Deep learning approaches for traffic prediction

Shao, Hongxin

2020

Shao, H. (2020). Deep learning approaches for traffic prediction. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/142029>

<https://doi.org/10.32657/10356/142029>

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

Downloaded on 26 Apr 2023 21:24:55 SGT

DEEP LEARNING APPROACHES FOR TRAFFIC PREDICTION

HONGXIN SHAO

INTERDISCIPLINARY GRADUATE SCHOOL

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

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

16/08/2019

.....
Date

.....
Hongxin Shao

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.

16/08/2019

.....
Date

.....
Soong Boon Hee

Authorship Attribution Statement

This thesis contains material from 3 papers published/submitted in the following peer-reviewed journals where I was the first author.

Chapter 3 is published as Hongxin Shao and Boon-Hee Soong. Traffic flow prediction with long short- term memory networks (LSTMs). In 2016 IEEE Region 10 Conference (TEN- CON), pages 2986–2989. IEEE, 2016.

The contributions of the co-authors are as follows:

- Prof Soong provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Prof Soong.
- I co-designed the study with Prof Soong and performed all the experiment work. I also analyzed the data.
- All experiment, including data collection and code writing was done by me.

Chapter 4 is submitted as Hongxin Shao and Boon-Hee Soong, “Exploring fully attention model for short-term traffic forecasting,” Journal of Sensor and Actuator Networks 2019.

The contributions of the co-authors are as follows:

- Prof Soong provided the initial idea of the work and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Prof Soong.
- I designed the experiment details and conducted the experiment.
- I analyzed the experiment results with the help of Prof Soong.

Chapter 5 is submitted as Hongxin Shao and Boon-Hee Soong, “Graph Diffusion Recurrent Neural Net- work for traffic forecasting,” Journal of Sensor and Actuator Networks 2019.

The contributions of the co-authors are as follows:

- Prof Soong and I proposed the idea and methodology
- I involved in the programming experiments and investigation. All the authors participated in the discussion of experimental results.
- The entire research project was conceived and supervised by Prof Soong.
- I wrote the original draft and the review & editing was with the help of Prof Soong.

16/08/2019

.....
Date

.....
Hongxin Shao

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my PhD supervisor, Prof. Soong Boon Hee. I have known him from my final year project during bachelor study. Along this long journey, he always provided me with enough patience, guidance and advice. I am grateful that he offered me an opportunity to pursue my passion in the research area and has given me enough academic freedom to explore on what interests me. I also appreciate his support and understanding when I decided to move to industry and converted to a part-time PhD student. I would also thank to my co-supervisor Prof. Wen Yonggang, mentor Prof. Soh Yeng Chai and former mentor Prof. King Jet for their advices.

I am grateful to Dr. Chan Yiu Wing Edwin, Dr. Sum Yee Loon and Dr. Huang Jiajia for sharing your experiences as senior PhD students. Their suggestions in both research and life have been constructive and invaluable.

I would like to thank Dr. Zhao Rui and Liu Yunxiang, with whom I started to know about artificial intelligence and deep learning. The night discussions and gaming time in the hostels have been memorable.

Meanwhile, I would like to thank Dr. Hu Hao, Dr Sun Meng, Dr. Li Beibei, Dr. Wang Wei, Dr. Gao Yumeng, Wang Guomin for being amazing friends in Infinitus Lab. Life is much easier and full of fun with them.

Also thanks to my excellent colleagues at Shopee, Dr. Wang Dongzhe, Dr. Chen Fengmin, Weng Quanchi, Gao Sa, Kou Wen, Raymond Hendy Susanto. I have learnt a lot from them and thanks for the encouragement and care during this thesis writing.

Last but not least, I would like to thank my parents. They nurtured me in a flexible environment and have always standed behind me for my decisions. Without their love and support, I could not finish this long journey.

WSN		Wireless Sensor Networks
ITS		Intelligent Transportation Systems
DL		Deep Learning
SAEs		Stacked Autoencoders
DBNs		Deep Belief Networks
MTL		Multitask Learning
RBN		Restricted Boltzmann Machine
NLP		Natural Language Processing
NMT		Neural Machine Translation
RNN		Recurrent Neural Network
GRU		Gated Recurrent Unit
ARIMA		AutoRegressive Integrated Moving Average
SVM		Support Vector Machine
SVR		Support Vector Regression
MAE		Mean Absolute Error
MAPE		Mean Absolute Percentage Error
RMSE		Root Mean Square Error
RW		Random Walk
WNN		Wavelet Neural Network
CNN		Convolutional Neural Network
LSTMs		Long-short Term Memory Networks
MLP		Multilayer Perceptron
BP		Backpropagation
ReLU		Rectified Linear Unit

Abstract

The rapid and continuous population growth and urbanization movements have resulted in the increase of number of vehicles on the road. Some detrimental effects on people's life are observed such as lost of productivity, air pollution and higher fuel consumption. There is a crucial need for intelligent transportation systems (ITS).

A well designed ITS can support many functions such as route planning and traffic management. They both rely on an accurate traffic prediction. In this thesis, we focus on the data-driven models for traffic prediction task. There are many existing works in this field, however they either require strong assumptions or are shallow in the model structure. Therefore, we explore the problem with deep learning approaches which demonstrate remarkable capability in extracting deeper representation in the data. Since traffic prediction depends on the historical values and is essentially a time series forecasting problem, we firstly investigate the temporal correlations among the data with LSTM networks on a one-step traffic flow prediction task.

Next we extend the task to multi-step traffic speed prediction task. However there are some potential issues in naive LSTM approach. It has difficulties for longer step decoding by relying on a compressed fixed-length vector solely. Also, the recurrent connection makes the computation very costly. Therefore we proposed a full attention model which discards the recurrent connection and uses attention to assist the fixed-length vector during decoding.

On the other hand, road conditions are always affected by their neighboring roads. Therefore we investigate the spatial dependencies among the neighboring nodes and proposed a graph diffusion recurrent neural network to incorporate both temporal and spatial features.

All three proposed models are verified in real world traffic dataset. The LSTM model is tested on the PeMS dataset which we crawled the data from a website

of US government project. We conducted experiments for the other two models on METR-LA dataset, which becomes popular recently and is used as a standard dataset for traffic prediction task. All approaches have demonstrated better performance in terms of MAE, RMSE and MAPE than strong baseline models.

Contents

Acknowledgements	i
Acronym List	iii
Abstract	iv
List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Background	1
1.2 Objectives	5
1.3 Main contributions	5
1.4 Thesis Outline	6
2 Literature Review	9
2.1 Parametric Methods	9
2.2 Non-parametric Methods	10
2.2.1 Machine Learning Methods	11
2.2.2 Artificial Neural Networks	12
2.3 Deep Learning models	14
2.4 Attention Mechanism	16
2.5 Spatial Dependencies	19
2.5.1 Convolutional Neural Networks	20
2.5.2 Graph Neural Networks	23
2.6 Summary	25
3 Traffic flow prediction with Long Short-Term Memory Networks	27
3.1 Introduction	28
3.1.1 Motivations	28
3.2 System Design	29
3.2.1 Introduction of RNNs	29
3.2.2 Long Short-Term Memory	31

3.2.3	Our proposed model	31
3.3	Experiments	34
3.3.1	Dataset Description	34
3.3.1.1	PeMS data collection	35
3.3.1.2	Data normalization	36
3.3.2	Evaluation metrics	36
3.3.3	Implementation Details	36
3.4	Experimental Results	37
3.4.1	Hyperparameter Analysis	38
3.5	Conclusions	41
4	Full attention model for traffic speed prediction	43
4.1	Introduction	43
4.1.1	Motivations	44
4.2	Methodology	44
4.2.1	Problem Definition	44
4.2.2	Transformer model	45
4.2.3	Proposed Model	45
4.2.4	Attention mechanism	47
4.2.5	Encoder-Decoder structure	50
4.3	Experiments	51
4.3.1	Dataset description	51
4.3.2	Preprocessing	52
4.3.3	Implementation Details	52
4.3.3.1	Optimizer	52
4.3.3.2	Regularization	53
4.3.4	Compared models	54
4.3.4.1	Parameters for comparing models	54
4.3.5	Metrics	54
4.4	Results	55
4.4.1	Metrics comparison	55
4.4.2	Forecast Visualization	57
4.4.2.1	Peak hours	57
4.4.2.2	Speed Fluctuation	57
4.4.3	Complexity analysis	59
4.5	Conclusions	62
5	Graph Diffusion Recurrent Network for traffic speed forecasting	63
5.1	Introduction	64
5.1.1	Motivation	64
5.1.2	Convolutional Neural Networks for traffic prediction	64
5.1.3	Graph Neural Networks	64
5.2	Methodology	65

5.2.1	Problem Definition	65
5.2.2	Notations	66
5.2.3	Proposed model	66
5.2.3.1	Graph Aggregator	67
5.2.3.2	Diffusion Convolution	69
5.2.3.3	Temporal Dynamic	69
5.3	Experiments	72
5.3.1	Preprocessing	72
5.3.2	Implementation details	73
5.3.3	Compared models	74
5.3.4	Metrics	74
5.4	Results and Discussion	78
5.4.1	Performance analysis	78
5.4.2	Complexity analysis	79
5.5	Conclusion	80
6	Conclusion and Future Work	81
6.1	Conclusion	81
6.2	Future works	83
	Author's Publications	85
	Bibliography	87

List of Figures

1.1	CO and CO_2 emission against traffic speed in tunnel [1]	2
1.2	Electronic Road Pricing in Singapore	3
1.3	Key components of Intelligent Transportation System	4
2.1	A simple artificial neural network architecture	12
2.2	Illustration of an autoencoder	15
2.3	Encoder-Decoder structure for NMT	17
2.4	Attention in encoder-decoder model	19
2.5	Attention with departure time feature	20
2.6	CNN architecture	21
2.7	Transform traffic condition to an image [2]	22
2.8	Highway and main road connection in a part of Singapore	23
3.1	Fully-connected neuron networks are insensitive to input sequence	28
3.2	RNN block	29
3.3	Unrolled RNN architecture	30
3.4	Our proposed LSTM model for traffic flow prediction	31
3.5	Structure of a LSTM Cell	33
3.6	Location of sensor with ID 1201100	34
3.7	Traffic prediction generated by our proposed model	39
3.8	RMSE and MAPE vs. number of time steps	40
3.9	RMSE and MAPE vs. number of layers	41
4.1	Overview of the proposed Encoder-Decoder Structure.	46
4.2	Attention mechanism.	49
4.3	Encoder and decoder blocks	50
4.4	Variation of learning rate value against the number of training steps	53
5.1	Our proposed GDRNN model with an encoder-decoder structure	66
5.2	Pairwise sum aggregator	68
5.3	GDRNN block	70

List of Tables

3.1	Performance comparison on PeMS dataset	38
3.2	Performance comparison on METR-LA dataset	38
4.1	Performance Comparison with different models for traffic speed forecasting at horizon 5-min, 10-min and 15-min.	56
4.2	Performance Comparison with different models for traffic speed forecasting at horizon 20-min, 25-min and 30-min.	56
5.1	Performance Comparison with different models for traffic speed forecasting. Our proposed methods consistently beat the models in all forecasting horizons with huge advantage.	71

Chapter 1

Introduction

In this Chapter, the background of current traffic problems is first described with an increasing need for better and more intelligent transportation management system. This motivates us to develop more sophisticated data-driven approaches for traffic prediction task. The research objectives are then presented, followed by the highlight of our key contributions. Last but not least, the thesis outline is illustrated with summary of each chapter.

1.1 Background

Due to the population growth, increase in living standard and the implementation of urbanization policies from the governments, the number of vehicles keeps increasing and this might have a detrimental effect on our daily life if the situation is not properly handled. Some possible consequences include, but are not limited to, loss in productivity (e.g., the traffic congestion will cause more time spent on the road), air pollution (e.g., more greenhouse gas emissions if the trip routing is not well planned), higher fuel consumption and accident risks [3, 4].

As pointed out by the study of [1], emission of pollutant gases is related to the traffic speed of vehicles on the road. Figure 1.1 shows the experiment measurements collected in a tunnel. The study takes into account a large range of vehicles and finds out that, a low traffic speed will cause high emission of pollutant gas while

FIGURE 1.1: CO and CO_2 emission against traffic speed in tunnel [1]

a very high speed also increases the emission slightly. The optimal emission level occurs at a moderate speed level at around 60 - 70 km/h.

In the study of Barth et al. [5], it is shown that transportation contributes to roughly a third of total CO_2 emission in United States. Also, a case study was carried out in Southern California, with the finding that CO_2 emission can be reduced by 20% if proper actions are taken [6]. The actions include: 1) control of traffic congestion, which allows vehicles to travel at a steady and faster speed; 2) management of the traffic speed, which regulates the car velocity on the road, without being too fast or too slow; 3) traffic smoothing, which prevents the vehicles from the stop-and-go scenarios during the traffic congestion.

In order to tackle the aforementioned problems, Intelligent Transportation System (ITS) is introduced with the goal "to improve decision making, often in real time, by transport network controllers and other users, thereby improving the operation of the entire transport system" [7]. By providing innovative and smart services, ITS aims to reduce the CO_2 emission and loss in productivity. The field has been popular over the past two decades and the success can be attributed to the following reasons:

Traffic Infrastructures are set up at regular distances or locations where traffic problem is severe, with the aim of better monitoring the traffic conditions of road segments. Sensors or cameras are attached on infrastructures of main roads. Figure 1.2 shows the Electronic Road Pricing (ERP) system in Singapore. It is used to manage the road congestion problems and installed at central areas, expressways

FIGURE 1.2: Electronic Road Pricing in Singapore

or arterial roads. During the operational hours, drivers who pass by are charged with a rate. This discourages the drivers to travel on ERP-installed road and find alternative routes.

Wireless Sensor Network is a network which collects the physical environmental condition from a group of spatially dispersed sensors and cooperatively passes the information to a central place. In ITS applications, Wireless Sensor Networks (WSNs) make use of numerous sensors to monitor and record the traffic conditions in the ambient environment. Some useful traffic data includes traffic speed, traffic flow and vehicle images.

Decision Making is the most important part and plays the central role in an ITS system. A reliable and sophisticated algorithm can ensure the successful delivery of the anticipated function. Data mining and machine learning methods [8, 9] are often applied to extract useful information in the data.

Computation Capability refers to the computational resources occurred in the process. For example, a large and complicated traffic network usually involves a huge number of data and there is a hardware requirement when applying the

FIGURE 1.3: Key components of Intelligent Transportation System

algorithm. The lower cost in random-access memory or central processing unit and popular use of graphics processing unit improve the computation efficiency and make the deployment of more sophisticated algorithm possible.

Figure 1.3 presents a comprehensive picture of how the four factors work together in an ITS system. The real-time traffic condition is collected by sensors installed on the traffic infrastructures. Leveraged on the large-scale sensor networks, data of nearby regions are sent back to central nodes instantly [10, 11]. Sophisticated strategies are then employed to serve the intentions such as traffic management, route planning and public safety.

In this thesis, we work on the decision making part of ITS. More specifically, the focus is on traffic prediction, one of the decision making applications. The aim of traffic forecasting is to mitigate the traffic network problem and alleviate congestion by an accurate prediction on future traffic state. Traffic managers as well as individual users can make use of this predictive information for plentiful purposes such as route planning and travel mode recommendation.

Traffic prediction focuses on predicting values of traffic variables such as traffic flow and traffic speed/time in the future based on the collected data at hand. Therefore the task is defined as a prediction of future traffic information based on the historical and current traffic information. A plethora of works have been proposed in this area and have achieved remarkable progress. However, most of them either requires strong assumptions or are superficial in model architecture, resulting in ineffectiveness of real world applications.

In this thesis, we are interested in traffic flow and traffic speed prediction, in the prediction horizon that ranges from 5 minutes to one hour. We want to develop

more sophisticated data-driven algorithms that are less dependent on external assumptions and are able to extract deeper hidden representations in the data. In particular, we focus on deep learning, an emerging popular area built on top of deep neural network models, for traffic prediction.

1.2 Objectives

The overall aim of this thesis is to develop deep learning approaches which can provide better prediction accuracy in traffic flow and speed. The specific research objectives are listed below:

- Develop predictive models which improves on the current statistical or machine learning models.
- Develop methods which provide both one-step and multi-step traffic prediction ability.
- Develop models which use the location information, with the assumption that neighboring road condition will affect the central road.
- Develop models that can be easily transferred or adjusted to traffic scenarios at another location

1.3 Main contributions

The main contribution of this thesis is development of data-driven models for traffic prediction task. We rely on different deep learning approaches to solve the problem. Deep learning, as an emerging popular technique, makes use of the abundant data sources and available computation resources. This fits well in the traffic scenario. The contributions are summarized below:

- We were among the first few groups to conduct research on traffic prediction with deep learning approaches. With the comprehensive understandings on both fields, we successfully adapted some deep neural network concepts on traffic prediction realm, such as attention and graph neural network.

- The existing works on traffic prediction require strong assumptions or are shallow in model structures. We proposed a Long Short-Term Network (LSTM) encoder-decoder model for one-step traffic prediction which is able to capture deeper hidden representations in the data and reduces the assumptions due to its data-driven nature. This work is published in [12].
- LSTM model has potential problems in multi-step traffic prediction. The encoder compresses the whole input sequence into one fixed-length vector and the decoder relies on this vector output for prediction. It is very hard to produce an accurate prediction at longer prediction horizon. Also, the recurrent connection in LSTM results in a sequential computation which is very expensive. We adapted the attention mechanism from deep learning field and it mitigates the reliance on fixed-length vector and enables parallel computation.
- Neighboring nodes' traffic will have impact on the central node. We proposed a model based on graph neural network and incorporated the traffic network topology in the model. This feature also enables our model to be easily transferable to different locations with replaceable graph structure of the traffic network.

1.4 Thesis Outline

Chapter 1 presents the background of this research and describes its objectives. The main contributions are also listed. In Chapter 2, related data-driven methods for traffic prediction are extensively reviewed. Merits and drawbacks of different models are presented. Some background knowledge of the techniques we used is also included. In Chapter 3, Chapter 4 and Chapter 5, three different deep learning based models are proposed. They focus to tackle different part in the task of traffic prediction. Lastly, a brief conclusion of our works and discussions for the future research directions are stated in Chapter 6. More details of each chapter are presented in the following:

- *Chapter 2: Literature Review:*

This chapter reviews the previous works concerning the traffic prediction task over the past few decades. They include parametric methods, non-parametric

methods and deep learning approaches. We introduce a few popular models for each category and use them as our baseline in the later chapters. Different approaches are compared and drawbacks of different models are stated. Techniques such as attention mechanism which have been applied abundantly in deep learning are introduced. Background and fundamental ideas of graph neural networks are presented as well. They work as the basic block of our proposed models.

- *Chapter 3: Traffic flow prediction with Long Short-Term Memory Networks:*

The method introduced in this chapter is our first attempt on the traffic prediction task, with a deep learning approach. We firstly examined two existing literatures on the deep learning approaches which are stacked autoencoders (SAEs) and Deep Belief Networks (DBNs). It is found out that, despite the successful application and claimed improvements on the performance metrics, those models neglected the intrinsic temporal correlations in the data which is quite important for time series task. Therefore, we investigated the use of Long Short-Term Memory Networks (LSTMs). This structure is designed for sequential data and have made remarkable improvements on areas of natural language processing (NLP). The experiments are conducted on a real-world traffic flow dataset PeMS and are designed to predict the future one-step traffic flow value with given values of historical traffic flow. Compared to a few baselines and existing deep learning approaches, the employment of LSTMs in our model has improved the performance significantly. Effects of different hyperparameters of model architecture are also analyzed. This part of work is published in paper “Traffic flow prediction with long short-term memory networks (LSTMs)” [12].

- *Chapter 4: Fully attention model for traffic speed prediction:*

In Chapter 3, LSTMs based model has been experimentally verified to be successful in traffic prediction task. However, there are some limitations on the model. Firstly, the model structure is based on recurrent connection, in which the next step computation has to wait for the previous computation. This impedes the parallel computing possibility and affects the efficiency during model training. Secondly, during the inference stage, the prediction relies only on the previous predicted output, this will probably introduce an error accumulation problem for a long sequence and does not make use of the

historical information sufficiently. To tackle these two potential limitations, we proposed a fully attention model for the task. The attention mechanism solves the problem by referring to every single step when the model is making prediction. A real-world traffic speed dataset METR-LA is used for the experiments. Comparison studies were carried out against LSTM model with attention mechanism, as well as other baselines. The result shows that our proposed model captures the change of trend well and outperform the other models significantly in short-term traffic speed prediction. This part of work is submitted as a paper “Exploring fully attention model for short-term traffic forecasting”.

- *Chapter 5: Graph Diffusion Recurrent Network for traffic speed forecasting:*

In this chapter, our proposed model is meant to deal with the spatial dependencies in the traffic data. Since traffic network is usually comprised of interconnected road segments, the traffic condition of nearby neighbors will have a significant impact on the central node. A few research works have applied convolutional neural networks (CNNs) to capture the spatial correlations. The traffic data are usually denoted into a two-dimensional matrix format and passed through a CNN model. However, there is a limitation on the CNN structure. It is designed for images which have grid Euclidean data structure. For traffic networks, it maintains a graph structure and it is hard to convert it to a grid structure where CNNs can directly be applied. Therefore, we explored a new areas of research, graph neural networks (GNNs). GNNs accept the input in a graph representation. It propagates and updates the content via a message-aggregation mechanism. Three variants of message-aggregation mechanisms are explored and all of them achieved remarkable improvements. This part of work is submitted in the paper “Graph Diffusion Recurrent Neural Network for traffic forecasting”.

Chapter 2

Literature Review

In this chapter, extensive reviews of the research works on traffic forecasting are presented. Firstly, the parametric methods, which are basically classical statistical models, are presented in 2.1. After that, non-parametric methods, which are mainly machine learning models, are discussed in 2.2. In recent years, deep learning models have performed well for sequential data, especially in the area of Natural Language Processing. Some of the models are examined in 2.3 and they have provided some insight and inspiration for the work in this thesis.

2.1 Parametric Methods

Parametric methods make the mathematical or statistical assumptions and aim to fit traffic data into the model. One of the most popular parametric methods is Auto-Regressive Integrated Moving Average (ARIMA) and it has 3 parameters [13, 14]:

- Auto Regressive (p): is the number of lagged observations used in the model and is represented by

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} \quad (2.1)$$

- Integrated (d): the degree of differencing by subtracting the current observation from previous observation d times, when the stationary condition is not met
- Moving Average (q): is the size of the window of errors

$$\hat{y}_t = c + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} + e_t \quad (2.2)$$

Levin et al. [15] examined ARIMA model on both traffic volume and occupancy at freeways and found out that the ARIMA with parameters of (0,1,1) was the best for different time intervals. Hamed et al. proposed a similar method on urban arterials for traffic volume forecast [16]. Later on, with the aim of improving the prediction accuracy, many research works were conducted with variants of ARIMA under similar traffic conditions, such as Seasonal ARIMA [17], ARIMA with historical data and spatial feature (STARIMA) [18], ARIMA with nonlinear GARCH (ARIMA-GARCH) [19] and ARIMAX with upstream sensor data [20].

Besides ARIMA variants, there are also other time-series models which are actively employed for traffic prediction. Okutani et al. proposed the Kalman filter approach for dynamic traffic volume prediction [21]. Later on a few variants were proposed such as adaptive Kalman filter [22] and Kalman filter with discrete wavelet decomposition [23]. However most of these parametric methods make strong assumptions for the data distribution and their structure is relatively simple.

2.2 Non-parametric Methods

Compared to parametric approaches, non-parametric models are more flexible since they have more freedom in choosing the structure and parameters.

They were proven to be strong in capturing the non-deterministic and complex non-linearity in traffic data. Generally non-parametric methods can be categorized into two classes: machine learning models and artificial neural networks.

2.2.1 Machine Learning Methods

A few machine learning methods were proposed in the traffic prediction task. One of the most popular method is support vector regression (SVR). It is a variant of support vector machine (SVM) algorithm which is well-know for its effectiveness and efficiency. By mapping a low-dimensional nonlinear data into a high dimension space, SVR then performs a linear regression to fit the data distribution. The general formula for SVR is shown as [24]:

$$y = w^T \phi(x) + b \quad (2.3)$$

, where $\phi(x)$ denotes a nonlinear transformation from the original input space x . w and b are the coefficients which are determined by minimizing the following loss function [25, 26]:

$$Loss = \frac{1}{2} ||w||^2 + C \frac{1}{N} \sum_{i=1}^N L_{\epsilon}(y_i, \hat{y}_i) \quad (2.4)$$

, where N is the number of samples, y_i is the actual value and $\hat{y}_i$ is the predicted value. C and ϵ are user-defined parameters. L_{ϵ} is defined as:

$$L_{\epsilon} = \begin{cases} 0, & \text{if } |y_i - \hat{y}_i| \leq \epsilon \\ |y_i - \hat{y}_i|, & \text{otherwise} \end{cases} \quad (2.5)$$

A few research works have modified SVR and applied the variants to traffic prediction. Lippi et al. [27] introduced two new support vector regression models and compared with the SARIMA model. Castro-Neto et al. [28] presented an online support vector machine for regression (OL-SVR) for short-term traffic flow prediction under both typical and atypical conditions. Other works include online learning weighted SVR [29], principal component analysis SVR [30] and hybrid Particle swarm optimization SVR [31].

Apart from SVR, several other machine learning algorithms are also explored in the traffic prediction task. Some works applied Gaussian mixture model (GMM) to the task and the model is also able to handle the incomplete data scenario [32–34]. Chen et al. [35] investigated the application of self-organizing maps (SOM)

in the task. Davis [36] and Chang et al. [37] explored k-nearest neighbors (KNN) algorithm in traffic forecasting and both achieved better results than the statistical model.

2.2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are a class of computing systems that have been extensively applied in numerous fields, including traffic prediction. It is inspired by the biological neural connections in human brain.

FIGURE 2.1: A simple artificial neural network architecture

A simple ANN architecture follows the diagram in Figure 2.1. There are three key components:

- **Input Layer:** raw features are fed into the input layer with numerical values. Each neuron in the input layer corresponds to one dimension of input feature.
- **Hidden Layer:** each hidden neuron is fully connected to all neurons from input layer and output layer. The hidden layers can be stacked a few times to get a deeper architecture.
- **Output Layer:** Similar as the input layer, the number of neurons in output layer corresponds to the feature dimension of expected output

For regression task, ANN seeks to find a mapping between input $\mathbf{x}$ and output $\mathbf{y}$. In the case of traffic prediction, x is usually historical data and y is the prediction at next time step. The transformation from input $\mathbf{x}$ to output $\mathbf{y}$ can be denoted by the equations [38]:

$$\mathbf{h} = \mathbf{W}_{xh}^T \mathbf{x}, \quad (2.6)$$

$$\mathbf{y} = \mathbf{W}_{hy}^T \mathbf{h}, \quad (2.7)$$

where $\mathbf{W}_{xh} \in \mathbb{R}^{f \times d}$ is the weights of the hidden layer, f is the dimension of input $\mathbf{x}$ and d is the dimension of hidden layer output $\mathbf{h}$. $\mathbf{W}_{hy} \in \mathbb{R}^{d \times o}$ is the weights of the output layer and o is the dimension of the output. The linear transformations are usually followed by a nonlinear activation function such as sigmoid, tanh or ReLU. According to the universal approximation theorem [39], a simple neural network can represent a plethora of functions. This makes neural network based models suitable choices for representing highly nonlinear traffic data.

Vlahogianni et al. [40] evaluated the neural network model on both univariate and multivariate traffic prediction, and found out that a simple one hidden layer could produce satisfactory results compared to ARIMA and state-space models. In another article [41], an ANN training method with hybrid exponential smoothing and Levenberg–Marquardt was proposed and further improves the generalization capability for traffic prediction.

2.3 Deep Learning models

Recently with the advancement in data collection, storage and processing technique, learning representation with a deeper ANN architecture becomes possible. This new realm is called deep learning (DL) and has drawn a lot of attentions from multiple research fields. With deeper architectures, DL models are able to capture a more complicated correlation among the data.

In [42], a Stacked Autoencoders (SAEs) is proposed as one of the first few deep learning models for traffic flow prediction. As the name suggests, SAEs stacks multiple layers of autoencoders. An autoencoder has an input layer, a hidden layer and an output layer. Different from ANN which takes supervised learning, an autoencoder undergoes unsupervised learning. During the training, an autoencoder tries to reconstruct the input. Given a set of training samples, $\{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^N\}$, where $\mathbf{x}^i \in \mathbb{R}^d$ is the input, an autoencoder first encodes $\mathbf{x}$ into a hidden representation $\mathbf{h}$, and then $\mathbf{h}$ is decoded to an output $\mathbf{y}$ [43]:

$$\mathbf{h} = f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \quad (2.8)$$

$$\mathbf{y} = g(\mathbf{W}_2 \mathbf{h} + \mathbf{b}_2) \quad (2.9)$$

, where $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices, $\mathbf{b}_1$ and $\mathbf{b}_2$ are bias vectors, f and g are activation functions. The network tries to minimize the reconstruction error $L(\mathbf{X}, \mathbf{Y})$ and the model parameters θ can be obtained as [43]:

$$\theta = \underset{\theta}{\operatorname{argmin}} L(\mathbf{X}, \mathbf{Z}) = \underset{\theta}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^N \|\mathbf{x}^i - \mathbf{y}^i\|^2 \quad (2.10)$$

The stacking of multi-layer autoencoders can be visualized in Figure 2.2. Encoder and decoder share a symmetrical structure.

FIGURE 2.2: Illustration of an autoencoder

Due to the large number of layers and parameters in SAEs, it is difficult to train directly with back-propagation. A greedy layerwise pre-training algorithm is proposed [44, 45]. The training procedures are described below:

- 1) Isolate the first autoencoder layer and train it by minimizing the reconstruction error, with training set as the input.
- 2) Freeze the weights of first autoencoder layer and use the output as input for second layer. Isolate and train the second layer.
- 3) Iterate 2) for the rest of the layers
- 4) Use the output of the last autoencoder layer as the final hidden representation.

After the pre-training, SAEs will output a hidden representation. This hidden representation will be passed to a one-layer neural network for traffic prediction task.

Around the same time, another unsupervised deep learning architecture Deep Belief Networks (DBNs) was proposed for traffic prediction [46, 47]. Similar to SAEs, DBNs are stacked version of another neural network module, Restricted Boltzmann Machines (RBMs). It also follows the greedy layerwise pre-training algorithm. Both methods share the same idea of using deep unsupervised learning to learn an informative hidden representation, and use that representation for regression layer input. The main difference is on the regression layer, where the authors proposed a novel Multitask Learning (MTL) architecture. A prediction for one road is considered as one task. By sharing the parameters among different tasks, i.e., different roads, the model obtained a better prediction capability. Based on this work, Koesdwiady et al. [48] proposed a DBNs with additional weather information and further improved the model performance.

2.4 Attention Mechanism

The idea of attention comes from the human visual behavior. When a person is looking at something, he is focusing on a particular part instead of everything his vision receives. This idea is then applied in neural machine translation (NMT), a sub-area of natural language processing (NLP).

NMT uses neural networks to translate sentence of one language to sentence of another language. A well-known structure in NMT is called encoder-decoder [49, 50] and is depicted in Figure 2.3. The English sentence "Economic growth has slowed down in recent years." passes into an encoder and all information is encoded to a vector. The vector then goes through a decoder and translates the sentence in French "La croissance économique a ralenti ces dernières années.". A common structure for the encoder and decoder is recurrent neural networks (RNNs) which has been proven to handle the sequential data well.

However, there is a critical problem with the encoder-decoder structure. The whole input sentence is compressed into a hidden vector. Also, the decoder relies solely on the hidden vector to decode all the information. Therefore, it poses a strong challenge to the information reliability on the hidden vector. Attention mechanism is invented to solve the problem in sequential modeling.

FIGURE 2.3: Encoder-Decoder structure for NMT

Given a query vector and a sequence of vectors, attention mechanism finds the relatedness of a query vector to every vector in a vector sequence and assign weights to them.

Let the input to the encoder be a sequence of sentence represented by $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$, where T is the length of the source sentence, x_i is the vector representing token i . On the other hand, $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_{T'})$ is the translated sequence where T' is the length of the target sequence.

As described above, the encoder will transform the input sequence into a vector c . If RNN is used, then the hidden state is calculated as [51]:

$$h_t = f(x_t, h_{t-1}) \quad (2.11)$$

and

$$c = q(h_1, \dots, h_T), \quad (2.12)$$

where h_i is a hidden state at time i and c is the generated vector at the end of the encoder sequence. c is then fed into the decoder. Therefore, the only input to the decoder is the final vector c .

Attention solves the problem by adding a context vector c_i at each decoding step. c_i is the weighted sum of the encoding output h_j , where $1 \leq j \leq T$. By doing so, information at every step of the encoder is considered instead of considering only the last compressed vector c [52].

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (2.13)$$

The weight α_{ij} is computed as [52]:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \quad (2.14)$$

where

$$e_{ij} = a(s_{i-1}, h_j) \quad (2.15)$$

is an alignment model. The alignment takes input the previous hidden state in the decoder as well as the hidden state at each encoder step. A few variants of attentions have been explored and applied in different areas. Normally, they can be categorized into the following three [53]:

- **dot** The alignment function is the dot production between the decoder state and encoder output.

$$e_{ij} = s_{i-1}^T h_j \quad (2.16)$$

- **general** The alignment takes a bi-linear form, where the decoder state is multiplied with a matrix W_a and the result is multiplied by the encoder output.

$$e_{ij} = s_{i-1}^T W_a h_j \quad (2.17)$$

- **concat** The encoder output and decoder hidden state are firstly concatenated. Then it is passed to a nonlinear hidden layer. After that, a linear projection is applied to get scalar value.

$$e_{ij} = v_a^T \tanh(W_a[s_{i-1}; h_j]) \quad (2.18)$$

The new encoder-decoder structure with attention mechanism can be viewed in Figure 2.4.

FIGURE 2.4: Attention in encoder-decoder model

There are several research works which have applied attention mechanism successfully in traffic prediction task. In [54], the author tries to predict the future one-step travel time. As shown in Figure 2.5, v_i is the departure time feature and h_i is the travel time feature after RNN computation at each time step. The concatenation alignment function is applied on $\mathbf{V}$ and $\mathbf{H}$. The attention mechanism allows the model to capture more important part of the input sequence based on the departure time information. Experiments shows an improvement of 3-4% on the prediction errors.

2.5 Spatial Dependencies

The traffic network in big cities is usually complicated with intersections of road segments. As many researchers have pointed out, a traffic pattern in one location of the network is usually affected by its nearby neighbors [55].

FIGURE 2.5: Attention with departure time feature

2.5.1 Convolutional Neural Networks

Recently, a deep learning architecture called Convolutional Neural Networks (CNNs) has drawn attention from the research field and is proven to be effective in extracting neighboring information in images. Though CNN is also one type of the neural networks, there are a few differences to regular neural networks introduced in section 2.2.2. In a regular neural networks architecture as shown in Figure 2.1, all the neurons in the hidden layer are connected to the previous layer. This will results in a large parameters to learn for the network. In the case of image input where features are represented in its pixel values, the input feature size is extremely large and a huge network easily lead to overfitting. Alternatively, CNNs discards the fully-connected connection and focus on a small region from the previous layer. In this way, the number of parameters is greatly reduced. At the same time, it is particularly useful to extract the correlation feature for neighboring pixels.

A simple CNN model architecture is shown in Figure 2.6. The brown block represents an input in three dimensions (width, height, depth). The CNN is the blue block with each neurons representing a CNN filter. The green block is the output after CNN and is usually followed by a fully-connected layer. As shown in the

FIGURE 2.6: CNN architecture

figure, each filter looks at the small rectangular portion of the input. The computation remains the same as regular neural networks but is now restricted to be local spatially. Therefore the output after the filter will contain the spatial dependencies inside this region. CNN contains multiple filters and they can be applied at the same region. Different filters can focus on one particular type of the spatial features. This makes CNN strong in capturing the local dependencies.

There are numerous applications of CNN during the past ten years. Most of them fall in the areas of handwriting recognition [39, 56, 57], image classification [58, 59] and object detection [60–62]. Those works focus mainly on the image task. Indeed, as early as in year 1995, Lecun et al. has pointed out that CNN can be applied in multiple areas such as images, speech and time-series [63]. The article points out that, similar as the strong two-dimensional local structures in images, there is strong one-dimensional local structure in time-series data. It is beneficial to extract the this highly correlated local feature before recognizing the temporal or spatial objects.

Inspired the success of CNNs in Computer Vision (CV), Wu et al. proposed a one-dimensional CNN to capture the spatial features of traffic[64]. The work is to predict the traffic flow of p locations $\{v_i\}_{i=1}^p$ in the next h time steps of $(t, t +$

FIGURE 2.7: Transform traffic condition to an image [2]

$1, \dots, t + h$). The historical data can be represented in a matrix[64]:

$$\mathbf{V} = \begin{bmatrix} v_1(t-n) & v_1(t-n+1) & \cdots & v_1(t-1) \\ v_2(t-n) & v_2(t-n+1) & \cdots & v_2(t-1) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ v_p(t-n) & v_p(t-n+1) & \cdots & v_p(t-1) \end{bmatrix} \quad (2.19)$$

The one-dimensional CNN works on the spatial axis, in this case the vector $T_q = [v_1(t-n+q), v_2(t-n+q), \dots, v_p(t-n+q)]^T$, where $0 \leq q \leq n-1$. The convolutional kernel computation can be written as[64]:

$$h_q^k = o_c(w_q^k * T_q^k + b_q^k), \quad (2.20)$$

where w_q^k is the weights, b_q^k is the bias, o_c is the nonlinear activation function and $*$ is the convolution sign. By incorporating the spatial feature with temporal feature and periodic (weekly/daily) features, the model gives a more accurate traffic flow prediction.

In another work [2], Ma et al. proposed a similar idea by converting the traffic network into a two-dimensional spatial-temporal matrix. The matrix is then used

FIGURE 2.8: Highway and main road connection in a part of Singapore

to generate a traffic image and CNN is applied on that. The full pipeline can be viewed at Figure 2.7.

2.5.2 Graph Neural Networks

Though CNN has shown excellent capability in plenty of research areas and the previous section has reviewed some implementations on traffic data, there are a few issues on applying it directly on the traffic data. Firstly, CNN works well on the Euclidean data such as image (two-dimensional grid) and text (one-dimensional grid). However, the traffic network behaves as a non-grid graph structure and it may be hard to find the Euclidean representation to fit in well. Figure 2.8 shows the traffic network around the middle part of Singapore. The yellow lanes are the highways and main roads which are interconnected irregularly. It makes more sense if the connections are represented by a graph instead of grids. Second, it is hard to find a localized convolutional filter to convert the traffic data into Euclidean domain. Graph neural networks (GNNs) which are designed for graph-structure data fits well in the case.

The very first idea of GNN was proposed in [65], which relates the neural network models to graph-domain data. Generally, every node in a graph can be defined by features of its own and connected neighboring nodes. Like other neural network models, the aim of GNN is to learn a hidden representation h_v that incorporates

the feature of node v and its connected neighbors' information. Every GNN model can be simplified with two basic functions, local transition function f and local output function g [66, 67]:

$$h_v = f(x_v, h_{ne}, x_{ne}) \quad (2.21)$$

$$o_v = g(h_v, x_v), \quad (2.22)$$

where x_v is the feature of node v , x_{ne} and h_{ne} are the features and hidden representations of the neighboring nodes of v .

Putting the above vector terms into a compact matrix form, the above formula can be written as [66, 67]:

$$H = F(H, X), \quad (2.23)$$

$$O = G(H, X) \quad (2.24)$$

Like existing deep learning models, GNN can have a deeper structure by iteratively computing the hidden representation[68]:

$$H^{t+1} = F(H^t, X) \quad (2.25)$$

Next, several types of the local transition function f will be introduced.

- **convolution** The work [69] of Kipf et al. extends the convolutional idea into graph-structured data using spectral approach. The spectral convolution is defined as:

$$g_\theta \star x = U g_\theta U^\top x, \quad (2.26)$$

where $x \in \mathbb{R}^N$ is the signal and g_θ is the filter defined in the Fourier domain. U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^\top$.

Based on this idea, Yu et al. [70] integrates the graph convolution and temporal convolution. Improvements on the experiments results show the potential of this application.

- **Aggregator** A general inductive framework for graph learning is proposed as GraphSAGE [68]. The idea is to aggregate the features from local neighbors.

$$\mathbf{h}_{\mathcal{N}_v}^t = \text{AGGREGATE}_t(\{\mathbf{h}_u^{t-1}, \forall u \in \mathcal{N}_v\}) \quad (2.27)$$

$$\mathbf{h}_v^t = \sigma(\mathbf{W}^t \cdot [\mathbf{h}_v^{t-1} \parallel \mathbf{h}_{\mathcal{N}_v}^t]), \quad (2.28)$$

where $\mathbf{h}_v^t$ is the hidden representation of node v at iteration t , $\parallel$ is the vector concatenation symbol and $\mathcal{N}_v$ is the set of neighboring nodes of node v .

Our implementation of graph diffusion recurrent networks in Chapter 5 are based on the aggregator idea. This framework defines only the general idea and there can be a few choices of aggregator functions. Therefore we explored three variants and compared the effectiveness of each function in the experiments later.

2.6 Summary

In this section, the connection of the above reviewed works and our works in the following sections are presented.

Section 2.1, 2.2 and 2.3 demonstrate a few popular models in traffic prediction task. They will work as the baselines in our experiments. Also, the existing neural network models has a drawback that does not consider the temporal order information. Based on this, we explored the Long Short-Term Memory networks in Chapter 3.

In section 2.4, the attention mechanism is introduced. Along with other works in NLP area, they inspire us to develop a fully attention model for traffic speed prediction which will be elaborated in Chapter 4.

Section 2.5 investigates a few works on capturing the spatial correlation. Works on emerging GNN architecture show significant improvement and flexibility on many areas. They inspire us to explore the spatial dependencies in traffic data with a graph approach. Detail of this part of our work is covered in Chapter 5.

Chapter 3

Traffic flow prediction with Long Short-Term Memory Networks

In this Chapter, we focus on one-step traffic flow prediction task with Long Short-Term Memory Networks (LSTMs), which is one of the deep learning architectures. Compared to existing works which either require strong assumptions or are shallow in model structure, there are some intrinsic benefits of LSTMs. Being deep in structure, the model is able to extract higher level of hidden representation in the data. Also, the temporal dependent structure design makes it especially appropriate in sequential data handling. Experiments were conducted on a few real world traffic datasets. We crawled the data from PeMS, a US government project and did the comparison studies against some strong baseline methods. In order to show the model's robustness to locations, experiment was conducted on another METR-LA. Moreover, the effect of hyperparameters of LSTMs are studied and presented.

The rest of the chapter is organized as follows. Section 3.1 presents the motivation of this work and the overview of the problem. In Section 3.2, details of our model structure are presented. Section 3.3 elaborates on the traffic flow dataset, metrics used and detailed implementation of proposed model. Experimental results and hyperparameter studies are covered in Section 3.4. Finally the conclusion is presented in Section 3.5.

FIGURE 3.1: Fully-connected neuron networks are insensitive to input sequence

3.1 Introduction

3.1.1 Motivations

Given a sequence of historical traffic flow values at a sensor point, the aim of traffic flow prediction is to predict the future traffic flows. In the previous chapter, various models for traffic prediction have been reviewed. There are obvious drawbacks with those models. For the parametric models, there are always some assumptions during the model building, this makes it difficult to adapt to new scenario and the structure is relatively simple. For research works with non-parametric machine learning models or simple artificial neural networks models, it has been experimentally proven to be superior to the parametric models. However, those models usually have a shallow structure and are not able to extract a higher level feature abstraction.

Recently, a few pioneering works on deep learning have brought insights to traffic prediction task, either using DBNs [46, 47] or SAEs [42]. DBNs and SAEs are great feature extractors and have been extensively applied in areas such as speech recognition [71–73] and image analysis [74–76]. However, they are insensitive to the input order, as shown in Figure 3.1. In both left and right of the figure, the yellow layers with neurons represent same fully-connected hidden layers. Inputs to the two neural networks are both x_1, x_2, x_3, x_4 but in different sequential order. However, the two networks will not be able to distinguish the sequential difference and produce the same outputs.

FIGURE 3.2: RNN block

In view of the ineffectiveness of current models, we explored Long Short-Term Memory (LSTM) networks, from the deep learning recurrent neural networks (RNNs) family on traffic flow prediction.

3.2 System Design

3.2.1 Introduction of RNNs

A recurrent neural network is a type of neural network that is specialized for processing a sequence of vectors. The main idea behind RNN is the built-in loop structure. Figure 3.2 represents a RNN block. The input and output at time step t of the RNN are x_t and h_t . At the same time, h_t will be fed back to the RNN block at the next time step with the input x_{t+1} . This loop structure in RNN allows the information to be passed from one step of the network to the next. The chain-like nature reveals that recurrent neural networks are intimately related to sequential

FIGURE 3.3: Unrolled RNN architecture

problem, such as speech recognition [77, 78], language modeling [79–81], and image captioning [82, 83].

Also, the reuse of RNN block allows the sharing of parameters for each value of the time index. Statistical strength across different time index is shared at neighboring time indexes and this could enhance the feature extraction process. All these inherent features have endowed RNN the capability to outperform simple multiple layer perceptron, as well as other deep learning architectures (e.g., SAE and DBN).

However, the traditional RNN faces two problems during training:

- **Vanishing Gradients** Gradients passed back are getting smaller over the time steps and eventually vanish.
- **Exploding Gradients** Gradients passed back are getting larger over the time steps and eventually explode.

An unrolled RNN structure is depicted in Figure 3.3. Assume there are t steps of input $x_1, x_2, \dots, x_t$ to the RNN. The loss at time step t to update the weights are calculated by back-propagation through time (BPTT). Since each input x_i has effect on the last time step, by chain rule of calculus, gradient signal progresses along the hidden layer and gets multiplied by the weight matrix of the neurons over t time steps. If the weight matrix is too small or large, the gradient signal will eventually become either too small that learning stops working, or too large that learning diverges.

FIGURE 3.4: Our proposed LSTM model for traffic flow prediction

3.2.2 Long Short-Term Memory

To address the gradient explosion or vanishing problem, long short-term memory (LSTM) was proposed [84]. The key idea of LSTM is a new structure called memory cell. It contains a neuron with a self-recurrent connection and three gates associated with it. The self-recurrent connection allows the state of memory cell to remain constant from one time step to another without outside interference. Input and output gates can allow the signal to come in or go out of the neuron or prevent it. Forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state.

3.2.3 Our proposed model

Suppose the historical traffic flow sequence collected at one sensor node are denoted as $x = x_1, \dots, x_T$, where x_i is the value of traffic flow at time step i , the task here

is to predict the next time step traffic flow x_{T+1} . An encoder-decoder model is proposed and illustrated in Figure 3.4. The encoder includes all blue LSTM blocks. It reads a sequence of traffic flow values $x = (x_1, x_2, \dots, x_T)$ and produces a fixed length vector c . This fixed length vector then goes to the decoder and generates the target x_{T+1} .

In the encoder, all LSTM blocks are concatenated according to the temporal sequence of the traffic flow. For each LSTM block, input x_t from current time step t and the previous hidden state vector h_{t-1} are fed into the block and produce the current hidden state vector h_t :

$$h_t = f(x_t, h_{t-1}) \quad (3.1)$$

where f is the computation function inside the LSTM block and will be elaborated later. At time step 0, there is no hidden state vector and this initial hidden state is initialized as a zero-value vector:

$$h_0 = 0 \quad (3.2)$$

At the last LSTM block, a fixed length vector c is obtained as

$$c = h_T, \quad (3.3)$$

which is the hidden state vector of the last LSTM block. With all the information flow up to time step t , the fixed length vector c is believed to encode all the temporal dependencies in the sequence and functions as a high-level representation of the input sequence. In the figure, only one layer of LSTM blocks are drawn for simplicity purpose. Normally, multiple layers can be stacked for highly non-linear data and the effect of layers on the model performance is explored in the experiments part later.

The decoder takes the fixed length vector c as the input and passes it to a linear regression layer. Here the linear regression layer is simply a fully-connected neural network layer. The output of linear regression layer is the desired output x_T .

Normally a long sequence of input is fed into the LSTM layer. If there is no control on the information, all information will proceed to the last time step and it will

FIGURE 3.5: Structure of a LSTM Cell

result in a jam with useless information. There are four gates in a LSTM cell to control the information flow. The structure of a LSTM cell is illustrated in Figure 3.5. The forget gate is used to prevent redundant information passing to the next time step:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \quad (3.4)$$

where x_t and h_{t-1} are input and hidden state vectors and W_f is weights matrix of forget gate. A sigmoid activation σ produces a value between 0 and 1. This will be the coefficient to control the forgetness of previous information.

Similarly, an input gate is present to control the information flow:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad (3.5)$$

LSTM also has a cell state C_t and is update by a tanh activation:

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \quad (3.6)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (3.7)$$

FIGURE 3.6: Location of sensor with ID 1201100

Finally, an output gate controls the output value:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (3.8)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (3.9)$$

3.3 Experiments

3.3.1 Dataset Description

We evaluate our model on the dataset crawled from Caltrans Performance Measurement System (PeMS) [85]. PeMS is a project sponsored by the California

Department of Transportation (Caltrans). It offers a website interface where user can search traffic data information (traffic flow, traffic speed, travel time, etc.) from its database. The data are collected from over 39,000 individual detectors along the freeway across all major metropolitan areas of the State of California. The data are then aggregated into a 5-min or 1-hour interval for each vehicle detection station (VDS) and stored in the database.

To evaluate that the proposed model is robust to different locations, it is further experimented on another dataset METR-LA [86], which is released recently and has become popular in traffic prediction area. It is used in many research works as a standard dataset for model comparison. It consists of high-resolution spatial-temporal transportation speed data from a project maintained by the LA county road network by Los Angeles Metropolitan Transportation Authority (LA-Metro). In total, 8,900 traffic loop-detectors were located on the highways and arterial streets to collect traffic speed data at the rate of one reading/sensor/min. The data used here is from 207 sensors during a period of 4 months from Mar 1st 2012 to June 30th 2012.

3.3.1.1 PeMS data collection

We evaluated data collected from 10 sensors which were placed at different locations, in order to prove that our model is robust to different locations and can be easily transferred to other places. Figure 3.6 shows one example of one VDS location. The VDS has ID of 1201100 and locates at lane 2 of I-405 mainline at Irvine Center Drive.

In order to download the data, a python library Requests is used. The POST method is used to request the PeMS website for relevant information. The data collection time range is one month from August 1st to August 31th 2014. Since weekday and weekend traffic present different patterns, here we only use the traffic flow data in weekdays. A simple partition splits the data into train/valid/test ratio of 70%/10%/20%.

A common issue with the collected data by roadside detector is data fidelity. Sometimes the detector is not working in normal condition, and the data has to be imputed by recalculation using neighbor roads information. Fortunately PeMS

provides its own test on the data and shows the percentage of data that is observed or imputed. In the experiment, we only selected the data which has an imputation value of 0%. Therefore the quality issue of data is removed.

3.3.1.2 Data normalization

A min-max normalization is applied to the raw traffic flow data based on the following formula:

$$z = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad (3.10)$$

where x is the raw traffic flow data and z is the value after min-max normalization. x_{min} and x_{max} are the minimum and maximum traffic flow value in the training set. After normalization, the value of z will be in the range of $[0, 1]$.

3.3.2 Evaluation metrics

Evaluation metrics are required to assess the effectiveness of different models. In a prediction task, two metrics are usually used: absolute percentage error (MAPE) and root mean square error (RMSE). Since the task here is one-step traffic flow prediction, the errors are calculated based on the model prediction traffic flow value and the actual next-step traffic flow value. The two metrics are defined as:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \frac{|x_i - \hat{x}_i|}{x_i} \quad (3.11)$$

$$\text{RMSE} = \left[\frac{1}{n} \sum_{i=1}^n (|x_i - \hat{x}_i|)^2 \right]^{\frac{1}{2}}, \quad (3.12)$$

where x_i is the real traffic flow observed by the detector and $\hat{x}_i$ is the predicted traffic flow by our model.

3.3.3 Implementation Details

During the experiments, the models are fine-tuned with optimal parameters using grid-search method. The implementation details are:

- **SVR:** A penalty parameter 2.3 and a 'rbf' kernel is used.
- **WNN:** The input layer size is set as 6. One hidden layer with 30 neurons is used. Learning rate value is 0.01.
- **SAEs:** The input layer size is 5. Three autoencoder layers are stacked with 40 neurones for each layer. Learning rate value is 0.01.
- **Our model:** There is one hidden layer of LSTM with 32 neurons and the input neuron number is 6. Learning rate is set to 0.01.

3.4 Experimental Results

We report the performance of our model in Table 3.1 and Table 3.2 for PeMS and METR-LA dataset respectively. We run our model 5 times independently with random seed and present the averaged results across different runs. The performances of baseline models are also presented and they include:

- **Random Walk:** RW is a simple method used in time series. It predicts the next time step traffic flow based only on the current traffic flow and a random change.
- **Support Vector Regression:** SVR is a machine learning method which has been introduced in Section 2.2.1.
- **Wavelet Neural Network:** WNN is one type of neural networks. However, the sigmoid activation function is replaced with wavelet basis and the output is a linear weighted combination of wavelet basis functions.
- **Stacked Autoencoders:** SAEs is an unsupervised deep learning feature extractor with a linear regression at the top. The model details have been reviewed in Section 2.3.

Our model obtained the best in MAPE and RMSE metrics for both dataset. It shows that the proposed model is robust to changes in locations. RW achieves a relatively low MAPE but a very high RMSE. This is due to the small random change added to the previous traffic flow value, which leads to a smaller relatively

Metrics		
Models	MAPE (%)	RMSE
RW	8.63	70.42
SVR	10.77	51.26
WNN	10.21	64.32
SAEs	7.92	56.31
Our model	6.15	49.23

TABLE 3.1: Performance comparison on PeMS dataset

Metrics		
Models	MAPE (%)	RMSE
RW	11.06	9.65
SVR	9.80	7.73
WNN	12.40	9.78
SAEs	9.49	6.69
Our model	9.01	6.02

TABLE 3.2: Performance comparison on METR-LA dataset

error. The high MAPE and low RMSE indicates big errors with smaller traffic flow. It shows that SVR is not able to linearly separate the data well even with a nonlinear mapping of original data. It is observed that neural network models have advantage in modeling the traffic flow. Besides our proposed model, SAEs also attains promising results, mainly due to its deep structure in effectively mapping non-linear functions of the features. However, the loss of sequential information marks the main huge difference from our proposed model.

Figure 3.7 plots generated traffic flow by our proposed model and the ground-truth values. The x-axis is the time during the day with the unit of hours. Y-axis represents the number of vehicles passed by during the last 15 minutes. In general, our model is able to follow the trend of change in traffic flow. Moreover, the traffic flow during peak hours in the morning (7 a.m.) and afternoon (4 p.m.) is well captured.

3.4.1 Hyperparameter Analysis

Hyperparameters are variables which determine the structure of a model. Since deep learning approaches basically stack layers of neurons, the number of neurons and number of layers are two primary model hyperparameters. Here we examine

FIGURE 3.7: Traffic prediction generated by our proposed model

the effect of different hyperparameter values on MAPE and RMSE, with the grid search method.

- **number of time steps:** In time series forecasting, one key feature is how many historical data is used. The number of time steps also corresponds to the number of LSTM blocks in our proposed model. Therefore, experiments on the effect of number of time steps were carried out and evaluated.

Figure 3.8 illustrates the values of RMSE and MAPE with different choices of time steps. Time step values were taken in the range of $[1, 2, \dots, 10]$ and each time step value k makes use of previous traffic flow values up to k . It is observed that, with time step value at 6, both RMSE and MAPE can achieve the lowest value. Also, similar trends are observed with increasing value of time steps. The metric generally goes down and then goes up after time step 6. This phenomenon could be explained by the model complexity.

FIGURE 3.8: RMSE and MAPE vs. number of time steps

Time steps are the input size to the model and determines number of LSTM blocks in each level. When the input size increases from 1, more previous traffic flow information is used and the model is able to provide a better result. Therefore, the error is decreasing. After time step 6, the model gets more and more complex. Overfitting problem occurs and is reflected in the increase of generalization error.

- number of hidden layers** In deep learning, many desirable results are obtained by stacking several layers of the same network, such as SAEs and DBN. Thus, we explored the effect of stacking multiple layers of LSTMs in our experiment. Figure 3.9 shows that both RMSE and MAPE increase when more LSTM layers are added to our model. One possible explanation might be due to the moderate size of our data set. When the number of hidden layers is increase, the model parameters also increase. With a relatively small training size, overfitting occurs easily. Thus, the optimal layer in our case would be one-layer LSTM.

FIGURE 3.9: RMSE and MAPE vs. number of layers

3.5 Conclusions

In view of the strong assumption required or shallow structure in existing research works, we proposed a LSTM encoder-decoder model to predict the one-step ahead traffic flow in this chapter. This is our first attempt to explore deep learning approaches in traffic prediction domain. The encoder-decoder framework consists LSTM blocks as the encoder and linear regression as the decoder. LSTM is able to exploit the long-term dependency in the traffic flow data sequence. As one of the neural network approaches, LSTM has a strong non-linear mapping power which is able to capture the latent feature representations and non-linearity in the traffic flow data.

We evaluated the performance of our model on two dataset: PeMS and METR-LA. and compared with several baselines (RW, SVR, WNN, and SAEs). Experimental results show that the proposed model is able to lower the MAPE and RMSE by 10% – 40% compared to the baseline models. It is also robust to changes in locations since the proposed model beat the baselines in both dataset. Also, we

experimentally evaluated the effects of two hyperparameters, number of time steps and number of hidden layers.

Chapter 4

Full attention model for traffic speed prediction

Many real world applications require multi-step predictions for accurate and flexible planning. In this chapter, we extend the previous one-step prediction work to multi-step prediction. A full attention model is proposed. It discards the recurrent connections in LSTMs and attains the sequential information by attention mechanism and a simple positional encoding. We run the experiments on a recently popular traffic speed dataset METR-LA. Comparison study is carried out against a few baseline models. Performance metrics and data visualization shows that our model is competent to capture the trends in the traffic speed.

The rest of the chapter is organized as follows. Section 4.1 presents the motivation of this work and the overview of the problem. In Section 4.2, details of our model structure are presented. Section 4.3 describes METR-LA dataset used, detailed implementation and baseline methods compared. Experimental results, visualizations and complexity analysis are reported in Section 4.4. Finally we draw the conclusion in Section 4.5.

4.1 Introduction

In this section, we demonstrate the motivations behind our proposed methodology and the related models inspiring our work.

4.1.1 Motivations

In the previous work described in Chapter 3, we proposed a LSTM based model for one-step ahead traffic prediction. However, the application of one-step traffic prediction is quite limited. A traffic management system or users who want to receive route advice would like to receive more information for decision making and they prefer a longer prediction horizon probably up to one hour.

The previously proposed LSTM encoder-decoder model has a few disadvantages in multi-step traffic prediction task. A potential issue is that the encoder has to compress the whole input sequence into a fixed-length vector. In a one-step prediction task, the decoder comprises only a linear regression layer. However, in the multi-step prediction, prediction at every decoding step relies on the compressed fixed-length vector. This poses the challenge that the model needs to be able to obtain a very informative fixed-length vector. Another possible issue is the computation complexity in LSTM model. It computes the hidden state at each time step by taking into account the current input and previous hidden state. So computation at next step needs to wait for the previous step to finish. This inherent recurrent nature impedes an efficient parallel computation.

With the two concerns in mind, we attempt to develop a model which is able to utilize more informative inputs during decoding and does not rely on the recurrent structure.

4.2 Methodology

In this section, we formally define the multi-step traffic speed prediction problem to be solved. Then we describe our proposed full attention model. The model overview is presented first. Then each part of the model will be elaborated with more details.

4.2.1 Problem Definition

Suppose there are N sensor detectors in a traffic network. Each sensor detector records a one-dimensional traffic speed value at each time step. Putting traffic

speed values of N sensors into a vector form, We can use $\mathbf{x}^{(t)} \in \mathbb{R}^N$ to represent the multivariate traffic signal at time step t . The goal of traffic flow prediction is to predict the future traffic speed condition based on the historical traffic signals. It can be formally defined as:

$$\left[\mathbf{x}^{(t-T'+1)}, \dots, \mathbf{x}^{(t)} \right] \longrightarrow \left[\mathbf{x}^{(t+1)}, \dots, \mathbf{x}^{(t+T)} \right], \quad (4.1)$$

where T' is the number of historical traffic speed values and T is the prediction horizon.

4.2.2 Transformer model

Recently, architectures which replace the recurrent units with solely attention mechanism have drawn the attention in research field and show a remarkable improvement especially in the area of Natural Language Processing [87–89]. Our proposed model is inspired by the Transformer model [90] in neural machine translation (NMT) task. It is the first work that argues, attention mechanisms can effectively be applied in sequence-to-sequence task without the use of recurrent structure.

We applied the transformer architecture in multi-step traffic speed prediction task. However, there are key differences between NMT task and traffic prediction task. In a Chinese-English NMT task, the input is a sequence of Chinese words and output is a sequence of English words. The word vectors of these two languages exist in different feature spaces. In our task, the input and output sequences are both traffic speed values from same data distribution, though at different time steps. Therefore, necessary modification on model structure is required.

4.2.3 Proposed Model

Inspired by the transformer model [90] in neural machine translation, we proposed a full attention model for short-term traffic speed forecast task. Figure 4.1 shows an overview of the proposed encoder-decoder structure. The functions of each part are explained as following:

FIGURE 4.1: Overview of the proposed Encoder-Decoder Structure.

Inputs: Inputs are the historical traffic signal at each encoder step and are defined as $[\mathbf{x}^{(t-T'+1)}, \dots, \mathbf{x}^{(t)}]$, where $\mathbf{x}^{(t)} \in \mathbb{R}^N$ is the measurement of traffic speed reading from N detector sensors at time step t .

Outputs: Outputs are the traffic speed signal at each decoder step. The ground-truth traffic speed values for future T steps are $[\mathbf{x}^{(t+1)}, \dots, \mathbf{x}^{(t+T)}]$, where $\mathbf{x}^{(t)} \in \mathbb{R}^N$. However, during training and inference stage, the future values remain unknown at current step. So we shift the sequence to the right by one place and add a zero vector in front as the starting symbol. The outputs therefore become $[0, \mathbf{x}^{(t+1)}, \dots, \mathbf{x}^{(t+T)}]$.

Input Embedding: In natural language processing, each token from a sentence normally goes through an input embedding layer which projects the token into a higher dimensional space. This transformation normally empowers the token more expressive power. In our case, the input at each time step $\mathbf{x}^{(t)}$ is in a simple concatenated vector form of all readings from N nodes. By passing the vector through an embedding layer would allow the traffic speed variables from different sensors to interact spatially. Here a one-layer feed forward neural network with weight $\mathbf{W} \in \mathbb{R}^{N \times d}$ is employed to obtain the d -dimensional ($d > N$) embedding vector.

$$\mathbf{z}^{(t)} = \mathbf{W}\mathbf{x}^{(t)} \quad (4.2)$$

Positional Encoding: As shown in the experiments of previous chapter, neural network models (WNN, SAEs) would perform worse without considering the temporal information. Since our model does not include the conventional recurrent connection and relies on attention solely, there is no temporal information associated. To incorporate the temporal features, we borrow the idea of position embeddings [91] $p = (p_1, \dots, p_T)$ where $p_i \in \mathbf{R}^d$ is the positional encoding at time step i and it is trained from a randomly initialized lookup table. The positional encoding is then added to the embedding layer output:

$$\mathbf{h}^{(t)} = \mathbf{z}^{(t)} + \text{PE}(\mathbf{x}^{(t)}) \quad (4.3)$$

Encoder: The input embedding with its positional encoding will be fed into the encoder and attention mechanism is applied here. We will explain more details later. The encoder outputs are passed to the decoder.

Decoder: At each step, decoder looks at both the current-step input as well as all outputs from the encoder. Two types of attention mechanisms are employed and will be covered later.

Output layer: Output vector from the decoder will have a different dimension as expected. Output layer will apply a linear projection to map it to a correct dimension space.

4.2.4 Attention mechanism

The main purpose of attention is to find the pairwise relatedness between a query vector and a sequence of vectors. A few types of attention mechanism have been reviewed in section 2.4. Here in our proposed model, we employed a self-attention mechanism. It has been widely used in lots of areas in NLP, such as machine comprehension [92–94], text summarization [95–97] and sequence role labeling [98, 99]. Different from other attention mechanisms where a query vector and a sequence of vectors are provided, query vector comes from the sequence of vectors for self-attention.

Given a sequence of vectors $\{\mathbf{x}^1, \dots, \mathbf{x}^N\}$, each vector $\mathbf{x}^{(i)}$ will undergo three different linear layers and produce its query, key and value vectors $\mathbf{q}^{(i)}$, $\mathbf{k}^{(i)}$, $\mathbf{v}^{(i)}$ respectively:

$$\mathbf{q}^{(i)} = \mathbf{W}_1 \mathbf{x}^{(i)} \quad (4.4)$$

$$\mathbf{k}^{(i)} = \mathbf{W}_2 \mathbf{x}^{(i)} \quad (4.5)$$

$$\mathbf{v}^{(i)} = \mathbf{W}_3 \mathbf{x}^{(i)} \quad (4.6)$$

Now suppose we want to find the self-attention for vector $\mathbf{x}^j$ (where $1 \leq j \leq N$) and the whole input sequence. Pairwise relatedness will be computed between $\mathbf{x}^j$ and $\mathbf{x}^i$. A scaled dot product of these two vectors is applied:

$$a_{ij} = \frac{\mathbf{q}^{(j)} \mathbf{k}^{(i)}}{\sqrt{d}}, \quad (4.7)$$

where a_{ij} is the relatedness value. Similarly, we calculate all the relatedness value of $\mathbf{x}^j$ and every single vector in the sequence. A softmax layer is then applied on the relatedness values to get pairwise weight, which is normalized into the range of $[0, 1]$ with sum one. A weighted sum is then taken to produce the final output, the final output now has all information from the input sequence $\mathbf{x}^1, \dots, \mathbf{x}^N$. An illustration of this attention computation can be visualized in Figure 4.2 and the mathematical formula is written as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \quad (4.8)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are the matrices forms of $\mathbf{q}, \mathbf{k}, \mathbf{v}$.

FIGURE 4.2: Attention mechanism.

The linear projection to query, key, value pairs in self-attention works similarly as the filters in convolutional neural networks. However, to have a better representative power, a CNN usually applies many filters and each filter is in charge of handling one feature type [100]. Therefore a multi-head attention is used and it projects the incoming vectors to queries, keys and values for h times. The attentions are then concatenated and pass to a linear layer to maintain the same dimension as one-head attention.

4.2.5 Encoder-Decoder structure

In this section, a detailed description especially the attention mechanisms will be covered for both encoder and decoder. A clear illustration is shown in Figure 4.3. Our encoder and decoder are stacked by multiple blocks. Each single block consists of two sublayer modules: attention module and a feed forward network module. We also borrow some of the naming conventions from NLP area. For example, source and target means the source and target sequence.

FIGURE 4.3: Encoder and decoder blocks

src-src Attention: applies a self-attention mechanism in the source sequence. An input at step t after embedding layer and positional encoding is given by $\mathbf{h}^{(t)}$. This vector representation does not carry any temporal interaction with other vectors at other time steps. The src-src attention mechanism is employed here to embed the contextual information from surrounding time steps.

trg-trg Attention: similar as src-src attention, trg-trg attention applies a self-attention mechanism in the target sequence. However, due to the nature of time-series forecasting, the information at future steps should remain unknown. Therefore the attention after current step is masked with zero.

src-trg Attention In src-trg attention, the target vector from the target side works as a query vector (e.g., $\mathbf{h}^{(t+1)}$ at time step $t+1$). The attention is calculated with regards to the whole source sequence $[\mathbf{h}^{(t-T'+1)}, \dots, \mathbf{h}^{(t)}]$.

Feed Forward Network: Each encoder and decoder block also contains a feed forward network. It consists of two linear transformations with a ReLU activation in between:

$$FFN(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2, \quad (4.9)$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d_{ff}}$ are the weights and b_1, b_2 are the biases of the two linear layers. $\mathbf{x}$ is the output from previous attention module.

Linear: The output of feed forward layer has a dimension of d_{ff} . In order to obtain a prediction of dimension N , we simply add a linear layer with weights $\mathbf{W} \in \mathbb{R}^{d_{ff} \times N}$.

4.3 Experiments

4.3.1 Dataset description

We conducted the experiments on a new public traffic dataset **METR-LA** [86]. METR-LA dataset has become popular recently and is used in many research works as a standard dataset for model comparison.

This high-resolution spatial-temporal transportation data comes from a project maintained by the LA county road network by Los Angeles Metropolitan Transportation Authority (LA-Metro). In total, 8,900 traffic loop-detectors were located on the highways and arterial streets to collect traffic speed data at the rate of one reading/sensor/min. In this experiment, traffic speed data from 207 sensors are selected, during a period of 4 months from Mar 1st 2012 to June 30th 2012.

The experiment was designed to predict the short-term traffic speed. We examine the model’s capability to predict the next 30-min traffic speed at an interval of 5 minutes, based on the past one hour traffic speed data. The dataset is split into a ratio of 70%/10%/20% for train/valid/test set.

4.3.2 Preprocessing

A standard scaling is applied to the raw traffic speed data based on the following formula:

$$z = (x - u)/s, \quad (4.10)$$

where x is the raw traffic data and z is the numeric value after applying standard scaling. x and u are mean and standard deviation of the training data.

4.3.3 Implementation Details

We trained our model with the PyTorch deep learning library. During the training process, teacher-forcing technique is applied and loss on validation set is monitored, with early stopping of patience 20.

4.3.3.1 Optimizer

We used the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. For better optimization, warmup strategy [101] is applied and the learning rate is varied according to the formula:

$$\text{lr} = d^{-0.5} \cdot \min(\text{step_num}^{-0.5}, \text{step_num} \cdot \text{warmup_steps}^{-1.5}) \quad (4.11)$$

As shown in Figure 4.4, the learning rate is increased linearly for the first warmup_steps. After that, it is decreased proportionally to the inverse square root of the step number. In the experiment, warmup_steps is set to 400.

FIGURE 4.4: Variation of learning rate value against the number of training steps

4.3.3.2 Regularization

Deep learning models are usually complicated and overfitting is one of the most common problems. In the experiment, we employed two techniques, residual connection [102] and dropout [103].

Residual Connection defines a simple short-cut connection for a sub-layer:

$$y = f(x) + x, \quad (4.12)$$

where $f(x)$ is the sub-layer and x is the input to the sub-layer. By applying this simple element-wise addition of input and output to the sub-layer, it helps to prevent exploding and vanishing gradient problems [102, 104]. In our model, the residual connection structure is introduced to every attention module and feed forward network module.

Dropout randomly drops out the neurons by a probability P_{drop} . It makes every neuron learn and contribute to the final decision and therefore prevents overfitting

in model training. We apply dropout to output of each sub-layer, attention module and positional encoding. $P_{drop} = 0.1$ is used.

4.3.4 Compared models

In order to prove the effectiveness of our proposed model, we compared the model with a few popular baselines, including conventional time-series regression models and recent deep learning models. The compared models include

- **ARIMA_{kal}**: Auto-Regressive Integrated Moving Average model with Kalman filter which is widely used in time series prediction.
- **FNN**: Neural network model with fully connection between consecutive hidden layers.
- **LSTM-Attn**: LSTM networks with attention mechanism from decoder hidden state to encoder outputs [52].

4.3.4.1 Parameters for comparing models

ARIMA_{kal} the orders are set as (3, 0, 1).

FNN There are two hidden layers and each layer contains 256 units. The initial learning rate is set to 0.01 and a scheduler is set to decrease the learning rate by $\frac{1}{10}$ every 20 epochs.

LSTM-Attn It follows the encoder-decoder structure with attention proposed in [52]. Both encoder and decoder contains two recurrent layers with 256 neurons each. The initial learning rate is 0.01 and reduced to $\frac{1}{10}$ every 10 epochs from epoch 15.

4.3.5 Metrics

Three common metrics are used to evaluate the performance of different models, including (1) Mean Absolute Error (MAE), (2) Mean Absolute Percentage Error (MAPE), and (3) Root Mean Square Error (RMSE). They are defined as following:

$$MAE = \frac{1}{n} \sum_{i=1}^n |x_i - \hat{x}_i|, \quad (4.13)$$

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{x_i - \hat{x}_i}{x_i} \right|, \quad (4.14)$$

$$RMSE = \left[\frac{1}{n} \sum_{i=1}^n (|x_i - \hat{x}_i|)^2 \right]^{1/2}, \quad (4.15)$$

where x_i is the real traffic flow observed by the detector and $\hat{x}_i$ is the predicted traffic flow by the model.

4.4 Results

4.4.1 Metrics comparison

Experiment results for different models are recorded in Table 4.1 and Table 4.2. The performance of each model is compared for different forecasting horizons, at a 5-minute interval.

In Table 4.1, the models' predicting capability is examined at the horizon of 5-min, 10-min and 15-min. In all three metrics of MAE, MAPE and RMSE, our proposed model achieves the best performance. In Table 4.2, the performance for future 20-min, 25-min and 30-min are reviewed. Our model performs best at 20-min and 25-min. The performance at 30-min is slightly worse than LSTM-Attn, though still comparable. This may be attribute to the complicated attention mechanism in the structure, which makes it hard to capture a long time dependency.

In general, neural network based models perform much better as compared to the traditional statistical model $ARIMA_{kal}$. This is accordant with the fact that neural network has a strong capability in mapping the complicated and non-linear data to a higher-dimensional space and obtain a more meaningful function in that space. Out of the three neural network models, FNN performs the worst, which is also expected. As explained in previous section, FNN contains a fully-connection structure. The sequential feature is lost and this feature is important in time

series prediction. Both LSTM-Attn and our model apply the attention mechanism. Though our model does not have a complicated recurrent structure like LSTM-Attn and is full attention-based, it still has a positional encoding layer to encode a simple sequential information. The attention mechanism also forces the model to learn the temporal correlation based on the positional encoding feature. The experiment result shows that, with a simple positional encoding, the attention mechanism can work well in predicting the short-term traffic speed.

TABLE 4.1: Performance Comparison with different models for traffic speed forecasting at horizon 5-min, 10-min and 15-min.

Models		ARIMA _{Kal}	FNN	LSTM-Attn	Our model
5 mins	MAE	3.99	3.67	3.08	2.95
	MAPE	11.6%	9.9%	8.03%	7.81%
	RMSE	8.21	7.94	5.62	5.06
10 mins	MAE	4.78	3.92	3.26	3.13
	MAPE	12.1%	10.3%	8.69%	8.45%
	RMSE	9.23	8.02	6.17	5.74
15 mins	MAE	5.15	4.23	3.40	3.33
	MAPE	12.7%	11.2%	9.26%	9.21%
	RMSE	10.45	8.17	6.59	6.33

TABLE 4.2: Performance Comparison with different models for traffic speed forecasting at horizon 20-min, 25-min and 30-min.

Models		ARIMA _{Kal}	FNN	LSTM-Attn	Our model
20 mins	MAE	5.43	4.31	3.52	3.49
	MAPE	13.5%	12.1%	9.71%	9.86%
	RMSE	10.91	8.24	6.90	6.81
25 mins	MAE	6.02	4.39	3.61	3.61
	MAPE	15.3%	12.9%	10.09%	10.08%
	RMSE	12.31	8.42	7.14	7.13
30 mins	MAE	6.90	4.49	3.70	3.73
	MAPE	17.4%	14.0%	10.44%	10.62%
	RMSE	13.23	8.69	7.35	7.44

4.4.2 Forecast Visualization

In order to have a clearer view on the forecasting capability of LSTM-Attn and our proposed full attention model, the predicted traffic speed values are plotted against different time of the day. We compare the two models under two different scenarios.

4.4.2.1 Peak hours

Traffic conditions change during different time of the day. Peak hours refers to the period of the day where traffic congestion is the highest on the road. It is normally occurs at three times of the day when people are on the way to and from work. There are many cars on the road and traffic conditions are usually bad. Therefore, the traffic speed recorded is slow. A good traffic prediction model should be able to identify the start and end of peak hours. As shown in Figure 4.5, the peak hours are in the morning (7:30 am - 09:30 am), noon (12 pm - 2 pm) and evening (4 pm - 7 pm). While both models are able to identify the alternating trends between peak and non-peak hours, our proposed model is able to capture the change in the speed better and give a more accurate prediction. An example is during the period of 2 pm to 3 pm, where there is a short change from noon peak hours to non-peak hours and fall back to peak hours again. Our model is capable of generate closer predictions. Another example is from 7 pm to 8 pm, where there is another short switching between peak to non-peak. Our model also captures the changes.

4.4.2.2 Speed Fluctuation

The road traffic condition is usually complicated. During special occasions such as traffic jam, there will be a lot of oscillations in the actual traffic speed and renders extra difficulties on model's forecasting capability.

The road condition is always complicated and less controllable. During special occasions such as traffic jam or traffic accidents, or simply due to stochastic nature of the traffic, there will be fluctuations in the actual traffic speed readings. This poses a challenge to the predictive models in different fluctuation situations. Here we look at two cases, one with a high frequency of fluctuation and the other less frequent.

(A) LSTM with attention

(B) Our model

FIGURE 4.5: Forecasting visualization for peak hours

In Figure 4.6, the traffic condition is terrible where it fluctuates randomly and intermittently during the daytime period. LSTM-Attn tends to generate predictions that are easily affected by the fluctuation and deviate largely from the actual traffic speed readings. On the other hand, our model is able to generate a more smooth prediction and capture the trend of change well. This is probably due to three attentions applied in our model. When the model is predicting, it looks at all previous readings and is therefore less affected by the high frequent fluctuation.

In Figure 4.7, an instance of regular fluctuation is inspected. It shows the traffic condition in a normal day with a few abrupt changes in the traffic speed. In general, our model shows the capability of producing smoother prediction. Moreover, at the minor fluctuation values, our model captures the trends and peak well. In contrast, LSTM-Attn's prediction is unstable and this makes it hard to apply in a real-world application. Again, it verifies the effectiveness of the multiple attention mechanisms.

4.4.3 Complexity analysis

In this section, complexity analysis in the form of big-O notation is presented for LSTM and full attention model. The complexity computation depends on the input sequence length and vector dimension. As defined in equation 4.1, $\mathbf{x}^{(t)}$ is the raw input data at each time step and T' is the input sequence length. The raw data cannot be directly used and normally goes through an embedding layer. Output from the embedding layer has a dimension of d is used here for complexity analysis. For simplicity, L is used for input sequence length instead of T' .

As shown in equation 3.4, the recurrent structure in an LSTM cell takes the current input and previous state followed by a matrix multiplication. Matrix multiplication is expensive in computation and has a complexity of $O(d^2)$. The total complexity with a sequence length of L is therefore $O(L \cdot d^2)$.

For the proposed full attention model, attention weights are computed in dot product form as shown in equation 4.7. A dot product computation is less expensive and takes the complexity of $O(d)$. With an entire sequence, attention calculation requires the vector at current time step to compute with vectors at any other time

(A) LSTM with attention

(B) Our model

FIGURE 4.6: Forecasting visualization under frequent fluctuation

(A) LSTM with attention

(B) Our model

FIGURE 4.7: Forecasting visualization under regular fluctuation

steps. Therefore, a sequence of length L needs L^2 times of attention computation. In total the computation complexity is $O(L^2 \cdot d)$.

In our task, L has a value of 6 and d has a value of 64 or 256. Therefore the full attention model has a much less computation complexity as compared to LSTM model.

4.5 Conclusions

In this chapter, a full attention model for short-term traffic speed prediction task is proposed to solve two potential issues when applying LSTM models to multi-step prediction. An encoder-decoder LSTM model maps the whole input sequence into a fixed-length vector and the decoder implements the prediction solely on this vector. It poses a strong challenge to the model to learn a informative fixed-length vector. Another problem is the expensive computation cost due to the recurrent connections in LSTM.

The proposed model replaces the LSTM with full attention and the attention also assists the fixed-length vector during decoding. The parallel computation of attention weights also reduces the time complexity. Experiments are carried out on a traffic speed dataset METR-LA. Results show that our proposed model reduces MAE, MAPE and RMSE in different prediction horizons from 5 mins to 30 mins. Visualization also shows that the proposed model is able to capture the trends of traffic speed changes. A time complexity analysis in terms of big-O notation further proves that the proposed model reduces the computation cost.

Chapter 5

Graph Diffusion Recurrent Network for traffic speed forecasting

In this chapter, we propose a graph diffusion recurrent network for traffic speed forecasting. In a traffic network where different road segments are interconnected, the traffic condition at one single node will be affected by the nearby nodes. Therefore the main objective of our proposed model is to explore the spatial correlations among the neighboring nodes. With the additional spatial features, we compare our model with a few strong baselines and also the state-of-the-art model. Experiment result shows that our model performs better than other models on the METR-LA dataset.

The rest of the chapter is organized as follows. Section 5.1 presents the motivation of this work and the overview of the problem. In Section 5.2, details of our model structure are presented. Section 5.3 elaborates on the experiment design. Experimental results, implementation details and visualizations are covered in Section 5.4. Finally the conclusion is presented in Section 5.5.

5.1 Introduction

5.1.1 Motivation

In Chapter 4, a fully attention model is proposed to deal with multivariate traffic speed prediction problem. However, a large focus was placed upon the attention part and the spatial correlation of nearby sensors are simply represented with an embedding layer. The fully-connected embedding layer only offers the linear combinations of nearby nodes' readings. A stronger feature extractor is required to effectively capture the spatial correlation.

5.1.2 Convolutional Neural Networks for traffic prediction

As shown in Chapter 2, CNNs are powerful in extracting the features within one local region from the images. Some research works have adapted CNNs into traffic prediction, applying it to incorporate the spatial correlation. For example, Wu et al. [105] attempted to combine LSTM with a 1-D convolutional neural networks (CNN) [106]. Meanwhile, Shi et al. [107] introduced convolutional LSTM which replaced the fully connected layer with convolutional operation.

Despite the performance gain from the proposed models, there are some drawbacks. Those models decoupled the traffic modeling into spatial and temporal aspects. This is inherently against the nature of the problem. On the other hand, although CNNs have been proven to work well with 2D-grid data, they cannot be directly applied to the general graph structure. Traffic network is a network with interconnected adjacent sensor nodes. It is more natural to represent the traffic network as a graph structure. Therefore, more attention should be paid to the spatial connectivity and distances in the graph structure.

5.1.3 Graph Neural Networks

Graph is a general type of data structure where a set of nodes and their relationships (edges) are presented. Due to its more natural and expressive power as graphs, it has recently gained increasing attention from many research fields, such as protein

interface prediction[108], natural language processing [109, 110] and knowledge graph [111, 112]. Graph neural network (GNN) is a deep learning model designed for graph domain. While it is hard to transform some problems to fit the deep learning models, GNN has a few advantages [113]. GNN is invariant to the input order, so it is more flexible to the input data structures. Edge information can work as a guide to the feature propagation. Some pioneering works [68, 114, 115] have explored the information propagation methods in different task and achieved state-of-the-art results.

In this chapter, we proposed a deep learning based model, Graph Diffusion Recurrent Neural Networks (GDRNN) to study the traffic speed forecasting problem. We formalize the task as a sequence-to-sequence problem and adapt the encoder-decoder structure here. Both spatial and temporal dependencies are studied. Recurrent Neural Networks (RNNs) is used to model the temporal dynamics. The spatial correlation is explored by applying graph neural networks (GNNs), where the traffic network is represented as a graph with each sensor as the node. Three variants of graph pooling variants are proposed and verified by experiments.

5.2 Methodology

5.2.1 Problem Definition

The goal of traffic flow prediction is to predict the multi-step ahead traffic flow given the historical traffic flow value. The sensor networks can be represented by a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ is the set of sensor nodes $|\mathcal{V}| = N$, $\mathcal{E}$ is the set of edges and $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the network structure.

The traffic flow can be denoted by $\mathbf{X} \in \mathbb{R}^{N \times d}$, where d is the features of each node. Let $\mathbf{X}^t$ be the graph signal at time t , the problem of traffic flow prediction can be formulated as:

$$\left[\mathbf{X}^{(t-T'+1)}, \dots, \mathbf{X}^{(t)}; \mathcal{G} \right] \longrightarrow \left[\mathbf{X}^{(t+1)}, \dots, \mathbf{X}^{(t+T)} \right], \quad (5.1)$$

where T is the number of time steps of prediction and T' is the number of historical time steps.

5.2.2 Notations

In this paper, we will use bold lowercase letters and bold uppercase letters for vectors and matrices respectively. The Fully-Connected (FC) layer with a nonlinear activation function α is denoted by $FC_\theta^\alpha(\mathbf{x}) = \alpha(\mathbf{W}\mathbf{x} + \mathbf{b})$, where $\mathbf{x}$ is the input and θ is the parameter ($\mathbf{W}$ represents the weight matrix and $\mathbf{b}$ is the bias to the FC layer). For activation functions, $h(\cdot)$ represents **ReLU** activation and $\sigma(\cdot)$ represents the sigmoid activation. An FC layer denoted by $FC_\theta(\mathbf{x})$ means no activation applied or in another meaning a linear transformation. $\oplus$ means matrix concatenation operation. A dot product between two vectors is denoted by $\langle \cdot, \cdot \rangle$.

5.2.3 Proposed model

FIGURE 5.1: Our proposed GDRNN model with an encoder-decoder structure

In order to solve the traffic forecasting problem, we proposed a graph diffusion recurrent neural network. It has an encoder-decoder structure, as illustrated in Figure 5.1. The input to the encoder are the traffic network graph signal $\mathbf{X}^t$ at each time step. The initial hidden state to the encoder are set to a zero hidden state. The last hidden state of the encoder as well as a special zero-input $\langle GO \rangle$ will be used to initiate the decoding process. The main block of our proposed

model consists of graph aggregator, diffusion convolution, and temporal dynamic. We will explain each part in details in the following subsections.

5.2.3.1 Graph Aggregator

In graph neural networks, the neighbor nodes' features are usually aggregated to the central node, so that each node will carry both its own and its neighbors' features. We define a general graph aggregator as $\mathbf{y}_i = \gamma_{\Theta}(\mathbf{x}_i, \{\mathbf{z}_{\mathcal{N}_i}\})$, where $\mathbf{x}_i$ is the original feature vector and $\mathbf{y}_i$ is the feature vector after aggregation for the central node i . $\mathcal{N}_i$ is the set of neighboring nodes of central node i and $\mathbf{z}_{\mathcal{N}_i}$ are the feature vectors of the neighboring nodes. $\gamma(\cdot)$ is the graph aggregation function. In this paper, we explored two types of graph aggregation functions.

Graph pooling aggregator does not consider individual connection of the central node and neighbor node. It uses a pooling operation to aggregate the features from all neighboring nodes. Some of the common pooling operations are max, average or sum pooling. Here, we adopted the average pooling and it is defined as below:

$$\mathbf{y}_i = \text{FC}_{\theta_o}(\mathbf{x}_i \oplus \text{pool}_{j \in \mathcal{N}_i}^{\text{avg}}(\text{FC}_{\theta_v}^h(\mathbf{z}_j))) \quad (5.2)$$

The neighboring nodes' feature will go through a fully-connected layer with *ReLU* activation followed by an average pooling operation. The feature vectors are then concatenated to the central node feature vector and go through another fully-connected layer with no activation function.

Graph pairwise sum aggregator looks at the pairwise connection of the central node and its neighbor node. As shown in Figure 5.2, each connection will be assigned with its own individual weights. Two variants of pairwise sum aggregator are explored.

FIGURE 5.2: Pairwise sum aggregator

The first variant Pairwise-Weighted is simple and straight forward. We inject the pairwise graph structure by a masked adjacency weight matrix. The aggregator is a feed forward neural network with bilinear matrix multiplication:

$$\mathbf{Y} = (\mathbf{A}\mathbf{W})\mathbf{X}, \quad (5.3)$$

where $\mathbf{W}$ is the learnable weights parameter between the nodes. $\mathbf{A}$ is the boolean adjacency matrix with 1 or 0, as a masking to $\mathbf{W}$. The input and output feature matrices of this aggregation function are $\mathbf{X} \in \mathbb{R}^{N \times d_i}$ and $\mathbf{Y} \in \mathbb{R}^{N \times d_o}$.

In the second variant Pairwise-Attention, we relied on a more complicated neural network model to calculate the pairwise weights:

$$w_{i,j} = \text{softmax}(\langle \text{FC}_{\theta_x}(\mathbf{x}_i), \text{FC}_{\theta_z}(\mathbf{z}_j) \rangle) \quad (5.4)$$

$$\mathbf{y}_i = \text{FC}_{\theta_o} \left(\mathbf{x}_i \oplus \sum_{j \in \mathcal{N}_i} w_{i,j} \text{FC}_{\theta_v}^h(\mathbf{z}_j) \right) \quad (5.5)$$

For a central node i and its neighbor node j , the feature vector is projected into another vector space with mapping function $FC_\theta : \mathbb{R}^{d_i} \rightarrow \mathbb{R}^{d_p}$. Pairwise weights are computed by applying dot products, followed by a softmax layer. To get the output feature, we pass the neighboring feature $\mathbf{z}_j$ to another FC layer with ReLU activation, $FC_{\theta_v} : \mathbb{R}^{d_i} \rightarrow \mathbb{R}^{d_v}$. The overall neighboring nodes' features are obtained by taking the weighted sum of this projection vector. The output feature $\mathbf{y}_i \in \mathbb{R}^{d_o}$ is obtained after a one more FC layer FC_{θ_o} .

5.2.3.2 Diffusion Convolution

Convolutional Neural Networks (CNNs) have been implemented successfully in Euclidean or grid-like structures, such as images. However the performance is not promising when it is introduced to the general graph structure data. Recently, there are some research works which have applied the graph convolution networks. Seo et al. [116] proposed Graph Convolutional Recurrent Neural Network (GCRNN), which replaced the fully-connected layers in LSTM [84] with the ChebNet operator [117]. Atwoord et. al [118] proposed a Diffusion Convolution Neural Network (DCNN) where a diffusion convolution is applied to general graph structure. Following this work, Li et al. [119] extends the DCNN to a diffusion convolution layer. It replaces the fully-connected layers in GRU with the diffusion convolution operator. In our model, we also apply diffusion convolution as one of our model components. The diffusion convolution layer maps a P -dimensional input feature to a Q -dimensional output feature.

$$\mathbf{H}_{:,q} = \mathbf{a} \left(\sum_{p=1}^P \mathbf{X}_{:,p} \star_{\mathcal{G}} f_{\Theta_{q,p,:}} \right), \text{ for } q \in \{1, \dots, Q\}, \quad (5.6)$$

where $\mathbf{X} \in \mathbb{R}^{N \times P}$ is the input and $\mathbf{Y} \in \mathbb{R}^{N \times Q}$ is the output feature. $f_{\Theta_{q,p,:}}$ are the learnable filter parameters and $\mathbf{a}$ is the activation function.

5.2.3.3 Temporal Dynamic

We leverage the Recurrent Neural Networks (RNNs) to model the temporal dynamics. More specifically, we use the encoder-decoder RNN structure with Gated Recurrent Units (GRU) [120], which is a simple and yet proved to be effective RNN

unit. Hence we use GRU for our temporal modeling. The key difference to conventional GRU structure is, we modified the gating mechanism by adding neighboring information and replaced the fully-connected layers with the diffusion convolution operator:

FIGURE 5.3: GDRNN block

$$\mathbf{r}^{(t)} = \sigma(\mathbf{W}_r [\Gamma_r(\mathbf{X}^{(t)}), \mathbf{H}^{(t-1)}] + \mathbf{b}_r) \quad (5.7)$$

$$\mathbf{u}^{(t)} = \sigma(\mathbf{W}_u [\Gamma_u(\mathbf{X}^{(t)}), \mathbf{H}^{(t-1)}] + \mathbf{b}_r) \quad (5.8)$$

5.3 Experiments

We conducted the experiments on a real-world large-scale traffic speed dataset **METR-LA** [86]. This high-resolution spatial-temporal transportation data was collected from the LA county road network by Los Angeles Metropolitan Transportation Authority (LA-Metro). About 8,900 traffic loop-detectors were located on the highways and arterial streets to collect traffic parameters such as occupancy, volume, and speed at the rate of one reading/sensor/min.

The experiment was designed to predict the future 15-minute, 30-minute and 60-minute traffic speed based on the past one hour traffic speed data. The traffic speed readings are aggregated at a 5-min interval. We selected the traffic speed data from 207 sensors, during a period of 4 months from Mar 1st 2012 to June 30th 2012. The dataset is split into a ratio of 70%/10%/20% for train/valid/test set.

5.3.1 Preprocessing

A standard scaling is applied to the raw traffic speed data based on the following formula:

$$z = (x - u)/s, \quad (5.11)$$

where x is the raw traffic data and z is the numeric value after applying standard scaling. x and u are mean and standard deviation of the training data.

Also, additional time feature is used in the model. Since the data is aggregated in 5-min interval, there are in total 288 time stamps in a day. We firstly map the time stamps in to the index of 0-287. Then the max-min scaling is applied to convert the time feature in the range of 0-1.

$$z_t = \frac{t - t_{min}}{t_{max} - t_{min}}, \quad (5.12)$$

where t_{max} is 287 and t_{min} is 0. t is the corresponding index of the time stamp. The converted time feature z_t is concatenated to the scaled traffic speed reading to obtain a two-dimensional feature input $[z; z_t]$.

Following [119][121], the adjacency matrix were built by using thresholded Gaussian kernel based on the distance between detector nodes:

$$W_{ij} = \begin{cases} \exp(-\frac{dis(v_i, v_j)^2}{\sigma^2}), & \text{if } dis(v_i, v_j) \leq k \\ 0, & \text{otherwise} \end{cases}$$

where $\mathbf{W}$ is the adjacency matrix, and $dis(v_i, v_j)$ is the distance between node i and j . σ is the standard deviation of the distances and k is the threshold with value 0.1.

5.3.2 Implementation details

We stack two layers of GDRNN with a hidden size of 64 both in the encoder and decoder. The three graph aggregator variants used are graph Ave-Pool, Pairwise-Weighted and Pairwise-Attention. For all aggregators, we set $d_v = 64$ and $d_o = 64$. In Pairwise2, the linear projection $d_p = 64$. During training, we used a batch size of 64 and Adam as the learning optimizer. The initial learning rate set to 0.01 with a decay ration of $\frac{1}{10}$ at training epoch [20, 30, 40, 50]. Scheduled sampling [83] technique is applied during training for faster and better loss convergence. During training, MAE was used as the loss function. Early stopping was performed by monitoring the loss on validation set. The input to the model is the traffic speed information over the last 12 time steps and the prediction is done for the future 12 horizons. We then evaluate the MAE, RMSE, MAPE on the hozion of 15-min, 30-min and 60-min for short, media and long-term traffic forecasting.

In order to have a fair comparison, we stack two layers for FC-LSTM and FNN. The hidden layer dimension is set to 128. The initial learning rate is set to 1e-4 and 1e-3 respectively.

In order to have a fair comparison, we tuned the hyperparameters for the neural network methods (FC-LSTM, FNN) to obtain each model's optimal results. There are 256 hidden neurons in the recurrent layer, with initial learning rate 0.1 and L1 weight decay 1e-5. For DCRNN, we follows the hyperparameters reported in [119].

5.3.3 Compared models

We compared our model with a few popular baselines, including conventional time-series regression models and recent deep learning models. The models include (1) $ARIMA_{kal}$: Auto-Regressive Integrated Moving Average model with Kalman filter which is widely used in time series prediction; (2) Feed forward Neural network (FNN): Feed forward neural network with two hidden layers and L2 regularization. (3) Recurrent Neural Network with fully connected LSTM hidden units (FC-LSTM) [50]. To prove the effectiveness of our proposed model in incorporating the graph structure, we also compare with the state-of-the-art model in METR-LA dataset, DCRNN[119].

5.3.4 Metrics

Three common metrics are used to evaluate the performance of different models, including (1) Mean Absolute Error (MAE), (2) Mean Absolute Percentage Error (MAPE), and (3) Root Mean Square Error (RMSE). They are defined as following:

$$MAE = \frac{1}{n} \sum_{i=1}^n |x_i - \hat{x}_i|, \quad (5.13)$$

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{x_i - \hat{x}_i}{x_i} \right|, \quad (5.14)$$

$$RMSE = \left[\frac{1}{n} \sum_{i=1}^n (|x_i - \hat{x}_i|)^2 \right]^{1/2}, \quad (5.15)$$

where x_i is the real traffic flow observed by the detector and $\hat{x}_i$ is the predicted traffic flow by the model.

(A) DCRNN for 15-min forecasting

(B) GDRNN for 15-min forecasting

FIGURE 5.4: Forecasting visualization at 15-min horizon

(A) DCRNN for 30-min forecasting

(B) GDRNN for 30-min forecasting

FIGURE 5.5: Forecasting visualization at 30-min horizon

(A) DCRNN for 60-min forecasting

(B) GDRNN for 60-min forecasting

FIGURE 5.6: Forecasting visualization at 60-min horizon

5.4 Results and Discussion

5.4.1 Performance analysis

Table 5.1 shows the performance of our model against the popular baseline models. We evaluate on all three variants of GDRNN: graph average pooling aggregator (Ave Pool), graph pairwise weighted aggregator (Pairwise-Weighted) and graph pairwise neural network aggregator (Pairwise-Attention). It is clear that all our model variants have consistently beaten the baseline models with obvious advantage. Comparing to the conventional linear time-series models, deep learning models (FNN, FC-LSTM and our models) achieved much better results, due to models' ability to represent the non-linearity in the data. Especially for long-term forecasting, the errors are much smaller. The better performance of FC-LSTM over FNN can be attributed to the recurrent unit structure which excels at modeling temporal dynamics.

We also compared our model with DCRNN[119], which is the state-of-the-art graph neural network based model on METR-LA dataset. With the extra use of graph structure information, both models achieves better result than normal deep learning models. All three variants are compared to DCRNN and show promising better performance. This shows the effectiveness of graph aggregation function in capturing the graph structure. However, the average pooling aggregator only takes the simple average of neighbor nodes' features, without considering the correlation between individual neighbor node and the central node. Different neighbor node may contribute differently to the central node. This probably explains why the pairwise aggregator outperforms the average pooling aggregator. Also compared to the Pairwise-Weighted, Pairwise-Attention has stronger feature capturing power due to its more complicated neural network aggregator structure.

Figure 5.4, 5.5 and 5.6 visualize traffic forecasting of DCRNN and GDRNN (pairwise2) at different horizons. In general, our model is able to give a more smooth prediction at different horizons. During peak hours such as 9 a.m., the traffic speed is slow probably due to traffic jam. The traffic speed will fluctuate heavily. Since the neighboring road intersection also experience the similar traffic jam scenario, by incorporating this information, our models are able to make smoother and more accurate prediction.

5.4.2 Complexity analysis

In this section, complexity analysis in the form of big-O notation is presented for LSTM and proposed graph neural network variants. Time complexity per layer is investigated and it depends on the input sequence length and vector dimension. As defined in equation 4.1, $\mathbf{x}^{(t)}$ is the raw input data at each time step. The raw data goes through an embedding layer and the output has a dimension of d which is used here for complexity analysis. Also L is used to represent the input sequence length.

As show in Chapter 4.4.3, the time complexity per layer for LSTM is $O(L \cdot d^2)$ and that for full attention model is $O(L^2 \cdot d)$. Also a FC layer is mathematically a matrix multiplication of weights and input in the form of $W^T X$. Therefore, the complexity is $O(d^2)$. Operations such as concatenation, and scalar product can be ignored.

For time complexity analysis of the proposed graph diffusion recurrent networks, we decouple them into three parts, graph aggregator, diffusion convolution layer and recurrent layer.

In Ave-Pool graph aggregator, there are two FC layers and the complexity is $O(d^2)$. In Pairwise-Weighted graph aggregator, the weighted adjacency matrix multiplication requires a complexity of $O(d^2)$. Pairwise-Attention graph aggregator include both attention mechanism and FC layer, therefore the time complexity is $O(L^2 \cdot d + d^2)$.

The convolutional layer in diffusion convolutional layer requires a complexity of $O(k \cdot L \cdot d^2)$ where k is the kernel size.

In every block of the recurrent layer, graph aggregators and convolutions are computed. Therefore, the complexity is increase by L times.

In conclusion, the time complexity for Ave-Pool model and Pairwise-Weighted is $O(L \cdot d^2 + k \cdot L^2 \cdot d^2)$. For Pairwise-Attention model, the complexity is higher in the form of $O(L^3 \cdot d + L \cdot d^2 + k \cdot L^2 \cdot d^2)$ due to the attention mechanism used. The better performance is achieved with the cost in higher computation time complexity.

5.5 Conclusion

In this chapter, we have proposed a graph diffusion recurrent neural network for multi-step traffic speed prediction task. It can utilize the neighboring nodes' traffic information to improve the prediction on the central node. In order to incorporate this spatial information, three graph aggregators are proposed. They take different forms of feature fusion from neighbor nodes. Ave-Pool uses an average pooling on neigh features. Pairwise-Weighted leverages the adjacency weight matrix and applies a mask on it. Pairwise-Attention learns the weights with neural network attention mechanism without prior weight matrix information. On the other hand, recurrent neural network is used to model the temporal dependencies on the node after it fuses the neigh information at each time step. Therefore the proposed model is able to capture both spatial and temporal dependencies in the data.

Experiments results on METR-LA traffic speed prediction dataset shows that all three proposed graph aggregators beat the state-of-the-art model in MAE, MAPE and RMSE. Discussion section shows that, the proposed models achieve better performance on the cost of higher computation.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

A few studies have shown that, the increase in number of vehicles and less regulations on the traffic would result to a few consequences in our daily life, such as lost of productivity, emission of greenhouse gases and higher fuel consumption. It is therefore imperative to have an intelligent transportation management system. An ITS normally consists of traffic infrastructures, wireless communications and data analytics. Enabled by the recent advances in WSN and IoT technologies, large-scale deployment of the sensors on traffic infrastructures and collection of the data become feasible and painless. On the other hand, data analytics, which explores the hidden correlation within the data and gives instructions to the decision making on traffic management, plays a more and more important role. Therefore, in this thesis, data-driven models which are based on deep learning approaches for traffic prediction are investigated.

In Chapter 2, existing data-driven approaches on traffic prediction are extensively reviewed. They cover from traditional statistical models, simple machine learning techniques to a few pioneering works with deep learning approaches. By reviewing the previous researches, we are able to realize the drawbacks of the existing works and draw insights on our proposed models in this thesis.

In Chapter 3, we proposed a LSTM based models for next-step traffic flow forecasting. Different from existing deep learning approaches, our model takes the temporal

information of historical traffic flow values. LSTM is powerful of capturing sequential orders of the traffic data with four gating mechanisms. We used LSTM as the encoder part and the decoder employs a linear projection layer. The model achieves significant improvements over several strong baselines as well as existing deep learning models. Results shows that the MAPE metrics can be reduced to 5.40% with our proposed model as compared with previous state-of-the-art 6.57%.

However, the recurrent connection in our proposed model remains a challenge in real-world use. With a recurrent connection, the computation at the next step has to wait for the finish of current computation. This poses a constraint if longer sequences are used as the input and a limitation for parallel computing. With the inspiration from NMT [90], we developed a fully attention model for traffic speed forecasting. Our model employs an encoder-decoder structure. Two attentions are used, one with self-attention among source and target sequences and the other with cross-attention between source and target sequences. The use of attention allows the prediction to attend to all information at previous steps and effectively improves the performance.

By intuition, the traffic condition of the current road will affect the nearby road segments. Thus, exploring the spatial dependencies of the whole traffic networks becomes purposeful. Due to the graph-like topology of most traffic networks, we did not apply CNN which is proven to handle grid-like data well. Instead, GNN models are used to combine neighboring nodes' information to the central node. Three variants of graph aggregation functions are explored. The first one applies a simple average pooling operation on neighboring features. The second one relies on a one-layer neural network to learn pairwise weights for neighboring features and a weight mask is used. The third one also learns a pairwise weights, however, the weights are determinend by a dot product of features between two nodes followed by softmax.

In summary, we successfully applied deep learning approaches to traffic prediction task. LSTM takes care of the sequential information well and GNN can capture the spatial dependency by incorporating neighboring nodes' information. Last but not least, attention mechanism is also shown to be effective in regularizing the prediction by attending to all historical data.

6.2 Future works

The thesis focuses on the spatial-temporal dependencies in the traffic data and has achieved significant improvement. However, there are still some potential areas worth investigating:

1. Combination of GNN and attention mechanism

In Chapter 4 and Chapter 5, since the two works are carried out in parallel, we investigate the performance of attention mechanism and GNN separately. However, in the fully attention model, the spatial dependency is only captured by a simple one-layer neural network and the GNN structure adopts the recurrent structure. The two models can be combined together and that may improve the performance further.

2. Add periodic information

Traffic data usually shows the periodic pattern. For example, the pattern of the traffic condition on a particular weekday will be similar as the same weekday of last week. Incorporating this information into the model will very likely improve the model performance.

3. Data imputation

Although deployment of sensor networks have offered a rich source of data, the quality of data is not guaranteed. Sensors tend to die easily when it is running out of battery or when it is exposed to a severe physical condition. In this thesis, we have a good quality of data source and preprocessing of the dataset is carried out so that some missing data is filtered. However, it is still worth investigation on the data imputation methods. One potential approach is to apply LSTM regarding on the historical data of the node itself. Another possible way is by inspecting the values recorded from the neighboring nodes.

4. External information

Some external sources of information can be leveraged to enhance the traffic prediction. For example, the traffic will be heavy on a national day since most of the people plan to go out. Another case could be the use of social media. In the daily life, people tend to complain in social media when the

road is congested. NLP techniques can be applied to analyze the sentiment of the posts and help the traffic prediction.

Author's Publications

Journal Articles

- **Hongxin Shao**, Boon-Hee Soong, “Exploring fully attention model for short-term traffic forecasting,” *Submitted to Journal of Sensor and Actuator Networks*.
- **Hongxin Shao**, Boon-Hee Soong, “Graph Diffusion Recurrent Neural Network for traffic forecasting,” *Submitted to Journal of Sensor and Actuator Networks*.

Conference Proceedings

- **Hongxin Shao**, Boon-Hee Soong, “Traffic flow prediction with Long Short-Term Memory Networks (LSTMs),” *2016 IEEE Region 10 Conference (TENCON)*, IEEE, 2016.

Bibliography

- [1] Ake Sjodin, Karin Persson, Kenth Andreasson, Bill Arlander, and Bo Galle. On-road emission factors derived from measurements in a traffic tunnel. *International Journal of Vehicle Design*, 20(1-4):147–158, 1998.
- [2] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.
- [3] Michel André and Ulf Hammarström. Driving speeds in europe for pollutant emissions estimation. *Transportation Research Part D: Transport and Environment*, 5(5):321–335, 2000.
- [4] Eva Ericsson, Hanna Larsson, and Karin Brundell-Freij. Optimizing route choice for lowest fuel consumption–potential effects of a new driver support tool. *Transportation Research Part C: Emerging Technologies*, 14(6):369–383, 2006.
- [5] Matthew Barth and Kanok Boriboonsomsin. Traffic congestion and greenhouse gases. 2009.
- [6] Matthew Barth and Kanok Boriboonsomsin. Real-world carbon dioxide impacts of traffic congestion. *Transportation Research Record*, 2058(1):163–171, 2008.
- [7] Kan Chen, John C Miles, et al. *ITS Handbook 2000: Recommendations from the World Road Association (PIARC)*. 1999.
- [8] Fei-Yue Wang. Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications. *IEEE Transactions on Intelligent Transportation Systems*, 3(11):630–638, 2010.

- [9] Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, and Cheng Chen. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, 2011.
- [10] Mohamed Amine Kafi, Yacine Challal, Djamel Djenouri, Messaoud Doudou, Abdelmadjid Bouabdallah, and Nadjib Badache. A study of wireless sensor networks for urban traffic monitoring: applications and architectures. *Procedia computer science*, 19:617–626, 2013.
- [11] Ondrej Karpis. Wireless sensor networks in intelligent transportation systems. *International Journal of Modern Engineering Research (IJMER)*, 3(2), 2013.
- [12] Hongxin Shao and Boon-Hee Soong. Traffic flow prediction with long short-term memory networks (lstm). In *2016 IEEE Region 10 Conference (TENCON)*, pages 2986–2989. IEEE, 2016.
- [13] Michael A Benjamin, Robert A Rigby, and D Mikis Stasinopoulos. Generalized autoregressive moving average models. *Journal of the American Statistical association*, 98(461):214–223, 2003.
- [14] Htrotugu Akaike. Maximum likelihood identification of gaussian autoregressive moving average models. *Biometrika*, 60(2):255–265, 1973.
- [15] Moshe Levin and Yen-Der Tsao. On forecasting freeway occupancies and volumes (abridgment). *Transportation Research Record*, (773), 1980.
- [16] Mohammad M Hamed, Hashem R Al-Masaeid, and Zahi M Bani Said. Short-term prediction of traffic volume in urban arterials. *Journal of Transportation Engineering*, 121(3):249–254, 1995.
- [17] Billy M Williams and Lester A Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.
- [18] Qing Yan Ding, Xi Fu Wang, Xiu Yuan Zhang, and Zhan Quan Sun. Forecasting traffic volume with space-time arima model. In *Advanced Materials Research*, volume 156, pages 979–983. Trans Tech Publ, 2011.

- [19] Chenyi Chen, Jianming Hu, Qiang Meng, and Yi Zhang. Short-time traffic flow prediction with arima-garch model. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 607–612. IEEE, 2011.
- [20] Billy M Williams. Multivariate vehicular traffic flow prediction: evaluation of arimax modeling. *Transportation Research Record*, 1776(1):194–200, 2001.
- [21] Iwao Okutani and Yorgos J Stephanedes. Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological*, 18(1):1–11, 1984.
- [22] Jianhua Guo, Wei Huang, and Billy M Williams. Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, 43: 50–64, 2014.
- [23] Yuanchang Xie, Yunlong Zhang, and Zhirui Ye. Short-term traffic volume forecasting using kalman filter with discrete wavelet decomposition. *Computer-Aided Civil and Infrastructure Engineering*, 22(5):326–334, 2007.
- [24] Chih-Chung Chang and Chih-Jen Lin. Training v-support vector regression: theory and algorithms. *Neural computation*, 14(8):1959–1977, 2002.
- [25] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.
- [26] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.
- [27] Marco Lippi, Matteo Bertini, and Paolo Frasconi. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):871–882, 2013.
- [28] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, 36(3):6164–6173, 2009.

- [29] Young-Seon Jeong, Young-Ji Byon, Manoel Mendonca Castro-Neto, and Said M Easa. Supervised weighting-online learning algorithm for short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1700–1707, 2013.
- [30] Xuexiang Jin, Yi Zhang, and Danya Yao. Simultaneously prediction of network traffic flow based on pca-svr. In *International Symposium on Neural Networks*, pages 1022–1031. Springer, 2007.
- [31] Wenbin Hu, Liping Yan, Kaizeng Liu, and Huan Wang. A short-term traffic flow forecasting method based on the hybrid pso-svr. *Neural Processing Letters*, 43(1):155–172, 2016.
- [32] Shiliang Sun, Changshui Zhang, and Guoqiang Yu. A bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems*, 7(1):124–132, 2006.
- [33] Sheng Jin, Dian-hai Wang, Cheng Xu, and Dong-fang Ma. Short-term traffic safety forecasting using gaussian mixture model and kalman filter. *Journal of Zhejiang University SCIENCE A*, 14(4):231–243, 2013.
- [34] Young-jin Kim, Ji-sun Hong, et al. Urban traffic flow prediction system using a multifactor pattern recognition model. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2744–2755, 2015.
- [35] Yudong Chen, Yi Zhang, and Jianming Hu. Multi-dimensional traffic flow time series analysis with self-organizing maps. *Tsinghua Science and Technology*, 13(2):220–228, 2008.
- [36] Gary A Davis and Nancy L Nihan. Nonparametric regression and short-term freeway traffic forecasting. *Journal of Transportation Engineering*, 117(2):178–188, 1991.
- [37] H Chang, Youngjoo Lee, B Yoon, and Sanghoon Baek. Dynamic near-term traffic flow prediction: system-oriented approach based on past experiences. *IET intelligent transport systems*, 6(3):292–305, 2012.
- [38] Mohamad H Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

- [39] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Eötvös Loránd University, Hungary*, 24:48, 2001.
- [40] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. *Transportation Research Part C: Emerging Technologies*, 13(3):211 – 234, 2005. ISSN 0968-090X. doi: <https://doi.org/10.1016/j.trc.2005.04.007>. URL <https://www.sciencedirect.com/science/article/pii/S0968090X05000276>.
- [41] Kit Yan Chan, Tharam S Dillon, Jaipal Singh, and Elizabeth Chang. Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and levenberg–marquardt algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):644–654, 2011.
- [42] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2014.
- [43] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.
- [44] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [45] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.
- [46] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, 2014.
- [47] Yuhan Jia, Jianping Wu, and Yiman Du. Traffic speed prediction using deep learning method. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1217–1222, Nov 2016. doi: 10.1109/ITSC.2016.7795712.

- [48] Arief Koesdwiady, Ridha Soua, and Fakhreddine Karray. Improving traffic flow prediction with weather information in connected cars: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 65(12):9508–9517, 2016.
- [49] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [50] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [51] Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. A recursive recurrent neural network for statistical machine translation. 2014.
- [52] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [53] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
- [54] Xiangdong Ran, Zhiguang Shan, Yufei Fang, and Chuang Lin. An lstm-based method with attention mechanism for travel time prediction. *Sensors*, 19(4): 861, 2019.
- [55] Huachun Tan, Yuankai Wu, Bin Shen, Peter J Jin, and Bin Ran. Short-term traffic prediction based on dynamic tensor completion. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2123–2133, 2016.
- [56] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, 2010.
- [57] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st*

- International Conference on Pattern Recognition (ICPR2012)*, pages 3304–3308. IEEE, 2012.
- [58] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [59] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [60] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.
- [61] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [62] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision*, pages 354–370. Springer, 2016.
- [63] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- [64] Yuankai Wu and Huachun Tan. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv preprint arXiv:1612.01022*, 2016.
- [65] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.
- [66] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

- [67] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.
- [68] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.
- [69] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [70] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640. AAAI Press, 2018.
- [71] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks for phone recognition. In *Nips workshop on deep learning for speech recognition and related applications*, volume 1, page 39. Vancouver, Canada, 2009.
- [72] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.
- [73] Seyed Hamidreza Mohammadi and Alexander Kain. Voice conversion using deep neural networks with speaker-independent pre-training. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 19–23. IEEE, 2014.
- [74] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.
- [75] Gary B Huang, Honglak Lee, and Erik Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2518–2525. IEEE, 2012.
- [76] Yan Xu, Tao Mo, Qiwei Feng, Peilin Zhong, Maode Lai, I Eric, and Chao Chang. Deep learning of feature representation with multiple instance learning for medical image analysis. In *2014 IEEE international conference on*

- acoustics, speech and signal processing (ICASSP)*, pages 1626–1630. IEEE, 2014.
- [77] A. Graves, N. Jaitly, and A. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Dec 2013. doi: 10.1109/ASRU.2013.6707742.
- [78] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013. doi: 10.1109/ICASSP.2013.6638947.
- [79] M. Sundermeyer, H. Ney, and R. Schlüter. From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529, March 2015. ISSN 2329-9290. doi: 10.1109/TASLP.2015.2400218.
- [80] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.
- [81] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE, 2011.
- [82] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.
- [83] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.
- [84] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [85] Chao Chen. Freeway performance measurement system (pems). 2003.

- [86] HV Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.
- [87] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*, 2017.
- [88] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.
- [89] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.
- [90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [91] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org, 2017.
- [92] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.
- [93] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017.
- [94] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusion-net: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*, 2017.

- [95] Kamal Al-Sabahi, Zhang Zuping, and Mohammed Nadher. A hierarchical structured self-attentive model for extractive document summarization (hssas). *IEEE Access*, 6:24205–24212, 2018.
- [96] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.
- [97] Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817, 2018.
- [98] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [99] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, 2018.
- [100] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [101] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*, 2018.
- [102] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [103] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [104] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

- [105] Yuankai Wu and Huachun Tan. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *CoRR*, abs/1612.01022, 2016.
- [106] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.
- [107] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.
- [108] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*, pages 6530–6539, 2017.
- [109] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *CoRR*, abs/1809.05679, 2018. URL <http://arxiv.org/abs/1809.05679>.
- [110] Guoshuai Zhao, Jun Li, Lu Wang, Xueming Qian, and Yun Fu. Graph-seq2seq: Graph-sequence-to-sequence for neural machine translation. 2018.
- [111] Daniel Neil, Joss Briody, Alix Lacoste, Aaron Sim, Paidi Creed, and Amir Saffari. Interpretable graph convolutional neural networks for inference on noisy knowledge graphs. *CoRR*, abs/1812.00279, 2018.
- [112] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357, 2018.
- [113] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. URL <http://arxiv.org/abs/1812.08434>.

- [114] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL <https://openreview.net/forum?id=SJU4ayYgl>.
- [115] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL <http://arxiv.org/abs/1511.05493>.
- [116] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.
- [117] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.
- [118] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.
- [119] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL <https://openreview.net/forum?id=SJiHXGWAZ>.
- [120] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL <https://www.aclweb.org/anthology/D14-1179>.
- [121] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In

Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden., pages 3634–3640, 2018. doi: 10.24963/ijcai.2018/505. URL <https://doi.org/10.24963/ijcai.2018/505>.

,