be removed while those higher than the threshold will form a cluster by combine the subspace adjoined to it. There are two main algorithms under this category: STING [31] and CLIQUE [32].

STING (Statistical Information Grid): STING is used as an information clustering algorithm with a hierarchical structure employed. The first level only has one cell which represents the whole space. The second level has four cells with each corresponding to one quadrant of the cell in the first level. This split will be continuously processed until a desirable number of layers is obtained. Statistical information of each cell is calculated and stored. The STING structure is widely used in auto-answering robots embedded in some service web as they can answer frequently asked questions efficiently. The basic steps of STING

could be summarized as follows:

- Step one: Determine a layer to begin with.
- Step two: Go through all the cells in this cell and a confidence index that this cell is relevant to the query will be calculated. After then, each cell will be assigned with a Boolean value indicating the relevance to the query based on the confidence index calculated.
- Step three: checking all the cells which labelled as relevant. If this cell is the bottom layer, output the information stored in this cell. If not, goes to the next level and repeat step two and three until reach the bottom cell.

The computational complexity is  $O(K)$ , where  $K$  is the number of cells in the lowest level and usually  $K \ll N$  (the number of data). This feature makes this algorithm efficient when dealing with the query. Besides, when data are updated, the information in the cell hierarchy do not need to be recalculated. Instead, an incremental update could be processed [33]. The disadvantage is that the confidence index is calculated in a possibility form, which may imply a loss of accuracy in query processing.

CLIQUE (Clustering in QUES): CLIQUE is a density-based and grid-based subspace clustering algorithm. It partitions the high-dimensional data space into non-overlapping rectangular units as introduced before. Dense units will be determined and connected in all subspace of interests. The advantage of this algorithm is this algorithm able to automatically find the subspaces of the highest dimensionality with arbitrary shape as long as high-density clusters exist. As in all grid-based clustering approaches, the weakness of this algorithm is the quality of the result highly related to the choice of the number and width of the petitions and grid cells.

Instead of assigning each point to just a single cluster, Expectation–Maximization (EM) clustering algorithm goes a step further and describe each cluster by its centroid, covariance, and weight.[57] The probability that a point belongs to a cluster is now given by a multivariate Gaussian probability distribution. This enables clusters overlapping each other as some points may assigned to multiple clusters.

Unlike clustering algorithms mentioned above, Affinity propagation does not require the number of clusters to be determined or estimated before running the algorithm.[58] Affinity propagation takes as input measures of similarity between pairs of data points, and simultaneously considers all data points as potential exemplars. Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerges.

#### ***D. Feature Selection***

As a learning machine, a neural network generates its output by interacting the input with the parameters stored inside the model. The quality of the training data set is essential in determine the quality of the neural network, as good training data can reduce the training time and enhance the performance of the network. However, in the cases with high-dimensional training data sets, it is not advised to use all the features as it will increase the computational complexity, resulting in extremely long training time and the overfitting problem. A good feature selection method should be able to figure out two types of data: irrelevant data and redundant data. Irrelevant features cannot involve in the learning process (e.g., student matric number is irrelevant to predict the student's GPA) and redundant features contain the same information and may mislead the learning process (e.g., purchase prices of a product and the amount of sales tax paid).

Two categories can be classified for the process of feature selection. Feature subset selection and feature ranking methods based on how the features are combined for evaluation. The feature subset selection method searches proper combinations of feature subsets by using some searching strategies such as a greedy forward selection, greedy backward elimination,

etc. Then, statistical measures or the supervised learning algorithms (e.g., the wrapper method) are used to evaluate those data subsets. The disadvantage of this method is that it generates  $2^N$  subsets from  $N$  features for evaluation. This method is obviously not applicable when dealing with high-dimensional data space. The feature ranking-based methods focus on individual features. In this method, each feature is ranked by a selection metric such as information gain, symmetric uncertainty, importance matrix, etc. and the top ranked features are selected as relevant features by a pre-defined threshold value. Compared to the feature selection method, this method is computationally cheaper, as the space complexity is not high. However, the disadvantage of this method is that it does not deal with redundant values.

### ***E. Ensemble Learning***

When a person is planning to purchase a high-valued product, e.g., a desktop, it is quite rare that he/she makes a purchase decision immediately based on the information received from the first shop entered. Instead, he/she may go to multiple similar shops and compare different models with reviews post by other buyers. Ensemble models in machine learning operate on a similar idea. Instead of adopting the output from a single model directly, ensemble learning summarizes multiple outputs by tuning different models before generating the output value. Many researchers have investigated the technique of combining the predictions of multiple predictors to produce a single prediction [34][35]. The ensembled predictors are generally more accurate than any of the individual predictors making up the ensemble.

Ensemble methods work best when the predictors are independent from one another. One way to get diverse classifiers is to train them using very different algorithms. This increases the chance that they will make different types of errors, improving the ensemble's accuracy. Another approach is to use the same training algorithm for every predictor, but to train them on different random subsets of the training set. These different random subsets can be obtained by using some techniques such as bagging and pasting. When sampling is performed with replacement, this method is called bagging [36] (short for bootstrap aggregating). When sampling is performed without replacement, it is called pasting [37]. In other words, both bagging and pasting allow training instances to be sampled several times across multiple predictors, but only bagging allows training instances to be sampled several times for the same predictor.

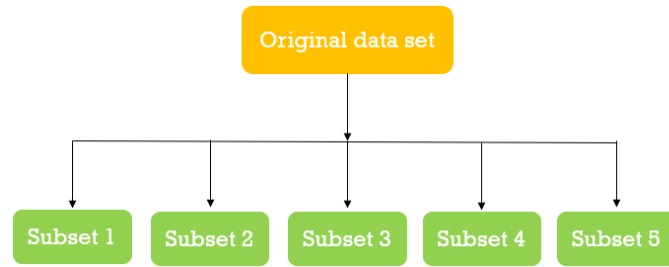


Fig.3 Structure of ensemble learning

After multiple predictors or classifiers are built, **four** simple but powerful aggregating techniques are frequently used: max voting, averaging, **median** and weighted averaging. The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a ‘vote’. The predictions which voted by the majority of the models are used as the final prediction. Similar to the max voting technique, averaging method take the average value of all the predictions as the output **and the median method take the median value as the output**. Weighted averaging is an extension of the averaging method. All models are assigned different weights defining the importance of each model for prediction. Outputs from these models are multiplied by this weight when calculating the average value.

### ***F. Traffic Light Control***

Traffic signals are designed to eliminate many conflicts by assigning right of way. A good signalized traffic control strategy can increase the intersection capacity and reduce the frequency of certain types of crashes such that reducing vehicle travelling delays, balancing traffic flow, and improving operational efficiency of an urban street network [38]. An engineering study of traffic conditions, pedestrian characteristics, and physical characteristics of location shall be performed to determine whether signal is warranted. This study shall include an analysis of factors related to the existing operation and safety at the study location and the potential to improve these conditions. There are two main types of current intersection signal control systems: 1. fixed-time, which includes staged-based and phase-based control system. 2. real-time adaptive signal control systems, such as SCAT [39] and SCOOT [40]

Fixed-time traffic light control strategy was widely utilized in the past when there is no huge demand on the transportation so it can solve almost congestion problem easily. However, this strategy can no longer meet the increasing transportation demand in the modern transportation system with exploding population of vehicles holders and limited road capacity. Now a days, many adaptive traffic signal controllers are developed based on the above control systems to optimize the travelling delays and control the traffic flow. Following are some state of art traffic light control algorithms formulated by other researchers with real case application.

Rongrong Tian, Xu Zhang [41] first use the TRANSYT traffic modelling software to find the optimal fixed-time signal plan. After then, they use VISSIM to affirm, evaluate the TRANSYT model. This model is used to assess the optimal signal plan. They also use VISSIM and VS-PLUS emulator to refine and evaluate an adaptive frame signal plan. The simulation result shows that delay in the adaptive signal control was shortened noticeably than that in the fixed time control.

Jianhua Guo et al [42] introduces a new method for area-wide traffic signal timing optimization under user equilibrium traffic. The optimization model was formulated as a multi-dimensional search problem which use the production of the sum of the travel time for each for each base station pair of regional urban street network and the variance of travel time of unit mileage for each base station pair as its objective. However, this objective function is not convex so the smallest product cannot mean that the two elements are both smallest. Multi-objective optimization model might offer more power in finding the best solution.

**Gustav Nilsson and Giacomo Como [43]** focused on a class of dynamic feedback traffic signal control policies that only requires information about the traffic volume in order to stabilize network. Stability is then proved by interpreting the generalized proportional allocation

controllers as minimizes of a certain entropy-like function that is then used as a Lyapunov function for the closed-loop system.

Junchen Jin and Xiaoliang Ma [44] proposed an adaptive group-based signal control approach capable of making decisions based on its understanding of traffic conditions at the intersection level. The control problem is formulated using a framework of stochastic optimal control for multi-agent system in which each signal group is modeled as an intelligent agent. The parameters were off-line optimized using a genetic algorithm. Simulation results shown that the proposed adaptive group-based control system outperforms the optimized GBVA control system mainly because of that's real-time adaptive learning capacity in response to the changes in traffic demand.

Nasser R. Sabar et al [45] proposed an adaptive memetic algorithm (MA) for optimizing signal timings in real world urban road networks using traffic volumes derived from induction loop detectors. This algorithm improves the current genetic algorithm (GA) by using a systematic neighborhood based simple descent algorithm as a local search to effectively exploit the search space around GA solutions and proposing an indicator scheme to control the local search application based on the diversity and the quality of the search process. This memetic algorithm accelerates the local search process compared to genetic algorithm.

Mohammad Aslani et al [46] utilized RL (Reinforcement learning) algorithms to design adaptive traffic signal controllers called actor-critic adaptive traffic signal controllers (A-CATs controllers). This controller takes traffic disruptions, discrete and continuous state actor-critic approaches and function approximation definitions into consideration. The simulation result shows the continuous A-CATs controller with the optimal function approximation outperforms the discrete one.



## 1.4 Contributions of this thesis

This thesis first provides the solution for the traffic congestion identification problem. By figuring out the congested region in the traffic network based on the vehicle data, economic benefit could be achieved since adaptive traffic light control algorithms could be applied on these targeted regions without significant degradation of network-wise performance. It is verified by the experiment carried out in this thesis that traffic delay time is reduced 75% compared to the case without knowing the congestion region. Also, a congestion level prediction model is proposed in this thesis. By using this model, potential traffic jam can be detected in some links and some prevention methods can be used to reduce or stop the negative impact of congestion for these links. Lastly, a traffic turning ratio prediction model is proposed in this thesis. The power of this model is shown in the experiment by integrating with the adaptive traffic light control strategy. With accurate turning ratio prediction, traffic light control algorithm works much more **efficiently**, and the total traffic queue time spend in the road can be reduced significantly.

## 1.5 Organization of the Thesis

The organization of the thesis is as follows. In chapter 2, part of the collected data with labels are trained with a simple feed-forward neural network such that all the newly collected data can be assigned with labels with respect to their current locations. Then, based on the data accumulated for every 15 seconds, the congestion level of each link is determined according to the average speed and vehicle density obtained. In the next step, two clustering algorithms are applied in solving the congestion region clustering problem. After discussing the result and performance of these two algorithms, a new hybrid clustering algorithm is developed which could highlight the over-saturated region with shorter computation time. Lastly, an RNN-based congestion level prediction model is constructed. The model could forecast the congestion level of each link that is very likely to occur in the following 15-second time step.

In Chapter 3 two machine learning models, i.e., a feed-forward neural network and an offline recurrent neural network (RNN), are trained for predicting the turning ratios. After that, performances of two models are discussed and compared. To increase the prediction accuracy of the RNN model, an ensemble learning method is introduced, which trains three independent RNN predictors instead of only one. This work could increase the stability of the prediction accuracy and robustness to the abnormal data collections. Finally, the well-trained model is integrated with an online optimization model, which aims at reducing the traffic delay time. The turning ratio prediction models are used in Chapter 4, together with real-time traffic signal control, to form a closed-loop adaptive traffic signal control strategy.

Chapter 5 summarizes all the results, and presents some envisioned future works, which could be carried out for other uses. All the tables and figures used in this thesis are listed in Appendix.

## Chapter 2: Traffic Congestion Region Identification and Prediction

In this chapter some concepts will be first defined. Then specific algorithms for congestion region identification will be introduced, and an improved clustering algorithm leveraging on existing ones will be discussed in detail. After providing detailed case studies with relevant comparisons, we provide some **concluding** remarks.

### 2.1 Basic concepts and problem statement

We first provide definitions of some key concepts that will be extensively used later.

- *Link and link number*: a *link* in a traffic network is defined as a uni-directional and one-lane road segment. In this thesis, each bi-directional road segment will be represented by two uni-directional links with opposite directions. A *link number* is a unique number assigned to a specific link. In this chapter, without loss of generality, we assume that the link label set is the set of consecutive discrete values, ranging from 1 to  $x$  (where,  $x$  is the total number of links in the target traffic network). The symbol used to represent a link is  $l_m$  where  $m$  is the corresponding link number

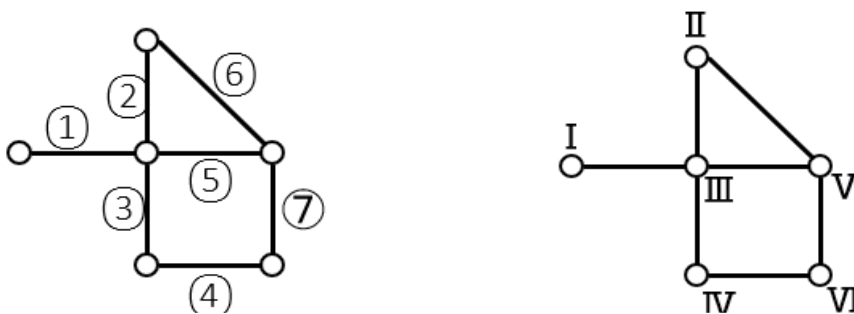


Figure 4: Example link number labelling (left) and Example node number labelling (right)

- *Node and node number*: a *node* in a traffic network is defined as an intersection, connecting with at least three links, having antagonistic traffic streams. It essentially represents either the beginning point or the endpoint of a link. Whenever a vehicle

reaches a node, a decision will be made as which link the driver intends to enter after leaving the current link. A node number is a unique number assigned to each node, taking from the set of consecutive discrete values ranging from 1 to  $x$  ( $x$  is the total number of nodes in the target traffic network). The symbol used to represent a node is  $N_m$  where  $m$  is the corresponding node number.

- The *number of nodes between two links*: it is defined as the minimum number of nodes required for a vehicle starting from one link to another. This number is used to represent the “distance” between two links while calculating the cost in the clustering algorithm. The symbol used to represent the number of nodes between two links is  $N_{xy}$  where  $x$  and  $y$  are the link numbers of these two links.

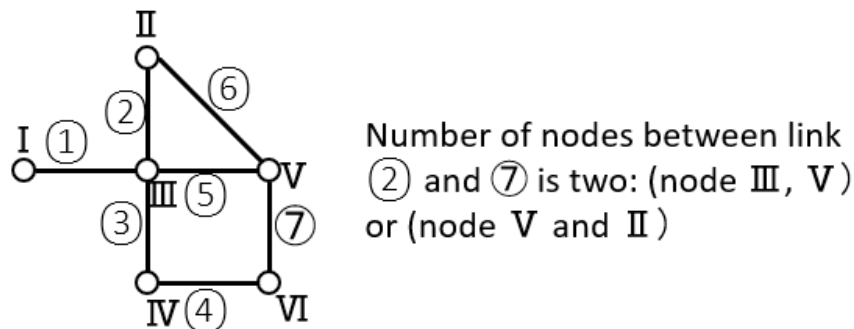


Figure 5: Example of calculating the number of nodes between two links

- *Link congestion level*: the link congestion level is defined to represent the traffic condition of this link. In this thesis, the traffic congestion condition is categorized into three levels, although more levels will not change the feasibility of our proposed approach. Each level is associated with two link characteristics: link speed and link available capacity (or link density). The first level expresses the under-saturated status where all the vehicle travelling in the target link drive at a free-flow speed and the link still has spare capacity for more vehicles to fit in, i.e., the link density is low. In this thesis, the label assigned to the first congestion level is 1 while calculating the cost in the clustering algorithm. The second level expresses the saturated status where the vehicles travelling in the target link drive at a saturation speed with no spare capacity for extra vehicles. The label assigned to the second congestion level is 2 while calculating the cost in the clustering algorithm. The third level expresses the over-saturated status where the driving speed is significantly

lower than the saturation speed and there is absolutely no spare capacity to accommodate any more vehicles. The value assigned to the third level is 3 while calculating the cost. A detailed rule on how to rigorously define these three levels will be discussed shortly. The symbol used to represent the congestion level of the link is  $C_m$  where m is the corresponding link number.

- *Road and road congestion level:* a *road* is a bi-directional route segment. As mentioned before, it is captured by two uni-directional links with opposite directions. The road congestion level is defined as the higher congestion level between the constituent links.
- The region, congestion level of region and congestion region: the region discussed in this thesis is defined as a cluster of roads in which all the roads are accessible from other roads in the same region. The congestion level of the region is defined as the average congestion levels of all the roads included in the region. A region is identified as a congestion region when the average congestion level is higher than a threshold value (e.g. 2.5).

## 2.2 Congestion level identification and prediction

The objective of this section is to formulate a new congestion measurement method in a proper way that meets most criteria discussed in the literature review. The criteria for a proper congestion measurement method, adopted in this thesis, are listed as follows:

1. Simple and clear for the audience to understand.
2. Generally applicable with typical traffic data to most traffic networks.
3. Descriptive and predictive

Based on these criteria, assessments are made to estimate the performance of the current commonly used congestion measures. Unfortunately, none of them provided a systematic and comprehensive analytical framework to quantify the relationship between the presence of public transport and the amount of traffic congestion. Recall the definitions of the congestion, the key is the relationship between the traffic flow and the link capacity since other features such as delay time and cost could be summarized as a consequence. The following figure depicts the fundamental diagram of the traffic flow versus density.

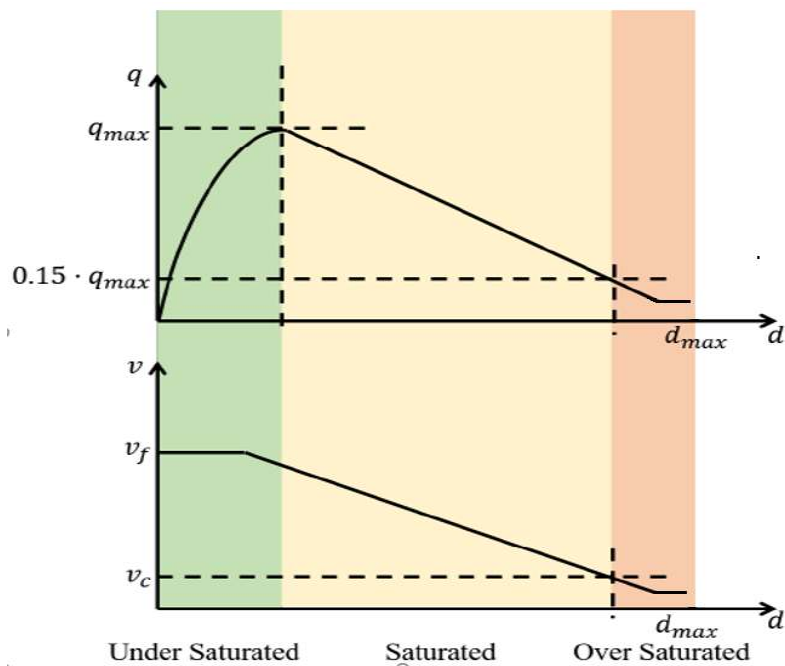


Figure 6: A Fundamental Diagram for the Congestion Identification

The first figure shows the relationship between the traffic flow which is denoted by  $q$  and the vehicle density which is denoted by  $d$ . It is obviously to be observed that the traffic flow is proportional to the vehicle density at the beginning until the traffic flow reaches its maximum number  $q_{max}$ . If keep increasing the vehicle density, the traffic flow will start to decrease until a certain value. The second figure shows the relationship between the average travelling speeds which is denoted  $v$  and the vehicle density. In the beginning, when the vehicle density is low, all the vehicles could travel at its free-flow speed. The speed will start decreasing when the number of vehicles reaches a threshold point. The average speed will decrease to a certain value and maintain it until the link reaches its maximum density. The traffic flow is estimated as follows:

$$q^{(t)} = v^{(t)} \cdot d^{(t)}$$

The token “t” indicates a specific time step as the average vehicle speed and vehicle density are calculated based on a fixed time period. Then, the partial derivative of the traffic flow rate over traffic density could be estimated as follows,

$$\left(\frac{\partial q}{\partial d}\right)^{(t)} = \frac{q^{(t)} - q^{(t-1)}}{d^{(t)} - d^{(t-1)}}$$

Based on the traffic flow-vehicle density curve shown in the fundamental diagram, a traffic condition-based congestion measurement is defined as follows:

$$\begin{cases} \frac{\partial q}{\partial d} \geq 0 \Leftrightarrow \text{Under congested,} \\ \left[\frac{\partial q}{\partial d} < 0\right] \wedge [q \geq 0.15 \cdot q_{max}] \Leftrightarrow \text{Saturated,} \\ \left[\frac{\partial q}{\partial d} < 0\right] \wedge [q < 0.15 \cdot q_{max}] \Leftrightarrow \text{Over saturated.} \end{cases}$$

If  $\frac{\partial q}{\partial d}$  is larger than zero, this link is labeled as undersaturated, as the green part and half of the yellow part shown in Fig. 9. If  $\frac{\partial q}{\partial d}$  is less than zero while the traffic flow rate is higher than 15% of its maximal value,  $q_{max}$ , the link is labelled as saturated. A threshold of 15% of  $q_{max}$  is set based on some empirical formula [47]. If  $\frac{\partial q}{\partial d}$  is less than zero and the traffic flow rate is lower than 15% of  $q_{max}$ , the link is labelled as oversaturated, as the red part shown in the fundamental diagram.

The performances of this method can be assessed based on the criterion.

1. Simplicity: this method relies on the fundamental diagram and two easily evaluated factors: the gradient and value of  $q$ .
2. Generality: this method is applicable to any traffic network, where its fundamental diagram is available.
3. Descriptive and Predictive: this method assigns traffic light colors to each congestion level which is easy to show on the map. In the meanwhile, this method identifies the congestion level based on the differentiation of traffic flow and vehicle density. This trend is further used for predicting the traffic condition in the next section.

In summary, the method proposed in this chapter qualifies three criteria above. While the output of this method is rank-based, the internal result is a specific value as calculated by the change of traffic flow divided by the change of vehicle density. Thus, the method proposed could be regarded as an acceptable method that could be applied to identify the traffic congestion **identification**.



## 2.3 Congestion Level Prediction

The objective of this section is to construct two well-trained, many-to-many recurrent neural networks which are able to predict the average speed and vehicle density respectively based on the historical data with an acceptable accuracy. As discussed in the literature, most participants in the traffic network follow a fixed periodic schedule. Thus, traffic congestion also shows patterns, especially during rush hours, e.g., 8 am to 9 am every Monday.

A general RNN model is shown below:

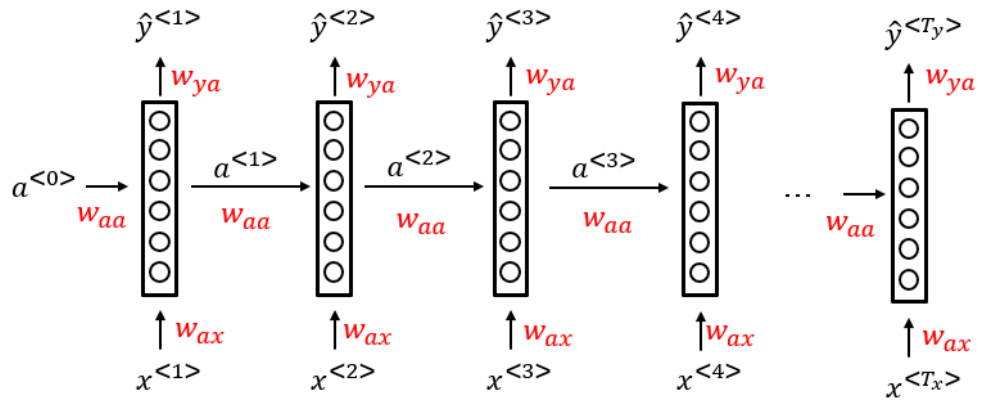


Figure 7: Unfolded general RNN model

$a^{<0>}$  is the initial memory which is pre-set to a default small non-zero value (-0.3~0.3).  $w_{aa}$  is the weight needed to be trained and multiplied by memory  $a^{<0>}$ .  $x^{<1>}$  is the first input value among the input sequence.  $w_{ax}$  is the weight needed to be trained and multiplied by input  $x^{<1>}$ . By summing up the above two results, the memory will be updated with the equation:  $a^{<1>} = g(w_{aa}a^{<0>} + w_{ax}x^{<1>} + b_a)$  where  $g$  is the activation function and  $b_a$  is pre-trained bias. Based on the new memory obtained, the first prediction result can be calculated with the equation:  $\hat{y}^{<1>} = h(w_{ya}a^{<1>} + b_y)$  where  $h$  is the activation function,  $w_{ya}$  is the weight needed to be trained and multiplied by memory  $a^{<1>}$ ,  $b_y$  is pre-trained bias. The above process will be repeated until a desired prediction result is calculated. In general, the process could be summarized as:

$$a^{<t>} = g(w_{aa}a^{<t-1>} + w_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(w_{ya}a^{<t>} + b_y)$$

To deal with the congestion prediction problem whose objective is to predict the congestion level in the next few time steps based on information of several previous time steps, a many-to-many RNN model is used, which is shown below. This model could be analyzed as two parts: encoder and decoder. These two components can be understood easily based on the general model introduced above. The encoder part (layers in the blue box) is the general RNN model with only the last output is calculated. The decoder part (layers in the green box) is the general RNN model with only one input and use the previous output as its new input.

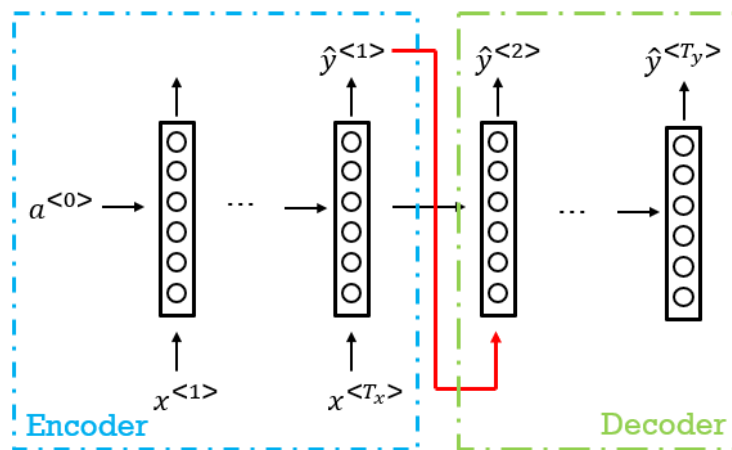


Figure 8: Encoder and Decoder parts

After training the above RNN model, the performance can be assessed by the testing data set which is neither used for training nor for parameter tuning. The equation for calculating the accuracy is defined below

$$acc = \frac{\text{No. of links whose predicted congestion level is correct}}{\text{total number of the links in the traffic network}}$$

## 2.4 Congestion region clustering

Previous works solve the problem as how to identify and predict the traffic congestion level. The objective of this section is to formulate a proper clustering algorithm based on the result obtained previously such that the congested region could be highlighted. Many clustering algorithms have been discussed in the literature review. From these clustering algorithms, K-means clustering and hierarchical clustering method are selected with two following reasons:

1. Clustering of congestion region is calculated based on the feature of the data instead of distribution. Density-based clustering methods and grid-based clustering methods which also take dense data into consideration are not suitable for this problem (please explain here why they are not suitable).
2. The size of a concerned congestion network is generally large. K-medoids is not suitable since this algorithm is only applicable and effective in small-scale problems.

When applying K-means clustering and hierarchical clustering algorithms, the key step is to calculate the similarity between two points. However, the general similarity formula only calculates the Euclidean distance between two points in the data space which is not feasible in this problem. New formulas for calculating the similarity between two points need to be developed first. There are two features need to be taken into consideration while calculating it:

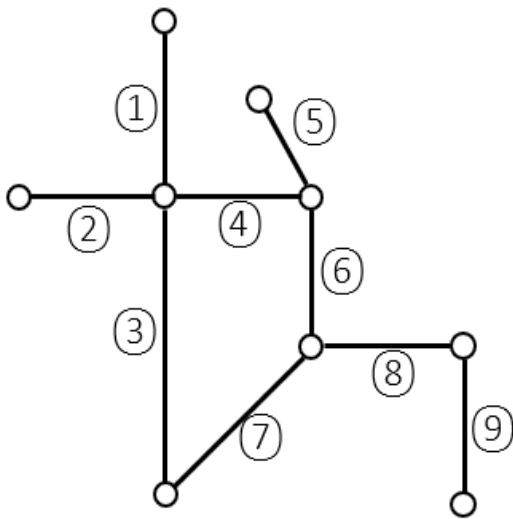
1. **Congestion level:** In this problem, congestion level is an abstract feature which is not computable. Thus, congestion levels should be transformed into specific values first before running the clustering algorithm. Concretely, in this thesis, the undersaturated status is assigned with 1, the saturated status is assigned with 2 and the oversaturated status is assigned with 3.
2. **Distance between two links:** the congestion situation in the links radiates its impact to its downstream and affected by its upstream link. Thus, neighboring links have a deeper impact on this link instead of the links in the distance. One way to calculate the distance

between two links is to calculate the geographic distance between the middle point of each link. However, the length of links in the traffic network varies a lot. Geographic distance between adjacent links may larger than the distance between two links which is far apart. Congestion region clustering is a link-based problem, the impact of the internal feature should be eliminated. Instead, the number of nodes between two links is used to measure the distance between two links.

Thus, the general formula used for calculating the dissimilarity between two links is shown below.  $N_{junctions}$  is the number of nodes between link  $i$  and the reference link  $m$ .  $C_i$  is the congestion level of link  $i$  and  $C_m$  is the congestion level of link  $m$ .  $\alpha$  and  $\lambda$  are hypermeters which could be set at users' preferences depending on which feature is more concerned. For instance, Increase  $\alpha$  will increase the dissimilarity between two links which is far part and increase  $\lambda$  will increase the dissimilarity between two links which has different congestion levels.

$$D_i^m = N_{junctions}^\alpha + (C_i - C_m)^\lambda$$

Besides, formulas for calculating the central point should also be developed since the general method will calculate the geographical center point and the link with the smallest distance will be selected as reference link in the next round. Obviously, this approach is not precise as this approach neglects the link distribution and could be influenced by some long links easily. By observing the congestion region, links in the center always hold smallest  $N_{max}$ , where  $N_{max}$  is the max number of nodes between this link towards all the other links in the same cluster. A simple example is shown below to make this concept understandable: figure in the left shows a congestion region with 9 links.  $N_{max}$  for each link is shown in the right table. Either Link 6 or link 7 can be selected as the central link in this case.



	$N_{max}$	w.r.t
link 1	4	link 9
link 2	4	link 9
link 3	3	link 9
link 4	3	link 9
link 5	3	link 9
link 6	2	link 1,2,3,9
link 7	2	link 1,2,3,9
link 8	3	link 1,2
link 9	4	link 1,2

### 2.4.1 K-means clustering

When applying the K-means clustering algorithm, besides the new dissimilarity formula and the reformulated method in searching for the central links, the last problem is how to choose the proper number of clusters. In general, there are two methods that are widely used for determining the number of clusters. The most widely used method is the elbow method. In this method, K will be set as 2 initially and the clustering algorithm will be run multiple times with each time increase K by 1. By plotting the global  $N_{max}$  among all the links for each K, an elbow point could be observed, the value of K corresponding to this elbow point is regarded as the best number of clusters. Thus, an adjusted K-means clustering could be applied with the following process:

- Step 1: Pre-set the number of clusters K equal to 2 initially and choose K links randomly as reference links
- Step 2: Assigning the rest links to one of these K links with respect to the dissimilarity calculated by the formula above. With the consideration of link connectivity, links will only be clustered with the reference link when the link is accessible for the chosen link.
- Step 3: After clustering all the links in the network. For each cluster, a link with the smallest  $N_{max}$  will be selected as the central link, which is also the new reference link. If multiple links have the same smallest  $N_{max}$ , all these links will be selected as reference links. An additional clustering algorithm will be applied on each reference

link, clusters with the smallest global  $N_{max}$  win the completion. The link which forms this cluster will be selected as a new reference link. If this result remains same, reference links will be selected randomly from them.

- Step 4: Repeating steps 2 and 3 until no changes to all the regions. Global  $N_{max}$  is recorded among all the links
- Step 5: Go back to step 1 and increase the number of clusters K by 1 until reaches a threshold value.
- Step 6: Plot the K -  $N_{max}$  curve, selecting the best number of clustering by observing the elbow point. Clusters obtained by using the corresponding K will be selected as the output of this algorithm.

## 2.4.2 Hierarchical clustering

The general hierarchical clustering algorithm clusters the data based on their distance. Similarly, an adjusted algorithm could be formulated based on the dissimilarity function formulated previously:

- Step 1: Calculate the dissimilarity between each two links in the traffic network based on the dissimilarity function. Cluster all the links with lowest dissimilarity, in this case, those links are neighboring links with same congestion level ( $N_{junctions} = 1$  ,  $C_i - C_m = 0$ ).
- Step 2: Cluster remaining link or clustering with second lowest dissimilarity.
- Step 3: Repeat step 2 by increasing the dissimilarity until all the links are included in one cluster and a hierarchical tree diagram could be obtained.
- Step 4: From the top of the hierarchical tree, select a suitable threshold that divides the links into a certain number of clusters.

## 2.4.3 Improved the K-means clustering algorithm

In practice, people pay more attention to the most congested region as it plays a dominant role in the traffic jam. And most traffic control optimizing algorithms are also

focusing on relieving the traffic condition in those areas. Thus, it is more meaningful to highlight the most congested region instead of taking all the links into consideration while applying the clustering algorithm. Based on this concern, additional constraints are needed while applying the adjusted K-means clustering algorithm. In this thesis, a threshold of the average congestion level among all the links in the cluster is selected as the constraint. Only the clusters with an average congestion level above the threshold are kept after running the algorithm. However, if this constraint is only applied at the last step, no result will be output because the average congestion level of all the clusters is likely to be lower than the threshold because they are generated by clustering all the links in the traffic network. And if we apply this constraint every time when assigning a link to a cluster, the computational cost will explode as many undersaturated links will be clustered multiple times during each clustering progress. The algorithm needs to be reformulated and an improved K-means clustering algorithm which only focuses on the most congested region, as proposed below:

- Step 1: Choose one over-saturated link as the reference/starting link.
- Step 2: Cluster its neighboring links (upstream/downstream) with the same congestion level (over-saturated).
- Step 3: Cluster all the neighboring links (upstream/downstream) of the cluster with a congestion level difference up to 1 while keeping the average congestion level above a pre-defined threshold.
- Step 4 (optional): If all adjacent links to the clustered links are not qualified for the constraint in step 3. A tolerance number of links is introduced, which will include multiple adjacent links first regardless of the constraint and cluster these links' adjacent links. If the clustering this link could maintain the average congestion level above the threshold value, these two links will be all clustered.
- Step 5: Repeat step 2-4 until all the adjacent links of this cluster are unqualified for the constraint
- Step 6: Choose another over-saturated but not clustered yet link as the new starting link and go back to step 2 until no over saturated link is left.

## 2.5 Simulation-based experiments

In this section, two case studies are carried out in order to test the performance of the methodology proposed above. The first experiment is to test the traffic congestion level identification algorithm in the on-campus traffic network. The second experiment is to test the traffic congestion level identification and traffic congestion region clustering algorithm in the Jurong area traffic network.

### 2.5.1 Case study one: Traffic congestion level identification (target: campus traffic network)

The first experiment is testing the congestion level identification algorithm based on the traffic data in the campus, the process is summarized as follow:

#### Phase 1: Data collection with the on-board unit (OBU)

In the first experiment, all on-campus data used were collected via OBU pre-installed in selected vehicles, including personal vehicles and school shuttle buses, running on the campus of Nanyang Technological University. Those data were sent from these vehicles to a server in the Smart Mobility Experience Lab (SMEL), as shown in figure 9.



Figure 9: Installed On-Board Unit



Data received were categorized into two types: pedestrian data and vehicle data. Only vehicle data were used in this thesis. Each set of vehicle data contains information including received time, vehicle ID, vehicle type, vehicle class, GPS data, heading, and speed, as shown in the table below. Vehicle ID is a unique series of numbers and vehicle types use numbers ranging from 0-255 to indicate vehicle type. For example, 4 indicates cars, 6 indicates buses. Heading determines the direction this vehicle is currently driving in. Heading starts with North at  $0^{\circ}$  and increases in a clockwise manner. This information is essential for determining each specific link that belongs to a road, as each road contains two directions (directed links). Our data were only collected when the vehicles were inside the campus transportation network which was the project region meant for testing. Data were only collected when the vehicles were inside the campus transportation network which was the project region meant for testing.

1	receivedT	type	vehicleType	vehicleClass	vehicleId	sourceRsuld	latitude	longitunde	speed	heading
2	1.5E+12	vehicle	6	0	27103368	26	1.3554393	103.68515	26.136	238.3
3	1.5E+12	vehicle	6	0	27100296	12	1.3464463	103.68727	0.792	130.7
4	1.5E+12	vehicle	6	0	26650760	27	1.3546828	103.68378	23.184	55.7375
5	1.5E+12	vehicle	6	0	27100296	12	1.3464046	103.68731	0.792	133.7375
6	1.5E+12	vehicle	6	0	27103368	26	1.355416	103.68512	26.568	235.325
7	1.5E+12	vehicle	6	0	26650760	27	1.3547011	103.6838	22.536	55.375
8	1.5E+12	vehicle	6	0	27100296	13	1.3463773	103.68734	0.792	132.85
9	1.5E+12	vehicle	4	0	20354256	1	1.3422466	103.68069	0.072	360

Table 2: Sample data set

## Phase 2: Link labelling

For the convenience of calculating and analysing, each link in the traffic network was pre-labelled with a unique number. Figure 10 shows the NTU campus main links with its pre-defined link numbers and nodes numbers,

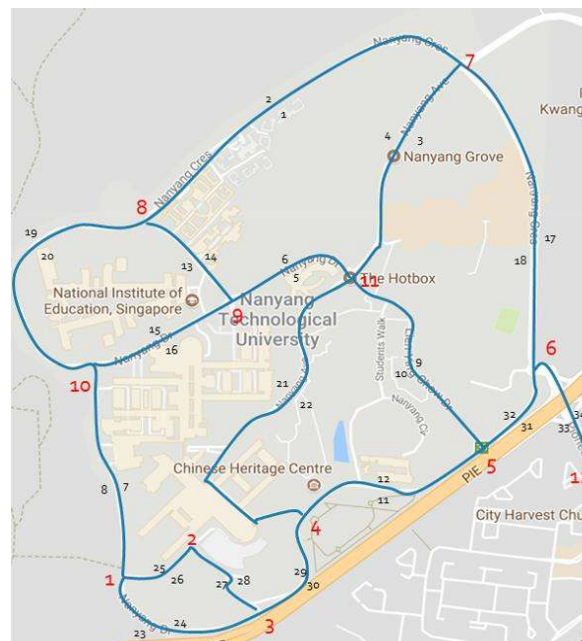


Figure 10: Labelled school campus

### Phase 3: Link Identification

It could be observed that the original BSM dataset does not have the feature of “link label”. Thus, the first step of this project is to associate the raw data with relevant link numbers accordingly. A well-trained FNN (feed-forward neural network) model is used here. The training process could be summarized in the following steps:

- Step 1: Select part of the collected data from the database with the requirement of covering all the links. Label all those data sets manually based on the GPS information and the heading direction. A small list of the training data is shown in Table 3:

latitude	longitude	speed	heading	link
1.355439	103.6852	26.136	238.3	1
1.346446	103.6873	0.792	130.7	9
1.354683	103.6838	23.184	55.7375	2
1.349464	103.685	14.256	304.9375	10
1.345095	103.6886	0.216	72.8875	32
1.351117	103.6798	20.016	192.85	14
1.345555	103.6825	33.552	238.8	22
1.348765	103.6818	20.016	255.025	16
1.350515	103.6858	14.688	193.15	3
1.34859	103.6865	0.432	141.875	9

Table 3: Sample training data set for training

- Step 2: To balance the weight of each input feature in the cost function, it is advised to adjust them into identical data sizes before proceeding with the data training session. During this step, all data will be normalized, with values ranging from -1 to 1. MATLAB has the inbuilt function “mapminmax” (mathematical function is shown below) which can transform data of the same type into the same range.

$$\text{mapminmax}(X) = \frac{X - \frac{X_{max} + X_{min}}{2}}{\frac{X_{max} - X_{min}}{2}} = \frac{2X}{X_{max} - X_{min}} - 1$$

$X_{max}$  is the max value in the training data set

$X_{min}$  is the min value in the training data set

- Steps 3: In order to get familiar with the neural networks, an FNN model is selected here and the normalized data obtained in steps 2 are used in training the model. The labelled feature is the output while other features are input. The well-trained network

structure is shown in figure 8., where three features are used: latitude, longitude and the heading angle. Thus, the size of the first layer is 3. There are 34 links in total for the NTU on-campus traffic network, so the size of the last layer is 34. Since the problem here is not complex, any reasonable combination of the hidden layer size could achieve high identification accuracy. Each input data set will activate one of the neurons in the last layer. The order of the neurons is the output of the network that also indicates the label of the link where the data set signals are derived.

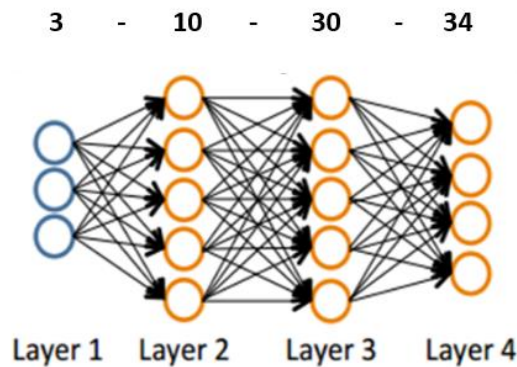


Figure 11. Well-tuned Feed-forward Neural Network

- Steps 4: Before putting this model into real practice, accuracy should be tested with the testing data set (testing data set is the data set that never be learned during the training phase). Figure 12 shows the sample comparison between the identification result and the true link number with 10 randomly selected inputs:

10x1 double		A1				
		A	B	C	D	
	1	1	latitude	longitude	heading	link
1	1	2	1.355439	103.6852	238.3	1
2	12	3	1.344256	103.6877	57.5375	12
3	3	4	1.354552	103.6872	236.0375	3
4	2	5	1.35203	103.6806	38.325	2
5	34	6	1.346812	103.6903	158.85	34
6	26	7	1.342129	103.6807	266.7875	26
7	26	8	1.341338	103.6793	279.3375	26
8	26	9	1.341352	103.6792	294.5	26
9	8	10	1.341981	103.6791	3.075	8
10	26	11	1.341318	103.6793	275.0125	26

↑  
Output matrix form MATLAB

↑  
Link number labelled manually

Figure 12. Comparison between output and label

It can be observed that the performance of this model is good. In general, this model achieved identification accuracy at 98.5% on the whole testing data set. Main errors were made in

some junction area or turning area where the boundary is not clearly divided. Since the errors do not have a significant influence on the result, the model is still considered adequate for this project.

#### Phase 4: Link Congestion Level Identification

In this project, a discrete-time framework is adopted. The congestion level is designed to describe the traffic condition for a certain period (e.g.,15 seconds in this project) to reduce the impact of outliers.

After accumulating the data for every 15 seconds, the algorithm introduced above is applied to identify the congestion level of the traffic network in NTU. The result is shown below. Table 4 is the output of the MATLAB codes and figure 13 visualized the result on the map to make the data easier to understand.

Origination	Destination	Logitude	Latitude	Congestion Level	link
1	2	1.341514	103.6802	2	24
2	1	1.341484	103.6803	2	32
2	3	1.341403	103.6818	1	29
3	2	1.341365	103.6818	1	23
1	3	1.339896	103.6806	2	36
3	1	1.339858	103.6806	2	34
1	10	1.344232	103.6787	2	8
10	1	1.344236	103.6788	2	7
10	9	1.347873	103.6802	1	15
9	10	1.347839	103.6802	1	16

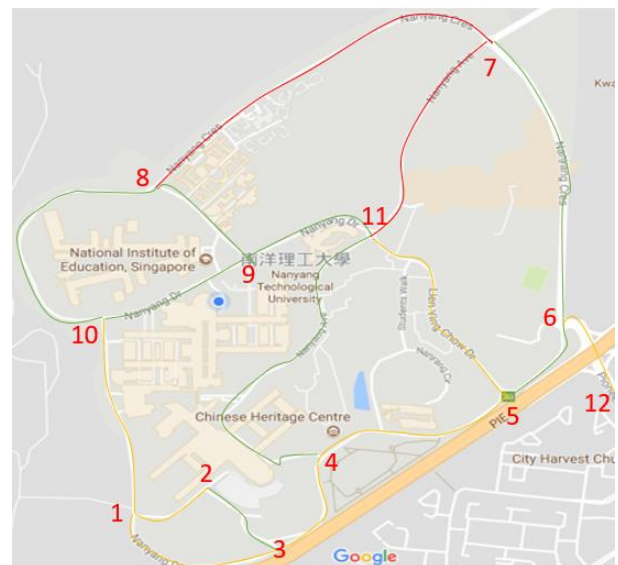


Table 4: Congestion level identified by MATLAB

Fig.13 Visualizing the identifying result on NTU campus

From the map above we could easily know that the link between nodes 8 and 7 as well as the link between nodes 7 and 11 are over-saturated, the link between nodes 10 and 1, the link between nodes 1 and 2, the link between nodes 3 and 4, and the link between nodes 5 and 11 are median congested while other links are not congested.

## 2.5.2 Case study two: Traffic congestion level prediction and congestion region clustering in the Jurong area traffic network

In the first experiment, not all vehicles on the campus are installed with OBU which means the data collected may not exactly represent the traffic condition. And the traffic condition in the campus are seldom congested which makes further research work difficult to process. Thus, another traffic network is considered: the Jurong area traffic network. The process is summarized as follow:

### Phase 1: Simulated traffic network setup and data collection

Due to the reason that the real traffic data is not accessible, a simulated traffic network is constructed with identical layout and size compared to the real transportation network in Jurong area. The software used is VISSIM which is a micro multi-model flow simulation software package. The overall view of this traffic network and one junction among it is shown below. The green and red bars restrict the traffic flow directions with practical consideration. This traffic network holds 66 junctions and 253 links in total.



Fig.14 overall view of Jurong area transportation network

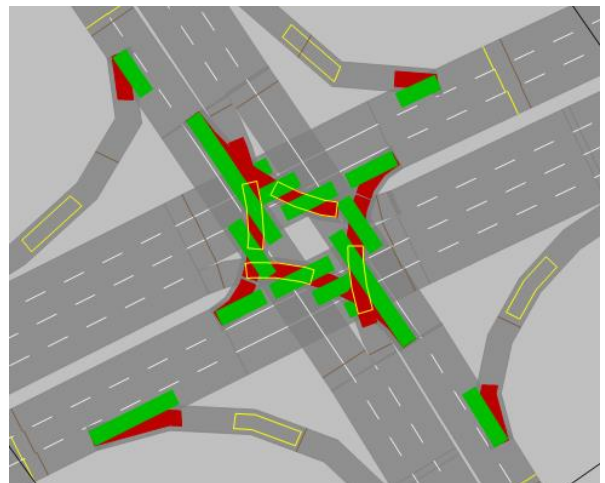


Fig.15 Single junction layout

Similarly, all the links need to be assigned with a unique number for calculating:

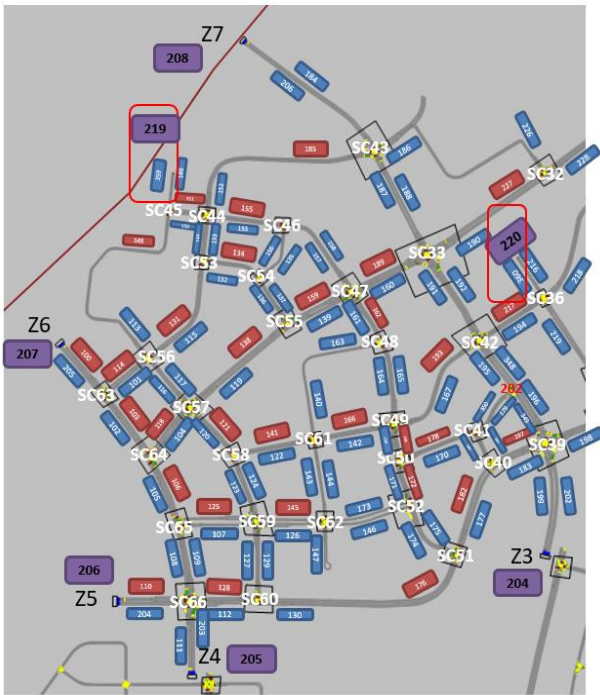


Fig.16 Jurong East & West Link Numbers & Junctions (Main area: D4,D5)

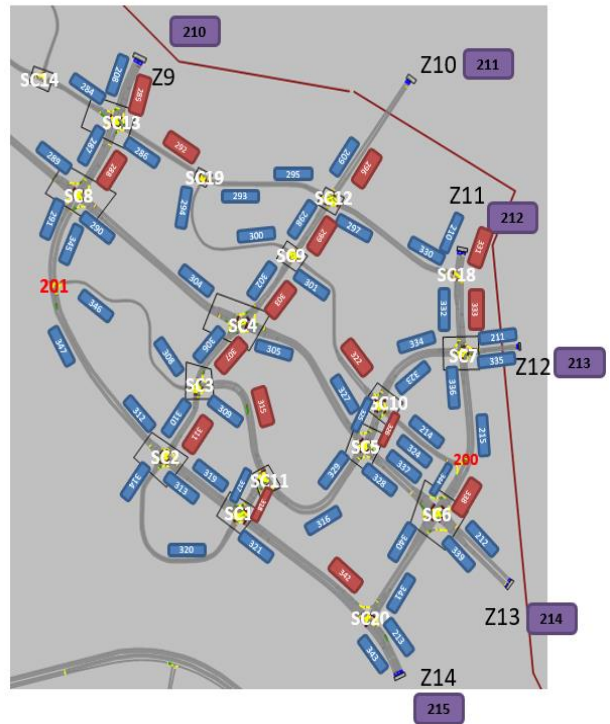


Figure 17: Jurong East & West Link Numbers & Junctions (Main area: E3, E4, partial D3, partial D4)

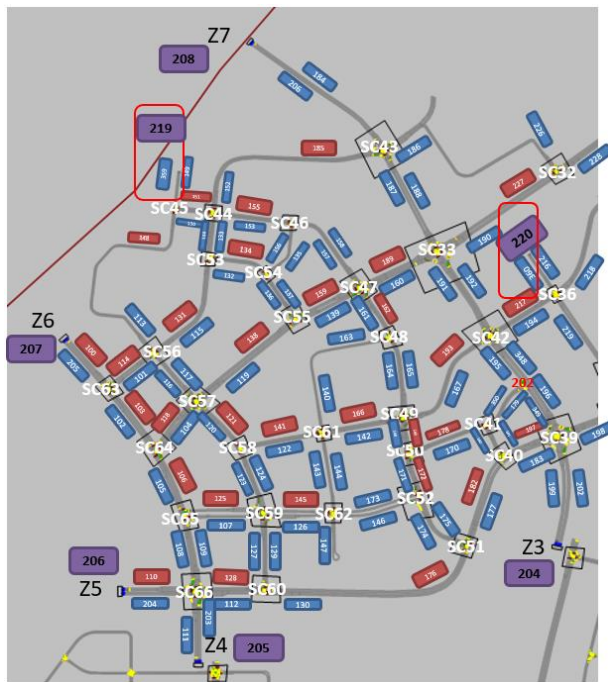


Figure 18: Jurong East & West Link Numbers & Junctions (Main area: Main area: D3,E3)

Vehicle data will be then auto-generated by the software. Data features are selected based on research needs. A sample collected data sheet is shown below:

1	LinkName	UpstreamLinkNumberList(left-straight-right)	DownstreamLinkNumberList(left-straight-right)	NumberOfLanes	AssignedLinkNumber(also the position in vector input)	Junction reference link (1=yes 0=no)	Length(meter)	MaxVolume(static)	upstream junction number	Upstream Junction Signal Controller	Downstream junction number	Downstream Junction Signal Controller	flowInNumber	OutLeftNum	OutStraight	OutRightNum
2	s03sgl_northwest	0-0-0	114-103-0	3->4	100	1	247.586	124.24	207	0	63	1	1	4	5	0
3	s03sg3_northeast	116-115-113	103-0-205	2->3	101	0	197.246	69.59	56	1	63	1	8	10	0	11
4	s03sg1_southeast	0-105-104	0-205-114	3->4	102	0	274.552	140.63	64	1	63	1	13	0	17	16
5	s04sg1_northwest	101-100-0	118-106-0	3->4	103	1	275.013	135.15	63	1	64	1	20	23	24	0
6	s04sg3_northeast	120-119-117	106-0-102	3->4	104	0	219.343	111.43	57	1	64	1	27	30	0	32
7	s04sg1_southeast	0-108-107	0-102-118	3->4	105	0	241.123	125.57	65	1	64	1	34	0	39	37
8	s05sg1_northwest	104-103-0	125-109-0	3	106	1	245.064	113.11	64	1	65	1	41	44	45	0
9	s05sg3_east	127-126-124	109-0-105	2->3	107	0	278.156	97.44	59	1	65	1	47	49	0	50
10	s05sg1_south	110-111-112	0-105-125	3->4	108	0	274.415	137.61	66	1	65	1	52	0	55	57

Table 5: Sample data collected from VISSIM

## Phase 2: Traffic congestion level prediction

From case study one, the same methodology could be applied in identifying the link congestion level. However, only identifying the congestion level for each link is far from being enough. It could be better if the congestion level can be forecasted. With high-accuracy congestion level predictions, emergent solutions could be put into use immediately to prevent the occurrence of potential congestion. As discussed in the literature review, a single layer many-to-many RNN model is used to predict the average speed and vehicle density separately as shown in the figure below:

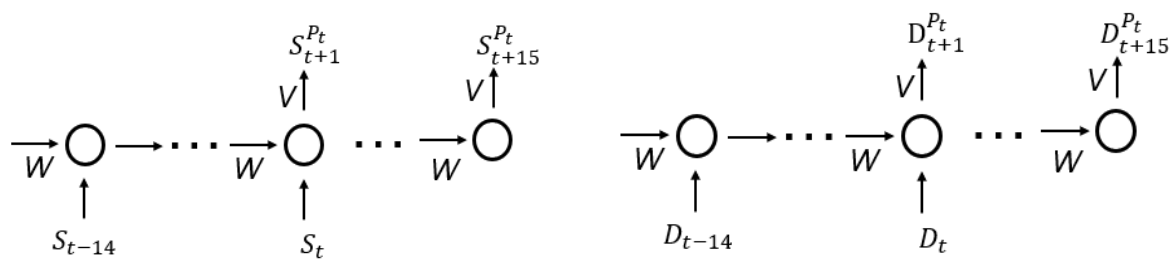


Fig.19 Two RNN models for speed and density prediction

The batch size of this RNN model is set at 15. This number is selected based on how many steps ahead are required to forecasting the congestion level. The longer the predicting time horizon, the larger the number. For example, the model constructed can be used to predict the congestion level up to 15 time steps later. Concretely, once after a time step ( $t$ ), the algorithm will take the current average speed  $S_t$  as well as all the average speed recorded in

the past 14 time steps  $\{S_{t-14}, \dots, S_{t-1}\}$  as input. Those 15 inputs will generate 15 outcomes corresponding to the predicted average speed in 1~15 time steps. Thus, for each future time  $(t+m)$ : the predicted outcome will be the average of the predicted results generated from the current time interval to the previous  $(15-m)$  time steps. The equation is shown below:

$$S_{t+m}^P = \frac{1}{16-m} \sum_{i=0}^{15-m} S_{t+m+i}^{P_{t-i}}$$

$$D_{t+m}^P = \frac{1}{16-m} \sum_{i=0}^{15-m} D_{t+m+i}^{P_{t-i}}$$

A prediction result is shown in figure 20, which compares the actual congestion level with the predicted congestion level for the next time step.

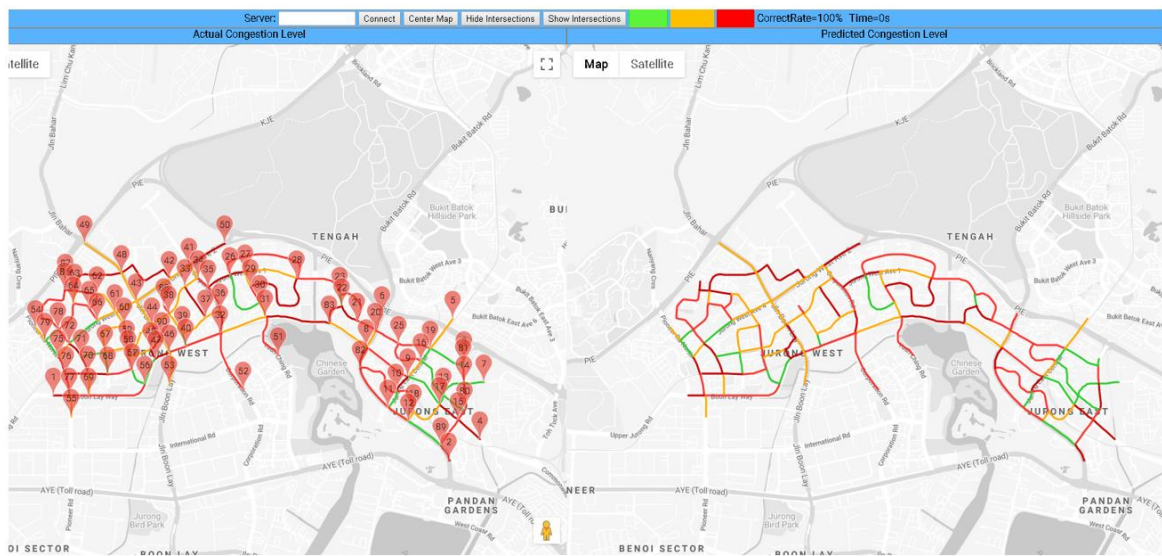


Figure 20: Comparison with actual congestion level (left) and predicted congestion level (right)

As shown in the table below, the prediction accuracy is very high when the prediction horizon is not too long. However, prediction accuracy drops fast when increasing the prediction horizon due to the traffic fluctuations.



Prediction Horizon	1 time step	2 time steps	3 time steps
Average Prediction Accuracy	98.8%	97.3%	95.6%
Prediction Horizon	4 time steps	5 time steps	6 time steps
Average Prediction Accuracy	92.4%	87.1%	80.4%

Table 6: Congestion level prediction accuracy with different time horizon

### Phase 3: Congestion region clustering

In this section, three clustering algorithms were used respectively: k-means clustering algorithm, hierarchical clustering algorithm and improved clustering algorithm based on the methodology discussed above.

When applying K-means clustering algorithm, the number of clusters should be decided in advance. In general, the optimal cluster number could be determined by using the elbow method. Figure below shows three results of the largest number of junctions between the links in the same cluster  $N_{max}$  after applying the K-means clustering algorithm with different pre-set numbers of clusters:

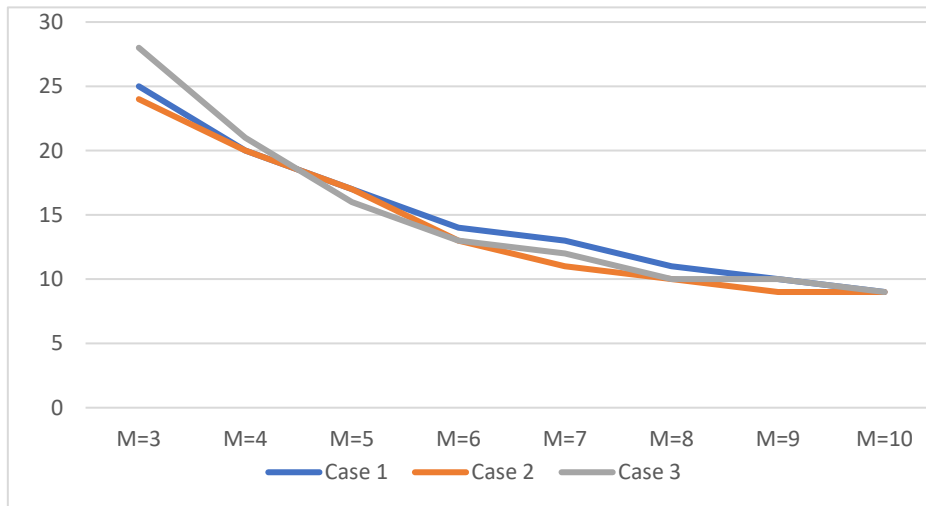


Fig.21. The largest number of junctions  $N_{junctions_{max}}$  VS different number of clusters

It could be observed from the above graph that the elbow point falls on M=6 with a high probability. Thus, in this case,(the Jurong area with 66 junctions and 253 links in total), M could be set at 6 generally when applying the K-means clustering algorithm. Figure 22 shows

the computational time used when applying K-means clustering algorithm and hierarchical algorithm:

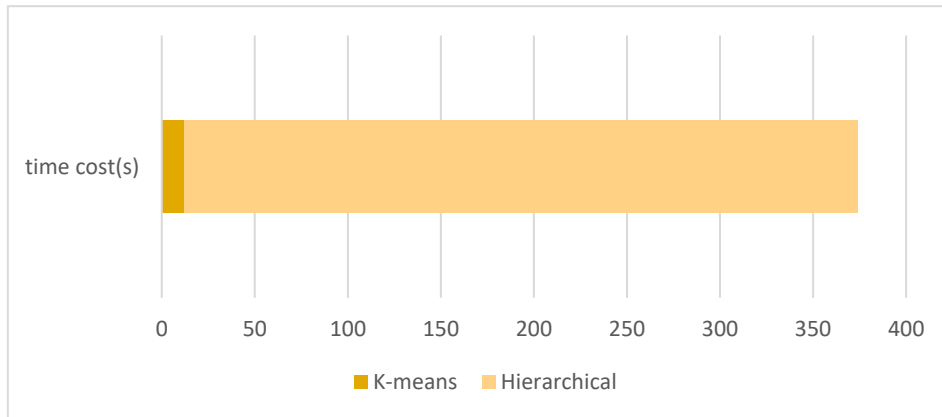


Figure 22: Computation time cost for each algorithm

It takes about 12 seconds for the K-means method to complete the clustering computation, while more than 350 seconds are needed for the hierarchical method. This result verifies the disadvantage of the hierarchical algorithm. Since the congestion region is needed to be updated once every 15 seconds, the K-means clustering algorithm is more suitable.

However, the clusters obtained by the K-means clustering algorithm are not significantly different from each other in terms of average regional congestion levels. It can be shown from the graph below. It could be found that the average congestion level fluctuates around 2 and these average congestion levels are similar to each other, which makes it impossible to figure out the most congested region.

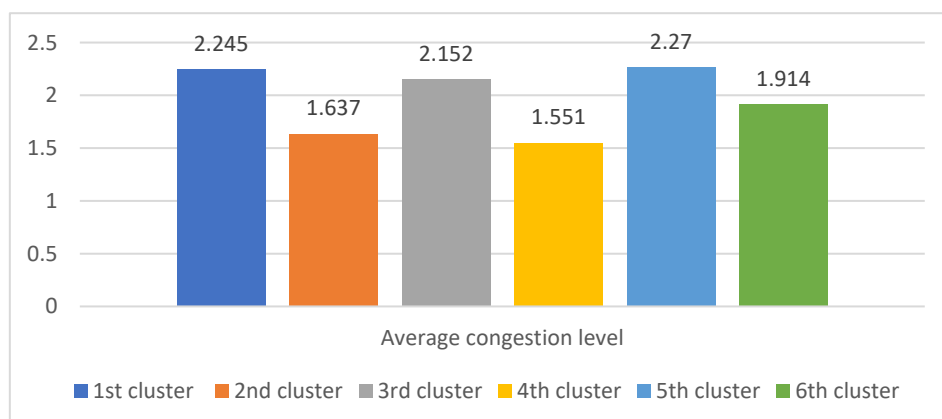


Figure 23: Average congestion level of each cluster

With the aim of identifying the over-saturated region, the improved clustering algorithm is used. Starting from the over-saturated links, a region is expanded by spreading upstream links and downstream links. Besides, by saving the time used on calculating the under saturated regions, the processing time is faster than the general K means clustering which only 9 seconds is needed on average to get the result.

## 2.6 Summary

This chapter aims to solve three problems: link congestion level identification, link congestion level prediction and congestion region clustering. For link congestion level identification, a new method is introduced which uses the difference of traffic flows and densities between the current traffic condition and that of the previous time step in deciding the traffic congestion level. The advantage of this method is that, by considering the differential value instead of static value, not only the current traffic congestion level can be identified but also the trend of changing can be detected. As a result, more information is generated which could be used in other related research projects. For the link congestion level prediction problem, a simple many-to-many single layer RNN model is used as the predictor. The model implemented is trained off-line. For the congestion region clustering problem, two general clustering algorithms are used whose performances are not satisfactory. Thus, an improved clustering algorithm is introduced which only focuses on the most congested region. The advantage of this algorithm is its (relatively) low computational complexity when compared with the other two, mainly owing to the step of ignoring the under-saturated links and only highlighting the most congested region which the traffic optimizing algorithm is of most interest. **Future work involves applying more prediction algorithms (SVM, ELM and GBDT) and clustering algorithm (EM clustering and Affinity propagation) to solve this problem and compare the performance with current work.**

## Chapter 3: Traffic Network Turning Ratio Prediction

This chapter contains four parts. The first part defines some basic concepts. The second part introduces two neural networks which can be tuned as a predictor. These two networks have been widely used and developed in many aspects. However, there is no relevant research that applies these two models in predicting the traffic turning ratio. And these works are done in Part three together with a case study on a 3\*3 simulated traffic network. The key challenge is how to tune the hypermeter of the network well in order to increase the prediction accuracy. The last part is the summary for the project based on the experiment result.

### 3.1: Basic concepts and problem statement

To make the subsequent technical development clear, some concepts are defined below.

- Turning ratio: whenever a vehicle reaches a node, it will decide which link to enter, after leaving the current link. In this thesis, all nodes are connected with four links, which means the vehicle will have three choices: turn left, go straight and turn right. Turning ratio is the percentage of the vehicles that make the same choices among all the vehicles leaving the current link. The symbol used to represent the turning ratio during time step  $t$  is  $\lambda(t)$ . The calculation formula is shown below

$$T.R_{Left} = \frac{\text{No. of vehicles that turn left}}{\text{Total vehicles that leaves current link}}$$

$$T.R_{Straight} = \frac{\text{No. of vehicles that go straight}}{\text{Total vehicles that leaves current link}}$$

$$T.R_{Right} = \frac{\text{No. of vehicles that turn right}}{\text{Total vehicles that leaves current link}}$$

$$T.R_{Left} + T.R_{Straigh} + T.R_{Right} = 1$$

- Traffic light status and assignments: Each signalized node in the traffic network control the traffic flow by adjusting the traffic light status for each direction. In this thesis, four statuses are used to describe different traffic light condition. These four statuses are shown below:

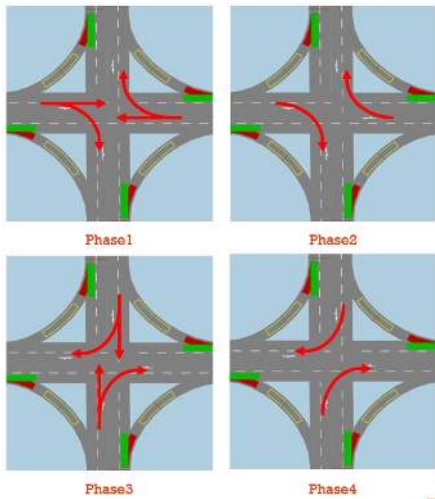


Figure 24: Four traffic light phases with traffic flow allowed

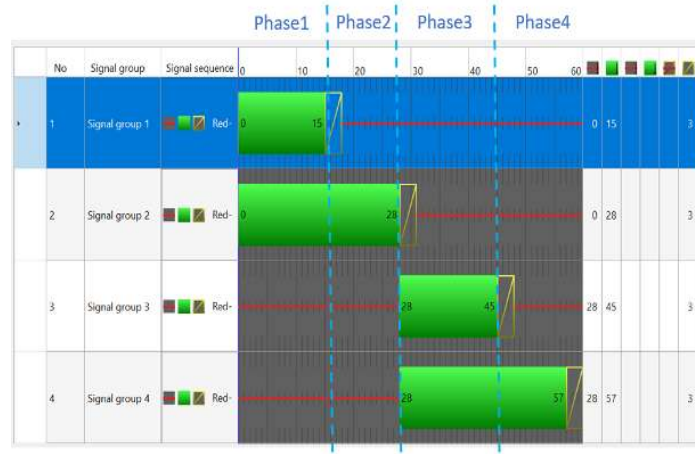


Figure 25: time length and schedule of each traffic light phase

Phase one will enable the vehicles to go straight and turn left in the horizontal direction. Phase two will enable the vehicles to go straight only in the horizontal direction. Phase three is identical to phase one except for the direction changes from being horizontal to being vertical. Phase four will enable the vehicles to go straight only on the vertical direction. When a fixed-time traffic light assignment is under consideration in this thesis for a performance comparison purpose, each phase is assumed to last for 15 seconds. The symbol used to represent the traffic light status during one time step is  $\theta(t)$ . The traffic light assignment is a sequence of pre-determined traffic light statuses. The traffic light in the node will operate based on this assignment.

- Supply function: In this thesis, the traffic network used is a region-based network. All the vehicles running inside is assumed to be supplied from the terminals of the traffic network. The number of vehicles supplied into the traffic network are different for different terminals and different time steps. Thus, a supply function is introduced to represent the number of incoming vehicles over time at each terminal. The symbol used to represent the supply function is  $s(t)$ .

The turning ratio is conceptually determined by each driver's driving habit (i.e., his/her origin-destination pair and the corresponding fixed yet unknown trajectory), the supply function and all previous traffic light assignments. Our conjecture is that, by a fixed

driving habit for each driver each link turning ratio is a function of the supply volume and past traffic light schedules, which could be written as:

$$\lambda(t) = f(s(0), s(1), \dots, s(t-1); \theta(0), \theta(1), \dots, \theta(t))$$

$$\lambda(t-1) = f(s(0), s(1), \dots, s(t-2); \theta(0), \theta(1), \dots, \theta(t-1))$$

So, a turning ratio at  $t$  is supposed to be a function of all historical turning ratios, flow-in vehicles and traffic light schedules:

$$\lambda(t) \leftarrow f\{[\lambda(0), \dots, \lambda(t-1)], [s(1), \dots, s(t-1)], [\theta(0), \dots, \theta(t)]\}$$

- Prediction accuracy: In order to visualize the performance of parameter selection while tuning the neural network, prediction accuracy is computed, which is defined as the number of turning ratios correctly predicted over the total number of turning directions. Since the turning ratio is a number between 0 and 1, the predicted turning ratio is said to be correct if the difference is less than 0.1. This standard is for testing only, it can be adjusted according to a user's expectation or requirement.

### 3.2: Machine learning algorithms for turning ratio prediction

In this section, the component and the architecture of the proposed FNN and RNN models are introduced. Here, turning ratio prediction is defined as predicting future turning ratios based on historical traffic information.

#### 3.2.1 Network-wide Traffic Data

Turning ratio prediction at one link normally utilizes a sequence of traffic information with  $N$  historical time steps as the input data, which are represented by a vector.

$$X_T = [X_{T-n}, X_{T-(n-1)}, \dots, X_{T-2}, X_{T-1}], X \text{ is a collection of } \{ \lambda, s, \theta \}$$

But the turning ratio of one link may be influenced by the traffic condition of nearby links or even locations faraway, especially when traffic jam propagates through the traffic network. To take these network-wide influences into account, it is better to take the network-wide traffic data as the input. Suppose the traffic network consists of  $P$  links and we need to predict the traffic turning ratio at time  $T$  using  $n$  historical time steps, the input can be characterized as a traffic data matrix:

$$X_T^P = \begin{bmatrix} x^1 \\ x^2 \\ \dots \\ x^P \end{bmatrix} = \begin{bmatrix} X_{T-n}^1 & X_{T-(n-1)}^1 & \dots & X_{T-2}^1 & X_{T-1}^1 \\ X_{T-n}^2 & X_{T-(n-1)}^2 & \dots & X_{T-2}^2 & X_{T-1}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ X_{T-n}^P & X_{T-(n-1)}^P & \dots & X_{T-2}^P & X_{T-1}^P \end{bmatrix}$$

Where each element  $x_t^p$  represent the turning ratio  $\lambda$  of the  $p^{\text{th}}$  link, traffic light assignment  $\theta$  and the supply function at the  $t^{\text{th}}$  time.

#### 3.2.2 An FNN model and parameter identification

Feed-forward neural networks [48] are designed with one input layer, one output layer and hidden layers [49]. The size of the input layer is equal to the number of input features and the size of the output layer is equal to the physical quantities of output. Thus, when constructing



an FNN model, the key is to find a proper number of hidden layers and their size respectively. Insufficient neurons in the hidden layers will lead to underfitting problems with low precision and too many neurons in the hidden layers will result in overfitting problems with high precision on the training data set and low precision on the testing data set. One common technique in searching proper parameters of hidden layers is Bayesian regularization, along with a Levenberg–Marquardt algorithm [50].

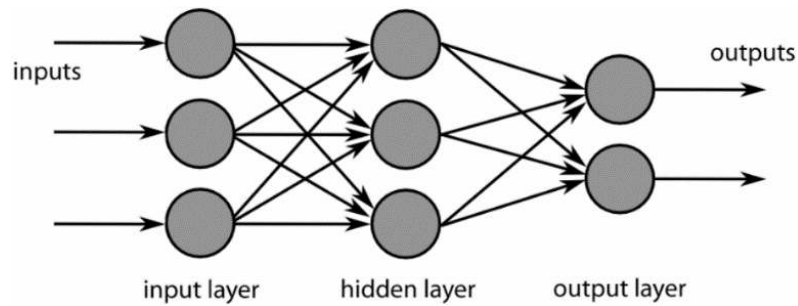


Figure 26: Basic structure of FNN

A general FNN model is shown above. In the FNN model, the input features will be placed in the neurons of the first layer. The data will be then propagated to the first hidden layers by multiplying the weights stored in the interconnections. Each neuron in the hidden layers is connected to every neuron in adjacent layers. These neurons will sum up the arrived weighted inputs of the previous hidden layer. After propagating the summation through an active function, the result will be transferred to all the neurons in the next hidden layer or output layer. In this work, the activation function used is an exponential sigmoid function, which has generally and traditionally been used to develop FNNs. The mathematical expression of sigmoid function is

$$f(x) = \frac{1}{1 + e^{-x}}$$

A bias term,  $b$ , is associated with each interconnection in order to introduce a supplementary degree of freedom. The expression of the weighted sum,  $S$ , to the  $i^{\text{th}}$  neuron in the  $k^{\text{th}}$  layer ( $k \geq 2$ ) is

$$S_{k,i} = \sum_{j=1}^{N_{k-1}} [(W_{k-1,j,i} I_{k-1,j}) + b_{k,i}]$$

To train this model, the collected data sets will be divided into three groups: the training data set, cross validation data set and testing data set. After partitioning the data sets, the training

set is used to adjust the parameters. The network is then trained until it correctly emulates the input/output mapping, by minimizing the average root mean square error. The testing set is used, during the adjustment of the network's parameters, to evaluate the algorithm's performance and stop the adjustment if the error on the testing set increases. Finally, the validation set measures the generalization ability of the model after the fitting process.

### 3.2.3 An RNN model and parameter identification

RNN is one of the powerful deep neural networks by using its internal memory with loops to deal with sequential data. Recall the structure of RNN shown in Figure 27:

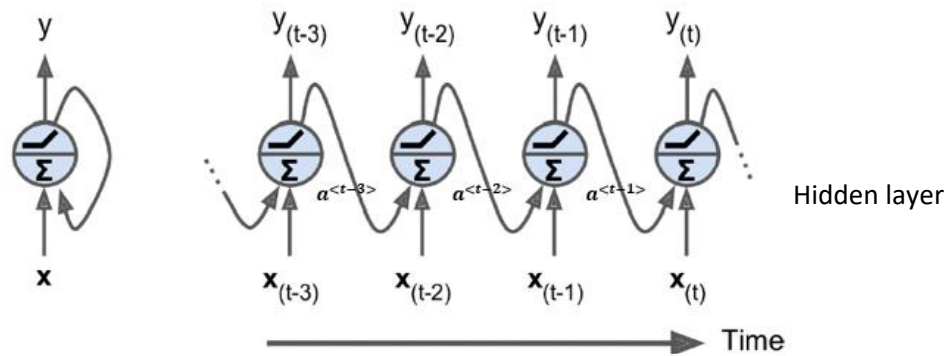


Figure 27: Basic structure of RNN

The right part of the figure shows the interval calculation process that, at each time iteration,  $t$ , the hidden layer maintains a hidden state,  $a^{<t>}$ , and updates it based on the layer input,  $x_t$ , and previous hidden state  $a^{<t-1>}$  by using the following equation:

$$a^{<t>} = g_a(w_{aa}a^{<t-1>} + w_{ax}x^{<t>} + b_a)$$

where  $w_{aa}$  is the weight matrix stored in the interconnection between two consecutive hidden states,  $w_{ax}$  is the weight matrix stored in the interconnection between the input layer and the hidden layer,  $b_a$  is the bias vector of the hidden layer.  $g_a$  is the activation function of the hidden layer which adding the non-linearity property to the model. The network output can be computed by the equation:

$$\hat{y}^{<t>} = g_y(w_{ya}a^{<t>} + b_y)$$

Where  $w_{ya}$  is the weight matrix stored in the interconnection between the hidden layer and the output layer,  $b_y$  is the bias vector of the output layer.  $g_y$  is the activation function of the output layer. The weight matrix and bias stored in the RNN will be trained and updated iteratively via the back-propagation method. Hidden layers will calculate and output a result for each input, and the last output,  $\hat{y}^{<t>}$  is the desired predicted turning ratio in the next time step. However, traditional deep RNN model suffering from the vanishing or exploding gradient problem. In the past decades, several recurrent architectures are proposed to handle this problem, like LSTM architecture and Gated Recurrent Unit (GRU). LSTM improve the RNN model by adding three gates in the hidden layers which enable the RNN deal with long-term dependencies to allow useful information pass along the LSTM. These three gates are called input gate, forget gate, and output gate which is denoted as  $i^{<t>}$ ,  $f^{<t>}$ ,  $o^{<t>}$  respectively. The interval structure of LSTM is shown in figure below and they can be calculated with following equations:

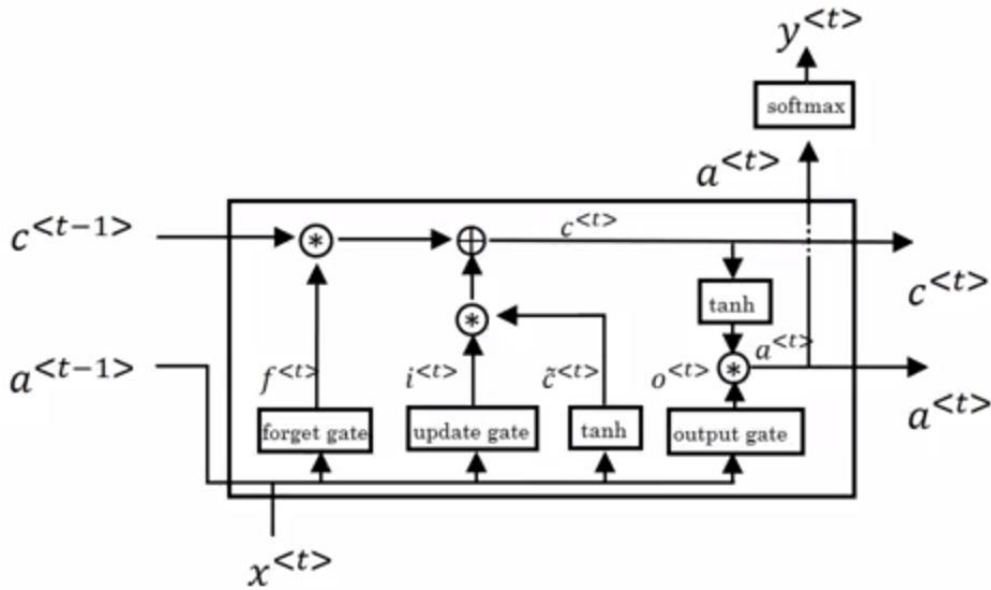


Figure 28: Basic structure of LSTM

$$c_N^{<t>} = \tanh(w_c[a^{<t-1>}, x^{<t>}] + b_c) \quad c_N^{<t>} \text{ is a candidate for replacing } c^{<t>}$$

$$\Gamma_f = \sigma(w_f[a^{<t-1>}, x^{<t>}] + b_f) \quad \text{forget gate}$$

$$\Gamma_i = \sigma(w_u[a^{<t-1>}, x^{<t>}] + b_i) \quad \text{input gate}$$

$$\Gamma_o = \sigma(w_o[a^{<t-1>, x^{<t>}] + b_o) \quad \text{output gate}$$

Where  $c_N^{<t>}$  is the primary hidden layer state ( $c_N^{<t>} = a^{<t>}$  for classic RNN model) which needed to be processed by the other parts in the LSTM cell.  $w_c, w_f, w_u, w_o$  are the weight matrices mapping the hidden layer input to the three gates and the input cell state,  $b_c, b_f, b_i, b_o$  are four bias vectors. Based on the above four equations, the cell output state  $C^{<t>}$ , and the new hidden layer state  $a^{<t>}$ , can be calculated with following equations:

$$C^{<t>} = \Gamma_i * c_N^{<t>} + \Gamma_f * c^{<t-1>} \\ a^{<t>} = \Gamma_o * C^{<t>}$$

### 3.3 Simulation-based experiments

In this section, an FNN and an RNN are tuned to solve realistic problems. A simulated 3\*3 traffic network is built in VISSIM, and all the data used for training the network are provided by this software. The procedures of tuning the networks are explained in detail.

#### 3.3.1 System setup

##### A. Simulated traffic network construction

In order to test the capability of FNN and RNN in predicting the traffic turning ratio, a simulated traffic network is constructed in the VISSIM. Considering the computational cost, the size of the network is set at 3\*3, it is neither too big to train nor too simple. This traffic network has 12 flow-in terminals, 9 junctions, and 48 links as shown below. Figure in the left shows the link number, node number, terminal number assignment. Figure in the right shows the built simulated traffic network with vehicles running inside. Vehicle data are generated and recorded per second.

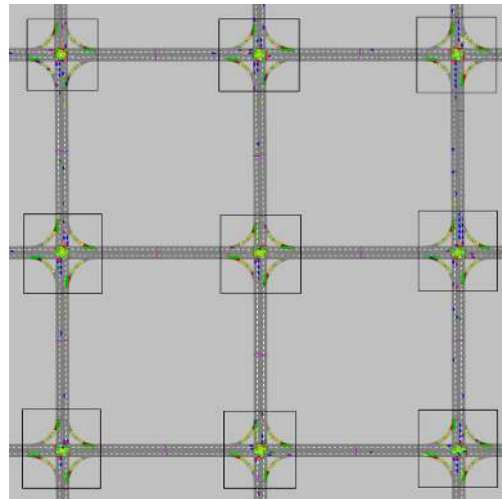
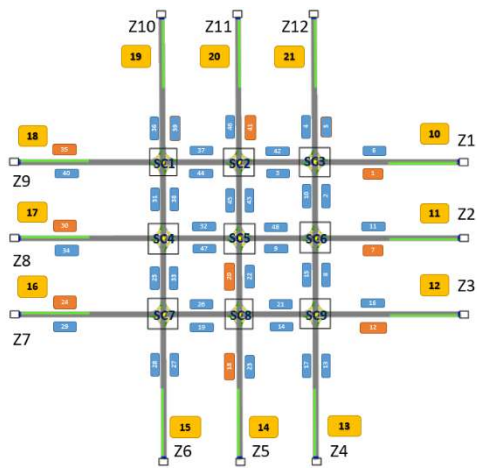


Figure 29: 3\*3 traffic network built in VISSIM

## B. Data preprocessing

In this experiment, raw data generated by the above traffic network contains 261 features:

- The number of flow-in vehicles from 12 terminals.
- Traffic light schedules at 9 junctions
- The current number of vehicles in all 48 links
- The number of vehicles leaving each link
- The number of leaving vehicles turn right, go straight and turn left respectively.

Part of features are collected for other research purposes which needs to be discarded and some important missing features need to be calculated and added into the training data set. The basic idea of feature selection is to choose the features that could provide enough information to the network such that the trained network can generated the output based on this information with high accuracy. Unlike picture processing or text processing, features in this problem hold their practical significance which is easier to decide if the feature is related to the turning ratio prediction. Thus, data preprocessing for this problem can be done manually which are summarized in the following steps:

- Sum up the vehicle information for every 15 seconds. This length of time is the time unit in formulating the traffic light assignment. By summing up the vehicle data, the change of

traffic flow due to traffic light assignment is easier to be observed.

- Third feature which contains the information of current number of vehicles in all 48 links is removed as they have no contributions to the prediction result based on the mathematical model implemented previously.
- It is observed that 24 links in the traffic network are connecting the terminals and half of them are exporting the vehicle form the network. Vehicle data of these 12 links is removed as they have no more impact on the vehicles still running inside the network.

### 3.3.2 Experimental procedure and data collection

#### A. Trained with FNNs

The first model used is a fully connected neural network, which only predicts turning ratios at a specific time step. Firstly, the model only uses one time-step information to predict the turning ratio in the next time step.

$$\lambda(t) \leftarrow \{ \lambda(t-1), s(t-1), \theta(t) \}$$

Searching for the best size of two hidden layers requires a lot of work. Figure 15 shows the prediction accuracy performed on the cross-validation data set with a different combination of two hidden layers.

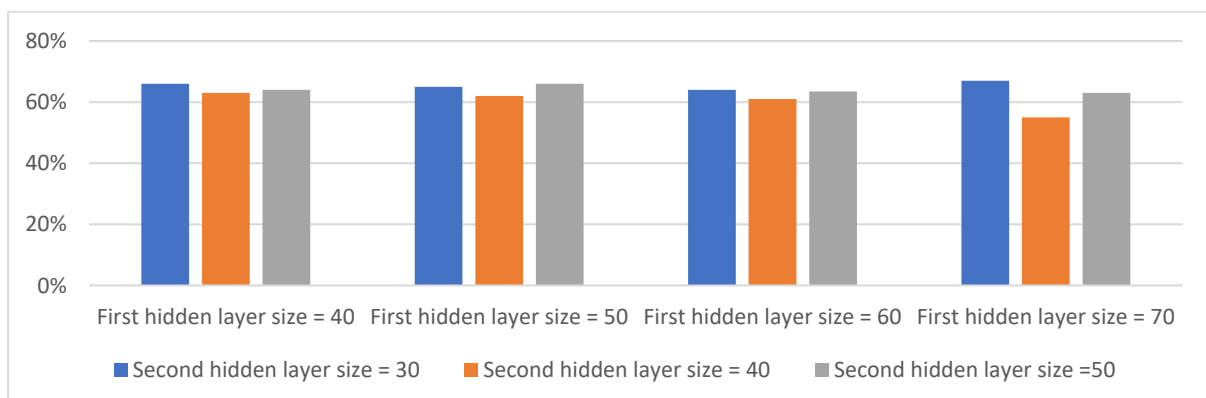


Figure 30: Prediction accuracy with different combination of hidden layer size

With the result obtained from Figure 30, when the size of the first hidden layer equals 70 and the size of the second hidden layer equals 30, the accuracy has the highest value at around 68%. Before settling down the structure of this FNN model, the performance should be double tested by the testing data set. Figure 16 shows the comparison between the predicted turning ratio (first row) and the actual turning ratio (second row). The accuracy applied to the testing data set is 65%. Not bad, the combination of these hidden layers could be selected. And the construction of the feed-forward network with the size of two hidden layers is shown in figure 31.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	9	1	1	1	1	9	10	1	1	9	2	1	3	10	1	8	1	1	1	4
2	10	1	1	1	1	10	10	1	1	9	1	1	6	10	1	9	1	1	1	1

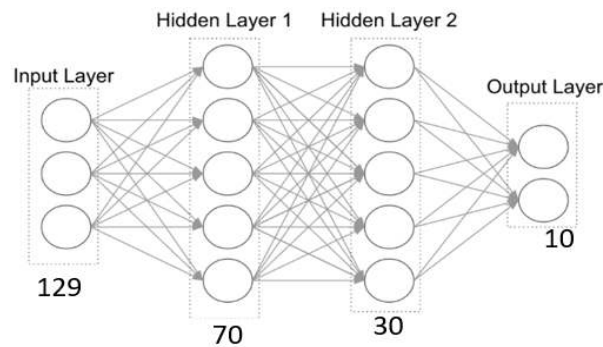


Figure 31: Comparison with the labeled value and the FNN structure

The number of outputs is ten, only one of them is activated when an input is provided. The sequence of the activated nodes (output equal to or close to 1) among these neurons is the output of the network. For example, if the second node is activated, it means the output of the network is 2 and it also indicates that only around 10% of the existing vehicles flow into this direction. If the last node is activated, it means the output of the network is 10 and all vehicles leaving this link flow into this direction.

Obviously, the accuracy achieved is not satisfactory. What could happen, if we increase the number of input features by looking at more time steps backward in time? For example, say:

$$\lambda(t) \leftarrow \{ \lambda(t-2), \lambda(t-1), s(t-2), s(t-1), \theta(t-1), \theta(t) \}$$

Following the same procedure, we could get the prediction accuracy graph shown below:

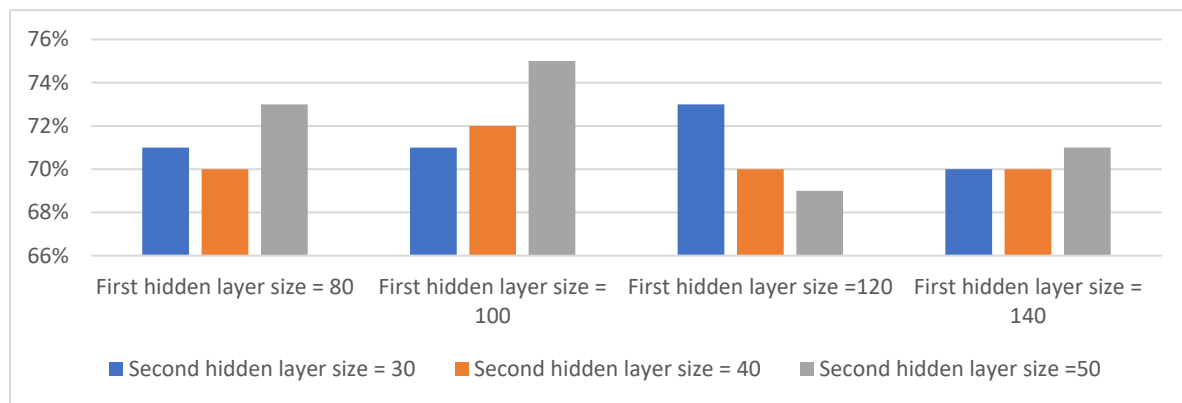


Figure 32: Prediction accuracy with different combination of hidden layer size

It is obvious to see that when the size of the first hidden layer is 100 and the size of the second hidden layer is 50, the accuracy reached is around 75%. Furthermore, the average accuracy increases roughly by 7% compared to the result obtained when only one single time step is used for prediction. Lastly, testing the structure set on testing data attains accuracy at 71%. Thus, the FNN model for two time-steps ahead is shown in Figure 33.

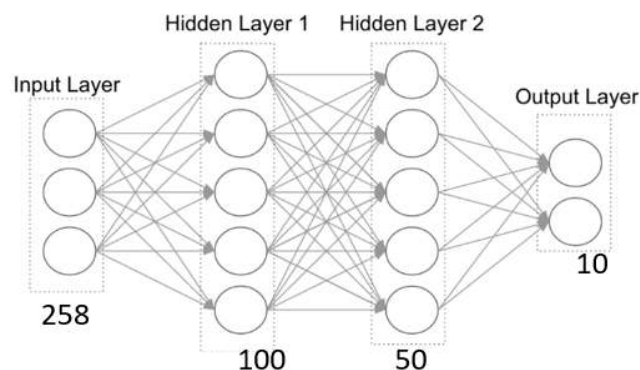


Figure 33: Construction parameters of the tuned FNN

The observation obtained in the previous two experiments suggests that, increasing the number of backward time steps is likely to show a decent way to increase the prediction accuracy. And the result of adding one more backward time step information matches this conjecture nicely, as shown in Figure 34.



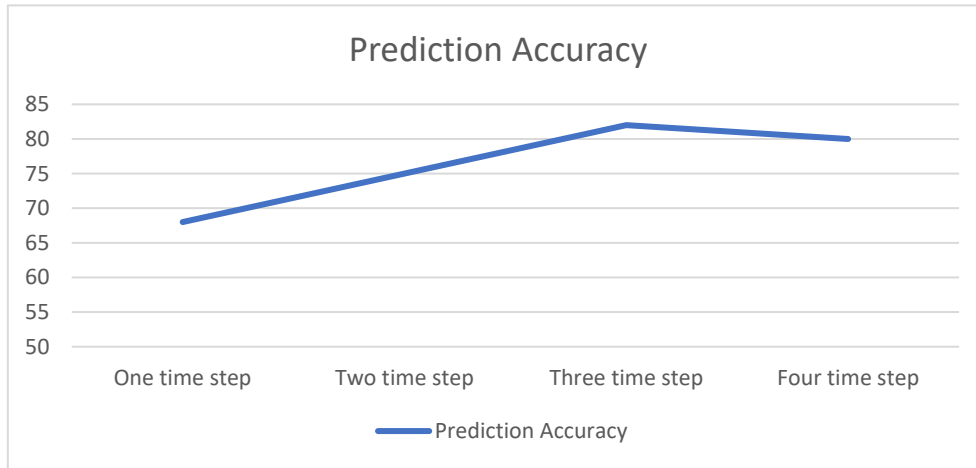


Figure 34: Prediction accuracy with respect to different historical time step used

However, Figure 34 also shows that the accuracy drops to 80% if four time steps are used as an input data set. The reason for this result is due to the lack of training data to match the number of model parameters increasing significantly with respect to the number of backward time steps used in the neural network model, set which results in the underfitting issue. After collecting enough training data, the accuracy rises to 88%!

What if all the historical time steps are used to predict the turning ratio at the next time step? In this project, this assumption could not be realized since it incurs a prohibitively high computational burden, making it unable to complete train the model training. However, it still could be illustrated if only a small number of time steps are involved. For example, the first two time steps could be used in predicting the turning ratio at the third time step.

Although only two previous time steps are considered, they include all the information. The prediction accuracy for this experiment is shown in Table 7.

	T=4	T=5	T=6	T=7	T=8	T=9	T=10
$\lambda(t), \lambda(t-1), \lambda(t-2), \lambda(t-3), \theta(t+1), 4000\text{samples}$	100%	100%	100%	95.56%	96.67%	96.67%	93.33%
$\lambda(t), \lambda(t-1), \lambda(t-2), \lambda(t-3), \theta(t+1), 2000\text{samples}$	100%	97%	100%	92.22%	90%	95.56%	93.33%
$\lambda(t), \lambda(t-1), \lambda(t-2), \theta(t+1), 4000\text{samples}$	100%	100%	100%	93.33%	96.67%	95.56%	93.33%

Table 7: Turning ratio prediction accuracy with different time horizon and sample quantity

In conclusion, it is possible to predict the turning ratio with a sufficiently high accuracy, if all the previous traffic light assignment information is known and used for training. However, in practice, it will certainly lead to high computational complexity. Thus, an acceptable trade-off between the computational complexity cost and prediction accuracy is needed.

### **B. Trained with RNNs**

The solution provided in section A seems decent with up to 90% prediction accuracy. However, the neural network model is a static model, which is only applicable to a specific time step. Thus, to cope with a long prediction horizon, many neural network models need to be constructed, which is not only computationally demanding, but also lacks of robustness to traffic disturbances. Thus, a dynamic prediction model is needed.

Recall that the problem is to use historical data to predict the turning ratio at the next time step, based on past data. The recurrent neural network is a possible choice in dealing with such a dynamic prediction problem

The off-line RNN is designed to be able to predict the turning ratio with any time step input. Thus, a three hidden layer RNN is used to confine the complexity of the problem. Each layer is also assigned with an LSTM unit to increase the performance on memorizing the essential information during the training phase. The primal model is shown in Figure 35. The first and second activation functions selected is ReLu to avoid the gradient vanishing problem and a sigmoid function is assigned to the final layer as it is required that the output should vary from 0 to 1, which is the range of each turning ratio.

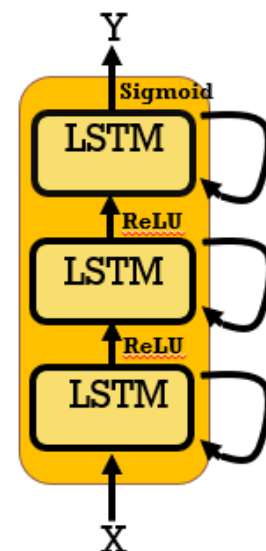


Figure 35: RNN model with three hidden layers

## Phase 1: Tuning the number of epochs

The first LSTM parameter to be tuned is the number of training epochs. The model will use a batch size of 5, and 100 neurons for each hidden layer. We will explore the effect of training this configuration for different numbers of training epoch.

### - Diagnosis of Epochs = 3

As introduced in the methodology, the number of training epochs is the number of used historical time steps. When the number of epochs equals 3, three historical data recorded as  $[X^{(t-2)}, X^{(t-1)}, X^{(t)}]$  are used to derive the corresponding output  $Y^{(t)}$ . After running the program, the recorded RMSE (root mean square error) during the training process for every 50 iterations is printed as follows:

```
Network setting: Epoch = 3, Batch size = 5, 100-100-100
TrainRMSE = 0.765673, TestRMSE = 2.480753
TrainRMSE = 0.594525, TestRMSE = 2.532568
TrainRMSE = 0.553451, TestRMSE = 2.435682
TrainRMSE = 0.518539, TestRMSE = 2.275689
TrainRMSE = 0.488321, TestRMSE = 2.214578
TrainRMSE = 0.476463, TestRMSE = 2.225736
TrainRMSE = 0.435786, TestRMSE = 2.147842
```

The results clearly show a downward trend in training RMSE over the training epochs and have a small impact on decreasing the test RMSE. It is a good sign, it shows that the developed model is able to solve this prediction problem. Next, we try to increase the number of epochs, and see how the quality of prediction may be affected.

### - Diagnosis of Epochs = 4

When epochs equal 4,  $[X^{(t-3)}, X^{(t-2)}, X^{(t-1)}, X^{(t)}]$  is used to derive its corresponding output  $Y^{(t)}$ . Running the program again, the output is shown below:

```
Network setting: Epoch = 4, Batch size = 5, 100-100-100
TrainRMSE = 0.682148, TestRMSE = 2.653387
TrainRMSE = 0.503291, TestRMSE = 2.316748
TrainRMSE = 0.436209, TestRMSE = 2.187441
TrainRMSE = 0.410452, TestRMSE = 2.073928
TrainRMSE = 0.400215, TestRMSE = 1.998349
TrainRMSE = 0.382912, TestRMSE = 1.943025
TrainRMSE = 0.378691, TestRMSE = 1.933408
```

The error is getting smaller compared to that of the previous experiment, it means increasing the number of epochs is a decent way to get a better prediction model.

- Diagnosis of Epochs = 5

$[X^{(t-4)}, X^{(t-3)}, X^{(t-2)}, X^{(t-1)}, X^{(t)}]$  is used to attach with output  $Y^{(t)}$ . The RMSE for both training set and test training set is shown below:

```
Network setting: Epoch = 5, Batch size = 5, 100-100-100
TrainRMSE = 0.712396, TestRMSE = 2.853255
TrainRMSE = 0.604337, TestRMSE = 2.375411
TrainRMSE = 0.538944, TestRMSE = 2.224692
TrainRMSE = 0.382902, TestRMSE = 2.346825
TrainRMSE = 0.340713, TestRMSE = 2.007812
TrainRMSE = 0.302451, TestRMSE = 1.893102
TrainRMSE = 0.260366, TestRMSE = 1.619003
```

It is observed that the error is higher than that of the previous experiment, but the error decreases rapidly after hundreds of iterations.

- Diagnosis of Epochs = 6

In this experiment,  $[X^{(t-5)}, X^{(t-4)}, X^{(t-3)}, X^{(t-2)}, X^{(t-1)}, X^{(t)}]$  is used to derive the output  $Y^{(t)}$ . The RMSE for both the training set and test training set is shown below:

```
Network setting: Epoch = 6, Batch size = 5, 100-100-100
TrainRMSE = 0.704826, TestRMSE = 3.193612
TrainRMSE = 0.580691, TestRMSE = 2.575439
TrainRMSE = 0.497732, TestRMSE = 2.128642
TrainRMSE = 0.428492, TestRMSE = 2.024609
TrainRMSE = 0.358942, TestRMSE = 1.943821
TrainRMSE = 0.299016, TestRMSE = 1.852743
TrainRMSE = 0.254031, TestRMSE = 1.583766
```

Although the RMSE for both the Training data and testing data is smaller than that of the previous experiment, the training time increases to more than 10 hours to train the model. Thus, after balancing the computational cost and performance, Epochs = 5 is selected as the LSTM parameter.

## Phase 2: Tuning the Batch Size

Batch size controls how often to update the weights of the network. And specially, the number of batches is normally selected as a factor of the size of the test and the training dataset. In this section, we will explore the effect of varying the batch size. We will hold the number of training epochs constant at 5. Since the size of the training data is 60, the candidates of batch size are selected as 5,10,15,20, respectively. Since the experiment with the batch size set at 5 has been carried out in the previous section, we can start from setting the batch size at 10 directly.

### - Diagnosis of 5 Epochs and Batch Size of 10

By holding the number of epochs at 5, we change the batch size to 10. Running the program again, and the RMSE for both the training set and testing set is shown below:

```
Network setting: Epoch = 5, Batch size = 10, 100-100-100
TrainRMSE = 0.675486, TestRMSE = 2.965245
TrainRMSE = 0.553349, TestRMSE = 2.478205
TrainRMSE = 0.480654, TestRMSE = 2.304337
TrainRMSE = 0.409402, TestRMSE = 2.006931
TrainRMSE = 0.348942, TestRMSE = 1.943826
TrainRMSE = 0.274579, TestRMSE = 1.623506
TrainRMSE = 0.237331, TestRMSE = 1.423683
```

It shows a faster decreasing trend in performance than that of a batch size of 5, and the computational time is around 10 minutes which is acceptable. Thus, setting the batch size at 10 is better than setting it at 5.

### - Diagnosis of 5 Epochs and Batch Size of 15

Repeating the previous experiment by changing the batch size to 15, The RMSE for both the training set and testing set is shown below:

```

Network setting: Epoch = 5, Batch size = 15, 100-100-100
TrainRMSE = 0.734766, TestRMSE = 3.659134
TrainRMSE = 0.584403, TestRMSE = 2.974042
TrainRMSE = 0.494283, TestRMSE = 2.594718
TrainRMSE = 0.332045, TestRMSE = 2.006931
TrainRMSE = 0.210385, TestRMSE = 2.204817
TrainRMSE = 0.173402, TestRMSE = 2.459373
TrainRMSE = 0.153928, TestRMSE = 2.387562

```

By increasing the batch size to 15, the RMSE of the training data set decreases rapidly while the RMSE of the test data set decreases first and then increases after hundreds of iterations. This result indicates that the model falls into overfitting problems, thus, setting the batch size to 15 seems not a wise choice.

In summary, by holding the number of epochs at 5, the best number of batch size is 10 as it has a better performance than setting the batch size at 5, while not encountering any overfitting problems. And the experiment is early stopped since the last experiment shows that keeping increasing the number of batch size has no improvement on the computational performance of the network.

### Phase 3: Tuning the number of neurons in the hidden layer

The last parameter that needs to be tuned is the number of neurons in the hidden layer. Since the work of this step is much larger than that of previous steps, it is better to decide other parameters first. From previous experiments, the most proper setting for the numbers of epochs and batch size are 5 and 10, respectively. Thus, while tuning the size of hidden layers in LSTM, these two parameters will be fixed at its best value. There are some general rules that can help to choose the range that may speed up the tuning process. The following formula provides a reference on deciding the range of the hidden layers' size :

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

$N_i$  = number of input neurons.

$N_o$  = number of output neurons.

$N_s$  = number of samples in training data set.

$\alpha$  = an arbitrary scaling factor usually 2-10.

Considering that the number of input features is 93 and the number of outputs is 72, and the number of samples in the training data set is 168,000, the suggested range of the neurons in the hidden layers is from 100 to 500. Thus, the candidates of the hidden layers' sizes are set at 100, 150, 200, 250, 300, 350, 400, 450, 500, respectively.

The tuning result is shown in the appendix. From the result, it is observed that the best combinations for the hidden layers' size is 400-300-150. Thus, the parameters for LSTM are all determined as shown in the figure below:

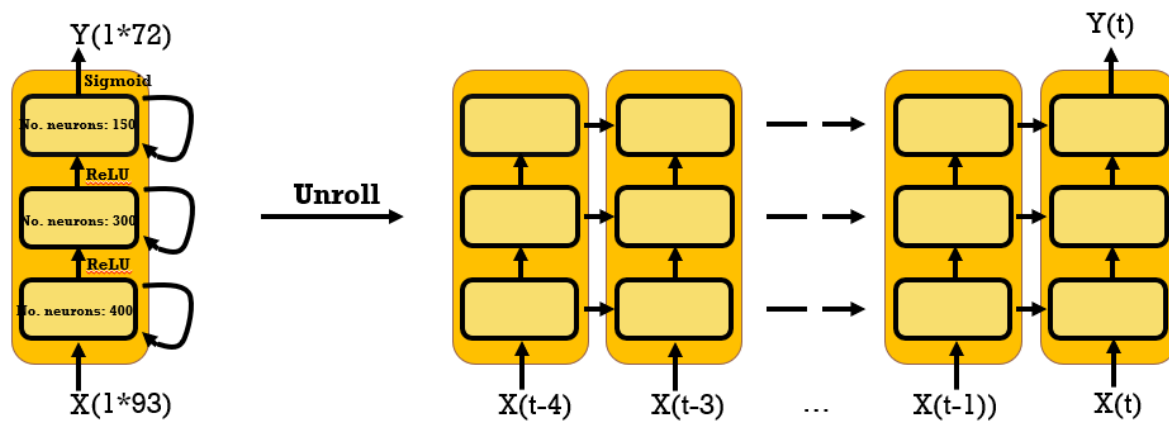


Figure 36: Tuned RNN with parameter labelled

Apparently, the accuracy is not satisfactory. Thus, an ensemble learning method is applied to boost the prediction accuracy.

#### Phase 4: Accuracy boost by applying ensemble learning

According to the literature review, building multiple different RNNs and taking the average value of the output could increase the accuracy. In order to build different prediction models, different training data sets are needed. However, the traffic data used in training the RNN is generated by a simulated traffic network. Different data subsets may not sufficiently differ from each other. Thus, instead of training multiple models with different data subsets, these models could be trained with different

output features in order to increase the differences among predictors. This idea can be realised based on the equation below:

$$T.R_{LEFT} + T.R_{STRAIGHT} + T.R_{RIGHT} = 1$$

From the above equation, the sum of the turning ratios for three directions of each link is 1. The predictors could be trained to predict either two of them. The third turning ratio could be calculated by this equation. Thus, three different predictors are trained as follows:

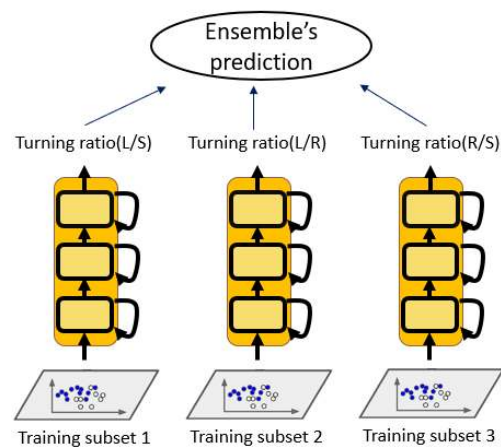


Figure 37: Ensemble learning model with three independent predictors included

Three different training data sets with the same size are collected from the VISSIM. The first dataset is trained to predict the left and straight turning ratio for all the links. Turning ratios for right direction is calculated instead of being predicted. With the same procedure, second dataset is trained to predict the turning ratio of going left and going right. The third dataset is trained to predict the turning ratio of going right and going straight.

After tuning these three models with the same steps discussed previously, three different results will be generated. Since no model is important than other two, the averaging method is used in calculating the final output. The accuracy after applying the ensemble learning method increased to 83%, which shows that the proposed ensemble method is sufficiently effective.



### **3.3.3 Comparisons and discussions**

From the above experiment, FNN could achieve higher accuracy up to 88% than RNN. However, the limitation of FNN is that this model is only applicable on specific time step, which means thousands of FNNs are needed to predict the traffic turning ratios for the entire predicting period. On the contrary, only one RNN is required to predict the turning ratios for the whole predicting horizon while the high computational complexity leads to a low prediction accuracy. However, the accuracy could be boosted by implementing multiple RNNs which are trained with different datasets and different output features in an ensemble strategy.

### 3.4 Summary

This chapter is purely application oriented. In this chapter, two neural networks are used in solving the traffic turning ratio prediction problem. By building a realistic 3\*3 simulated traffic network in VISSIM, we were able to obtain sufficient traffic data to train these two networks. The first model constructed is the feed-forward neural network. After training the FNN to predict the turning ratio for the whole prediction horizon, the accuracy of the tuned model is low, which is just a bit better than random guess. Thus, this network is only trained to predict the traffic turning ratio for a specific time. The result is promising, as it could achieve a very high prediction accuracy. In addition, an RNN is also trained to solve this problem. Due to the computational complexity, although the performance of RNN is much better than FNN, the prediction accuracy is still not satisfactory. Thus, an ensemble learning techniques is applied to boost the prediction accuracy. Considering that the training data set is generated by software which means all the vehicles are running in an ideal environment without disturbance, although reducing the outliers could enhance the performance of the neural network, the difference between different data sets may not be obvious. In order to increase the differences among local predictors, different output features were assigned to these data sets. The result is promising as it makes the prediction accuracy increased by 9%. This result also shows the potential of getting even higher accuracy by constructing more different predictors and aggregating all the predictions, which will be part of my future research topics.

## **Chapter 4: Closed-loop traffic light control: a realistic case study**

Two experiments are carried out in this chapter. In the first experiment, the traffic congestion region identification model is integrated with an adaptive traffic light control strategy, which aims to compute an optimal traffic light schedule based on real-time traffic conditions. However, this strategy requires a high computational cost which cannot be applied to all the intersections in the traffic network. Thus, instead of utilizing this optimization algorithm globally, only the intersections in the most congested region need to be adjusted and other regions remain using the default fixed-time traffic light schedule. Total queue delays inside the network after integrating this model is recorded and compared with other algorithms. In the second experiment, a closed-loop traffic light control algorithm is proposed by integrating the traffic turning ratio prediction model with the adaptive traffic light control strategy. As the turning ratio prediction plays an important role in this strategy, this experiment will explore if this turning ratio prediction model can increase the efficiency of this traffic light control strategy in reducing the delay time. And by feeding back the new traffic data, the off-line traffic turning ratio prediction model will be retrained which enables the prediction model to be adaptive to the traffic environment.

## 4.1 System setup and problem statement

Since the first experiment aims to figure out the performance of applying traffic light optimal control strategy locally, the size of the tested traffic network should be large enough. The traffic network in the Jurong area with 66 junctions is proper to this experiment. While the second experiment aims to test the performance of the integrated closed-loop traffic control algorithm. The size of the tested network should not be too large, as it will take long time to compute the result. Thus, a 3\*3 simulated traffic network is selected.

Both experiments are tested on the simulated traffic networks built in VISSIM. The data used is generated by VISSIM as well. The program runs 3600 second each time and the traffic light control algorithm adjusts the traffic light for every 15 seconds. Traffic data are collected from VISSIM.

### ***A. Integration with traffic congestion identification model***

The process of this experiment is designed as follows. After running the simulated traffic network built in VISSIM, for every 15 seconds or one time step, Traffic data (vehicle speed, vehicle density etc.) will be collected and transferred to the traffic congestion identification model. Part of the links will be clustered and identified as the most congested region based on the clustering algorithm proposed above. This result will be further transferred to the traffic light control module, which computes an optimal traffic light assignment for the next 15 seconds and feedback to the VISSIM. This process is also presented by the block diagram in figure 38 below:

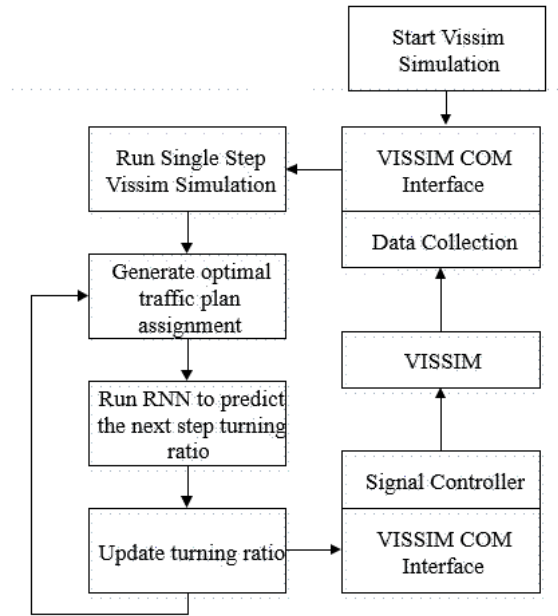


Figure 38: Block diagram for the first experiment process

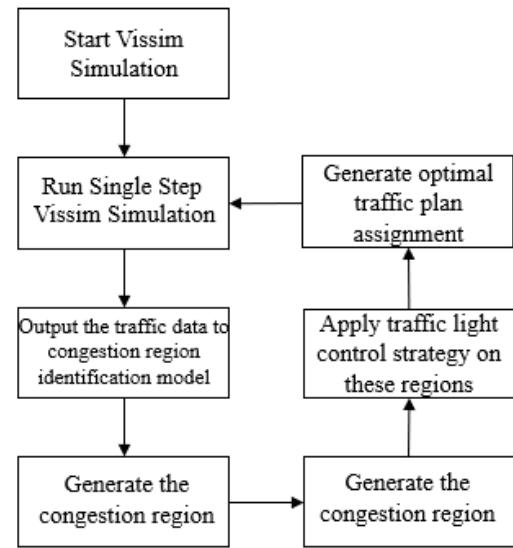


Figure 39: Block diagram for the second experiment process

### ***B. Integration with turning ratio prediction model***

The process of this experiment is designed as follows. After running the simulated traffic network built in VISSIM for a certain period, current traffic data together with the historical data within 5 time steps will be accumulated and transferred to the turning ratio prediction model. Based on the data provided by VISSIM, the predicted turning ratio at next time step for each link is calculated. This result will be further transferred to the traffic light control strategy. After that, new traffic light plan assignments can be computed and fed back to VISSIM. Then the aforementioned process is repeated for every subsequent time step. The following block diagram in the figure 39 above also shows the process:

However, this process uses off-line tuned traffic turning ratio prediction model. This model cannot adaptive to the changing environment. In order to keep the prediction model being undated, this model needs to be retrained with new data constantly. Generally, this closed-loop traffic light control strategy is put in use only during daytime. Thus, time in the night can

be utilized to retrain the model by using the traffic data recorded during the daytime. New process is shown in the block diagram below:

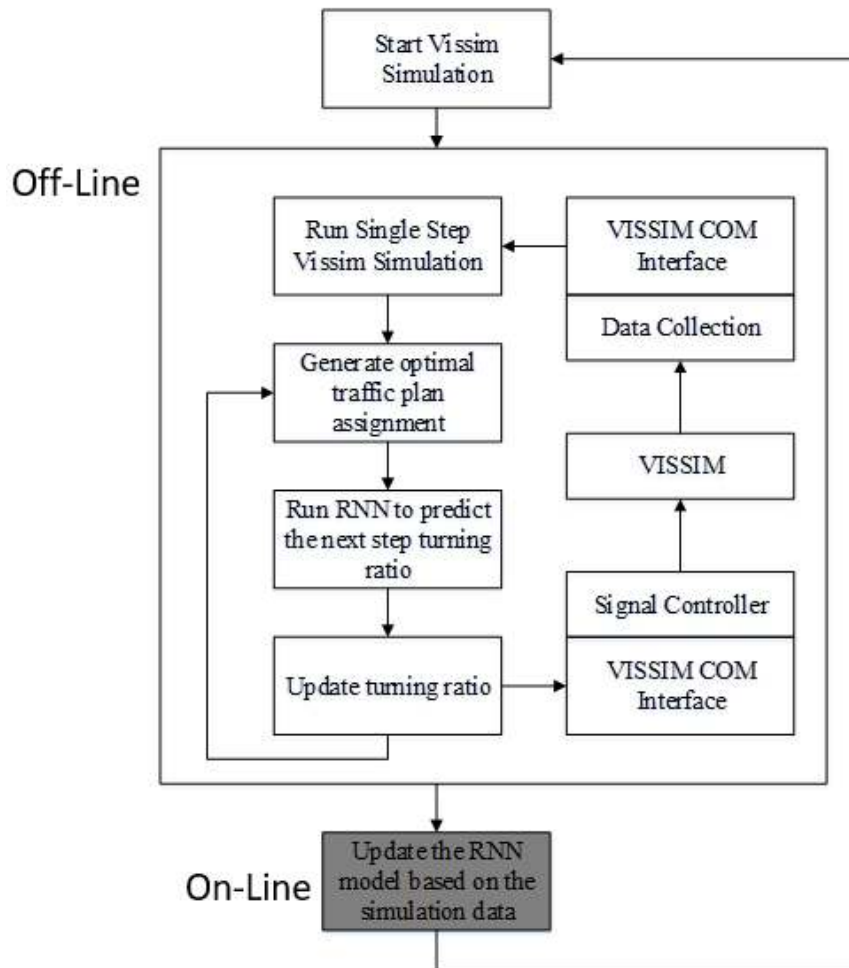


Figure 40: Improved Block diagram for the second experiment process

## 4.2 Experimental results

### A. Integration with traffic congestion identification model

The proposed congestion region identification strategy has been integrated with a traffic light scheduling algorithm [reference??], which applies different types of traffic light control schemes in regions according to their congestion levels, where pure local strategies such as the fixed-time scheme or back-pressure scheme is applied in a least congested region, and a full-fledged optimization scheme is applied to the most congested regions. The simulation result is shown in figure 16, where the total queue delay in the traffic network reduces 75% compared to the situation where only a pre-time (or fixed-time) schedule is assigned. In conclusion, identifying the congested regions in a traffic network is potentially able to increase the efficiency of real-time optimal traffic light scheduling algorithms.

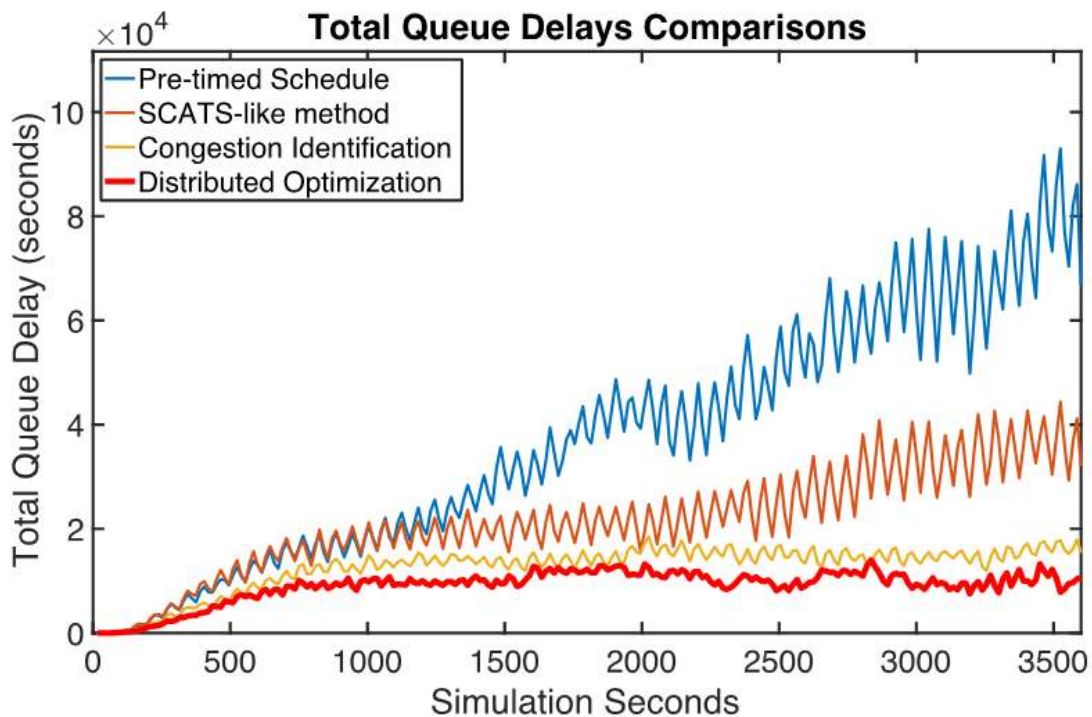


Figure 41: Simulation result for the traffic light control algorithm with different method integrated

## B. Integration with turning ratio prediction model

In this closed-loop traffic light control simulation case study, a traffic network developed in VISSIM is treated as a ground truth. The traffic light control system constantly generates data, simulating sensor data collection in a real traffic network, while the link turning ratios used in the control algorithm are provided by the RNN-based predictor. After sufficient data are collected, the VISSIM simulation is paused or switched to a fixed-time scheme, while the RNN-based predictor is re-trained based on newly collected data. Once the re-training is done, the optimal traffic light controller is back online again, and the procedure repeats. The resulting network delays are shown in Figure 42 below:

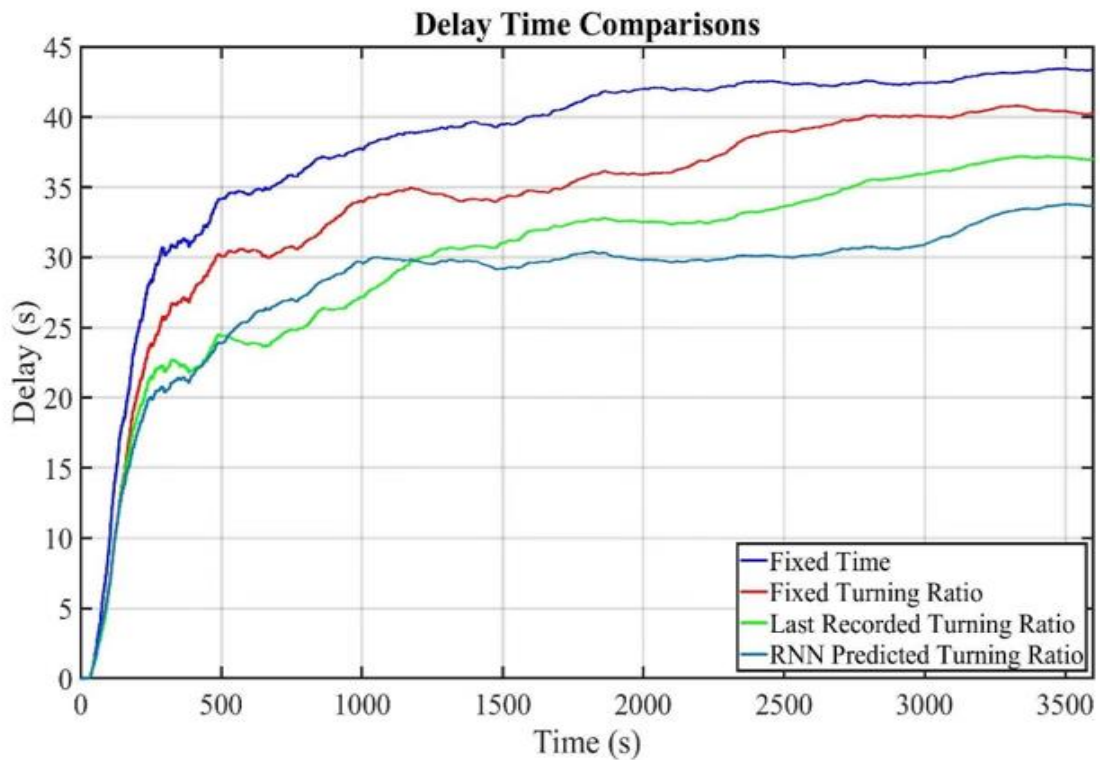


Figure 42: Simulation result for the traffic light control algorithm with different turning ratio prediction model integrated



### **4.3 Comparisons and discussions**

From the results of these two experiments, a promising trend of decreasing the delay time can be observed. Thus, by integrating the congestion identification model and turning ratio prediction model, the performance of the adaptive traffic control strategy is improved noticeably. Further improvements could be carried out on implementing the traffic signal controller with genetic algorithms and predict the turning ratios for multiple steps.

# Chapter 5: Conclusions and Recommendations

## 5.1 Conclusions

This research aims to provide solutions for the congestion region identification and traffic network turning ratio prediction. In the first part, a proper way of determining the congestion level is proposed. Two different clustering algorithms are used in identifying the congestion regions. Either method provides a feasible result which helps to reduce the traffic delay time by 4 times. In the second part, two neural network models are constructed. The FNN could achieve higher prediction accuracy but with high computational complexity. The RNN is more convenient to use but with relatively low accuracy. After that, a commonly used ensemble learning method is applied which successfully increases the prediction accuracy of the RNN model by 9%. Lastly, the RNN model is integrated with a real-time optimal traffic light control strategy and significantly reduces the overall traffic delay time in the simulation, when compared to several other methods.

## 5.2 Recommendations

More detailed and complete literature review needs to be discussed as many state-of-art algorithms are not included. Those algorithms should also be further applied into above cases with clear and qualitative comparisons.

The evolution of the traffic congestion regions deserves to be further explored, as such knowledge could help a network-wise real-time traffic signal control strategy generate timely optimal traffic light schedules to reduce or eliminate the impact of the traffic congestion.

Furthermore, a self- adaptive RNN model needs to be further developed, which could take the traffic data collected during the day time, and autonomously update the network parameter in a regular basis. The self-adaptation could increase the turning ratio prediction accuracy and robustness of the closed-loop traffic light control system to network changes.

## Chapter 6: Appendix

### 6.1 Tuning result for different combination of hidden layers size

- Number of neurons in first hidden layer = 100, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	58.52%	57.61%	62.57%	59.37%	63.55%	61.74%	62.32%	59.10%	63.29%
150	60.36%	68.92%	59.84%	53.32%	59.52%	54.19%	58.97%	61.96%	61.88%
200	54.67%	63.5%	59.7%	66.1%	52.06%	55.89%	50.67%	56.67%	57.24%
250	61.82%	69.67%	64.86%	52.9%	66.43%	64.42%	56.15%	58.3%	54.48%
300	61.09%	60.05%	62.62%	60.21%	62.85%	56.62%	51.15%	66.22%	60.83%
350	57.88%	61.95%	63.56%	51.83%	69.5%	66.91%	67.58%	66.49%	55.48%
400	58.23%	63.92%	55.86%	59.42%	55.84%	64.24%	57.8%	59.23%	63.95%
450	48.21%	59.53%	60.02%	63.54%	64.58%	64.25%	66.32%	59.82%	53.71%
500	56.34%	51.23%	72.49%	63.41%	61.19%	65.79%	54.32%	67.4%	53.36%

- Number of neurons in first hidden layer = 150, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	56.42%	51.12%	63.59%	64.32%	62.85%	64.73%	68.22%	65.12%	53.21%
150	62.41%	64.51%	55.64%	52.63%	55.24%	59.25%	53.34%	63.63%	62.81%
200	64.62%	54.02%	54.23%	64.83%	50.75%	52.53%	54.62%	53.69%	52.2%
250	63.8%	64.35%	58.42%	54.04%	63.26%	65.42%	52.56%	57.2%	51.63%
300	62.74%	69.67%	67.53%	66.51%	72.45%	66.52%	50.1%	68.27%	63.75%
350	54.56%	66.5%	63.42%	61.03%	63.69%	63.41%	65.62%	63.45%	53.44%
400	53.2%	57.95%	64.87%	56.45%	54.04%	69.28%	57.5%	54.33%	63.31%
450	58.01%	53.54%	63.43%	62.56%	63.57%	54.64%	63.65%	56.66%	55.25%
500	57.35%	55.77%	56.75%	65.45%	71.67%	65.64%	52.35%	64.33%	56.63%

- Number of neurons in first hidden layer = 200, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	53.45%	56.11%	63.49%	54.01%	64.33%	54.77%	64.84%	57.8%	64.23%
150	69.86%	63.34%	57.45%	54.64%	50.34%	67.37%	64.34%	64.72%	61.51%
200	61.51%	53.72%	56.65%	62.54%	52.46%	53.43%	58.76%	55.07%	52.56%
250	62.56%	62.3%	50.62%	63.53%	62.72%	63.81%	62.59%	53.62%	53.59%
300	63.04%	68.68%	69.64%	62.62%	68.83%	63.72%	51.34%	64.2%	63.05%
350	56.06%	64.56%	62.62%	63.85%	64.78%	64.63%	65.93%	54.73%	52.98%
400	68.41%	66.31%	59.45%	55.39%	64.03%	69.42%	64.39%	67.02%	64.44%
450	61.26%	67.65%	62.76%	51.54%	67.36%	57.44%	53.05%	52.76%	52.26%
500	61.41%	55.93%	67.72%	60.61%	65.07%	65.17%	51.72%	63.86%	57.54%

- Number of neurons in first hidden layer = 250, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	61.18%	63.63%	65.41%	55.42%	55.52%	54.52%	64.51%	52.42%	62.87%
150	62.03%	64.52%	54.36%	53.37%	55.61%	64.14%	70.22%	64.56%	62.16%
200	56.92%	68.23%	54.07%	64.31%	55.83%	54.8%	52.65%	53.53%	63.23%
250	64.42%	62.62%	63.12%	53.05%	62.31%	58.43%	52.04%	63.32%	62.82%
300	58.51%	61.52%	55.63%	57.38%	57.31%	58.53%	62.45%	63.21%	60.38%
350	54.3%	62.09%	62.57%	56.52%	64.05%	63.41%	65.55%	63.16%	54.56%
400	52.04%	57.21%	53.63%	57.51%	54.72%	62.42%	59.42%	65.12%	63.54%
450	61.23%	54.52%	70.02%	63.34%	62.62%	63.51%	62.52%	54.51%	65.12%
500	53.19%	54.52%	66.31%	63.86%	67.84%	65.73%	65.43%	67.52%	59.97%

- Number of neurons in first hidden layer = 300, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	53.45%	55.13%	64.63%	63.63%	62.64%	69.32%	64.66%	65.02%	64.53%
150	63.56%	65.45%	53.1%	55.63%	55.73%	53.75%	63.93%	64.83%	56.34%
200	64.83%	57.83%	57.3%	62.73%	56.92%	53.95%	63.75%	53.65%	53.66%
250	66.05%	62.3%	55.07%	53.76%	63.82%	64.83%	54.83%	55.28%	57.03%
300	61.62%	70.33%	66.53%	66.73%	63.67%	67.54%	50.05%	66.32%	66.45%
350	58.52%	67.52%	63.42%	65.43%	63.42%	66.42%	64.42%	64.42%	54.05%
400	63.04%	56.83%	63.76%	52.77%	54.62%	63.95%	61.12%	55.83%	60.16%
450	63.28%	52.44%	61.31%	58.42%	62.31%	57.63%	63.53%	71.64%	65.28%
500	55.36%	56.34%	52.78%	65.61%	64.37%	65.62%	55.62%	67.31%	61.42%

- Number of neurons in first hidden layer = 350, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	63.61%	57.51%	67.52%	62.98%	60.5%	63.51%	61.61%	63.61%	62.62%
150	61.29%	64.82%	58.66%	57.5%	58.03%	59.41%	59.99%	63.81%	63.63%
200	64.61%	57.53%	56.72%	63.77%	63.5%	55.02%	58.09%	53.51%	64.67%
250	65.4%	52.47%	68.44%	64.44%	63.66%	66.32%	59.08%	53.41%	60.33%
300	66.7%	65.63%	67.31%	64.62%	64.96%	68.55%	63.52%	65.24%	60.52%
350	57.34%	65.72%	62.11%	60.41%	65.61%	62.03%	64.6%	63.99%	63.24%
400	56.12%	58.82%	63.54%	58.42%	59.63%	67.92%	54.03%	52.51%	63.16%
450	59.52%	64.56%	69.44%	62.04%	63.62%	57.72%	65.71%	54.71%	64.73%
500	60.52%	61.94%	65.86%	65.9%	66.72%	66.66%	63.72%	65.72%	63.72%

- Number of neurons in first hidden layer = 400, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	63.42%	63.31%	70.12%	62.31%	67.63%	68.34%	66.98%	57.15%	63.63%
150	63.42%	66.73%	63.62%	67.05%	74.34%	62.15%	69.62%	65.81%	63.75%
200	59.42%	70.53%	72.41%	67.42%	66.08%	68.64%	67.53%	72.64%	68.23%
250	71.82%	63.67%	70.86%	66.93%	64.51%	67.84%	64.33%	62.52%	67.42%
300	66.93%	65.73%	66.88%	63.04%	60.36%	63.77%	67.64%	71.64%	66.42%
350	57.8%	65.04%	65.62%	63.84%	70.53%	72.42%	64.75%	66.46%	68.09%
400	73.27%	64.63%	72.06%	65.74%	68.62%	64.53%	70.07%	65.64%	69.35%
450	63.27%	63.57%	59.85%	63.73%	63.56%	64.73%	68.42%	62.52%	67.53%
500	64.63%	67.08%	70.52%	67.66%	67.09%	71.05%	53.53%	67.3%	63.76%

- Number of neurons in first hidden layer = 450, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	63.42%	62.52%	59.04%	65.62%	70.42%	67.31%	65.52%	68.88%	57.32%
150	64.52%	68.21%	64.52%	64.03%	67.23%	71.3%	64.41%	67.63%	65.62%
200	66.73%	65.72%	57.23%	66.72%	65.52%	67.42%	62.77%	67.82%	63.63%
250	63.45%	69.76%	64.73%	64.72%	68.53%	65.53%	68.64%	60.05%	72.73%
300	67.54%	66.67%	67.55%	70.55%	67.85%	65.57%	62.74%	66.82%	65.04%
350	59.74%	64.42%	66.53%	70.05%	67.24%	70.63%	69.82%	68.63%	67.44%
400	58.42%	57.45%	66.82%	63.74%	69.53%	64.26%	63.63%	68.42%	64.82%
450	60.7%	62.86%	57.74%	57.57%	65.64%	59.64%	62.05%	69.66%	67.83%
500	58.42%	56.5%	59.63%	64.52%	62.53%	63.74%	62.73%	63.52%	68.42%

- Number of neurons in first hidden layer = 500, the first row shows the size of the second hidden layer and the first column shows the size of third hidden layer.

Prediction Accuracy	100	150	200	250	300	350	400	450	500
100	63.52%	63.62%	65.74%	66.73%	66.72%	67.42%	60.24%	58.42%	62.2%
150	62.04%	63.42%	70.6%	62.53%	63.88%	66.63%	67.73%	57.74%	65.71%
200	69.53%	58.53%	57.42%	63.52%	62.52%	62.78%	69.42%	70.65%	64.56%
250	60.46%	68.62%	63.62%	62.52%	70.73%	63.64%	64.74%	63.83%	62.74%
300	59.42%	62.41%	64.62%	67.41%	64.62%	68.62%	63.62%	60.63%	67.74%
350	58.72%	65.62%	64.5%	64.86%	64.82%	66.55%	60.73%	56.53%	60.63%
400	61.22%	63.65%	63.41%	67.99%	67.92%	69.42%	66.31%	61.62%	63.43%
450	65.73%	60.92%	58.48%	62.85%	62.73%	64.07%	59.83%	63.62%	63.26%
500	67.35%	62.74%	70.23%	68.62%	63.45%	62.67%	63.73%	67.72%	62.84%

# Bibliography

- [1] L.G. Anthopoulos, The rise of the smart city, in: Understanding Smart Cities: A Tool for Smart Government or an Industrial Trick? Public Administration and Information Technology, vol. 22, Springer, Cham, 2017.
- [2] A. Das, P.K. Dash, B.K. Mishra, An intelligent parking system in smart cities using IoT, in: Exploring the Convergence of Big Data and the Internet of Things, IGI Global, Hershey, PA, 2018, pp. 155–180.
- [3] Y. Zhang, R. Su, and K. Gao, “Urban road traffic light real-time scheduling,” in Decision and Control (CDC), 2015 IEEE 54th Annual Conference on. IEEE, 2015, pp. 2810–2815.
- [4] ECMT (ed.) (1999) The spread of congestion in Europe, Report on the 110th Round Table on Transport Economics, Paris: OECD Publication Service.
- [5] Weisbrod, G., Vary, D., and Treyz, G. (2001) Economic Implications of congestion, NCHRP Report 463, Washington, DC.: Transportation Research Board.
- [6] Victoria Transport Policy Institute (VTPI) (2005) Congestion reduction strategies: identifying and evaluating strategies to reduce congestion, in: Online TDM Encyclopaedia, Victoria, British Columbia, Canada: Victoria Transport Policy Institute.
- [7] Downs, A. (2004) Still stuck in traffic: coping with peak-hour traffic congestion, Washington, D.C.: The Brookings Institution.
- [8] Turner, S.M. (1992) Examination of indicators of congestion level, Transportation Research Record: Journal of the Transportation Research Board, No. 1360, pp.150-157.
- [9] Turner, S.M., Lomax, T.J. and Levinson, H.S. (1996) Measuring and estimating congestion using travel time-based procedures, Transportation Research Record: Journal of the Transportation Research Board, No. 1564, pp. 11-19.
- [10] Schrank, D. and Lomax, T. (1997) Urban roadway congestion: 1982-1994, volume 1 &2, Research Report 1131-9, Texas: Texas Transportation Institute.
- [11] <https://mobility.tamu.edu/umr/>
- [12] Texas Transportation Institute (2005) The Keys to Estimating Mobility in Urban Areas Applying Definitions and Measures That Everyone Understands, The Texas A&M University System

- [13] Roess, R.P., Messer, C.J., Mcshane, W.R., Fruin, J.J., Levinson, H.S. May, A.D. and Dudek, C.L. (1985) Highway Capacity Manual, Special Report 209, Washington, D.C.: Transportation Research Board.
- [14] Williams, B.M., Hoel, L.A., 2003. Modelling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results. *Journal of Transportation Engineering – ASCE* 129, 664–672.
- [15] Smith, B.L., Williams, B.M., Oswald, R.K., 2002. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C* 10, 303–321.
- [16] Ghosh, B., Basu, B., O’Mahony, M.M., 2005. Time-series modelling for forecasting vehicular traffic flow in Dublin. *Transportation Research Board Annual Meeting*, Washington, DC.
- [17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [18] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Acoustics, speech and signal processing (icassp), 2013 IEEE international conference on*. IEEE, 2013, pp. 6645– 6649.
- [19] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.
- [20] D. Eck and J. Schmidhuber, “A first look at music composition using lstm recurrent neural networks,” *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, 2002.
- [21] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.
- [22] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,” *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.
- [23] Y. Duan, Y. Lv, and F.-Y. Wang, “Travel time prediction with lstm neural network,” in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 1053–1058. 2

- [24] Y.-y. Chen, Y. Lv, Z. Li, and F.-Y. Wang, "Long short-term memory model for traffic congestion prediction with online open data," in Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on. IEEE, 2016, pp. 132–137.
- [25] Y. Wu and H. Tan, "Short-term traffic flow forecasting with spatial temporal correlation in a hybrid deep learning framework," arXiv preprint arXiv:1612.01022, 2016.
- [26] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in Proceedings of the 2017 SIAM International Conference on Data Mining. SIAM, 2017, pp. 777–785.
- [27] X. Song, H. Kanasugi, and R. Shibasaki, "Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level." in IJCAI, 2016, pp. 2618–2624.
- [28] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "," arXiv preprint arXiv:1705.02699, 2017.
- [29] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in Chinese Association of Automation (YAC), Youth Academic Annual Conference of. IEEE, 2016, pp. 324–328.
- [30] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "Lstm network: a deep learning approach for short-term traffic forecast," IET Intelligent Transport Systems, vol. 11, no. 2, pp. 68–75, 2017.
- [31] Wang, Wei et al. "STING: A Statistical Information Grid Approach to Spatial Data Mining." VLDB (1997).
- [32] Agrawal, R., Gehrke, J., Gunopulos, D. & Raghavan, P. (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications.. In L. M. Haas & A. Tiwary (eds.), SIGMOD Conference (p./pp. 94-105), : ACM Press. ISBN: 0-89791-995-5
- [33] Jaiwei Han and Micheline Kamber, "Datamining: Concepts and Techniques", Morg Kaufman Publishers, 2001.
- [34] Ali, K., & Pazzani, M. (1996). Error reduction through learning multiple descriptions. Machine Learning, 24, 173–202.
- [35] Alpaydin, E. (1993). Multiple networks for function learning. In Proceedings of the 1993 IEEE International Conference on Neural Networks, Vol. I, pp. 27–32 San Francisco.
- [36] Breiman, L. (1996c). Stacked regressions. Machine Learning, 24(1), 49–64



- [37] Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156 Bari, Italy.
- [38] S.W. Chiou, TRANSYT derivatives for area traffic control optimisation with network equilibrium flows, *Trans. Res. B.* 37 (3) (2003) 263–290.
- [39] P. Lowrie, “Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic,” Roads and Traffic Authority NSW, Darlinghurst, NSW Australia, 1990.
- [40] P. Hunt, D. Robertson, R. Bretherton, and M. C. Royle, “The scoot on-line traffic signal optimisation technique,” *Traffic Engineering & Control*, vol. 23, no. 4, 1982.
- [41] Rongrong Tian , Xu Zhang, “Design and Evaluation of an Adaptive Traffic Signal Control System – A Case Study in Hefei”, China 2016 ,International Symposium of Transport Simulation (ISTS’16 Conference), June 23~25, 2016
- [42] Jianhua Guo, Ye Kong, Zongzhi Li, Wei Huang, Jinde Cao, Yun Wei , “A model and genetic algorithm for area-wide intersection signal optimization under user equilibrium traffic”, International Association for Mathematics and Computers in Simulation (IMACS), 0378-4754 (2017).
- [43] Gustav Nilsson \_ Giacomo Como, “On Generalized Proportional Allocation Policies for Traffic Signal Control”, *International Federation of Automatic Control*, 50-1 (2017) 9643–9648.
- [44] Junchen Jin , Xiaoliang Ma, “A group-based traffic signal control with adaptive learning ability”, *Engineering Applications of Artificial Intelligence* 65 (2017) 282–293.
- [45] Nasser R. Sabar , Le Minh Kieu , Edward Chung , Takahiro Tsubota , Paulo Eduardo Maciel de Almeida, “A memetic algorithm for real world multi-intersection traffic signal optimisation problems”, *Engineering Applications of Artificial Intelligence* 63 (2017) 45– 53
- [46] Mohammad Aslani, Mohammad Saadi Mesgari, Marco Wiering, “Adaptive traffic signal control with actor-critic methods in a real world traffic network with different traffic disruption events”, *Transportation Research Part C* 85 (2017) 732–752
- [47] R. P. Roess, E. S. Prassas, and W. R. McShane, *Traffic engineering*.
- [48] Schmitz, J.E., Zemp, R.J., Mendes, M.J., 2006. Artificial neural networks for the solution of the phase stability problem. *Fluid Phase Equilib.* 245, 83–87.

- [49] Chouai, A., Laugier, S., Richon, D., 2002. Modeling of thermodynamic properties using neural networks: application to refrigerants. *Fluid Phase Equilib.* 199, 53–62.
- [50] Levenberg, K., 1944. A method for the solution of certain problems in least squares. *Q. Appl. Math.* 2, 164–168.
- [51] Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20, 273–297.
- [52] Huang, G.B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multi class classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42(2), 513–529.
- [53] Breiman, L. (2001). Random forests. *Mach. Learn.*, 45, 5–32.
- [54] Friedman, J.H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4), 367–37
- [55] Ye, J., Chow, J.-H., Chen, J., & Zheng, Z. (2009). Stochastic gradient boosted distributed decision trees. In D.W.-L. Cheung, I.-Y. Song, W.W. Chu, X. Hu, & J.J. Lin (Eds.), *CIKM* (pp. 2061–2064). ACM.
- [56] Chapelle, O., & Chang, Y. (2011). Yahoo! Learning to rank challenge overview. In O. Chapelle, Y. Chang, & T.-Y. Liu (Eds.), *Yahoo! Learning to rank challenge*. In *JMLR Proceedings*: 14 (pp. 1–24).
- [57] Dellaert, Frank (2002). "The Expectation Maximization Algorithm".
- [58] Brendan J. Frey; Delbert Dueck (2007). "Clustering by passing messages between data points". *Science*. 315 (5814): 972–976