**UNIVERSITY OF SOUTHAMPTON**

Faculty of Physical Sciences and Engineering
Electronics and Computer Science

# A Deep Learning Method for Traffic Prediction

*by* **Yanzhe (Solomon) Zhang**

April 29, 2024

Project Supervisor: Professor Lie-Liang Yang

A progress report submitted for the award of MEng
Electronic Engineering with Artificial Intelligence

-

University of Southampton

Abstract

Faculty of Physical Sciences and Engineering
Electronics and Computer Science

**A Deep Learning Method for Traffic Prediction**

by Yanzhe (Solomon) Zhang

This is the progress report of the Part III Project.

In contemporary urban environments, efficient urban mobility is imperative for daily routines. Traffic congestion stands as a pervasive challenge in urban areas, leading to delays, frustration, and inefficiencies in transportation systems. This project aims to explore traffic prediction using deep learning techniques, specifically recurrent neural networks (RNNs), to provide accurate insights into future traffic conditions.

As reached halfway through the project, the dataset had been found, data had been preprocessed, and an LSTM model had been constructed. In the next term, it is planned to optimize the hyperparameters of the model and add spatial aspects to it.

**Statement of Originality**

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

***You must change the statements in the boxes if you do not agree with them.***

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

**I have acknowledged all sources, and identified any content taken from elsewhere.**

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

**I have not used any resources produced by anyone else.**

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

**I did all the work myself, or with my allocated group, and have not helped anyone else.**

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

**The material in the report is genuine, and I have included all my data/code/designs.**

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

**I have not submitted any part of this work for another assessment.**

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

**My work did not involve human participants, their cells or data, or animals.**

# Contents

# Chapter 1: Introduction

## 1.1 Project Problem

In modern, fast-paced lives, urban mobility is an integral aspect that significantly impacts our daily routines. One of the most pervasive challenges in urban areas is the issue of traffic congestion, causing delays, frustration, and inefficiencies in transportation systems. This poses high requirements for traffic management and navigation systems addressing these problems. In the topic of traffic management and planning, accurate prediction of vehicles and proactively suggesting optimal routes in urban road networks is one of the important tasks to improve traffic efficiency, and safety, and to mitigate traffic congestion. This project seeks to explore the realm of traffic prediction, employing deep learning techniques to provide accurate insights into future traffic conditions.

This project involves the development of a sophisticated model using recurrent neural network (RNN)-based methods. The model will not only consider historical traffic data but also incorporate environmental factors such as time of day, whether it is a weekday, and whether it is a holiday. By embracing a comprehensive approach, we aim to create a predictive system that can adapt to various conditions.

## 1.2 Goals

The main milestones to be done for this project iterates:

- Gather appropriate datasets consisting of road information, and congestion situations in a time sequence.
- Modify the datasets to find some features that are vital for prediction.
- Create a deep learning model to predict the average travel time of vehicles at a certain time in each road section.
- Evaluate the performance and accuracy of the model.

- Find ways to improve the performance, and evaluate again.
- Create a pathfinding algorithm, and use the result of the prediction to make decisions on optimal routes.

# Chapter 2: Theoretical Background

## 2.1 Machine Learning

Machine learning focuses on developing algorithms and models that enable computers to learn patterns from data and make predictions or decisions without being explicitly programmed for the task [7]. The use of statistical techniques such as linear regression allows it to improve its performance over time as it is exposed to more information. There are three commonly used machine learning approaches [1] including supervised learning, where the algorithm is trained on labeled data; unsupervised learning, where the algorithm discovers patterns in unlabeled data; and reinforcement learning, where the algorithms learn through trial and error based on feedback from its actions.

### 2.1.1 Performance Measure

In both machine learning and deep learning, the cost function is often the target to minimize. This function measures how well a model is performing on the training data. In this project, mean squared error (MSE) and mean absolute percentage error (MAPE) have been considered. The functions of the two indicators are shown below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{2.1}$$

MSE calculates the square of the difference between the predicted value $\hat{y}_i$ and the actual value $y_i$, and takes the mean of it [1]. The value is in the range of $[0, +\infty)$, when the predicted value equals the real value, MSE is 0. As the difference gets larger, MSE increases.

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{2.2}$$

Instead of the actual value of the loss, MAPE is more focused on percentage [3]. The range of MAPE is also $[0, +\infty)$, where a 0% MAPE means a perfect match, and above 100% would be considered as bad. Note that when the real value is in the denominator, which means it is not useable for any data set that contains a real value of 0.

### 2.1.2 Gradient Descent

Gradient descent is an optimization algorithm used to minimize the cost function iteratively. The gradient or derivative of the cost function at a point tells the direction of the steepest increase of the function. Hence if the parameters are updated in the opposite direction of the gradient, the model will be closer, and potentially reach the mimimum cost. The size of the the step of each update is controlled by the learning rate.

For example in linear regression, we shell fit the data using a polynomial function [1]:

$$\hat{y}(x, \mathbf{w}) = \omega_0 + \omega_1 x^1 + \omega_2 x^2 + \cdots + \omega_n x^n = \sum_{i=0}^{n} \omega_i x^i \tag{2.3}$$

Where the function has an order of n, with parameters $\omega_i$ controlling each term. These parameters are also denoted as vector $\mathbf{w}$ for future convenience.

The cost function using MSE hence can be denoted as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}(x_i, \mathbf{w}) - y_i)^2 \tag{2.4}$$

A step of gradient descent can be represented as below, repeat the process until achieving a certain level of accuracy, or other convergence criteria.

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \alpha \times \nabla(\text{MSE}) \tag{2.5}$$

### 2.1.3 One-hot Encoding

Sometimes the data used in machine learning contains categorical variables, which represent categories or labels, such as colors and the class of a road. Each value of these variables is independent, the size does not matter. To let the model understand this better, one-hot encoding could be introduced.

One-hot encoding is a technique used to represent those categorical variables as binary vectors. The length of the vectors is equal to the number of unique categories, and each position in the vector corresponds to a specific category. If the data is in a category, the corresponding position is set to be 1, otherwise, fill with 0. With this implemented, each category is treated as an independent entity and does not impose any ordinal relationship.

## 2.2 Deep Learning

Deep learning is a subset of machine learning, which includes the use of neural networks. It is designed to solve complex problems with large datasets. Popular deep learning architectures include convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers.

### 2.2.1 Neural Network

At the core of deep learning are artificial neural networks (ANN), which are inspired by the structure and functioning of the human brain. The structure of a neural network is illustrated in Figure 2.1. It is composed of layers of interconnected nodes (neurons). The input layer is where the data is fed into the network, and the output layer produces the final result. Between the input and output layers, there are one or more hidden layers each node in the hidden layers has a usually non-linear activation function. Section 2.2.2 talks more about them. Lines between layers are linear transformations, with the parameter vector $\mathbf{w}$, and bias b.

What the network trying to do is the same as machine learning: find a function that best fits the data, so it can make predictions. The process could be expressed as:

$$\mathbf{y} = \mathbf{w}^{II} \times A_k(\mathbf{w}^{\mathbf{I}}\mathbf{x} + b_k^I) + b^{II} \tag{2.6}$$

The inputs fed into the network first go through a linear transformation and then map to an activation function. At last, go through a linear transformation again and output. By altering the parameters, it is possible to construct any curve.

Deep learning emphasizes the use of deep neural networks, which means networks with multiple hidden layers. These networks are capable of learning complex datasets with non-intuitive characteristics. The depth allows the network to automatically learn features at different levels of abstraction.

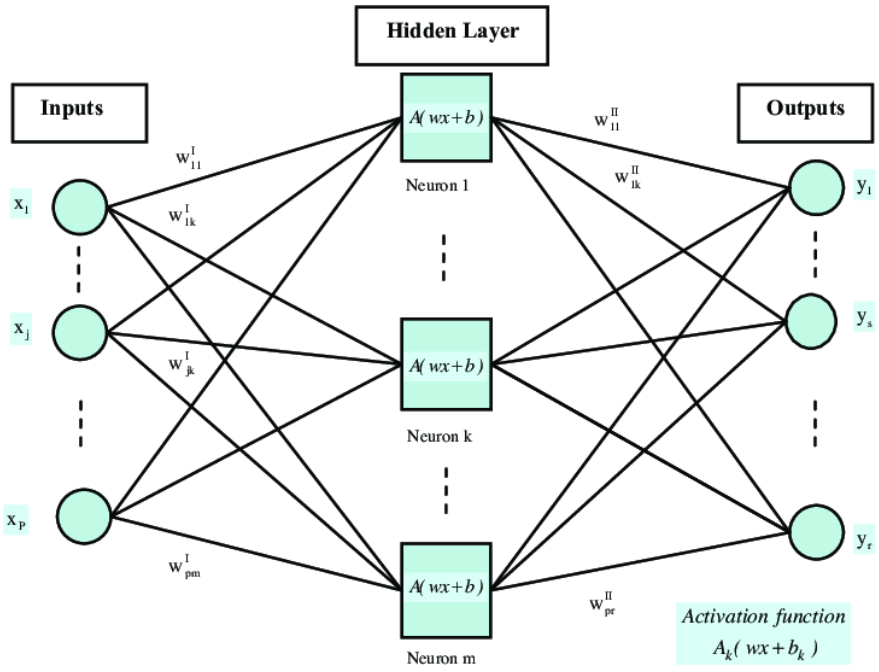



FIGURE 2.1: A simple ANN structure with one hidden layer (Adapted from [4])

Backpropagation is a key used in training deep neural networks. It is a supervised learning algorithm that adjusts the parameters **w** by propagating the error backward from the output layer to the input layer. This allows the network to update parameters to optimize predictions. Gradient descent is one of the backpropagation methods. Knowing the error from prediction, it measures which connection in the hidden layer.

### 2.2.2 Activation Functions

The choice of activation function is vital in neural networks by introducing non-linearity into the model [2]. It enables the network to learn complex patterns and relationships in the data.

#### 2.2.2.1 Rectified Linear Unit (ReLU)

ReLU is one of the most commonly used activation functions. It replaces all negative values in the inputs with zero, which can be represented by:

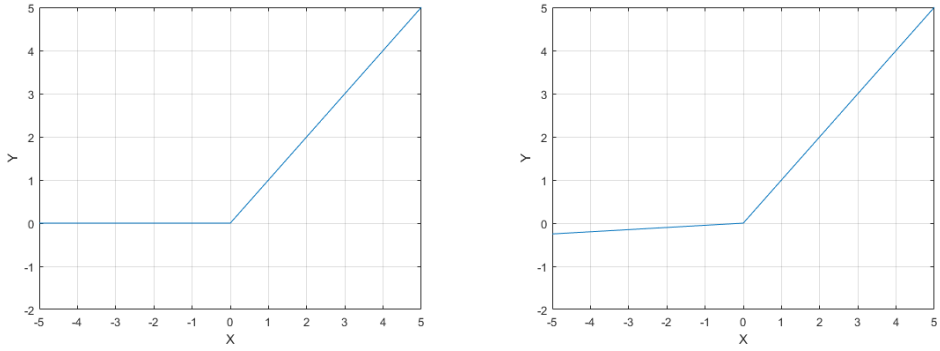
$$f(x) = \max(0, x) \tag{2.7}$$

During the training, it will not activate all neurons at the same time, which gives an advantage that it is computationally efficient. However, when $x < 0$, the gradient is zero. As training progresses, neurons may become inactive and weights fail to update.

To solve this problem, a variant of ReLU, Leaky ReLU, could be used instead. Leaky ReLU allows a small, non-zero gradient when the input is negative.

$$f(x) = \max(\alpha x, x) \tag{2.8}$$

Where $\alpha$ is a small positive constant. A plot of both functions is shown in Figure 2.2




(A) ReLU (B) Leaky ReLU

FIGURE 2.2: ReLU activation functions

It is also possible to use the exponential linear units (ELU) to address the problem of inactive neurons. It uses $e^x - 1$, which gives a similar curve, but has a small negative value at $x < 0$.

#### 2.2.2.2 Other Functions

There are many other activation functions such as sigmoid, hyperbolic tangent (Tanh), Softmax, etc. Those are suitable for different purposes. For example, sigmoid have a limited output range, suitable to use before output. However, the curve gets too smooth at two ends, which causes a low learning efficiency. It is more used in classification problems.

## 2.3 Recurrent Neural Networks (RNNs)

RNNs are a type of ANN designed for sequence data where the order of the data points is crucial [6]. This makes RNN a good approach for this project. Similar to the simple ANN, RNN is constructed by multiple RNN cells. The formula of each cell is given by:

$$\mathbf{h}_t = A(\mathbf{w}^{\mathbf{I}}\mathbf{x}_t + \mathbf{w}^{\mathbf{II}}\mathbf{h}_{t-1} + b) \tag{2.9}$$

Where $\mathbf{x}_t$ is the input at time $t$, $\mathbf{h}_t$ is the output at $t$, hence, $\mathbf{h}_{t-1}$ is the output at time $t-1$. Everything else is the same as the simple ANN. The essence of RNN is that the output of the current moment will take place in the calculation of the next moment as one of the inputs. Figure 2.3 illustrates the idea of RNN.

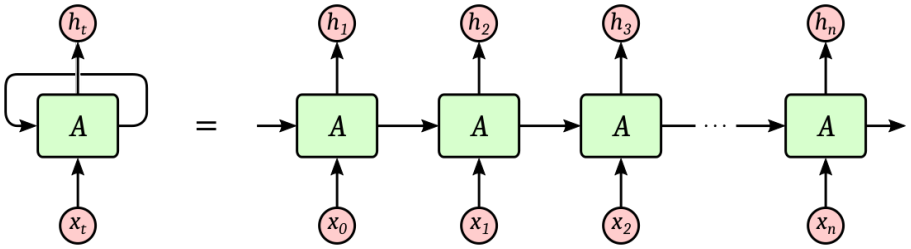



FIGURE 2.3: RNN as a neural network very deep in time (Adapted from [5])

RNNs are susceptible to the vanishing/exploding gradient problem [8]. Since the RNN uses the backpropagation to minimize the cost function, and the error has been calculated from outputs going back through the network to update the weights, those weights in the feedback loop ($\mathbf{w_{rec}}$) are multiplied a lot of times during the backpropagation. If $\mathbf{w_{rec}} < 1$, the gradient will be vanishing, since it approaches 0. In contrast, If $\mathbf{w_{rec}} > 1$, then the weight will tend to be very large, i.e. exploding.

## 2.4 Long Short-Term Memory (LSTM)

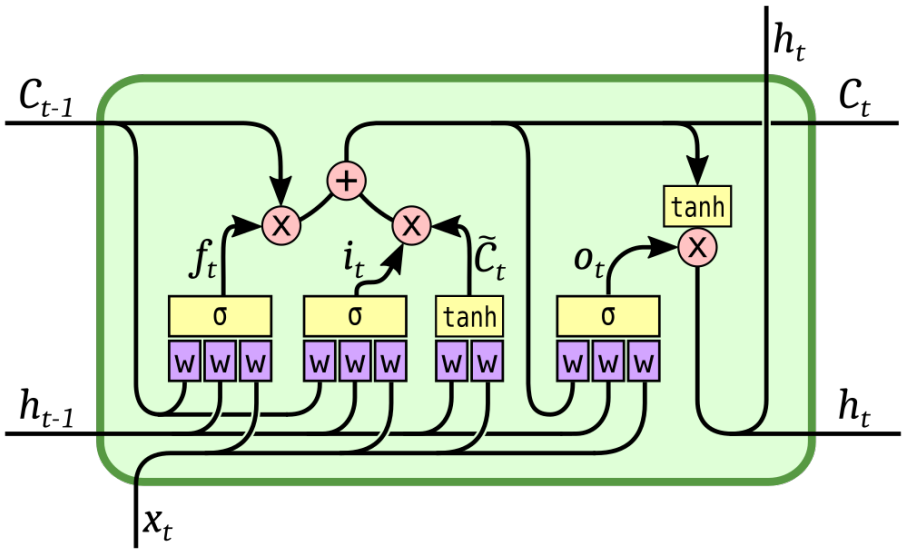



FIGURE 2.4: LSTM with peephole connections (Adapted from [5])

To address the vanishing/exploding gradient problem, multiple variations of RNN have been proposed. Long Short-Term Memory (LSTM) networks are one of the popular architectures [9]. The structure of LSTM is shown in Figure 2.4.

The LSTM network introduces three gates to manage the weights of samples in the sequenced data. $f$ is the forget gate, $i$ is input gate, and $o$ is forget gate. $\sigma$ is a signoid function that gives a value between 0 and 1, which can act like a switch. $\mathbf{h}$ and $\mathbf{c}$ stands for hidden state and cell state. Each carries a memory path to pass down information. The equation of each value is given by:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{w}_{xi}\mathbf{x}_t + \mathbf{w}_{hi}\mathbf{h}_{t-1} + b_i) \\
f_t &= \sigma(\mathbf{w}_{xf}\mathbf{x}_t + \mathbf{w}_{hf}\mathbf{h}_{t-1} + b_f) \\
\mathbf{o}_t &= \sigma(\mathbf{w}_{xo}\mathbf{x}_t + \mathbf{w}_{ho}\mathbf{h}_{t-1} + b_o) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{w}_{xc}\mathbf{x}_t + \mathbf{w}_{hc}\mathbf{h}_{t-1} + b_c) \\
\mathbf{c}_t &= f_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{2.10}$$

These gates, or rather sigmoid functions, let LSTM choose to ignore a sample if the current sample has been considered as not important. Otherwise, the LSTM will discard the information before the sample, and only retain the information of current time $t$.

In the first version of LSTM, there is no output gate [8], and the equations looks like this:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{w}_{xi}\mathbf{x}_t + \mathbf{w}_{hi}\mathbf{h}_{t-1} + b_i) \\
f_t &= \sigma(\mathbf{w}_{xf}\mathbf{x}_t + \mathbf{w}_{hf}\mathbf{h}_{t-1} + b_f) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{w}_{xc}\mathbf{x}_t + \mathbf{w}_{hc}\mathbf{h}_{t-1} + b_c) \\
\mathbf{h}_t &= f_t \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{h}}_t
\end{aligned} \tag{2.11}$$

It is clearer that in this set of equations, $\mathbf{i}$ controls the weight of short-term memories, and $f$ controls longer memories. Those two gates are independent. While training, it can find suitable parameters for $\mathbf{i}$ and $f$, which means $\mathbf{i}$ and $f$ will use different $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to give different control strategies.

The addition of the output gate and use of $\mathbf{c}$ is to keep the longer memories more effectively. $\mathbf{c}$ does not exit the output gate, hence it is not affected if the current output gate is approaching 0. Even though the current $f$ is tend to 0, which means $\mathbf{c}_{t-1}$ has been forgotten, $\mathbf{h}_{t-1}$ still contains information about $\mathbf{c}_{t-1}$. This allows the long-term memories to pass down through the time sequence.

# Chapter 3: Data Preparation

## 3.1 Dataset Explained

The dataset was found and downloaded on the internet [11], which is real data gathered in the city of Guiyang. It contains both static and dynamic data throughout the years of 2015-2017, separated into three sets.

The first datasheet contains the information of a total of 132 links, the structure is shown in Figure 3.1. link_ID is a string of unique IDs for each link; the length and width of the links are double values in meters; link_class is the classification of the road, but since all the links in this data set are in class 1, this column has been eliminated for further processing.

| 1 | link_ID | length | width | link_class |
|---|---|---|---|---|
| 2 | 4377906289869500514 | 57 | 3 | 1 |
| 3 | 4377906284594800514 | 247 | 9 | 1 |
| 4 | 4377906289425800514 | 194 | 3 | 1 |
| 5 | 4377906284525800514 | 839 | 3 | 1 |
| 6 | 4377906284422600514 | 55 | 12 | 1 |

FIGURE 3.1: First few samples of link info

The second datasheet records the upstream and downstream relations of each link, forming the topology of the map. The structure is shown in Figure 3.2.

| 1 | link_ID | in_links | out_links |
|---|---|---|---|
| 2 | 4377906289869500514 | 437790628552580514 | 4377906281969500514 |
| 3 | 4377906284594800514 | 437790628451460514 | 4377906285594800514 |
| 4 | 4377906289425800514 | | 4377906284653600514 |
| 5 | 4377906284525800514 | 437790628123460514 | 4377906280334600514 |
| 6 | 4377906284422600514 | 3377906289434510514#4377906287959500514 | 4377906283422600514 |

FIGURE 3.2: First few samples of link topology

The third datasheet is the records of the travel_time (i.e. the average time of vehicles stays on the link) every two minutes (Figure 3.3). A higher value of travel_time means it takes a longer time for vehicles to pass the link, which implies the road is more congested. Hence, it is the value to predict. The datasheet also includes the record time interval and the date, including year and month, of the record. There are a total of 25,999,606 records, distributed mainly from March to June of 2016 and 2017.

| 1 | link_ID | date | time_interval | travel_time |
|---|---|---|---|---|
| 2 | 9377906285566510514 | 2016-05-21 | [2016-05-21 23:20:00,2016-05-21 23:22:00) | 17.6 |
| 3 | 3377906288228510514 | 2016-05-21 | [2016-05-21 18:46:00,2016-05-21 18:48:00) | 3.5 |
| 4 | 3377906284395510514 | 2016-05-21 | [2016-05-21 07:06:00,2016-05-21 07:08:00) | 10.0 |
| 5 | 4377906284959500514 | 2016-05-21 | [2016-05-21 14:34:00,2016-05-21 14:36:00) | 3.5 |
| 6 | 9377906282776510514 | 2016-05-21 | [2016-05-21 05:04:00,2016-05-21 05:06:00) | 1.5 |
| 7 | 3377906287674510514 | 2016-05-21 | [2016-05-21 16:04:00,2016-05-21 16:06:00) | 34.1 |

FIGURE 3.3: First few samples of records

## 3.2 Data Preparation and Evaluation

Before the training of the model, the data needs to be analyzed first. For the RNN-based model, the data should be sorted in the sequence of time. With this implemented, the feature of time itself will not give much further characteristics, hence digging of the data to get more key features is vital before the training. This would potentially help to reach a better performance.

### 3.2.1 Static Data

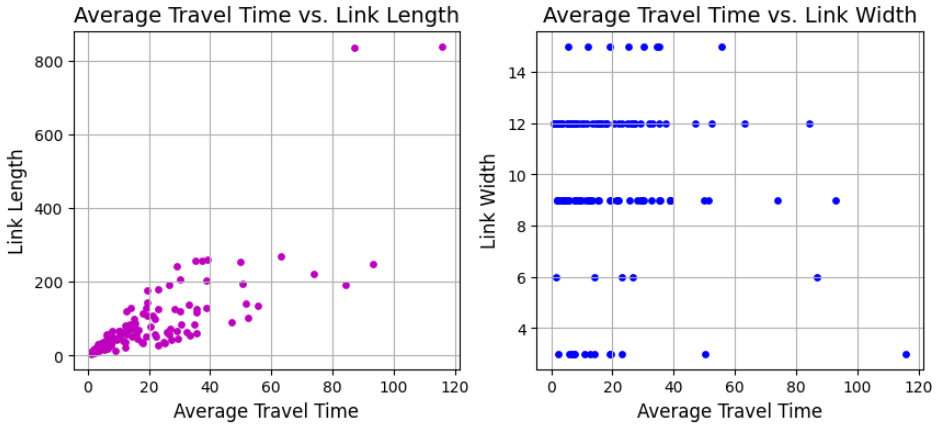



FIGURE 3.4: Length & Width vs. Average Travel Time

The relation of the length and width of the links and the travel time has been studied, and the plot is shown in Figure 3.4. We can see that length has a strong positive correlation with travel time, whereas width seems not relevant at all.

The upstream and downstream of links also are vital for prediction. It is planned to take them into account in the future.

### 3.2.2 Records

The time indication features i.e. years, months, etc. are integer encoded. This might lead to the model incorrectly assuming magnitude relationships between categories. To eliminate this misunderstanding, one-hot encoding is used. The year contains only 2016 and 2017, which can be represented by 0 and 1, using one dimension of a sample. The month includes every month from March to July, a total of 5 months, and five dimensions are occupied.

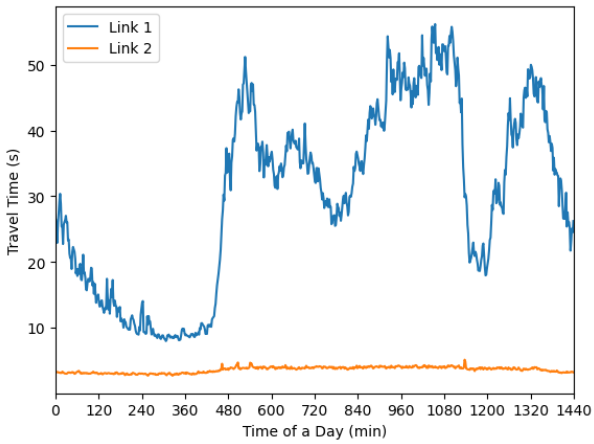


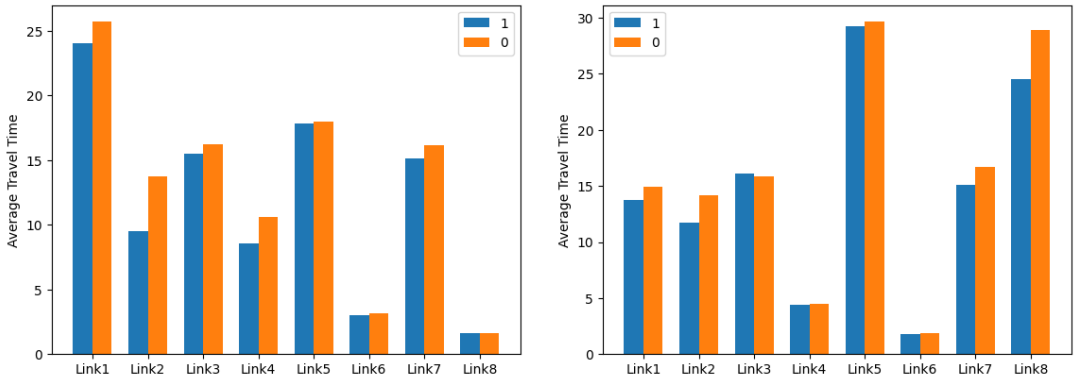

FIGURE 3.5: Time of day vs. average travel time of two randomly choosed links

The time of the day also reveals characteristics for some of the links. Figure 3.5 shows the plot of two randomly chosen links. Link1 varies a lot at different times. In contrast, Link2 gives a much flattened curve. Hour of day is decided to use as a feature, since it is also discrete, one-hot encoding is implemented.

Whether it is a holiday also affects people's behaviour. Hence it would be reasonable to add a feature to tell if it is a holiday. Weekdays are also identified since some of the samples reveal periodicity throughout the week. By plotting them with the travel time on Figure 3.6, it can be seen that the weekdays are generally higher in terms of travel time than holidays and weekends.

With everything above done, the data now have a dimension of 72 for each link. The structure is shown in Table 3.1. There are 132 different links, each link is trained separately.

It has been noticed that most links have missing data at various time segments. Therefore, multiple methods were attempted to fill in the missing values. For the case where the missing values are sparse, i.e. two time segments before and after the sample are




(A) Holiday (1); Working day (0)

(B) Weekend (1); Weekday (0)

FIGURE 3.6: 8 randomly choosed links are used to compare (A) holiday (B) weekend with normal working day

| | length | type | | length | type |
|---|---|---|---|---|---|
| year | 1 | bool | is_holiday | 1 | bool |
| month | 5 | bool | is_weekend | 1 | bool |
| weekday | 7 | bool | width | 1 | int |
| day_of_month | 31 | bool | length | 1 | int |
| hour_of_day | 24 | bool | speed | 1 | float |
| time_of_day | 1 | int | travel_time | 1 | float |

TABLE 3.1: Dimensions of samples for a single link

not missing, the average of the two segments before and after is used to fill in the gap. For large chunks of missing, drop the chunk in the training set, and fill the mean of the data of historical simultaneous moments in the test set. If fill the training set, it may cause large errors.

# Chapter 4: Design & Implementation

To implement the neural network, a deep learning library is needed. Tensorflow with Keras API is chosen. Appendix A shows the code to implement the LSTM model.

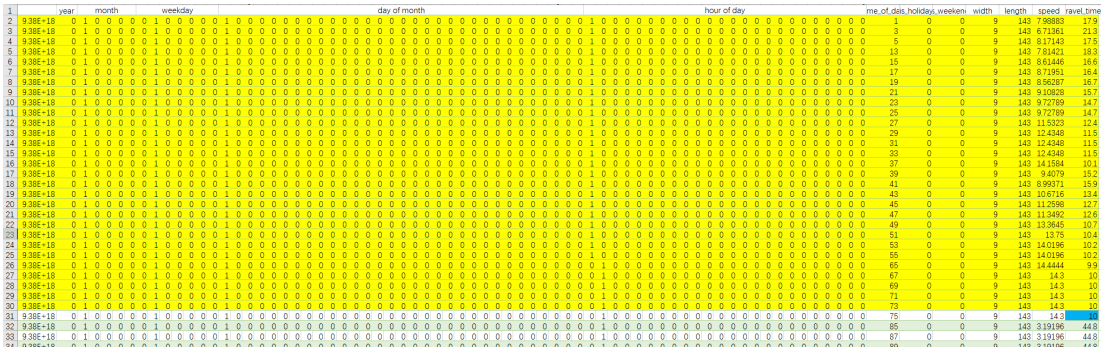


FIGURE 4.1: Yellow represents input structure; blue is the label as target

It is decided that to predict one sample, data from the previous 1 hour is used. For each sample, this gives an input shape of [30, 75], 30 being the previous 30 samples and 75 being the number of features. Output shape is [1] since only the travel time will be predicted. Figure 4.1 represents an example of one pair of inputs.

The data have then been normalized, shuffled, and separated into training and validation sets, and it is ready to be fed into the model. The separation rate has been set to 0.9.

A one-layer LSTM model is constructed, with a dropout layer of 0.2 and a dense layer, shown in Figure 4.2.

After 70 epochs, the model reaches its point of early stopping, with an MSE loss of $6.17 \times 10^{-4}$.

A plot of predicted value vs. target value is shown in Figure 4.4. For this plot, the closer the points to the line $y = x$, the better the prediction is.

```
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 140)               120960

 dropout_3 (Dropout)         (None, 140)               0

 dense_3 (Dense)             (None, 1)                 141

=================================================================
Total params: 121101 (473.05 KB)
Trainable params: 121101 (473.05 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

FIGURE 4.2: A LSTM model constructed using Keras

```
Epoch 1/100
566/566 [==============================] - 8s 13ms/step - loss: 0.0035
Epoch 2/100
566/566 [==============================] - 7s 13ms/step - loss: 0.0016
Epoch 3/100
566/566 [==============================] - 8s 13ms/step - loss: 0.0014
Epoch 4/100
566/566 [==============================] - 7s 13ms/step - loss: 0.0012
Epoch 5/100
566/566 [==============================] - 7s 13ms/step - loss: 0.0011
Epoch 6/100
566/566 [==============================] - 7s 13ms/step - loss: 0.0011
Epoch 7/100
566/566 [==============================] - 7s 13ms/step - loss: 0.0010
Epoch 8/100
566/566 [==============================] - 7s 13ms/step - loss: 0.0010
Epoch 9/100
566/566 [==============================] - 7s 13ms/step - loss: 0.0010
Epoch 10/100
566/566 [==============================] - 7s 13ms/step - loss: 9.9365e-04
Epoch 11/100
566/566 [==============================] - 7s 13ms/step - loss: 9.8852e-04
Epoch 12/100
566/566 [==============================] - 7s 13ms/step - loss: 9.7626e-04
Epoch 13/100
...
Epoch 73/100
564/566 [============================>.] - ETA: 0s - loss: 6.1713e-04Restoring model weights from the end of the best epoch: 70.
566/566 [==============================] - 7s 13ms/step - loss: 6.1686e-04
Epoch 73: early stopping
```

FIGURE 4.3: Result

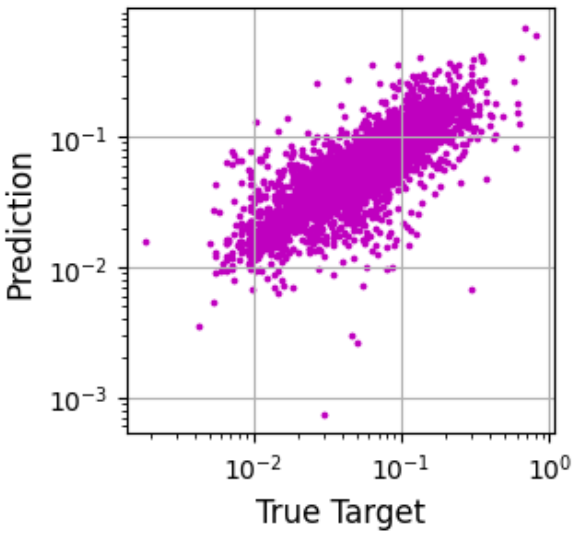



FIGURE 4.4: Predicted vs. target value

# Chapter 5: Conclusions & Proposal

## 5.1 Conclusions

A Gantt Chart of all the work done is shown in Appendix A, Figure B.1. Up to this point, the background of the LSTM model has been well-researched. The dataset has been found, and the data are well understood. Several method to preprocess the data in temporal aspects has been go through, to reconstruct a better dataset fed into the model. A simple LSTM model has been built, given a basic working skeleton of this project.

## 5.2 Future Work

The chart also states the work not completed and planned to be done in the future.

The deep learning library still needs to be more familiarized. It is the priority to have a better structure for the network by altering the number of LSTM layers, batch sizes, and so on. These hyperparameters could be optimised using Bayesian optimisation.

The information on upstream and downstream has not been used for now, consider using spatio-temporal graph convolutional networks (STGCN) method [12][13] to extract traffic's spatial characteristics as well, to potentially reach better performance. Hence, further background research on Bayesian optimisation and STGCN is required.

The LSTM model is sensitive to missing values, consider the LSTM-M model [10] to reduce its impact.

A navigation algorithm based on the A* method will be implemented to show how the model can benefit daily lives.

# References


[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. springer, 2006.

[2] A. Geron. Hands-on machine learning with scikit-learn, keras and tensorflow, 2019. URL https://www.vlebooks.com/Product/Index/1609613.

[3] R.J. Hyndman. *Forecasting: principles and practice, 2nd edition*. OTexts: Melbourne, Australia, 2012.

[4] Ozan Kocadağlı and Barış Aşıkgil. Nonlinear time series forecasting with bayesian neural networks. *Expert Systems with Applications*, 41:6596–6610, 11 2014. doi: 10.1016/j.eswa.2014.04.035.

[5] Jakub Kvita. Visualizations of rnn units diagrams of rnn unrolling, lstm and gru., 2016. URL https://kvitajakub.github.io/2016/04/14/rnn-diagrams/.

[6] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

[7] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press Cambridge, 2012.

[8] Antônio H. Ribeiro, Koen Tiels, Luis A. Aguirre, and Thomas B. Schön. Beyond exploding and vanishing gradients: analysing rnn training using attractors and smoothness, 2020.

[9] Hochreiter Sepp and Schmidhuber Jürgen. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[10] Yan Tian, Kaili Zhang, Jianyuan Li, Xianxuan Lin, and Bailin Yang. Lstm-based traffic flow prediction with missing data. *Neurocomputing*, 318:297–305, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.08.067. URL https://www.sciencedirect.com/science/article/pii/S0925231218310294.

[11] Tianchi. Traffic prediction dataset, 2018. URL https://tianchi.aliyun.com/dataset/dataDetail?dataId=1079.

[12] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. *CoRR*, abs/1709.04875, 2017. URL http://arxiv.org/abs/1709.04875.

[13] Zou Zhene, Peng Hao, Liu Lin, Xiong Guixi, Bowen Du, Md Zakirul Alam Bhuiyan, Yuntao Long, and Da Li. Deep convolutional mesh rnn for urban traffic passenger flows prediction. In *2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1305–1310, 2018. doi: 10.1109/SmartWorld.2018.00227.

# Appendix A: Listings

```
def normalize(vector):
    scaler = preprocessing.MinMaxScaler()
    return scaler.fit_transform(vector)

def sort_samples_by_link(data):
    sorted_data = {}
    for record in data:
        if record[0] in sorted_data.keys():
            sorted_data[record[0]].append(record[1:])
        else: sorted_data[record[0]] = [record[1:]]
    return sorted_data

def create_inout_sequences(data, shape):
    x = np.zeros((data.shape[0]-shape[0], shape[0], shape[1]))
    y = np.zeros((data.shape[0]-shape[0], 1))
    for i in range(data.shape[0]-shape[0]):
        train_seq = data[i:i+shape[0]]
        train_lbl = data[i+shape[0]:i+shape[0]+1][0][-1]
        x[i] = train_seq
        y[i] = train_lbl
    return x, y

ExpandedDataLoaded = sort_samples_by_link(load_expanded_data())
useData = np.array(ExpandedDataLoaded['4377906280329500514']) # Use one link

useData[:, 68] = normalize(useData[:, 68].reshape(-1, 1)).reshape(1, -1)
useData[:, 71:] = normalize(useData[:, 71:])

x, y = create_inout_sequences(useData, [30, 75])
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,
                                                    shuffle=True)
model = Sequential()
earlystop = EarlyStopping(monitor='loss', min_delta=0,
                          patience=3, verbose=1, restore_best_weights=True)
model.add(LSTM(units=140, return_sequences=False,
               input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(units=10, activation='sigmoid'))
model.compile(optimizer='adam', loss=tf.keras.losses.MeanSquaredError())
model.summary()

model.fit(X_train, y_train, epochs=100, batch_size=100, callbacks=earlystop)
```

LISTING A.1: Code of the LSTM model's skeleton

# Appendix B: Charts

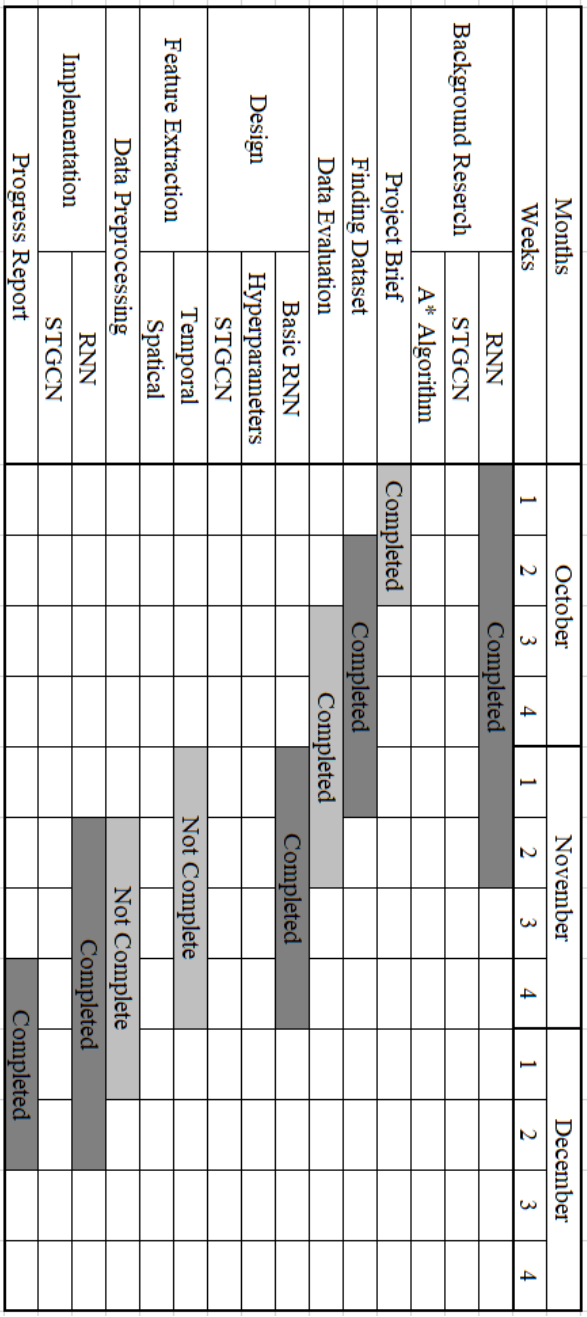



FIGURE B.1: Gantt Chart