Traffic flow prediction with graph convolutional networks

MC Oosthuizen

orcid.org/0000-0003-2514-8135

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor:	Prof AJ Hoffman
Co-supervisor:	Prof MH Davel

Graduation:	June 2023
Student number:	28392310

Declaration

I, Marko Oosthuizen, hereby declare that the dissertation titled "Traffic flow prediction with graph convolutional networks" is my own original work and has not already been submitted to any other university or institution for examination.

M.C. Oosthuizen

Student number: 28392310

Signed on the 29th day of November 2022 at Potchefstroom.

Abstract

Traffic flow prediction is a complex task utilising both spatial and temporal information in order to predict the expected speed of traffic at different points in a transportation network. We study the use of graph convolutional networks for traffic speed prediction. Different models are investigated utilising internationally benchmarked data before the most promising approach (Graph WaveNet) is applied to a new database of South African road network data. This task is relevant, since traffic forecasting enables better road network management, reducing congestion. This in turn reduces travel time and pollution.

Different traffic forecasting models were considered before selecting Graph WaveNet as the best available traffic forecasting tool in terms of performance, availability of code, and training time. Graph WaveNet was first analysed using well-known datasets containing fixed sensor readings, while the South African datasets were being constructed. It was found that the models produced perform well when it comes to handling missing data and utilising the historic trend of data when making predictions. While Graph WaveNet performs well overall, we found that there is still room for improvement during times of congestion in road networks.

Graph WaveNet was successfully implemented on new Floating Car Data (FCD) (not fixed sensor data) recorded on South African road networks utilising only self-learned adjacency matrices. The South African models produced shared many similarities with the earlier models: the models successfully utilised the historic trend of the data, and performance degraded during periods of congestion in the road network. While the final error is not directly comparable, the new models achieved an MAE of 4.84, 4.99 and 5.10 over a 15-minute, 30-minute and 60-minute prediction horizon on the Johannesburg dataset, compared to an MAE of 2.69, 3.05 and 3.49 over the same prediction horizons on the METR-LA dataset.

Keywords: *traffic, prediction, congestion, graph convolutional network*

Contents

List of Figures **x**

List of Tables **xvi**

List of Acronyms **xxi**

1 Introduction **1**

1.1 Overview 1

1.2 Problem statement 2

1.3 Research scope 3

1.4 Aims and objectives 3

1.5 Research methodology 4

1.6 Design 4

1.7 Dissertation overview 5

1.8 Publications 6

2 Background **7**

2.1 Overview 7

2.2 Traffic prediction 7

2.2.1 Complexities in traffic prediction 8

2.2.2 Temporal dependency representation 9

2.2.3 Spatial dependency representation . . . 10
2.3 Traffic modelling approaches . . . 10
2.3.1 Temporal dependency modelling . . . 10
2.3.2 Spatial dependency modelling . . . 11
2.3.3 Spatiotemporal modelling . . . 12
2.4 Training . . . 13
2.4.1 Supervised learning . . . 13
2.4.2 Stochastic gradient descent . . . 14
2.4.3 Hyperparameter tuning techniques . . . 15
2.5 Traffic datasets . . . 15
2.6 Conclusion . . . 16

3 Data analysis **17**
3.1 Overview . . . 17
3.2 Dataset overview and characteristics . . . 18
3.2.1 Zero values present in datasets . . . 19
3.2.2 Traffic network node connections . . . 19
3.2.3 Average daily and weekly traffic speed analysis . . . 22
3.3 Conclusion . . . 27

4 System selection **28**
4.1 Overview . . . 28
4.2 System introduction . . . 28
4.2.1 STTN . . . 29
4.2.2 Graph WaveNet . . . 31
4.2.3 STAWnet . . . 32

4.2.4 GMAN . . . 33
4.2.5 STNN . . . 35
4.3 Tool comparison . . . 37
4.3.1 Published performance . . . 37
4.3.2 Availability of tools . . . 39
4.3.3 Recreation of published results . . . 39
4.3.4 Training time comparison . . . 42
4.3.5 STNN data partitioning and performance calculation . . . 42
4.4 Conclusion . . . 43

5 Graph WaveNet **44**
5.1 Overview . . . 44
5.2 Graph WaveNet convolution elements . . . 44
5.2.1 1×1 convolutional layer . . . 45
5.2.2 Causal convolution . . . 45
5.2.3 1-D Convolution . . . 45
5.2.4 2-D Convolution . . . 46
5.2.5 Dilated convolution . . . 46
5.3 Graph Convolutional Network . . . 46
5.3.1 Feature propagation . . . 47
5.3.2 Feature transformation and nonlinear transition . . . 48
5.4 Graph WaveNet . . . 48
5.4.1 Problem definition . . . 48
5.4.2 Model description . . . 49
5.5 Available improvements to Graph WaveNet . . . 53
5.6 Conclusion . . . 55

6 Experimental protocol **56**

6.1 Overview . . . 56

6.2 Performance metrics . . . 56

6.3 Validation loss calculation corrections . . . 57

6.3.1 Batch problem 1 . . . 57

6.3.2 Batch problem 2 . . . 58

6.4 Hyperparameter optimisation protocol evaluation . . . 59

6.4.1 Baseline implementation . . . 59

6.4.2 Baseline implementation: reduced parameters . . . 61

6.4.3 Baseline implementation: increased epochs . . . 63

6.4.4 Dynamically extended training . . . 64

6.4.5 Dynamically extended training: reduced parameters . . . 66

6.4.6 Results comparison . . . 67

6.5 Missing data . . . 69

6.6 Conclusion . . . 69

7 Impact of congestion on traffic speed prediction **72**

7.1 Overview . . . 72

7.2 Binned traffic speed analysis . . . 73

7.3 Congestion analysis . . . 76

7.3.1 Thresholds . . . 77

7.3.2 Per time step analysis . . . 78

7.3.3 Recurrent congestion analysis all days . . . 80

7.3.4 Recurrent congestion analysis workdays . . . 83

7.4 Congested data boosting . . . 85

7.4.1 Experiment design . . . 85

7.4.2 Results . . . 85
7.5 Congested data only . . . 90
7.5.1 Experiment design . . . 90
7.5.2 Results . . . 90
7.6 Conclusion . . . 94

8 Additional analysis **95**
8.1 Overview . . . 95
8.2 Trend analysis . . . 95
8.2.1 Discussion . . . 98
8.3 Individual kernels . . . 98
8.3.1 Experiment design . . . 99
8.3.2 Results . . . 99
8.4 Encoding time as cyclical continuous feature . . . 102
8.4.1 Experiment design . . . 103
8.4.2 Results . . . 103
8.5 Conclusion . . . 105

9 Application to SA data **106**
9.1 Overview . . . 106
9.2 INRIX data . . . 106
9.2.1 Road segments . . . 107
9.2.2 Data sampling . . . 107
9.2.3 South African datasets . . . 108
9.3 Data analysis . . . 108
9.3.1 Missing data . . . 109

9.3.2 Average speed . . . 109
9.4 South African data models . . . 113
9.4.1 Experiment design . . . 113
9.4.2 Results . . . 114
9.5 SA data models performance analysis . . . 118
9.5.1 Binned traffic speed analysis . . . 118
9.5.2 Trend analysis . . . 120
9.6 Conclusion . . . 125

10 Conclusion **126**

10.1 Overview . . . 126
10.2 Key findings . . . 126
10.3 Outcomes . . . 129
10.4 Future work . . . 129
10.5 Conclusion . . . 130
References . . . 131

A Supplemental content **135**

A.1 Appendix: Chapter 7 . . . 135
A.2 Appendix: Chapter 9 . . . 138
A.3 Appendix: Publication . . . 142

List of Figures

2.1 Illustration of spatial correlation in a traffic network. Road segments 1 and 2 are correlated while road segments 1 and 3 are not, illustrating the non-Euclidean nature of a traffic network. Adapted from [7]. 9

3.1 METR-LA and PEMS-BAY sensor positions. Reproduced from [7] 18

3.2 All road-wise connections between nodes in the Los Angeles Metropolitan Transportation Authority (METR-LA) dataset. 20

3.3 All road-wise connections between nodes in the PEMS-BAY dataset. . . . 20

3.4 Reduced road-wise connections between nodes in the METR-LA dataset. These are the connections used by Graph WaveNet and DCRNN. The connections are reduced based on the threshold used for the thresholded Gaussian kernel ($k = 0.1$). 21

3.5 Reduced road-wise connections between nodes in the PEMS-BAY dataset. These are the connections used by Graph WaveNet and DCRNN. The connections are reduced based on the threshold used for the thresholded Gaussian kernel ($k = 0.1$). 21

3.6 Boxplot of the average network speed per day for the METR-LA dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right. 23

3.7 Boxplot of the average network speed per day for the PEMS-BAY dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right. 24

3.8 Boxplot of the average network speed per hour of day for the METR-LA dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right. 24

3.9 Boxplot of the average network speed per hour of day for the PEMS-BAY dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right. 25

3.10 METR-LA average network speed variance per hour for each day of the week (Training and validation data partitions only). 25

3.11 PEMS-BAY average network speed variance per hour for each day of the week (Training and validation data partitions only). 26

4.1 Spatial-Temporal Transformer Network (STTN) overall architecture. This consists of an input layer, k spatiotemporal hidden layers, and an output layer which implements two 1×1 convolutional layers. 30

4.2 Framework of Graph WaveNet. This consists of an input layer, k spatiotemporal hidden layers, and an output layer. 32

4.3 Framework of STAWnet. Each spatiotemporal block consists of a gated Temporal Convolutional Network (TCN) and a dynamic attention network (DAN). 33

4.4 Framework of Graph Multi-Attention Network (GMAN). This consists of a spatiotemporal embedding, an encoder and a decoder each with k spatiotemporal attention blocks, and two fully connected layers. 34

4.5 Framework of Spacetime Neural Network (STNN). This consists of k spacetime modules and a fully connected output layer. Each spacetime module contains a spacetime attention block and a spacetime convolution block. . . 36

5.1 Framework of Graph WaveNet. d=input dimensions, n=number of nodes, c=the 'nhid' hyperparameter. 50

6.1 Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs. Each data point represents the best validation loss achieved for the run. The orange line shows how the best validation loss improves as more runs are completed. 60

6.2 Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs. Each data point represents the epoch on which the best validation loss was achieved for the run. 60

6.3 Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs with the number of possible parameters reduced. Each data point represents the best validation loss achieved for the run. The orange line shows how the best validation loss improves as more runs are completed. 62

6.4 Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs with the number of possible parameters reduced. Each data point represents the epoch on which the best validation loss was achieved for the run. 62

6.5 Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 100 epochs. Each data point represents the best validation loss achieved for the run. The orange line shows how the best validation loss improves as more runs are completed. 63

6.6 Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 100 epochs. Each data point represents the epoch on which the best validation loss was achieved for the run. 64

6.7 Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training on METR-LA dataset. Each data point represents the best validation loss achieved for the run. The orange line how shows the best validation loss improves as more runs are completed. . 65

6.8 Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training on METR-LA dataset. Each data point represents the epoch on which the best validation loss was achieved for the run. 65

6.9 Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training and a reduced number of parameter values to search over on METR-LA dataset. Each data point represents the best validation loss achieved for the run. The orange line shows the how best validation loss improves as more runs are completed. 66

6.10 Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training and a reduced number of parameter values to search over on METR-LA dataset. Each data point represents the epoch on which the best validation loss was achieved for the run. 67

7.1 Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon, binned per average network speed: METR-LA, validation data partition. 74

7.2 Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon, binned per average network speed: PeMS-Bay, validation data partition. 74

7.3 Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon, binned per sensor speeds for METR-LA, validation data partition. 75

7.4 Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon binned per sensor speed for PEMS-BAY, validation data partition. 75

7.5 METR-LA training set average network speed per time of day and day of the week. 76

7.6 Histogram of METR-LA training partition speed readings of all sensors. . . 78

7.7 Histogram of METR-LA validation partition average network speed readings of all sensors. 79

7.8 Graph WaveNet per time step congestion analysis: METR-LA, validation set. Mean results over all 12 prediction horizons. 79

7.9 Graph WaveNet per time-step congestion analysis: METR-LA, validation set. Results for the 12th prediction horizon. 80

7.10 Histogram of METR-LA validation partition average daily network speed readings of all sensors 82

7.11 Graph WaveNet recurrent congestion analysis: METR-LA, validation set. Mean results over all 12 prediction horizons. 82

7.12 Graph WaveNet recurrent congestion analysis: METR-LA, validation set. Results for the 60-minute prediction horizons. 83

7.13 Graph WaveNet workday recurrent congestion analysis: METR-LA, validation set. Mean results over all 12 prediction horizons. 84

7.14 Graph WaveNet workday recurrent congestion analysis: METR-LA, validation set. Results for the 60-minute prediction horizons. 84

7.15 Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA training data partition. 88

7.16 Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA validation data partition. 89

7.17 Mean performance of Graph WaveNet optimised on congested data only from the training data partition for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA training data partition. 92

7.18 Mean performance of Graph WaveNet optimised on congested data only from the training data partition for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA validation data partition. 93

8.1 Trend analysis for sensor 407339 in the PEMS-BAY dataset. Sensor 407339 is the sensor for which prediction errors are the largest. Predicted and true traffic speed values are based on the PEMS-BAY validation data partition while the trend is calculated using the training data partition. 96

8.2 Trend analysis on sensor 400073 in the PEMS-BAY dataset. Sensor 400073 is randomly selected. Predicted and true traffic speed values are based on the PEMS-BAY validation data partition while the trend is calculated using the training data partition. 97

8.3 Architecture of the Graph WaveNet model with an additional individual kernels layer inserted. The TCN used in the individual kernels block applies a unique kernel to the input from each of the individual kernels in a traffic network. 100

8.4 Encoding time in a cyclical way. 102

9.1 Boxplot of the average network speed per day for the JHB dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right. 110

9.2 Boxplot of the average network speed per day for the CPT dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right. 110

9.3 Boxplot of the average network speed per hour for the JHB dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right. 111

9.4 Boxplot of the average network speed per hour for the CPT dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right. 112

9.5 JHB average network speed variance per hour for each day of the week (Training and validation data partitions only). 112

9.6 CPT average network speed variance per hour for each day of the week (Training and validation data partitions only). 113

9.7 Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per sensor speed: JHB, validation data partition. 119

9.8 Trend analysis for the sensor in the JHB dataset for which model 1 trained in Section 9.4 performs the worst. Predicted and true traffic speed values are based on the JHB validation data partition while the trend is calculated using the training data partition. 121

9.9 Trend analysis for the sensor in the JHB dataset for which model 1 trained in Section 9.4 performs the best. Predicted and true traffic speed values are based on the JHB validation data partition while the trend is calculated using the training data partition. 122

9.10 Trend analysis for the sensor in the CPT dataset for which model 1 trained in Section 9.4 performs the worst. Predicted and true traffic speed values are based on the CPT validation data partition while the trend is calculated using the training data partition. 123

9.11 Trend analysis for the sensor in the CPT dataset for which model 1 trained in Section 9.4 performs the best. Predicted and true traffic speed values are based on the CPT validation data partition while the trend is calculated using the training data partition. 124

A.1 Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per sensor speed: CPT, validation data partition. 139

A.2 Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per average network speed: JHB, validation data partition. 140

A.3 Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per average network speed: CPT, validation data partition. 141

List of Tables

3.1 Characteristics of the PEMS-BAY and METR-LA datasets 17

4.1 Original dataset partitioning reported on in the literature. Partitioning is done in chronological order: training, validation, testing. 38

4.2 Performance of top 5 models, based on METR-LA dataset leader board, for which code is available on the METR-LA dataset when predictions are made 60 minutes into the future. 38

4.3 Performance of top 5 models, based on PEMS-BAY dataset leader board, for which code is available on the PEMS-BAY dataset when predictions are made 60 minutes into the future. 38

4.4 Published performance of selected models on the METR-LA dataset test partition. MAE and RMSE given in miles per hour. 39

4.5 Published performance of selected models on the PEMS-BAY dataset test partition. MAE and RMSE given in miles per hour. 39

4.6 Hyperparameters used when recreating the published results of the evaluated tools 40

4.7 Recreated results for 15-minute, 30-minute, and 60-minute predictions of tools on METR-LA test data partition. MAE and RMSE given in miles per hour. 40

4.8 Recreated results for 15-minute, 30-minute, and 60-minute predictions of tools on PEMS-BAY test data partition. MAE and RMSE given in miles per hour. 40

4.9 Difference between published and recreated results for the METR-LA dataset (published − recreated). Published results are given in Table 4.4 and recreated results in Table 4.7. MAE and RMSE given in miles per hour. 41

4.10 Difference between published and recreated results for the PEMS-BAY dataset (published − recreated). Published results are given in Table 4.8 and recreated results in Table 4.8. MAE and RMSE given in miles per hour. 41

4.11 Training time comparison of implemented tools 42

4.12 STNN recreated results for the sake of comparability with results in Tables 4.7 and 4.8 respectively. Data partitioning is done to match the other models, and performance calculations are done in the same way as the rest of the models. MAE and RMSE given in miles per hour. 43

5.1 Dimensions of output from each TCN block to skip connection. 53

6.1 Hyperparameter optimisation parameters 61

6.2 Hyperparameter optimisation parameters reduced 61

6.3 Hyperparameter optimisation approaches results and compute time comparison. MAE and RMSE given in miles per hour. 68

6.4 Performance achieved with Graph WaveNet when hyperparameter optimisation is done compared to the performance achieved by the model trained using published hyperparameters from Section 4.3.3. Results are given for the METR-LA validation and training data partitions. MAE and RMSE given in miles per hour. 68

6.5 Performance of Graph WaveNet models trained on data for which missing values are interpolated to reduce training difficulty (the mean performance of the three models is also given). Results are given for METR-LA training and validation data partitions. The performance of the original vanilla Graph WaveNet model is also given. Original model: OM. MAE and RMSE given in miles per hour. 70

6.6 Mean performance for all twelve prediction horizons of Graph WaveNet models trained on data for which missing values are interpolated to reduce training difficulty to Graph WaveNet (the mean performance of the three models is also given). Results are given for METR-LA training and validation data partitions. The performance of the original vanilla Graph WaveNet model is also given. Original model: OM. MAE and RMSE given in miles per hour. 70

7.1 Performance of Graph WaveNet when evaluated on recurrent congestion during workdays. Results are given for the METR-LA training partition. MAE is given for mean performance on all twelve prediction horizons and for the 60-minute prediction horizon. 86

7.2 Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds and recreated Graph WaveNet performance. Results are given for the METR-LA training and validation data partitions. MAE and RMSE given in miles per hour. 87

7.3 Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds and recreated Graph WaveNet performance. Results are given for congested data only extracted from the METR-LA training and validation data partitions using a fixed daily threshold for work days as described in Section 7.4. MAE and RMSE given in miles per hour. 87

7.4 Performance of Graph WaveNet optimised only on congested data from the training data partition for three different seeds (the mean performance of the three models is also given) along with recreated Graph WaveNet performance. Results are given for the METR-LA training and validation data partitions. MAE and RMSE given in miles per hour. 91

7.5 Performance of Graph WaveNet optimised on congested data only from the training data partition for three different seeds (the mean performance of the three models is also given) along with recreated Graph WaveNet performance. Results are given for congested data from the METR-LA training and validation data partitions. The congested data used is described in Section 7.5. MAE and RMSE given in miles per hour. 91

8.1 Minimum, maximum, mean, and variance of MAE when considering Graph WaveNet model performance on individual sensors in the PEMS-BAY en METR-LA traffic networks using the validation data partition from each of these datasets. MAE given in miles per hour. 98

8.2 Performance of individual kernels models for 15 minutes, 30 minutes, and 60 minutes predictions on METR-LA validation and training partitions along with mean performance of three individual kernels models and original Graph WaveNet model performance. Original model: OM. MAE and RMSE given in miles per hour. 100

8.3 Mean performance of individual kernels models for 5 minutes to 60 minutes predictions on METR-LA validation and training partitions along with mean performance of three individual kernels models and original Graph WaveNet model performance. Original model: OM. MAE and RMSE given in miles per hour. 101

8.4 Minimum, maximum, mean, and variance of MAE for individual sensors in the METR-LA valisation data partition. Results are given for the individual kernels models and original Graph WaveNet model. Original model: OM, individual kernels model: IK. MAE given in miles per hour. 101

8.5 Graph WaveNet models trained using time encoded as cyclical feature performance on METR-LA validation and training data partitions. The performance of the original vanilla Graph WaveNet model is given for reference. Original model: OM. MAE and RMSE given in miles per hour. . . . 104

8.6 Graph WaveNet models trained using time encoded as cyclical feature mean performance for 12 prediction horizons on METR-LA validation and training data partitions. The performance of the original vanilla Graph WaveNet model is given for reference. Original model: OM. MAE and RMSE given in miles per hour. 104

9.1 Basic characteristics of the JHB and CPT datasets constructed using car vehicle data from INRIX. 108

9.2 Graph WaveNet performance on training and validation data partitions of JHB dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour. 114

9.3 Graph WaveNet mean performance on training and validation data partitions of JHB dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour. 115

9.4 Graph WaveNet performance on training and validation data partitions of CPT dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour. 115

9.5 Graph WaveNet mean performance on training and validation data partitions of CPT dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour. 115

9.6 Graph WaveNet performance on the testing data partition of JHB dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour. . . . 116

9.7 Graph WaveNet performance on the testing data partition of CPT dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour. . . . 117

A.1 Performance of each model trained on congested boosted data in Section 7.4. The MAE is given for each prediction horizon on congested data only and not congested data data. Congested data is extracted as described in Section 7.4. Results are given for the METR-LA validation data partition. 137

A.2 Performance of each model trained on congested data only in Section 7.5. The MAE is given for each prediction horizon on congested data only and not congested data. Congested data is extracted as described in Section 7.5. Results are given for the METR-LA validation data partition. 137

A.3 ANOVA analyses and Welch's t-test performed on normalised data from Tables A.1 and A.2 results. 138

List of Acronyms

ANOVA analysis of variance

API Application Programming Interface

CalTrans California Transportation Agencies

CNN Convolutional Neural Network

DCRNN Diffusion Convolutional Recurrent Neural Network

DNN Deep Neural Network

FCD Floating Car Data

GCN Graph Convolutional Network

GMAN Graph Multi-Attention Network

GNN Graph Neural Network

GPS Global Positioning System

GRU Gated Recurrent Unit

ITS Intelligent Transportation Systems

LSTM Long-Short Term Memory

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

METR-LA Los Angeles Metropolitan Transportation Authority

MLP multilayer perceptron

PeMS Performance Measurement System

ReLU Rectified Linear Unit

RMSE Root Mean Square Error

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SOTA state-of-the-art

STAWnet Spatio-Temporal Attention Wavenet

STNN Spacetime Neural Network

STTN Spatial-Temporal Transformer Network

TCN Temporal Convolutional Network

TISA Traveller Information Services Association

TMC traffic message channel

Chapter 1

Introduction

1.1 Overview

The ability to predict future traffic flow from prior sensor data is one of the key elements of an intelligent transportation system. Models that perform well for traffic flow prediction are able to capture both spatial dependencies (the interconnectedness of roads) and the time-dependent nature of the task (traffic information before and after an event). Various new graph-convolutional models for traffic flow prediction have recently emerged, producing state-of-the-art (SOTA) results.

In this study, we investigate the problem of implementing a traffic prediction method on a South African road network. At the start of the study, there were no publicly available traffic prediction systems or publicly available datasets for South African road networks. Thus, in parallel with this project, South African road network traffic data was recorded by Schalk Rabe a fellow Masters degree student.

Traffic congestion is a problem in cities resulting in lost time, reduced productivity, increased pollution and wasted energy. Traffic congestion is caused by the rapid growth of the vehicle population in cities, resulting in a constantly increasing burden on the transportation infrastructure [1]. To combat traffic congestion, intelligent transportation

systems are used to improve operational efficiency and increase the capacity of transportation systems [2]. An intelligent transportation system consists of advanced sensing and communication techniques, information processing and traffic management technologies that can analyse data collected from diverse sources for informed decision-making [1].

A fundamental part of intelligent transportation systems is traffic state prediction. This involves directly determining the state of traffic at a future time when given continuous feedback of traffic data [2]. Congestion alleviation through traffic prediction is much more affordable than constructing new roads. Other congestion mitigation measures such as restricting the number of private vehicles allowed on the road network are infeasible in countries with poor public transportation infrastructure [3]. However, traffic prediction is much more complicated than conventional time series analysis since it is subject to external spatial factors [3]. Traffic on one road in a network influences the traffic on a different road in the same network and the entire road network is affected by weather conditions. Fortunately, the predictive power of deep neural networks (DNNs) enables the modelling of complex non-linear traffic patterns in a road network and has thus become the most prominent traffic prediction method [3].

1.2 Problem statement

South African road networks are affected by congestion due to population growth and poor road maintenance. To alleviate congestion on South African road networks in a cost-effective manner, traffic prediction techniques can be used to allow informed traffic management decisions to be made. However, at the start of this study, no publicly available datasets or traffic flow prediction systems were available specifically for South African road networks.

1.3 Research scope

The initial scope of the investigation is limited in terms of the following:

- Only methods with available codebases will be implemented. We further limit methods to ones that implement a form of graph convolutional operation since graph convolutional models were the state of the art at the start of this study as could be seen on available leaderboards.
- Datasets used for experimentation should have publicly available leaderboards to ensure the use of SOTA methods. For this reason, we restrict datasets to PEMS-BAY and METR-LA as discussed in Section 2.5.
- South African datasets are recorded only from national principal arterial roads and provincial arterial roads to make the datasets comparable to the METR-LA and PEMS-BAY datasets which consist only of highway data. This is discussed in Chapter 9.

1.4 Aims and objectives

The main aim of this study is to investigate the ability of SOTA traffic flow prediction methods to perform well in a South African road network context. To achieve this aim the following research questions are identified:

1. What are the best available traffic prediction methods?
2. Which of these traffic prediction methods performs best and can be re-implemented reliably?
3. For these methods, what is the effect of congestion on prediction capabilities?
4. Can the performance of the best available method be improved?

5. Do the characteristics of South African road network data differ from those of internationally available traffic datasets?

6. How well does the chosen model perform when implemented on South African road networks?

7. Is a traffic dataset collected from Floating Car Data (FCD) suitable for use in the selected traffic prediction method?

1.5 Research methodology

An exploratory research study is proposed in which existing traffic prediction techniques will be implemented. This will further the understanding of traffic flow prediction methods. The results of this study will be evaluated quantitatively.

1.6 Design

Specifically, the following will form part of the study:

1. Literature review: Gain a deeper understanding of spatial and temporal correlation modelling techniques used in traffic prediction and the complexities involved. Give the background needed for understanding deep learning models and select benchmark datasets to use for experimentation.

2. Dataset preparation: Analyse and understand the structure of the selected benchmark datasets while waiting for South African datasets to become available.

3. Software implementation: Recreate published results for best performing models on chosen datasets and select a model for further experimentation.

4. Model analysis: Perform an in-depth analysis of the chosen model.

5. Experimental setup: Determine the experimental protocol to use when optimising and training traffic prediction models.

6. Congestion analysis: Analyse model performance during traffic congestion. Traffic prediction is of utmost importance during times of congestion, as little action is required from either road users or the traffic management system when traffic is flowing normally.

7. Experimentation: Explore possible improvements to the chosen model.

8. South African road network implementation: Apply the chosen and analysed traffic prediction tool using FCD collected from the South African road networks and analyse the results.

1.7 Dissertation overview

The structure of this dissertation is as follows:

- Chapter 2: provides the background necessary for the rest of the dissertation and discusses various approaches to traffic prediction.
- Chapter 3: analyses datasets used for experimentation.
- Chapter 4: describes and implements available traffic prediction tools and selects a tool for further experimentation.
- Chapter 5: describes the chosen traffic prediction tool in depth.
- Chapter 6: the experimental protocol used for experimentation is chosen.
- Chapter 7: analyses model performance during traffic congestion and experimentally explores possible model performance improvements during periods of recurrent congestion in a road network.

- Chapter 8: experimentally explores possible improvements to model performance by tweaking model architecture and input data representation.
- Chapter 9: using the knowledge acquired up to this point, implements a traffic prediction tool on South African road networks and analyses the results.
- Chapter 10: concludes the study, summarises the key findings and suggests future work.

1.8 Publications

As part of this study, the following publication was produced: "A Comparative Study of Graph Neural Network Speed Prediction during Periods of Congestion" [4], included as Appendix A.3. There is an overlap between this publication and some of the material included in Chapter 7.

The publication was developed as part of this study and produced according to the following timeline:

- Jan 2021: Study started.
- Jan 2022: Research for Chapter 7 initiated.
- Aug 2022: Selected content from Chapter 7 submitted to the International Conference on Neural Computation Theory and Applications 2022.
- Oct 2022: Paper published.
- Nov 2022: Dissertation completed.

Chapter 2

Background

2.1 Overview

In this chapter, traffic prediction is introduced along with the complexities involved and methods for representing temporal and spatial dependencies. Temporal, spatial, and spatiotemporal modelling approaches used in traffic prediction are discussed. Essential concepts regarding training deep learning models are introduced followed by an overview of datasets commonly used for traffic prediction.

2.2 Traffic prediction

One of the most prominent issues in Intelligent Transportation Systems (ITS) is traffic pattern prediction [5]. Traffic pattern prediction is the process of analysing traffic data, such as speed and density data, of a road network, extracting traffic patterns, and predicting future traffic conditions. The traffic prediction problem can be defined as learning the parameters of a function that takes as input historical traffic data and returns the future traffic values for the road network [3].

Traffic flow prediction is one of the most frequent use cases of deep learning in ITS [6].

Predictions are usually made at fixed intervals into the future, such as at 5-minute to 60-minute intervals. Recent approaches in this domain focus on capturing spatial, temporal and external patterns [6].

The first traffic prediction models used were classical statistical models such as the Autoregressive Integrated Moving Average family of models [3]. However, statistical models are relatively weak regarding traffic prediction since they assume traffic data is stationary, (the mean, variance and autocorrelation stay unchanged over time) failing to handle complex traffic data [3]. Deep Neural Network (DNN) models are flexible, can learn from the data, and certain DNNs can explicitly capture different aspects of traffic data, making them the go-to techniques for traffic prediction tasks since DNNs with the right architecture are good at capturing both non-stationary and non-linear aspects of traffic data.

2.2.1 Complexities in traffic prediction

Predicting the future traffic speed of a traffic network is challenging due to complex spatial and temporal dependencies. Non-stationarity of traffic data is caused by traffic incidents such as rush hours and traffic accidents [7]. Also, nodes in the traffic network (the locations where traffic data are recorded) have unique and complex correlations [7]. Figure 2.1 gives an example of these complex correlations. Road segments 1 and 2 have a solid correlation while road segments 1 and 3 do not, even though road segments 1 and 3 are close in the Euclidean space. We can also observe that the future traffic speed of this specific instance is affected more by downstream traffic than by upstream traffic by evaluating the traffic speed plots of road segments 1 and 2 [7]. This illustrates the complex spatial structure of traffic: it is possible that two parallel roads in different directions can be spatially close and have little or no effect on one another [7].

Traffic predictions are made based on these complex spatial and temporal dependencies using the spatiotemporal information of a traffic network. Spatiotemporal data relates to both space and time. In terms of traffic prediction, spatiotemporal data can consist of

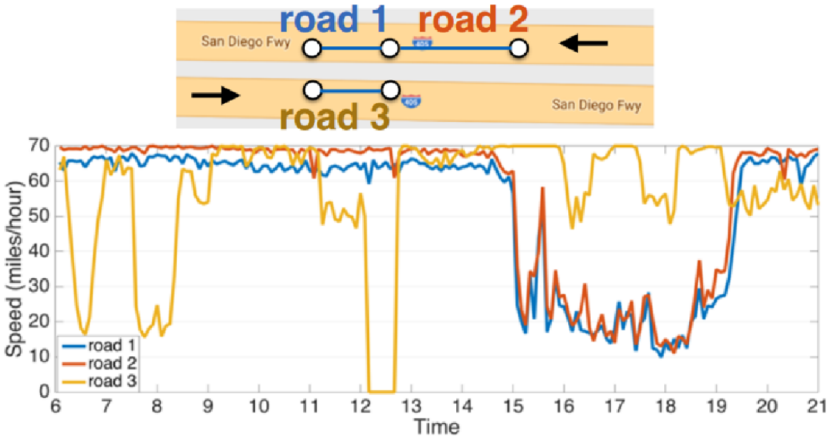


Figure 2.1 – Illustration of spatial correlation in a traffic network. Road segments 1 and 2 are correlated while road segments 1 and 3 are not, illustrating the non-Euclidean nature of a traffic network. Adapted from [7].

links, road segments, regions, and weather conditions [8]. External information such as weather conditions are represented only as independent features in most studies and are thus not mentioned further [8].

A model captures spatial or temporal aspects when it or its sub-components are specifically designed for capturing spatial and/or temporal dependencies, or when the data is modelled so that it includes the spatial and/or temporal information [3].

2.2.2 Temporal dependency representation

Based on the survey done by Lee et al. [8], there are only two input representation types for representing temporal dependencies: sequentiality and periodicity. Sequentiality refers to presenting temporal data by stacking consecutive temporal units into a single vector [8], with recent time slots more relevant than distant time slots [5]. Temporal periodicity represents the hourly, daily or weekly change patterns in the data that are extracted [8].

2.2.3 Spatial dependency representation

Traffic flow is greatly affected by the topological structure of the underlying road network [5]. The traffic flow on a road impacts its adjacent roads due to directional spatial correlations [5]. Spatial dependencies can be represented via stacked vectors, grid representations, and graphs [8]. A stacked vector representing spatial dependency is constructed by stacking the data of multiple spatial units into a single vector according to predefined rules based on domain knowledge or preference [8]. Grid representations present spatial information as 2-dimensional Euclidean data that can be used in deep networks as image-like representations [8]. However, spatial structure in traffic is non-Euclidean and directional [7], which is why graph representations of traffic networks have become popular. Graph representation has been widely adopted for spatial dependency modelling since any spatial unit can be represented as a graph node. Graphs can express intricate relationships between nodes and are thus a good representation method for traffic networks that possibly have complex latent dependencies [8].

2.3 Traffic modelling approaches

Here an overview is given of temporal and spatial dependency modelling approaches used by recent traffic prediction methods. Temporal and spatial dependencies can either be modelled using separate approaches described in Sections 2.3.1 and 2.3.2 respectively or at the same time, as described in Section 2.3.3.

2.3.1 Temporal dependency modelling

Older traffic prediction models such as Diffusion Convolutional Recurrent Neural Network (DCRNN) [7] implement Recurrent Neural Networks (RNNs) for temporal dependency modelling. The most effective RNNs used are called gated RNNs [9]. Commonly used gated RNNs are Long-Short Term Memory (LSTM) networks [10] and Gated Recurrent

Unit (GRU) networks [11] [9].

RNN-based approaches used for capturing temporal dependencies suffer from time-consuming iterative propagation for capturing long sequences. Convolutional Neural Network (CNN)-based approaches that give the advantage of parallel computing when used with Graph Convolutional Networks (GCNs) [12] require the use of many layers to capture long sequences since they adopt standard 1D convolution [13]. One way to increase effectiveness and efficiency is to use dilated causal convolutions to implement what is known as a Temporal Convolutional Network (TCN) [14] as done by Wu et al. [13] for the Graph WaveNet model. Another way to increase effectiveness is to use transformers. Transformers are highly parallelisable due to the self-attention mechanism, and long-range dependencies can be captured from input sequences with varying lengths using just one layer [15]. This is the reason for Xu et al. [15] implementing a temporal transformer in the Spatial-Temporal Transformer Network (STTN).

2.3.2 Spatial dependency modelling

To capture the non-linearity of traffic patterns, neural networks are incorporated; nevertheless, their fully connected architectures are costly both computationally and regarding memory [15]. The complex geographical patterns in traffic flows cannot be captured efficiently due to the absence of assumptions regarding the topological information of a traffic network [15].

Given their potent feature extraction capabilities in numerous applications, CNNs have been used for traffic prediction since their introduction [15]. Spatial characteristics are extracted using CNNs by transforming traffic networks into regular grids [15]. However, the fundamental topological information that characterises irregular traffic networks is lost as a result of the grid conversion [15].

To extend deep learning to non-Euclidean domains, Graph Neural Networks (GNNs) are developed. GCNs are a variation of GNNs that extend traditional convolutions to the graph domain [12]. GCNs are now frequently used to simulate the geographical rela-

tionships of traffic flows and to investigate the underlying traffic topology [15]. DCRNN adapts to traffic flow directions by using diffusion graph convolutions on a directed graph for spatial dependency modelling [7]. But, it is assumed that the graph structure reflects the true dependency relationship amongst the nodes, while connected nodes can have a weak inter-dependency relationship and unconnected nodes can have a stronger inter-dependency relationship [13]. To overcome this, Graph WaveNet implements a learnable embedding for each node in the graph to increase the accuracy of traffic predictions with hidden spatial patterns [13]. But the spatial dependencies learned by Graph WaveNet are fixed once trained. Instead of using predetermined graph structures and local nodes, STTN effectively models dynamically directed spatial relationships in high-dimensional latent subspaces [15].

2.3.3 Spatiotemporal modelling

Most traffic prediction models make use of two separate techniques for spatiotemporal modelling. Temporal modelling techniques and spatial modelling techniques are described in Sections 2.3.1 and 2.3.2 respectively. DCRNN, for instance, implements an RNN for temporal modelling and a GCN for spatial modelling while Graph WaveNet implements a TCN for temporal modelling and a GCN for spatial modelling.

Certain limitations exist in GNN-based traffic prediction models commonly used [16].

(i) These models place a significant amount of reliance on the graph topology, which differs across various road networks. As a result, rather than displaying the innate characteristics of the transportation system, they are constrained to a certain road network [16].

(ii) Each of these models consists of distinct components that separately collect characteristics from the spatial and temporal dimensions before combining them into a single feature map to produce a prediction. But, this configuration implicitly presupposes that locational correlations remain constant throughout time. However, different correlations may exist between two sites at different periods [16].

To attempt to overcome these limitations the spatiotemporal correlation learning paradigm, called Spacetime Interval Learning is implemented by Yang et al. [16]. The paradigm fuses spatial and temporal dimensions into a single manifold and attempts to capture correlations that reflect the distance between two traffic events in spacetime [16]. To capture these correlations, the Spacetime Neural Network (STNN) is created, which combines spacetime attention blocks and spacetime convolution blocks.

STTN, Graph WaveNet, STAWnet, and STNN are described in more detail in Sections 4.2.1, 4.2.2, 4.2.3, and 4.2.5 respectively.

2.4 Training

Building algorithms that rely on a set of examples of a phenomenon in order to be useful is the focus of the computer science subfield of machine learning [9]. Machine learning can also be defined as the process of collecting data to build a dataset, and then building a statistical model based on the dataset to solve a particular problem [9]. In deep learning, the majority of model parameters are learnt from the outputs of earlier layers rather than directly from the features of training samples[9].

Training a model can be done using supervised, semi-supervised, unsupervised, or reinforcement learning. In this study, we focus only on supervised learning.

2.4.1 Supervised learning

In supervised learning, each example in the dataset is associated with a label. The label can be a real integer, an element from a finite set of classes, a vector, a matrix, a graph, or any more complicated structure [9]. A supervised learning algorithm's objective is to create a model from the dataset that receives a feature vector as input and returns data that enables inferring the label for this feature vector [9].

2.4.2 Stochastic gradient descent

Gradient descent is a method for locating a local minimum of a function that involves starting at a random location and moving forward in steps that are proportional to the function's approximate or actual gradient at that location [9]. Gradient descent can be used to find optimal parameters for neural networks.

Gradient descent is, however, slow when implemented on large datasets and is also sensitive to the learning rate [9]. A variation of gradient descent, known as minibatch Stochastic Gradient Descent (SGD), accelerates computation by approximating the gradient using smaller batches (subsets) of the training data [9].

SGD has undergone numerous "upgrades." AdaGrad is a kind of SGD algorithm that scales the learning rate for each parameter inversely proportional to the square root of the sum of all of their historical squared values in accordance with their gradient history [17]. 'Momentum' is a technique that accelerates SGD by directing the gradient descent in the necessary direction and minimising oscillations [9].

SGD variations like RMSprop and Adam are widely utilised in neural network training. RMSprop alters AdaGrad by converting the gradient accumulation into an exponentially weighted moving average [17]. Adam, the name of which is derived from "adaptive moments", can be seen as a variant of RMSProp and momentum with a few distinctions [17]. In Adam, momentum is directly integrated as an estimate of the gradient's first-order moment (with exponential weighting) [9]. And to account for their initialisation at the origin, the estimates of the first-order moments (the momentum term) and the (uncentered) second-order moments are corrected for bias by Adam [17].

Hyperparameters, such as the learning rate mentioned above, are parameters of a deep learning model or machine learning model that are not optimised by the learning algorithm. Hyperparameters need to be tuned by experimentally finding the best combination of values, one per hyperparameter, to achieve the best possible performance from a model [9]. To tune hyperparameters, a hyperparameter tuning technique is used.

2.4.3 Hyperparameter tuning techniques

The simplest hyperparameter tuning technique is the grid search. To perform a grid search, you select a discrete set of values for each hyperparameter. You then train the model on all possible combinations of hyperparameters in the discrete sets. Based on the performance of each model on the validation set, you select the best model or you narrow down the search by reducing the range of values for each parameter and creating new discrete sets to perform another grid search if the model performance achieved is not good enough.

In contrast to grid search, random search allows you to specify a statistical distribution for each hyperparameter from which values are randomly picked and the overall number of combinations you want to test [9].

Bayesian search techniques use the results from past evaluations to choose the next hyperparameter values to evaluate [9]. This form of optimisation limits the number of optimisation runs by choosing hyperparameter values based on values that resulted in the model performing well in the past.

2.5 Traffic datasets

Benchmark datasets help to accelerate research by making findings easy to compare, while a lack of benchmark datasets can cause research to be data-specific instead of generalisable [8]. There is no standard benchmark dataset in traffic prediction, meaning that findings are not easy to share and integrate [8].

The closest to a benchmark dataset that is available for traffic flow and speed prediction is the California Transportation Agencies (CalTrans) Performance Measurement System (PeMS) dataset which is frequently used in the traffic prediction field [3], [8], [18]. This is because of the CalTrans dataset's public availability, simple structure and long historical data [3]. A specific subset of the CalTrans PeMS dataset that is commonly used is called

PEMS-BAY [7], [16], [19], [20]. Another commonly used dataset that resembles the PeMS dataset is the Los Angeles Metropolitan Transportation Authority (METR-LA) dataset [7], [13], [16], [20]. The PEMS-BAY and METR-LA datasets are discussed in more detail in Chapter 3.

Many authors also use data from Beijing [3]. A popular paper by Yu et al. [21] makes use of BJER4 collected by the Beijing Municipal Traffic Commission. Numerous other authors also use data from Beijing but, uncertainty exists over whether or not all the Beijing datasets come from one unified data source [3].

There are leaderboards available for the METR-LA[1] and PEMS-BAY[2] datasets which make finding current SOTA models easy. Unlike the METR-LA and PEMS-BAY datasets, there are no leaderboards available for Beijing datasets such as BJER4.

2.6 Conclusion

The building blocks used in traffic prediction methods and difficulties in traffic prediction are identified and discussed. Various spatial and temporal dependency modelling techniques used in recent work were compared and essential concepts used in deep learning that are implemented in this study are introduced.

Traffic flow prediction was introduced as well as the complexities involved in traffic prediction. Methods for representing temporal and spatial dependencies were mentioned and temporal, spatial and spatiotemporal modelling approaches used in traffic prediction were discussed. Essential concepts regarding training deep learning models were introduced followed by an overview of datasets commonly used for traffic prediction.

[1] https://paperswithcode.com/sota/traffic-prediction-on-metr-la
[2] https://paperswithcode.com/sota/traffic-prediction-on-pems-bay

Chapter 3

Data analysis

3.1 Overview

In this section, we discuss the METR-LA and PEMS-BAY datasets. Both of these datasets, as summarised in Table 3.1, were released by Li et al. [7] and are popular when evaluating traffic prediction models [16]. The datasets are sorted in chronological order and then partitioned: 70% training, 10% validation, and 20% testing. Only the training and validation data partitions of the datasets are used in this chapter to perform the analysis.

Table 3.1 – Characteristics of the PEMS-BAY and METR-LA datasets

	METR-LA	**PEMS-BAY**
Sensors	207	326
Period	4 months	6 months
Resolution	5 minutes	5 minutes
Speed unit	miles per hour	miles per hour
Number of time steps	34 272	52 116
Number of traffic data points	6 519 002	16 937 179

3.2 Dataset overview and characteristics

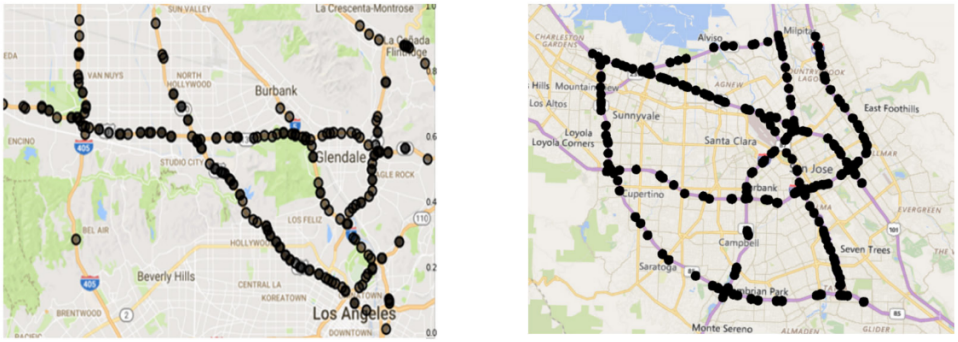
The METR-LA dataset consists of four months of traffic speed data, collected from 1 March 2012 to 27 June 2012, aggregated into 5-minute windows recorded using 207 sensors on the highways of Los Angeles. The PEMS-BAY dataset consists of six months of traffic speed data, collected from 1 January 2017 to 30 June 2017, aggregated into 5-minute windows recorded using 326 sensors in the Bay area of Los Angeles by CalTrans Performance Measurement System (PeMS).

Along with the sensor readings, the longitude and latitude of the sensors are given. The sensor positions of both datasets are visualised in Figure 3.1.

A directed adjacency matrix, constructed from the pairwise road network distances between sensors, adjusted using a thresholded Gaussian kernel using Equation 3.1 [7], is also available for each dataset. All self-loops are included in the adjacency matrix.

$$\mathbf{W}_{ij} = \mathsf{exp}(-\frac{\mathsf{dist}(v_i, v_j)^2}{\sigma^2}) \text{ if } \mathsf{dist}(v_i, v_j) \leq k, \text{ otherwise } 0 \tag{3.1}$$

Here $\mathbf{W}_{ij}$ is the edge weight and $\mathsf{dist}(v_i, v_j)$ the road network distance between sensor v_i and v_j, σ is the standard deviation of distances and k is the threshold [7]. DCRNN and Graph WaveNet both use $k = 0.1$ for the threshold. When there is more than one route between two sensors the shortest distance path is used.



(a) METR-LA (b) PEMS-BAY

Figure 3.1 – METR-LA and PEMS-BAY sensor positions. Reproduced from [7]

3.2.1 Zero values present in datasets

The METR-LA dataset contains 403 570 zero values (7.11%). Making up a large portion of these zero values is 1 482 time instances where all sensor values are zero (5.41%) of the dataset. Thus, most of the zero values are due to time instances where all sensor values are zero at the same time. The PEMS-BAY dataset (training and validation data partitions) contains 304 zero values (the percentage is negligible) and no time instances where all sensor values are zero.

The authors of the DCRNN paper [7] claim that the zero values in the datasets represent intervals in time that no cars passed the sensors and not stationary traffic [22]. This is, however, hard to believe when it comes to the instances where no traffic passed any sensor during a given 5-minute interval in the entire METR-LA dataset. These values are therefore treated as missing data, as discussed in Section 6.5.

In Wu et al.'s original description of Graph WaveNet [13] it is stated that missing values are excluded both from training and testing. What this means in practice is that the loss function used by Graph WaveNet merely assigns 0 loss to predictions for which the true sensor reading is 0. Zero values are still used as input features for predictions.

3.2.2 Traffic network node connections

Figures 3.2 and 3.3 visualise all of the connections between nodes in the traffic networks. There are many connections in the traffic networks due to all of the smaller roads that are present, as can be seen in Figure 3.1.

Figures 3.4 and 3.5 visualise the road-wise connections used to construct the weighted adjacency matrices for the METR-LA and PEMS-BAY dataset. The connections are reduced using the Gaussian kernel of Equation 3.1 and a threshold of 0.1.

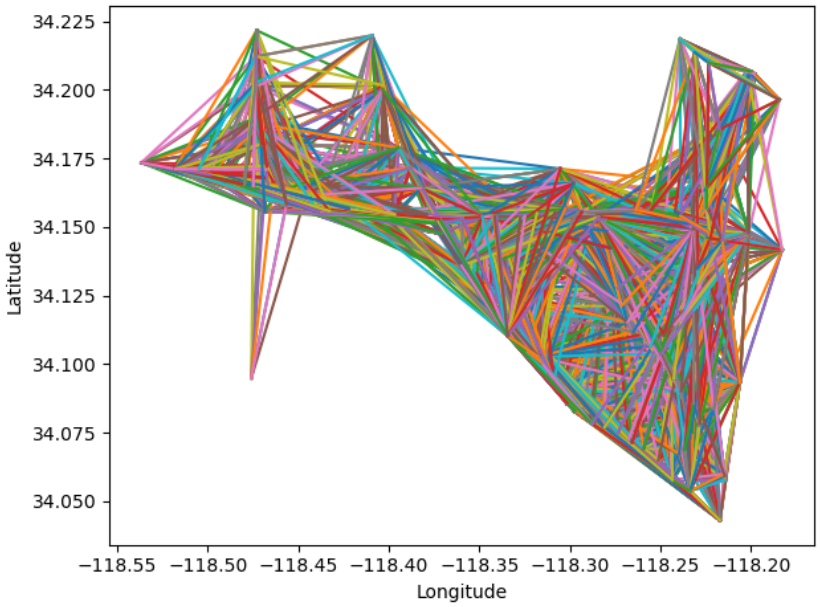


Figure 3.2 – All road-wise connections between nodes in the METR-LA dataset.

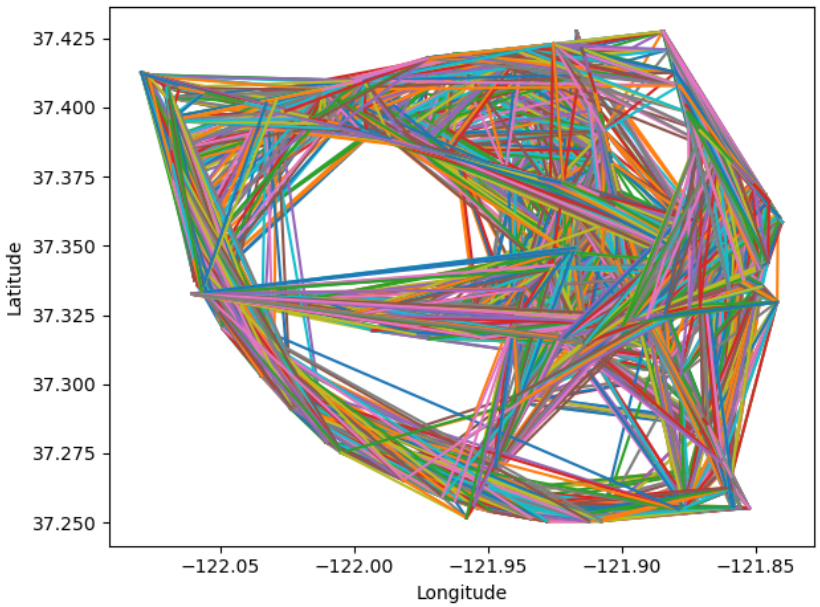


Figure 3.3 – All road-wise connections between nodes in the PEMS-BAY dataset.

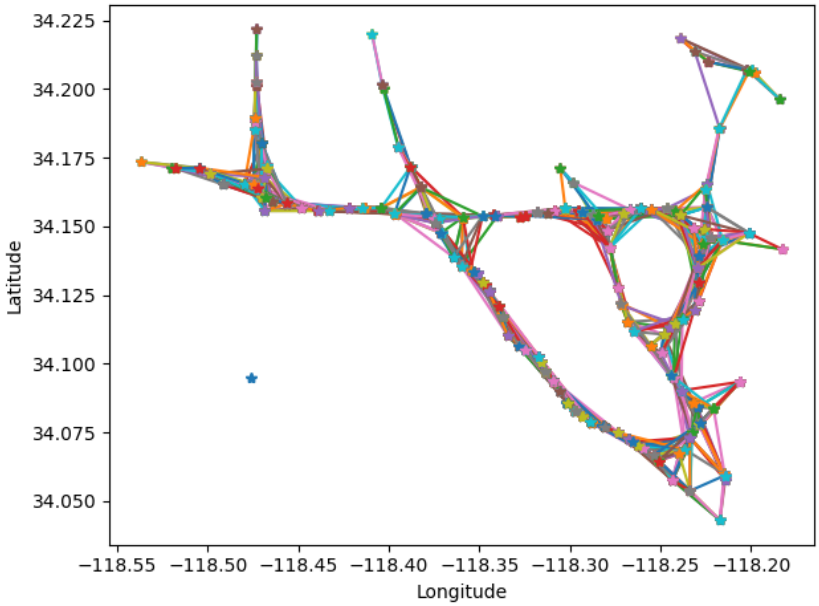


Figure 3.4 – Reduced road-wise connections between nodes in the METR-LA dataset. These are the connections used by Graph WaveNet and DCRNN. The connections are reduced based on the threshold used for the thresholded Gaussian kernel ($k = 0.1$).

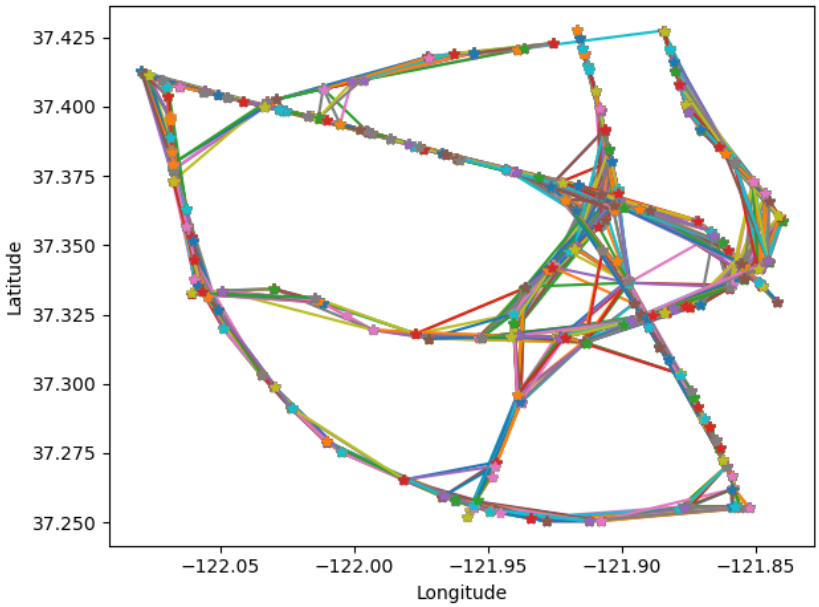


Figure 3.5 – Reduced road-wise connections between nodes in the PEMS-BAY dataset. These are the connections used by Graph WaveNet and DCRNN. The connections are reduced based on the threshold used for the thresholded Gaussian kernel ($k = 0.1$).

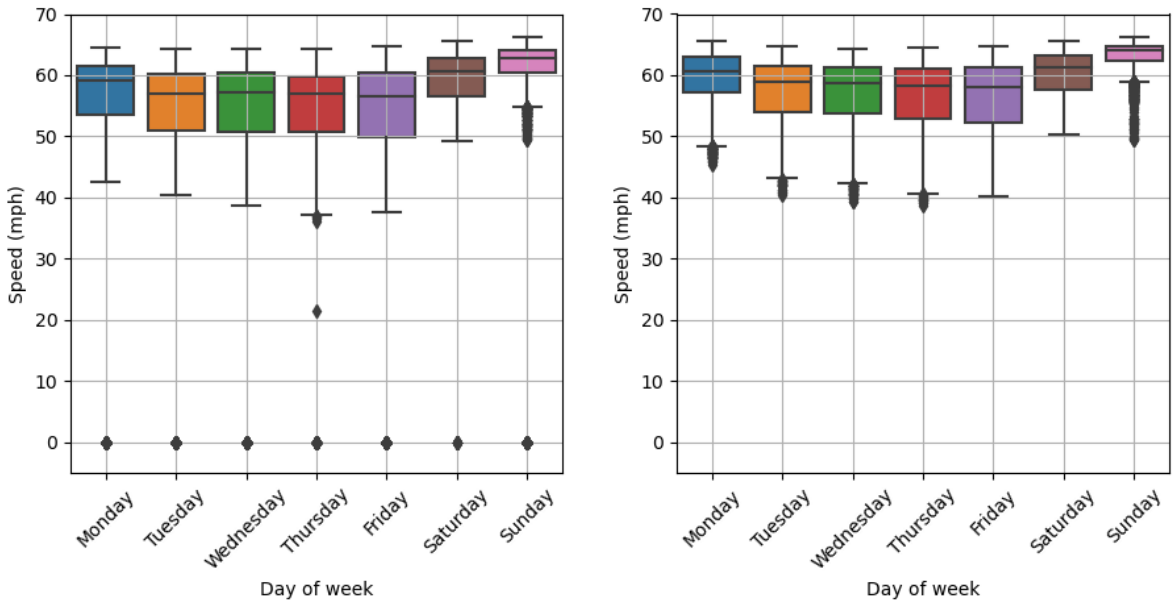
3.2.3 Average daily and weekly traffic speed analysis

Here we obtain a deeper insight into the behaviour of each dataset with regard to the variation in average network-wide speed for each day of the week and for different times of the day. To do this we first look at the boxplot of the average network speed for each day of the week and then for each hour of the day. A boxplot gives us the median value, interquartile range (IQR), and minimum and maximum values that are not outliers. The IQR is the difference between the 3rd quartile value and the 1st quartile value. Outliers are defined as values 1.5 IQR below the 1st quartile or 1.5 IQR above the 3rd quartile. We then finally take a look at the variance in average network traffic speed during each day of the week for each of the datasets.

Figures 3.6 and 3.7 show boxplots of the average network speed for the METR-LA and PEMS-BAY datasets per day of the week. (Black dots are outliers.) From the boxplots, we can see that the average traffic speed is higher and varies less on weekends. We can also see from Figure 3.6 that the zero values in the METR-LA dataset have a significant impact on the average measured network speed as well as the variance of the data. The PEMS-BAY dataset, on the other hand, is not significantly affected by the few zero values that are present in the dataset.

Figures 3.8 and 3.9 show boxplots of the average network speed per hour of the day for the METR-LA and PEMS-BAY datasets, again including zero values on the left and excluding zero values on the right. We can see that for the METR-LA dataset zero values impact the variance of the data during all times of day and not just times that you would expect the network to carry little traffic such as very early in the morning.

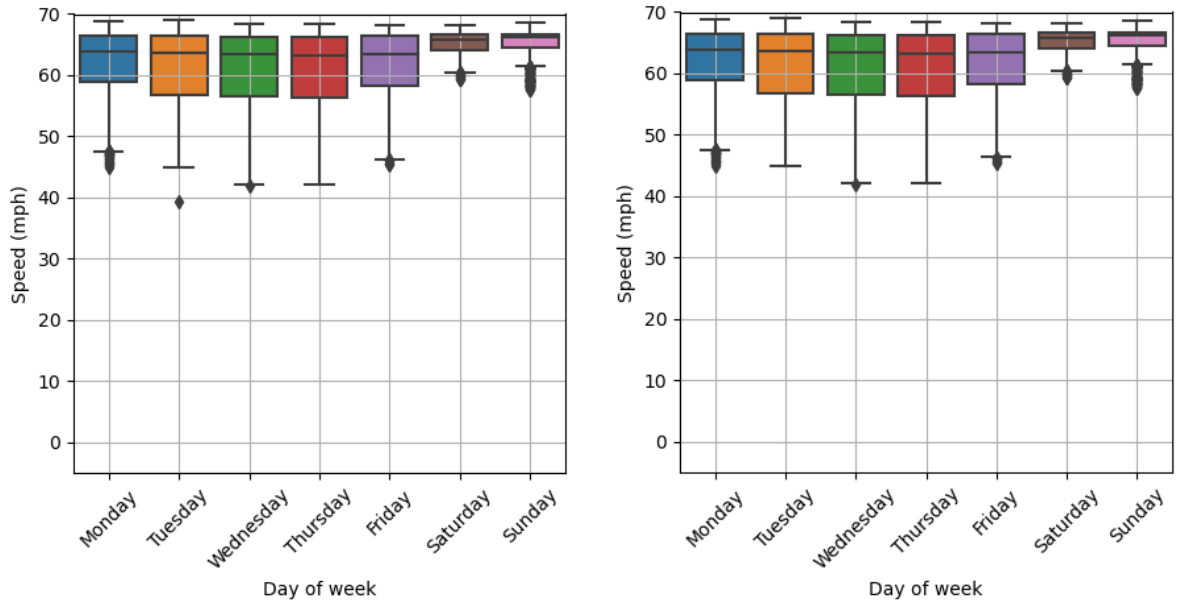
By comparing Figure 3.10a and 3.11a we can see that the average network speed of the PEMS-BAY dataset varies much less than the traffic speed of the METR-LA dataset. The variance of the METR-LA dataset can be reduced significantly as seen in Figure 3.10b by excluding the time instances for which sensor readings are zero. When the zero values are excluded from the METR-LA dataset the variance of the dataset resembles that of the PEMS-BAY dataset.



(a) With zero values

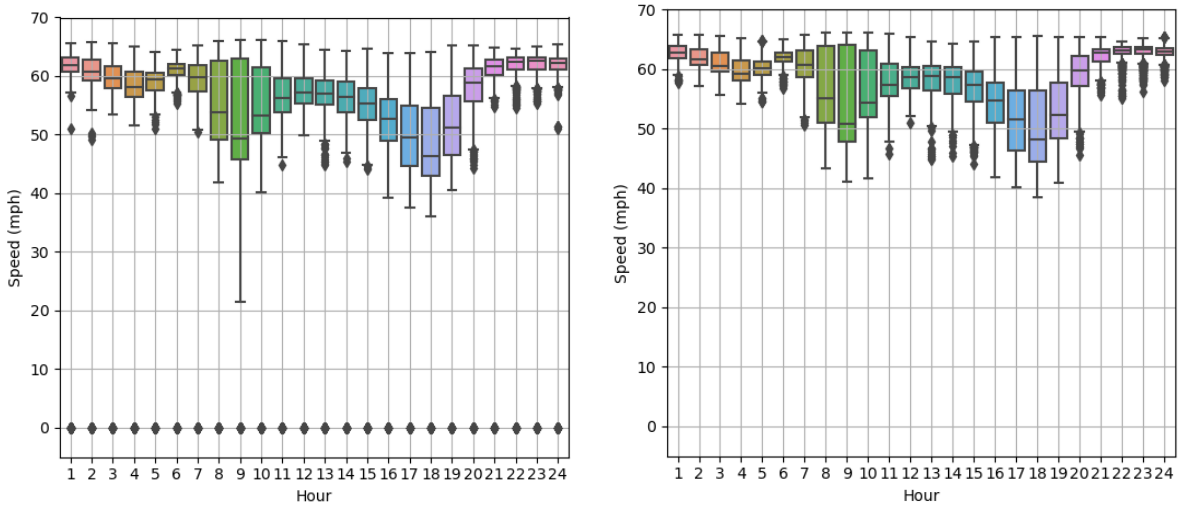
(b) No zero values

Figure 3.6 – Boxplot of the average network speed per day for the METR-LA dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right.



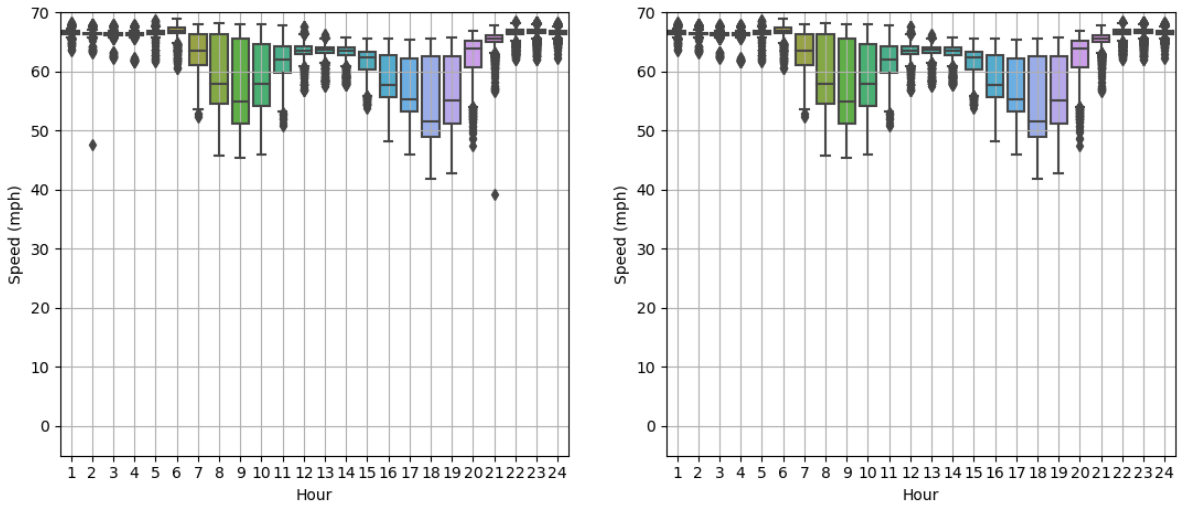
(a) With zero values **(b)** No zero values

Figure 3.7 – Boxplot of the average network speed per day for the PEMS-BAY dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right.



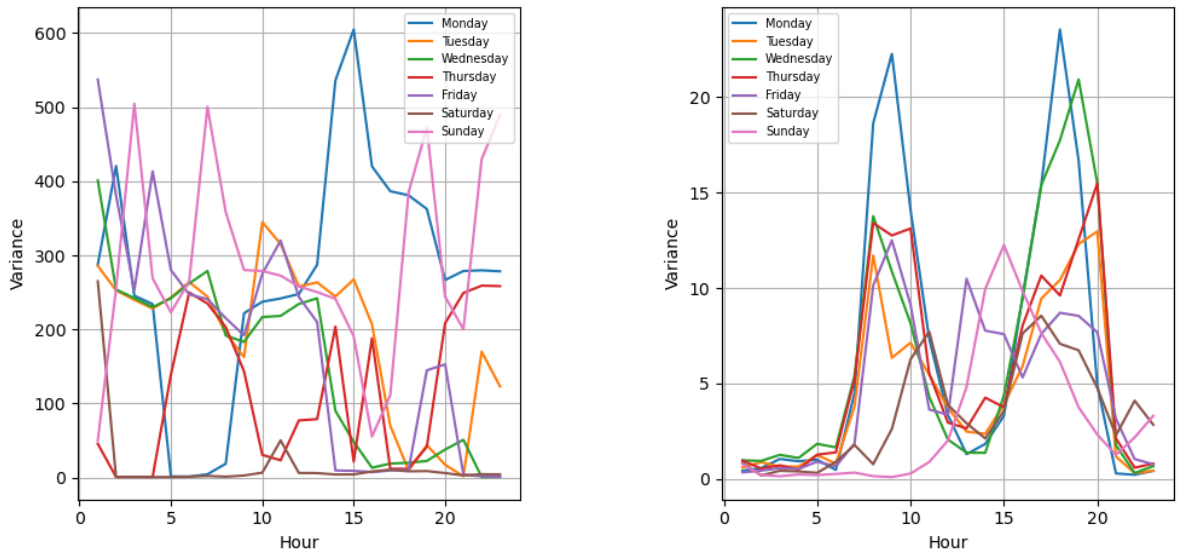
(a) With zero values **(b)** No zero values

Figure 3.8 – Boxplot of the average network speed per hour of day for the METR-LA dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right.



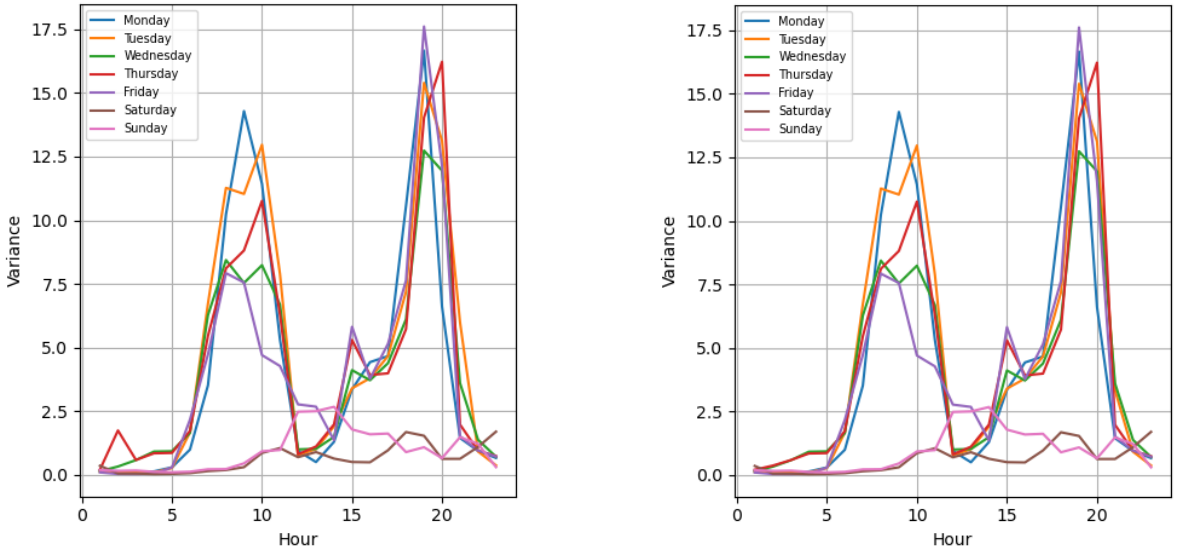
(a) With zero values **(b)** No zero values

Figure 3.9 – Boxplot of the average network speed per hour of day for the PEMS-BAY dataset (Training and validation data partitions only). Including zero values on the left. Excluding zero values on the right.



(a) With zero values **(b)** No zero values

Figure 3.10 – METR-LA average network speed variance per hour for each day of the week (Training and validation data partitions only).



(a) With zero values

(b) No zero values

Figure 3.11 – PEMS-BAY average network speed variance per hour for each day of the week (Training and validation data partitions only).

3.3 Conclusion

In this chapter, we saw that zero values in the METR-LA dataset have a significant effect on the data variance of the traffic network. We, therefore, treat these values as missing data, as discussed in Section 6.5.

We also determined that there is a noticeable difference between traffic data recorded on weekends and traffic data recorded during the rest of the week. This is expected since there are much fewer people travelling to and from work during the weekend.

While analysing the data we were unable to find a good reason in the available literature for the use of a threshold of 0.1 for the Gaussian kernel used to construct the adjacency matrices of the PEMS-BAY and METR-LA datasets.

Chapter 4

System selection

4.1 Overview

In this chapter, a tool is selected for further experimentation. First, models are selected based on published results and the consulted leaderboards, previously introduced in Section 2.5. The published results of the selected models are then recreated if possible, using available software for the tools. The best-performing tool is then selected based on re-created results for further experimentation.

4.2 System introduction

What are the best available traffic prediction methods for implementation in South Africa? To answer this question traffic prediction leaderboards for the METR-LA[1] and PEMS-BAY[2] datasets were consulted (2021-10-19). The published performance of models is also taken into account for models not featured on the leaderboards. A method is chosen based on whether or not there is an implementation available to the public for the method. Methods are also chosen based on their published performance, that is whether or not

[1]https://paperswithcode.com/sota/traffic-prediction-on-metr-la
[2]https://paperswithcode.com/sota/traffic-prediction-on-pems-bay

they are SOTA. The chosen methods are discussed in this section, before evaluation.

4.2.1 STTN

In order to dynamically model directed spatial relationships and capture real-time conditions and traffic flow directions, a spatial transformer version of the graph neural network is used in the STTN model [15].

As can be seen in Figure 4.1, STTN consists of stacked spatiotemporal blocks and a prediction layer [15]. Each spatiotemporal block consists of a spatial transformer and a temporal transformer. In each spatiotemporal block, the input node and the graph adjacency matrix are both used to extract spatial features by the spatial transformer. The spatial transformer consists of a spatiotemporal position embedding layer, a fixed graph convolution layer, a dynamic graph convolution layer, and a gate mechanism for information fusion. The fixed graph convolution layer is based on Chebyshev polynomial approximation [23] (which falls into the spectral-based approaches category) and captures the stationary spatial dependencies from the road topology [15]. The learnt linear mappings that translate the input features of each node to the high-dimensional latent subspaces are fed into the dynamic graph convolution layer [15]. These mappings are realised using feed-forward neural networks. The spatial features and input node features are combined to generate the input to the temporal transformer. The prediction layer implements two 1×1 convolutional layers to aggregate the spatiotemporal features, obtained by stacking spatiotemporal blocks, for traffic prediction.

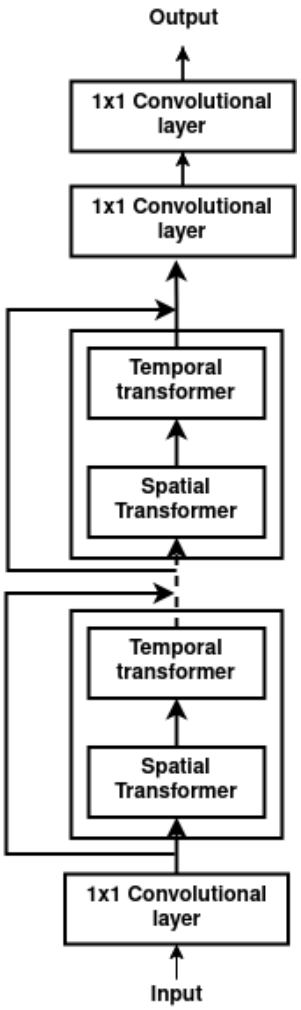


Figure 4.1 – STTN overall architecture. This consists of an input layer, k spatiotemporal hidden layers, and an output layer which implements two 1×1 convolutional layers.

4.2.2 Graph WaveNet

Graph WaveNet, as seen in Figure 4.2, developed by Wu et al. [13], uses a self-adaptive adjacency matrix in the graph convolutional layer that preserves hidden spatial dependencies. To increase effectiveness and efficiency the temporal convolution layer is assembled using dilated causal convolutions.

The graph convolution layer makes use of both predefined spatial dependencies and self-learnt hidden graph dependencies. When the graph structure of the traffic network is unavailable only the self-adaptive adjacency matrix is used to capture the hidden spatial dependencies. The graph convolution used is a generalised form of diffusion convolution and falls into the spatial-based approach category. The self-adaptive adjacency matrix is learnt via stochastic gradient descent and does not require any prior knowledge. By using stochastic gradient descent the model discovers hidden spatial dependencies by itself [13]. To do this, two node embeddings—the source and target node embeddings—are randomly initialized with learnable parameters. The source node embedding and target node embedding are multiplied to derive the spatial dependency weights between the source nodes and the target nodes [13]. The Rectified Linear Unit (ReLU) activation function is used to eliminate weak connections and the SoftMax function is used to normalise the self-adaptive adjacency matrix.

Dilated causal convolution is used in the temporal convolution layer to capture temporal trends of nodes [13]. Dilated causal convolution facilitates parallel computing and solves the gradient explosion problem by allowing for an exponentially larger receptive field by increasing layer depth and being able to handle long-range sequences non-recursively [13].

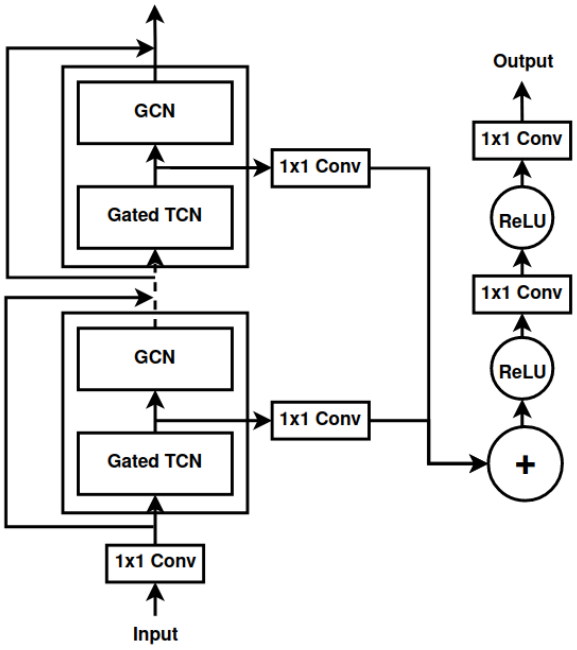


Figure 4.2 – Framework of Graph WaveNet. This consists of an input layer, k spatiotemporal hidden layers, and an output layer.

4.2.3 STAWnet

Spatio-Temporal Attention Wavenet (STAWnet) proposed by Tian et al. [20], as seen in Figure 4.3, implements multiple stacked spatiotemporal blocks and output layers. A spatiotemporal block consists of a gated temporal convolution network and a dynamic attention network. Dilated causal convolution with a gate mechanism is used for extracting temporal dependencies as for Graph WaveNet. The self-attention network is used on graph-structured data to uncover patterns in the model for modelling dynamic spatial interdependence [20]. The core concept of attention is to dynamically assign various weights to various nodes.

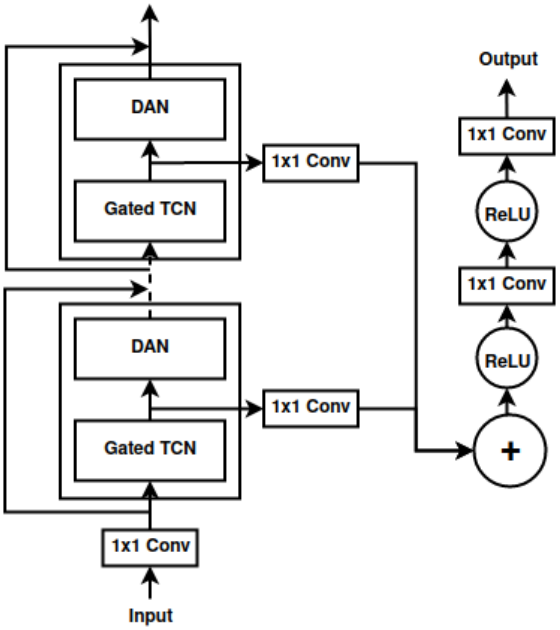


Figure 4.3 – Framework of STAWnet. Each spatiotemporal block consists of a gated TCN and a dynamic attention network (DAN).

4.2.4 GMAN

Graph Multi-Attention Network (GMAN), as seen in Figure 4.4 follows the encoder-decoder architecture. Between the encoder and decoder, a transform attention layer is introduced to transform the encoded past traffic features into future representations [19]. The encoder and decoder are composed of spatiotemporal attention blocks (ST-attention blocks). In the ST-Attention blocks, a gated fusion mechanism is used to adaptively fuse spatial and temporal representations, while a spatial attention mechanism models dynamic spatial correlations and a temporal attention mechanism models non-linear temporal correlations [19]. The transform attention mechanism alleviates the effect of error propagation by modelling the direct relationships between historical and future time steps [19].

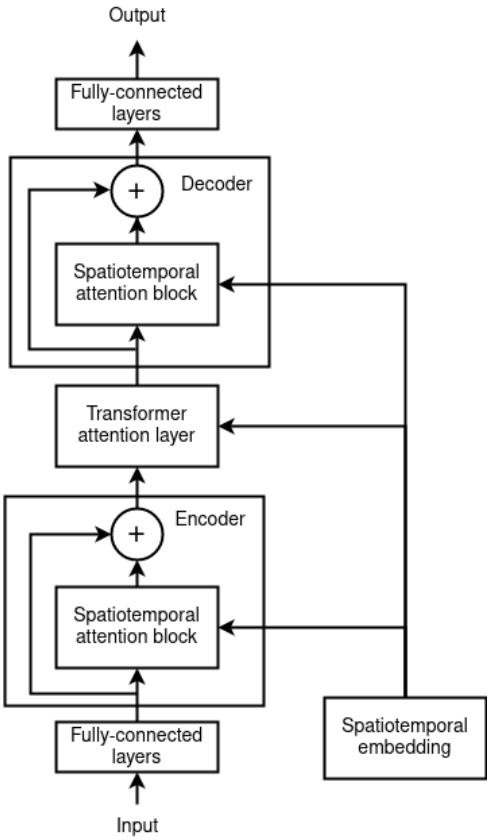


Figure 4.4 – Framework of GMAN. This consists of a spatiotemporal embedding, an encoder and a decoder each with k spatiotemporal attention blocks, and two fully connected layers.

4.2.5 STNN

STNN, as seen in Figure 4.5, makes use of a spatiotemporal correlation learning paradigm, called spacetime interval learning, that fuses spatial and temporal dimensions into a single manifold [16]. The spacetime interval learning paradigm ignores sensors that do not significantly contribute to the final forecast by extracting traffic data from surrounding sensors within a specified time window of each target sensor, in the local-spacetime context, which is equivalent to the receptive field of sensors [16]. In the local-spacetime context, the spacetime interval learning paradigm then learns to correlate a node at a particular time with a different node at a different time [16]. The spacetime interval learning paradigm:

- Simulates spatiotemporal correlations in the local-spacetime context of each target node, which renders it independent of the graph structure and applicable to all traffic flows in general [16].
- Changes traffic prediction from network-level to node-level, enabling the model to forecast traffic for numerous locations of interest concurrently on various machines. [16].
- Unlike existing models, the spatial and temporal dimensions are fused into a single manifold to capture varying spatial correlations between locations across time [16].
- Implements a spacetime attention block that highlights the interval between events capturing pair-wise influences, and a spacetime convolution block that aggregates the learnt features from spatial, temporal, and spatiotemporal aspects to capture many-to-one influences [16].

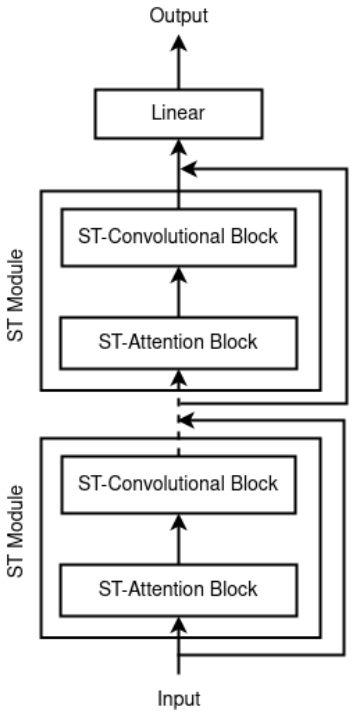


Figure 4.5 – Framework of STNN. This consists of k spacetime modules and a fully connected output layer. Each spacetime module contains a spacetime attention block and a spacetime convolution block.

4.3 Tool comparison

In this section, we compare the published system performance of the systems described above on the METR-LA and PEMS-BAY datasets. We also attempt to recreate the published results of each tool. We compare the recreated results with the published results and then compare the training time of the tools for which results could be recreated.

4.3.1 Published performance

Here we determine how the results from the different tools compare when implemented on the same datasets. The metrics used for comparing the performance of different models are Mean Absolute Error (MAE) [24], Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE) [24] for all the nodes in the traffic network. These metrics are discussed in more detail in Section 6.2. Two available leaderboards, introduced in Section 2.5, are used to determine what the current SOTA models are. Table 4.2 and Table 4.3 list the top 5 models for which code is available, as seen on the leaderboards of the METR-LA dataset[3] and the PEMS-BAY dataset[4], respectively. The METR-LA and PEMS-BAY leaderboards list the top traffic prediction models based on their ability to predict traffic speed 12 steps into the future with data aggregated into 5-minute windows, meaning that predictions are made 60 minutes into the future. The leaderboards for these two datasets are used since they are the most widely used datasets in recent years in terms of the traffic prediction task [3].

The published 15-minute, 30-minute and 60-minute traffic speed prediction results from the leaderboard models where available are given in Table 4.4 and Table 4.5, along with the results from STTN [15] on the PeMS-Bay dataset. The results compared in Table 4.4 and Table 4.5 are obtained by implementing the prediction models on the datasets divided in chronological order as given in Table 4.1.

As seen in Table 4.4 and 4.5 the accuracy of predictions decrease as the prediction horizon

[3]https://paperswithcode.com/sota/traffic-prediction-on-metr-la
[4]https://paperswithcode.com/sota/traffic-prediction-on-pems-bay

increases. And, the order of model performance is similar for both datasets where the same models are used.

Table 4.1 – Original dataset partitioning reported on in the literature. Partitioning is done in chronological order: training, validation, testing.

	Partition sizes		
Tool	**Training**	**Validation**	**Testing**
DCRNN [7]	70%	10%	20%
GMAN [19]	70%	10%	20%
STAWnet [20]	70%	10%	20%
Graph WaveNet [13]	70%	10%	20%
STNN [16]	70%	20%	10%
STTN [15]	70%	10%	20%

Table 4.2 – Performance of top 5 models, based on METR-LA dataset leader board, for which code is available on the METR-LA dataset when predictions are made 60 minutes into the future.

Model	**MAE (mph) 60-minute prediction**
STAWnet	3.44
Graph Wavenet incrementally improved	3.47
Graph WaveNet	3.53
DCRNN	3.6
STGCN	4.45

Table 4.3 – Performance of top 5 models, based on PEMS-BAY dataset leader board, for which code is available on the PEMS-BAY dataset when predictions are made 60 minutes into the future.

Model	**MAE (mph) 60-minute prediction**
STNN	1.86
GMAN	1.86
STAWnet	1.89
Graph WaveNet	1.95
DCRNN	2.07

Table 4.4 – Published performance of selected models on the METR-LA dataset test partition. MAE and RMSE given in miles per hour.

	15 min			30 min			60 min		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
STNN [16]	2.27	4.46	5.80%	2.56	5.29	6.84%	3.01	6.23	8.50%
STAWnet [20]	2.70	5.22	6.98%	3.04	6.14	8.22%	3.44	7.16	9.82%
Graph WaveNet [13]	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
DCRNN [7]	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.59	10.50%

Table 4.5 – Published performance of selected models on the PEMS-BAY dataset test partition. MAE and RMSE given in miles per hour.

	15 min			30 min			60 min		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
STNN [16]	1.20	2.41	2.53%	1.50	3.26	3.33%	1.86	4.22	4.30%
GMAN [19]	1.34	2.82	2.81%	1.62	3.72	3.63%	1.86	4.32	4.31%
STAWnet [20]	1.31	2.78	2.76%	1.62	3.70	3.67%	1.89	4.36	4.47%
Graph WaveNet [13]	1.30	2.74	2.76%	1.63	3.70	3.67%	1.95	4.52	4.63%
STTN [15]	1.36	2.87	2.89%	1.67	3.79	3.78%	1.95	4.50	4.58%
DCRNN [7]	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%

4.3.2 Availability of tools

The availability of code for the evaluated tools is an important aspect when it comes to the ease with which the tools can potentially be implemented. The official source code is available for STNN[5], GMAN[6], STAWnet[7], and Graph WaveNet[8]. An implementation is available for STTN[9] as well but this is not referenced by the original proposers of STTN.

4.3.3 Recreation of published results

Here we attempt to recreate the published results/performance of the tools to determine the validity of the available implementations. The results are obtained by implementing the tools as described by the publishers with settings as specified in the code or in the published papers, depending on where the information is available. The hyperparameters

[5]https://github.com/libingixn/STNN
[6]https://github.com/zhengchuanpan/GMAN
[7]https://github.com/CYBruce/STAWnet
[8]https://github.com/nnzhan/Graph-WaveNet
[9]https://github.com/wubin5/STTN

used are given in Table 4.6. No conflicting values are observed between the published work and available implementations.

Table 4.6 – Hyperparameters used when recreating the published results of the evaluated tools

	Graph WaveNet	**STAWnet**	**STNN**
Learning rate	0.001	0.001	0.001
Dropout rate	0.3	0.3	0.3
Hidden dimensions	32	32	
Batch size	64	64	80
Weight decay rate	0.0001	0.0001	0
Epochs	100	100	50
Seed	99	99	1

Tables 4.7 and 4.8 list the recreated results for 15-minute, 30-minute, and 60-minute predictions. The input data used is aggregated into 5-minute windows meaning that a 60-minute prediction indicates a 12-step prediction when implementing the tools.

Table 4.7 – Recreated results for 15-minute, 30-minute, and 60-minute predictions of tools on METR-LA test data partition. MAE and RMSE given in miles per hour.

	15 min			**30 min**			**60 min**		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
STNN	2.29	4.45	5.80%	2.59	5.22	6.84%	3.03	6.26	8.49%
STAWnet	2.72	5.26	6.97%	3.08	6.22	8.30%	3.50	7.27	9.96%
Graph WaveNet	2.69	5.13	6.76%	3.05	6.12	8.17%	3.49	7.21	9.82%

Table 4.8 – Recreated results for 15-minute, 30-minute, and 60-minute predictions of tools on PEMS-BAY test data partition. MAE and RMSE given in miles per hour.

	15 min			**30 min**			**60 min**		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
STNN	1.21	2.43	2.51%	1.49	3.23	3.23%	1.87	4.18	4.32%
STAWnet	1.32	2.81	2.77%	1.64	3.75	3.69%	1.95	4.46	4.52%
Graph WaveNet	1.30	2.72	2.71%	1.62	3.66	3.64%	1.93	4.43	4.52%

The differences between the published and recreated results are given in Tables 4.9 and 4.10.

- The recreated results of Graph WaveNet are better than the published results. The software used is the version that was last updated on 22 December 2019.

Table 4.9 – Difference between published and recreated results for the METR-LA dataset (published − recreated). Published results are given in Table 4.4 and recreated results in Table 4.7. MAE and RMSE given in miles per hour.

	15 min			30 min			60 min		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
STNN	-0.02	0.01	0.00	-0.03	0.07	0.00	-0.02	-0.03	0.01
STAWnet	-0.02	-0.04	0.01	-0.04	-0.08	-0.08	-0.06	-0.11	-0.14
Graph WaveNet	0.00	0.02	0.14	0.02	0.10	0.20	0.04	0.16	0.19

Table 4.10 – Difference between published and recreated results for the PEMS-BAY dataset (published − recreated). Published results are given in Table 4.8 and recreated results in Table 4.8. MAE and RMSE given in miles per hour.

	15 min			30 min			60 min		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
STNN	-0.01	-0.02	0.02	0.01	0.03	0.10	-0.01	0.04	-0.02
STAWnet	-0.01	-0.03	-0.01	-0.02	-0.05	-0.02	-0.06	-0.10	-0.05
Graph WaveNet	0.00	0.02	0.05	0.01	0.04	0.03	0.02	0.09	0.11

- The STAWnet recreated results are almost as good as the published results. The software used is the version that was last updated on 27 May 2021.
- The recreated results for STNN are approximately the same as the published results. The software used is the version that was last updated on 8 September 2021.
- We were unable to recreate the published GMAN performance with the available implementation. The software used is the version that was updated on 20 September 2021. Problems reported on GitHub[10] with no reply from the author include the inability to recreate results, training taking an extremely long time, and inconsistencies between what is published and the actual code. The comments on GitHub suggest results cannot be recreated with the available software implementation.
- We were not able to recreate the published STTN results using the available 3rd party software. The software used is the version that was last updated on 22 April 2021. The available software seems to be incomplete or a work-in-progress.

[10]https://github.com/zhengchuanpan

4.3.4 Training time comparison

In Table 4.11 we compare the training time of the models for which results could be recreated. As seen in Table 4.11, STNN takes much longer to train than the other methods. STAWnet and Graph WaveNet have very similar training times, with STAWnet requiring only slightly more time to train.

Table 4.11 – Training time comparison of implemented tools

	Epochs	**METR-LA total training time (seconds)**	**PeMS-Bay total training time (seconds)**	**METR-LA training time per epoch (seconds)**	**PeMS-Bay training time per epoch (seconds)**
STNN	50	12 789	294 244	256	5 885
STAWnet	100	5 941	14 432	59	144
Graph WaveNet	100	4 991	11 601	50	116

4.3.5 STNN data partitioning and performance calculation

STNN partitions the datasets differently than the rest of the models. To ensure that the results are truly comparable, STNN is trained again with the same hyperparameters as before but with the dataset divided in chronological order as follows: 70% training, 10% validation, and 20% testing. This is not a 100% fair comparison since no hyperparameter optimisation was done. The system could perhaps do better with some optimisation, or it could be that it is optimistic since a part of the test set was originally used to do the previous hyperparameter optimisation. However, this removes at least one element of the variability: reporting on a completely different test set. (A full hyperparameter evaluation would have been too expensive at this point.)

The performance calculations used by STNN are also changed to match those of Graph WaveNet and STAWnet. Results published by STNN are calculated by averaging performance up to the prediction horizon reported on. Graph WaveNet and STAWnet, on the other hand, report on performance for specific prediction horizons.

With the above two changes implemented, the performance of STNN degrades significantly as can be seen when comparing the results in Table 4.12 with the published results of STNN in Tables 4.4 and 4.5 respectively.

Table 4.12 – STNN recreated results for the sake of comparability with results in Tables 4.7 and 4.8 respectively. Data partitioning is done to match the other models, and performance calculations are done in the same way as the rest of the models. MAE and RMSE given in miles per hour.

	15 min			**30 min**			**60 min**		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
METR-LA	2.67	5.25	7.30%	3.20	6.49	9.28%	3.98	8.05	12.29%
PeMS-Bay	1.41	2.96	2.97%	1.83	4.10	4.16%	2.37	5.26	5.71%

4.4 Conclusion

It was possible to recreate the results for Graph WaveNet, STAWnet, and STNN. For both STTN and GMAN it was not possible to recreate the published results without altering or rewriting the available code. For GMAN, similar issues to ours were observed by other forum users. For STTN the available code is not currently being improved by the third-party publisher even though it is not working properly at this point.

Graph WaveNet and STAWnet are the best-performing models for which results were successfully recreated. STNN does not perform as well as the other two models when the performance of the model is calculated correctly. STNN is not only the worst-performing model, but it also takes the longest to train. STAWnet performs as advertised, is easy to implement and the model can be implemented without a predefined graph. Graph WaveNet performs the best, based on the recreated results and can also be implemented without the use of a predefined graph.

Based on these results, Graph WaveNet is selected as the best available tool for implementation and experimentation, while STAWnet also remains a viable option.

Chapter 5

Graph WaveNet

5.1 Overview

In this chapter, we describe the Graph WaveNet system in detail by first introducing the main elements of the network (Sections 5.2 and 5.3) and then the final system (Section 5.4). We also discuss work that aimed to improve Graph WaveNet after its initial definition (Section 5.5).

5.2 Graph WaveNet convolution elements

CNN blocks make up most of the Graph WaveNet model. Here the workings of the various convolution implementations used in Graph WaveNet are explained. These include 1×1 convolutional layers, causal convolution, 1-D convolution, 2-D convolutional layers and dilated convolution.

5.2.1 1×1 convolutional layer

A 1×1 convolutional layer (or a convolutional layer with 1×1 kernels) is equivalent to a fully connected layer. It is usually used to control the number of channels between layers in a network and to control the model complexity [25]. A 1×1 convolutional layer has c 1×1 kernels that are all d deep, where d is the number of input channels, and c is the number of output channels.

5.2.2 Causal convolution

Causal convolution is used for temporal data to ensure that any prediction made by the model cannot depend on any future time steps [26]. For 1-D data, implementing causal convolution is done by moving the output of a regular convolution forward a few time steps [26]. For a convolution kernel of size k, zero padding of $k-1$ must be added in the past direction of the input sequence. This is specific to causal convolution (padding in the past direction and not the future direction).

5.2.3 1-D Convolution

The output value of a simple/vanilla 1-D convolutional layer with input size $(|\mathbf{B}|, |\mathbf{C}_{in}|, l_{in})$ and output size $(|\mathbf{B}|, |\mathbf{C}_{out}|, l_{out})$ can be described as [27]:

$$\mathsf{out}(\mathbf{B}_i, \mathbf{C}_{out_j}) = \mathsf{bias}(C_{out_j}) + \sum_{k=0}^{\mathbf{C}_{in}-1} \mathsf{weight}(\mathbf{C}_{out_j}, k) \star \mathsf{input}(\mathbf{B}_i, k) \tag{5.1}$$

where $\star$ is the cross-correlation operator, weight is the learned filter values, input is the 1-D input signal, $|\mathbf{B}|$ is the batch size, $|\mathbf{C}|$ is the number of channels, l is the sequence length, and i and j specify the particular input batch and channel. A learnable bias can also be added to the output, given by $\mathsf{bias}(C_{out_j})$. The output sequence length is given by Equation 5.2 [27]. In Equation 5.2, $\mathsf{padding}$ refers to the number of values the input sequence is padded with, and $\mathsf{dilation}$ is described in Section 5.2.5.

$$l_{out} = \lfloor \frac{l_{in} + 2 \times \mathsf{padding} - \mathsf{dilation} \times (\mathsf{kernel\_size} - 1) - 1}{\mathsf{stride}} + 1 \rfloor \tag{5.2}$$

5.2.4 2-D Convolution

The output value of a simple/vanilla 2-D convolutional layer with input size $(|\mathbf{B}|, |\mathbf{C}_{in}|, h_{in}, w_{in})$ and output size $(|\mathbf{B}|, |\mathbf{C}_{out}|, h_{out}, w_{out})$ can be described as [28]:

$$\mathsf{out}(\mathbf{B}_i, \mathbf{C}_{out_j}) = \mathsf{bias}(\mathbf{C}_{out_j}) + \sum_{k=0}^{\mathbf{C}_{in}-1} \mathsf{weight}(\mathbf{C}_{out_j}, k) \star \mathsf{input}(\mathbf{B}_i, K) \tag{5.3}$$

where $\star$ is the 2-D cross-correlation operator, $|\mathbf{B}|$ is the batch size, $|\mathbf{C}|$ is the number of channels, h is the height of the input planes, w is the input plane width, and i and j specify the particular input batch and channel. A learnable bias can also be added to the output, given by $\mathsf{bias}(C_{out_j})$. The output sequence height is given by Equation 5.4 and the output sequence width by Equation 5.5.

$$h_{out} = \lfloor \frac{h_{in} + 2 \times \mathsf{padding}[0] - \mathsf{dilation}[0] \times (\mathsf{kernel\_size}[0] - 1) - 1}{\mathsf{stride}[0]} + 1 \rfloor \tag{5.4}$$

$$w_{out} = \lfloor \frac{w_{in} + 2 \times \mathsf{padding}[1] - \mathsf{dilation}[1] \times (\mathsf{kernel\_size}[1] - 1) - 1}{\mathsf{stride}[1]} + 1 \rfloor \tag{5.5}$$

5.2.5 Dilated convolution

By inserting holes between the components of the kernel, dilated convolutions enlarge the kernel [29]. The dilation rate d indicates how much the kernel is widened. There are $d-1$ spaces between kernel elements [29]. Stacking dilated causal convolutions exponentially increases the receptive field of a network while preserving input resolution [13]. Dilation enables convolution networks to capture longer sequences with fewer layers, saving computation resources [13].

5.3 Graph Convolutional Network

The Graph Convolution Network was introduced in 2017 by Kipf et al. [12].

For an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, with n vertices/nodes $v_i \in \mathbf{V}$, edges $(v_i, v_j) \in \mathbf{E}$, an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ (binary or weighted), and a matrix of node feature vectors

$\mathbf{X} \in \mathbb{R}^{n \times d}$ where d is the degree, the layer-wise propagation rule of a multilayer Graph Convolutional Network is given by Equation 5.6 [12].

$$\mathbf{H}_{(k+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_k\mathbf{W}_k) \tag{5.6}$$

Here, $\tilde{\mathbf{A}}$ is the adjacency matrix with added self-connections. $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ is the degree matrix which is used to normalise the adjacency matrix to avoid/reduce vanishing or exploding gradients. $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is a constant and stays the same for all layers. $\mathbf{W}_k$ is a layer-specific trainable weight matrix. σ is an activation function that introduces a non-linearity. $\mathbf{H}_k \in \mathbb{R}^{n \times c}$ is the matrix of activations in the k'th layer with c channels. The input to the first layer is given by $\mathbf{H}_0 = \mathbf{X}$.

Similar to typical CNNs and MLPs, GCNs learn a feature representation for the features of each node in the input over multiple layers, which is then used as input for the next layer (such as a linear classifier) [30]. Each graph convolution layer has three stages: feature propagation, linear transformation, and a point-wise nonlinear activation [30].

5.3.1 Feature propagation

The features $\mathbf{h}_{(k,i)}$ of each node v_i are averages with the feature vectors in their local neighbourhood at the beginning of each GCN layer k.

$$\bar{\mathbf{h}}_{(k,i)} \leftarrow \frac{1}{d_i + 1}\mathbf{h}_{(k-1,i)} + \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}}\mathbf{h}_{(k-1,j)} \tag{5.7}$$

Equation 5.7 can be expressed more compactly over the entire graph as a matrix operation. Let $\mathbf{S}$ denote the normalised adjacency matrix with added self-loops:

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}} \tag{5.8}$$

Updating all the nodes simultaneously using Equation 5.7 now becomes a sparse matrix

multiplication given by Equation 5.9 [30]. This step smooths the hidden representations locally along the edges of the graph.

$$\bar{\mathbf{H}}_k \leftarrow \mathbf{S}\mathbf{H}_{(k-1)} \tag{5.9}$$

5.3.2 Feature transformation and nonlinear transition

After the feature propagation, a GCN layer is equivalent to a standard multilayer perceptron (MLP) [30]. Each layer has a learned weight matrix $\mathbf{W}_k$, and a linear transformation is applied to the smoothed hidden feature representations. A nonlinear activation function such as ReLU [31] is then applied [30]:

$$\mathbf{H}_k \leftarrow \mathsf{ReLU}(\bar{\mathbf{H}}_k \mathbf{W}_k) \tag{5.10}$$

As with a standard MLP, the final layer in a GCN can have a classification function to predict the class of each node or a regression function to predict the future values of each node.

5.4 Graph WaveNet

Here the Graph WaveNet model architecture is explained in detail by describing how the components discussed in Sections 5.2 and 5.3 fit together. The flow and processing of data in the Graph WaveNet model are also described.

5.4.1 Problem definition

A graph is represented by Equation 5.11. For prediction, $\mathbf{V}$ is the set of sensors in the traffic network (the nodes) $|\mathbf{V}| = n$, and $\mathbf{E}$ is the set of road segments (the edges)

[13]. The weighted adjacency matrix of the graph that represents the nodes' proximity (a function of their road network distance [7]) is given by $\mathbf{A} \in \mathbb{R}^{n \times n}$.

$$\mathbf{G} = (\mathbf{V}, \mathbf{E}) \tag{5.11}$$

The traffic flow observed on $\mathbf{G}$ can be represented as a graph signal $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the number of features of each node [7]. At each time step t the graph signal (dynamic feature matrix) is represented by $\mathbf{X}^{(t)}$. The traffic prediction objective, represented by Equation 5.12, is to learn a function h that maps s' historical graph signals to s future graph signals given a graph $\mathbf{G}$ [7].

$$[\mathbf{X}^{(t-s'+1)}, \ldots, \mathbf{X}^{(t)}; G] \xrightarrow{h} [\mathbf{X}^{(t+1)}, \ldots, \mathbf{X}^{(t+s)}] \tag{5.12}$$

5.4.2 Model description

The framework of the Graph WaveNet model is shown in Figure 4.2. The Graph WaveNet model consists of a preprocessing layer, spatiotemporal layers, and an output layer [13]. Each spatiotemporal layer consists of a graph convolution layer and a gated temporal convolution layer.

Input

The input is a $b \times d \times n \times l$ tensor, where n is the number of nodes, l is the sequence length, d is the number of node features (two in this case since Graph WaveNet uses speed and time of day, but the day of the week is also an option for an additional parameter), and b is the batch size. The input is padded to allow for causal convolution and convolution with a 1×1 kernel is applied to the input to create c channels with identical dimensions (c is a hyperparameter "nhid"). After preprocessing we are left with a $b \times c \times n \times (l + \mathsf{pad})$ tensor. (Graph WaveNet uses dilated causal convolution with a kernel size of 1×2 thus,

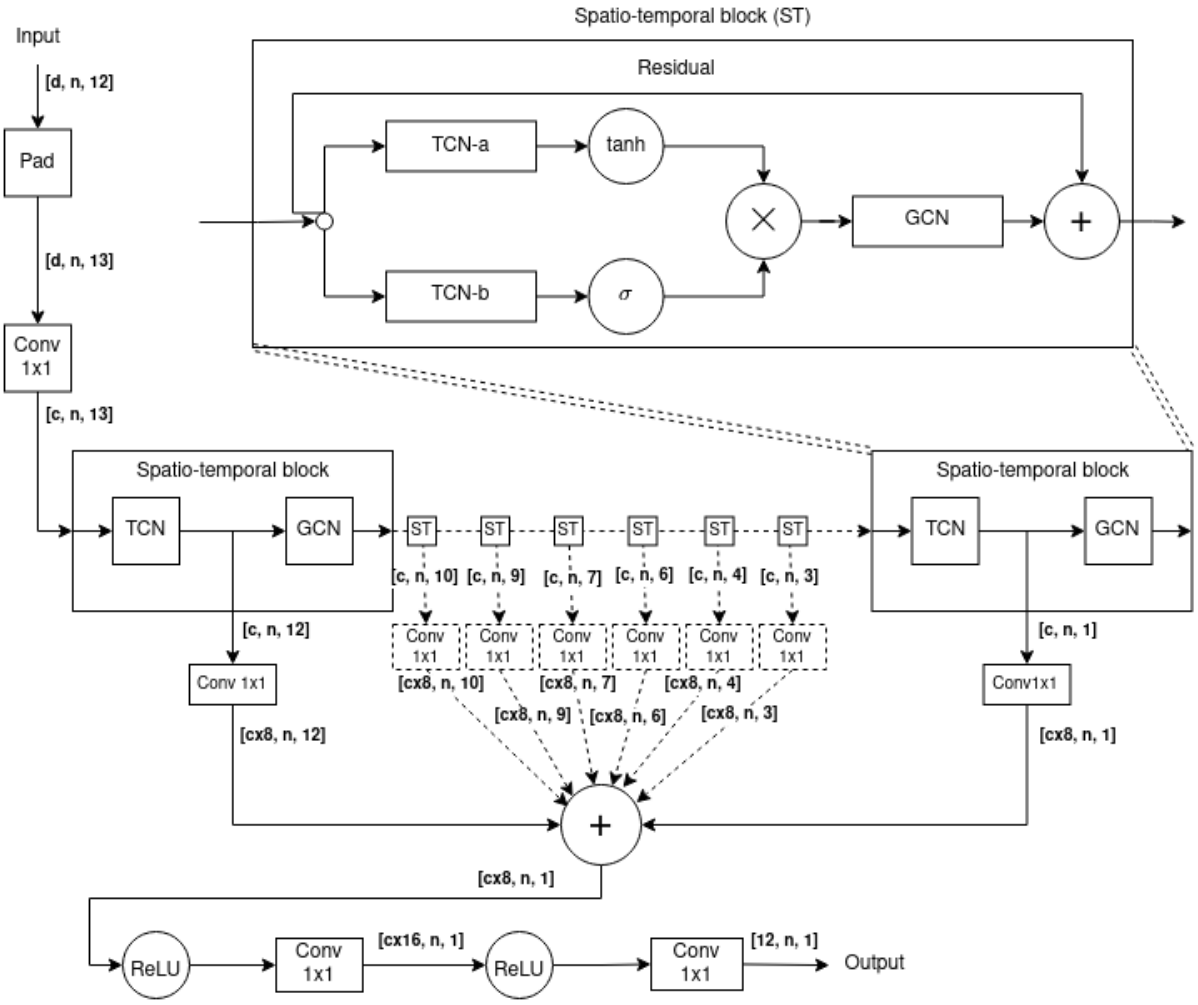


Figure 5.1 – Framework of Graph WaveNet. d=input dimensions, n=number of nodes, c=the 'nhid' hyperparameter.

pad = 1.)

Hidden layers

The receptive field size of Graph WaveNet is designed to accommodate an input sequence with a length of 12 so that in the last spatiotemporal layer the temporal dimension of the output is equal to one. This is done to generate the entire output sequence $[\mathbf{X}^{(t+1)}, \ldots, \mathbf{X}^{(t+s)}]$ as a whole, rather than generating $\mathbf{X}^t$ recursively through s steps.

The dilation in each layer is: [1, 2, 1, 2, 1, 2, 1, 2]. This dilation causes the temporal dimension to decrease as follows: $[13 \rightarrow 12 \rightarrow 10 \rightarrow 9 \rightarrow 7 \rightarrow 6 \rightarrow 4 \rightarrow 3 \rightarrow 1]$. This means that there are 8 hidden layers.

Temporal Convolution layer

The two TCNs are the first part of each hidden layer as seen in Figure 4.2. Three copies are made of the input. Two of the copies go to the two TCNs and the third is saved for later.

Each of the TCNs applies causal convolution (2-dimensional dilated causal convolution with a kernel size of 1×2) and produces c channels of output. The output from each TCN is a tensor of size $b \times c \times n \times (l + pad - d)$, where d is the dilation factor.

The tangent hyperbolic function is applied to the output of one of the TCNs and the sigmoid function to the output of the other. The element-wise product of the two outputs is then computed. The sigmoid function squashes the output to values between 0 and 1 and weakens the output from the tangent hyperbolic function. This is called 'gating'.

Gating mechanisms are important in RNNs and are also a powerful tool for controlling the flow of information in a TCN [13]. The element-wise multiplication of the output from the two TCNs is then the output of the gated TCN block. Two copies of the gated temporal features are created. One copy is passed to a final output queue (indicated by

the skip connections in Figure 4.2) and the other is passed to the GCN.

Graph Convolution layer

Before this step, the model only deals with node feature data over time and not the adjacency matrix or nodes and edges. The Graph Convolution layer/block now iterates through each time step individually to discover spatial features. GCN does not change the dimensions of the data; it performs a transformation called diffusion convolution. The GCN first picks a fixed time step and from that time step retrieves the corresponding $n \times c$ feature matrix.

Graph WaveNet makes use of a generalised form of the diffusion convolution layer presented by Li et al. [7] for its graph convolution layers but to discover hidden spatial dependencies an adaptive adjacency matrix is also learned through stochastic gradient descent. The adaptive adjacency matrix given by Equation 5.13 implements ReLU to eliminate weak connections and Softmax to normalise the entries of the matrix [13].

$$\tilde{\mathbf{A}}_{adp} = SoftMax(ReLU(\mathbf{E}_1\mathbf{E}_2^T)) \tag{5.13}$$

By combining the predefined spatial dependencies and hidden spatial dependencies in the diffusion process, the graph convolution operation is given by Equation 5.14 [13].

$$\mathbf{X}_{:,p\star\mathbf{G}} f_{\mathbf{W}} = \sum_{j=0}^{J-1} (\tilde{\mathbf{D}}_O^{-1}\tilde{\mathbf{A}})^j \mathbf{H}_k \mathbf{W}_{j,1} + (\tilde{\mathbf{D}}_I^{-1}\tilde{\mathbf{A}}^T)^j \mathbf{H}_k \mathbf{W}_{j,2} + \tilde{\mathbf{A}}_{adp}^j \mathbf{H}_k \mathbf{W}_{j,3} \tag{5.14}$$

If we do not have an adjacency matrix/no predefined spatial dependencies the Graph Convolution operation simplifies to Equation 5.15 [13].

$$\mathbf{X}_{:,p\star\mathbf{G}} f_{\mathbf{W}} = \sum_{j=0}^{J-1} \tilde{\mathbf{A}}_{adp} \mathbf{H}_k \mathbf{W}_{(k,3)} \tag{5.15}$$

The diffusion step used is $J = 3$. The copy of the original input is now added to the

output from the graph convolution layer to get the final output of the hidden layer.

Skip connection and output

Each time the TCN block of a layer produces an output, with dimensions as seen in Table 5.1, a copy of the output is passed to a 1×1 convolution layer that produces $c \times 8$ output channels and is stored in an output queue. (This step is called a residual skip connection. It is used by ResNet architecture to avoid the vanishing gradient problem and to build deeper networks.) All of the outputs in the queue are then summed together producing a tensor with dimensions $[b, c \times 8, n, 1]$. Then ReLU is applied, followed by convolution with a 1×1 kernel, and $c \times 16$ output channels. After that, ReLU is applied again, followed by a final convolution with a 1×1 kernel to produce the traffic speed predictions for the batch: $[b, 12, n, 1]$. All of these steps are visualised in Figure 5.1.

5.5 Available improvements to Graph WaveNet

Changes made to Graph WaveNet by other researchers are acknowledged in this section. Shleifer et al. [22] incrementally improved Graph WaveNet's performance on the traffic prediction task through a series of modifications:

Hyperparameters:

Table 5.1 – Dimensions of output from each TCN block to skip connection.

Layer	Output dimensions
1	$[b, c, n, 12]$
2	$[b, c, n, 10]$
3	$[b, c, n, 9]$
4	$[b, c, n, 7]$
5	$[b, c, n, 6]$
6	$[b, c, n, 4]$
7	$[b, c, n, 3]$
8	$[b, c, n, 1]$

- Implementing learning rate decay by multiplying the learning rate by 0.97 after each epoch.
- Increasing the number of convolution channels from 32 to 40.
- Clipping gradients to $L2 = 3$ gives better results than clipping gradients to $L2 = 5$ as originally done by Graph WaveNet.
- Replacing the 0s in the training data with the average speed in the training data.

Skip connections: In each block, Shleifer et al. [22] add the output of the Gated TCN to the output of the GCN, providing a skip connection around the GCN. (The original Graph WaveNet model already provides a skip connection around the GCN and does not provide one after the GCN. Looking at the code provided, it is not clear that any additional change was made by Schleifer et al. [22] in this regard.) **Pretraining:** They improve the final model's performance by pretraining the model for shorter prediction horizons.

Shleifer et al. [22] also tried various approaches/modifications that did not work. Here are some of the experiments that failed to improve the results:

1. Adding the day of the week information.
2. Using 75 minutes of historic data instead of 60.
3. Cyclical learning rates.
4. Transfer learning from PEMS-BAY to METR-LA.
5. Larger batch sizes and half-precision.
6. Using transformers instead of 1D convolution.
7. Training using the information discarded by the dilation in the TCN.
8. Initialising the learned adjacency matrix from the single value decomposition of the road network.

9. Using RMSE instead of MAE for training.

10. Reducing/increasing the size of the learned node embedding.

11. Adding batch normalisation after the graph convolution step.

5.6 Conclusion

The building blocks of Graph WaveNet are explained, followed by an in-depth description of the Graph WaveNet model. Available minor improvements of Graph WaveNet (added after its initial definition) are then acknowledged.

Chapter 6

Experimental protocol

6.1 Overview

This chapter describes the protocol used for experiments in this study. The performance metrics used throughout this study are given in Section 6.2. Changes made to the available Graph WaveNet tool to improve hyperparameter optimisation efficiency and loss calculation accuracy are described in Section 6.3. A hyperparameter optimisation strategy is chosen by comparing different optimisation approaches in Section 6.4. A possible solution to the problems caused by zero values in the dataset (as previously described in Chapter 3) is implemented in Section 6.5.

6.2 Performance metrics

The performance metrics used throughout this document are Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), as defined in Equations 6.1, 6.2, and 6.3, respectively. In these equations, x_i is the predicted speed, and y_i is the true speed over n measurements.

$$\mathsf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{6.1}$$

$$\mathsf{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}} \tag{6.2}$$

$$\mathsf{MAPE} = \frac{\sum_{i=1}^{n} \left| \frac{y_i - x_i}{y_i} \right|}{n} \times 100 \tag{6.3}$$

6.3 Validation loss calculation corrections

The changes made to the available Graph WaveNet tool to improve hyperparameter optimisation efficiency and loss calculation accuracy are explained in the following subsections.

6.3.1 Batch problem 1

Graph WaveNet excludes predictions made for zero values in the dataset when calculating MAE, MAPE, and RMSE. Consider the MAE given by Equation 6.6. Since there are missing/zero values in the data, only the value for which data is available is considered, meaning that the MAE represents only one prediction. Now consider the MAE given by Equation 6.9. There are no missing values, meaning the MAE represents 10 predictions.

If we calculate the average MAE of the 2 "batches" using Equation 6.12 the MAE representing only one predicted value is given the same weight as the MAE value representing 10 predicted values. To overcome this, the MAE must instead be calculated as in Equation 6.13. The incorrect method of calculating the MAE of the validation loss as given by Equation 6.12 is what is used in the available Graph WaveNet code. This negatively affects hyperparameter optimisation because it results in a false correlation being learnt between validation loss and batch size. We calculate the MAE using Equation 6.13 in all

experiments. Since this change does not affect achievable performance apart from during hyperparameter optimisation, the recreated published results in Section 4.3.3 were not recreated again.

$$true_1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 10] \tag{6.4}$$

$$pred_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 8] \tag{6.5}$$

$$MAE_1 = MAE(true_1, pred_1) = 2 \tag{6.6}$$

$$true_2 = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12] \tag{6.7}$$

$$pred_2 = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13] \tag{6.8}$$

$$MAE_2 = MAE(true_2, pred_2) = 1 \tag{6.9}$$

$$true_t = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] \tag{6.10}$$

$$pred_t = [1, 2, 3, 4, 5, 6, 7, 8, 9, 8, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] \tag{6.11}$$

$$MAE_f = \frac{(MAE_1 + MAE_2)}{2} = 1.5 \tag{6.12}$$

$$MAE_t = MAE(true_t, pred_t) = 1.0909 = \frac{(MAE_1 \times 1 + MAE_2 \times 10)}{11} \tag{6.13}$$

6.3.2 Batch problem 2

To ensure all batches have the same size the last sample in the data partition is used to pad the data partition. The number of values padded is given by Equation 6.14 which could cause a false relationship between batch size and validation loss during hyperparameter optimisation, especially since the validation set is much smaller than the training data partition.

$$\mathsf{num\_padding} = \mathsf{batch\_size} - (\mathsf{data\_size}\, mod(\mathsf{batch\_size})) \tag{6.14}$$

This problem is overcome by instead padding with 0 values meaning that the padded values do not affect the calculated MAE if the MAE is calculated correctly as described above. This method is implemented in all of the experiments.

6.4 Hyperparameter optimisation protocol evaluation

Here different approaches to hyperparameter optimisation is explored to obtain a better understanding of Graph WaveNet's training process. To compare the different approaches each approach is implemented for 40 runs. Weights & Biases[1] is used for hyperparameter optimisation. Bayesian search is implemented to reduce the time needed for optimisation. A single seed (99) is used for each of the implementations. Early stopping is implemented to ensure that the Bayesian optimisation is done using only the best validation loss from each run.

6.4.1 Baseline implementation

Each run is allowed to train for 50 epochs using hyperparameters in the ranges as seen in Table 6.1. By evaluating Figure 6.1 we can see that the optimisation algorithm quickly converges and does not then improve for the last 20 runs. By considering Figure 6.2 we see that the best validation loss of each run is achieved close to the last epoch. On average the best validation loss is achieved on epoch 38, and the best validation loss was achieved on epoch 31 for the best run. The optimisation process might thus benefit from increasing the number of epochs.

[1] https://wandb.ai/site

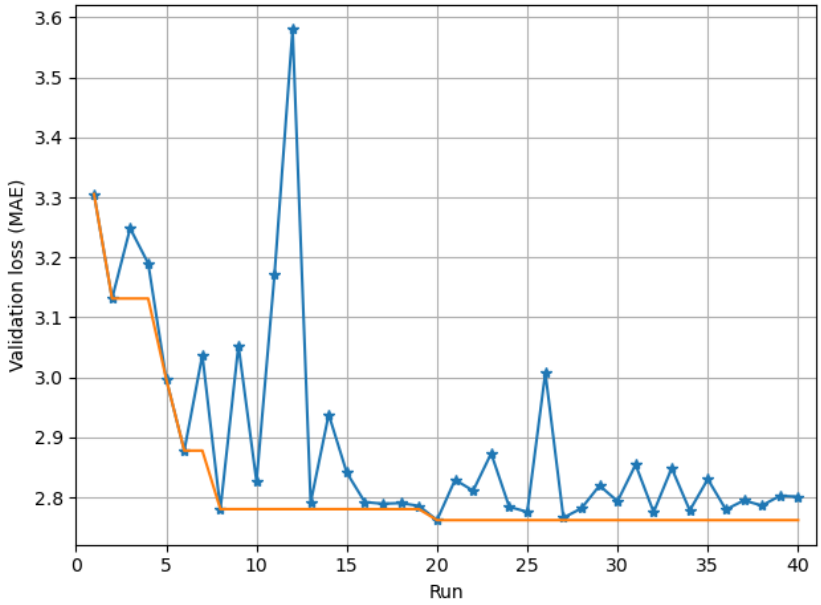


Figure 6.1 – Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs. Each data point represents the best validation loss achieved for the run. The orange line shows how the best validation loss improves as more runs are completed.

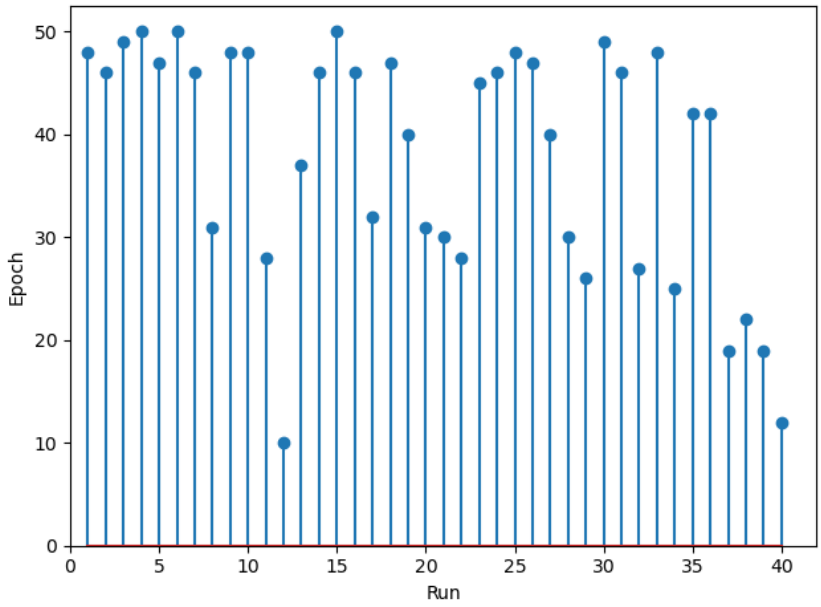


Figure 6.2 – Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs. Each data point represents the epoch on which the best validation loss was achieved for the run.

Table 6.1 – Hyperparameter optimisation parameters

Parameter	Min	Max	Distribution
nhid	20	120	int_uniform
batch_size	16	128	int_uniform
learning_rate	0.0001	0.0100	uniform
dropout	0.01	0.60	uniform
weight_decay	0.00	0.01	uniform

6.4.2 Baseline implementation: reduced parameters

Each run is allowed to train for 50 epochs using hyperparameters in the ranges as shown in Table 6.2. The number of possible hyperparameter combinations is reduced by reducing the number of possible values for the batch size and nhid (the number of channels used). We also specifically include the parameter values used to achieve the published Graph WaveNet results. This is done to ensure that the optimisation process is able to select these values if they are truly the best combination. In Section 6.4.1 the optimisation algorithm can assign any integer value to these parameters within the range given by the minimum and maximum values. Here we only allow the algorithm to use specific values for these parameters given in the form of a list of integers. This is done to restrict the search space, allowing more emphasis to be placed on finding the ideal learning rate, dropout, and weight decay combination, for a smaller set of nhid, and batch size combinations.

By evaluating Figure 6.3 we can see that the optimisation algorithm again quickly converges and does not then improve for the last 26 runs. By considering Figure 6.4 we see that the best validation loss of each run is again achieved close to the last epoch. On average the best validation loss is achieved on epoch 37, and the best validation loss was achieved on epoch 27 for the best run.

Table 6.2 – Hyperparameter optimisation parameters reduced

Parameter	Min	Max	Distribution
learning_rate	0.0001	0.01	uniform
dropout	0.01	0.6	uniform
weight_decay	0	0.01	uniform
nhid	[20, 32, 40, 60, 80, 100, 120]		
batch_size	[16, 32, 64, 128]		

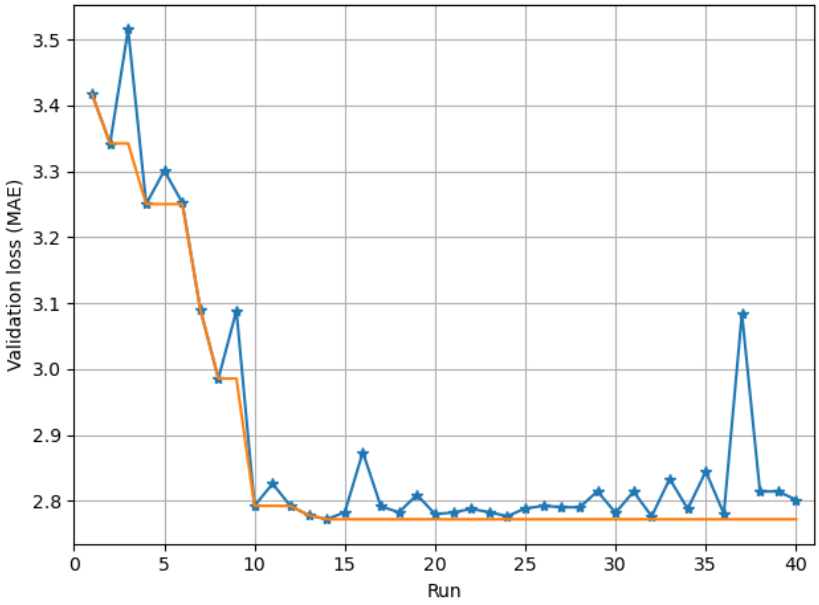


Figure 6.3 – Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs with the number of possible parameters reduced. Each data point represents the best validation loss achieved for the run. The orange line shows how the best validation loss improves as more runs are completed.

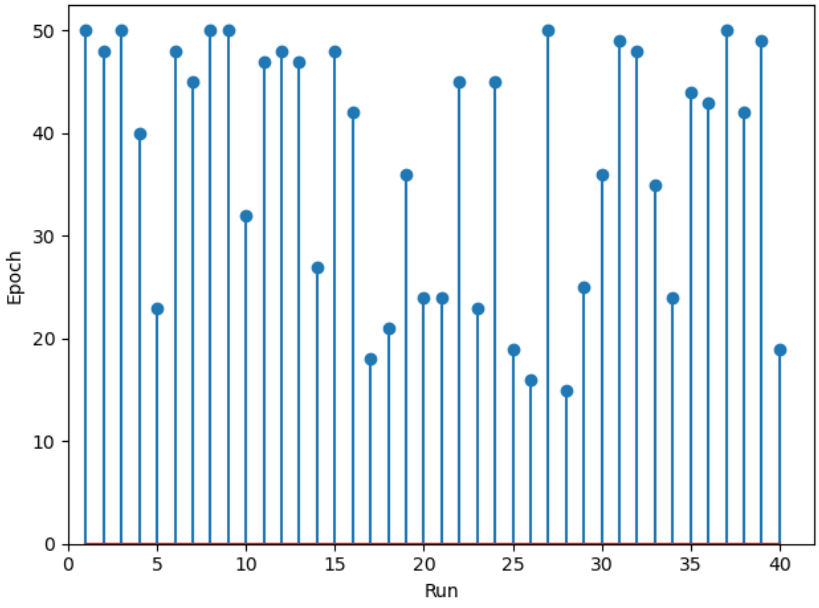


Figure 6.4 – Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 50 epochs with the number of possible parameters reduced. Each data point represents the epoch on which the best validation loss was achieved for the run.

6.4.3 Baseline implementation: increased epochs

Each run is allowed to train for 100 epochs using hyperparameters in the ranges as seen in Table 6.1. We implement early stopping to ensure that the Bayesian optimisation is done using only the best validation loss from each run.

By evaluating Figure 6.5 we can see that the optimisation algorithm again quickly converges and does not then improve for the last 17 runs. By considering Figure 6.6 we see that the best validation loss of each run is achieved on average at epoch 65, and on epoch 51 for the best run. Thus, the optimisation algorithm would probably not greatly benefit from increasing the number of epochs.

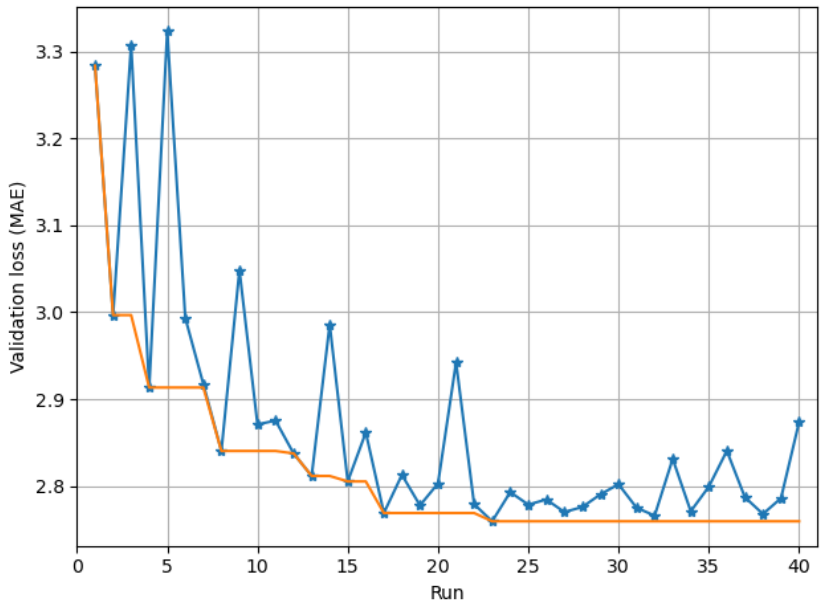


Figure 6.5 – Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 100 epochs. Each data point represents the best validation loss achieved for the run. The orange line shows how the best validation loss improves as more runs are completed.

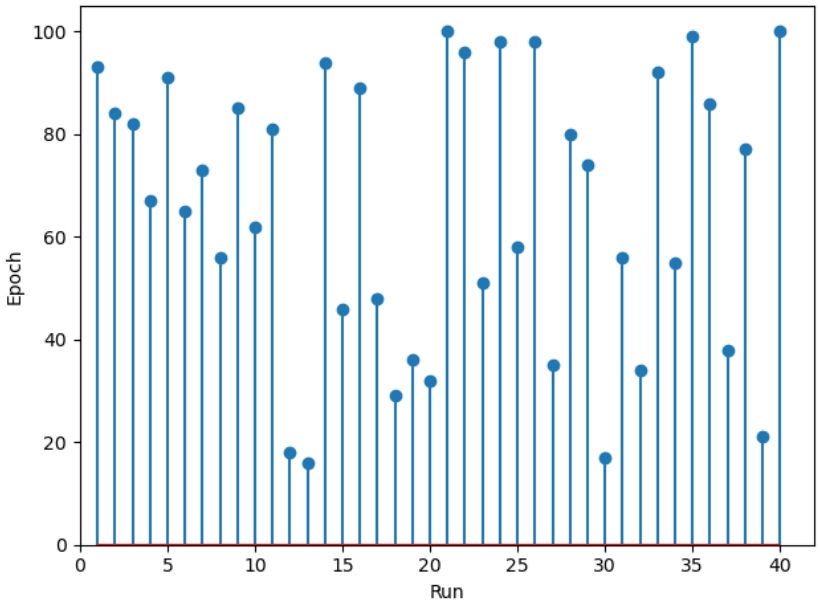


Figure 6.6 – Graph WaveNet Bayesian hyperparameter optimisation on METR-LA dataset, runs trained for 100 epochs. Each data point represents the epoch on which the best validation loss was achieved for the run.

6.4.4 Dynamically extended training

Each run is trained for at least 50 epochs. After 50 epochs if the validation loss has not improved during the last 20% of epochs, training is stopped, otherwise, training is continued for an additional 20% of epochs. This process is repeated on the last epoch of each run until there is no improvement in the last 20% of epochs. This technique is used to ensure the convergence of each run.

By evaluating Figure 6.7 we can see that the optimisation algorithm struggles to converge. By considering the information represented by Figure 6.8 we determine that the best validation loss of each run is achieved on average at epoch 95, and on epoch 60 for the best run.

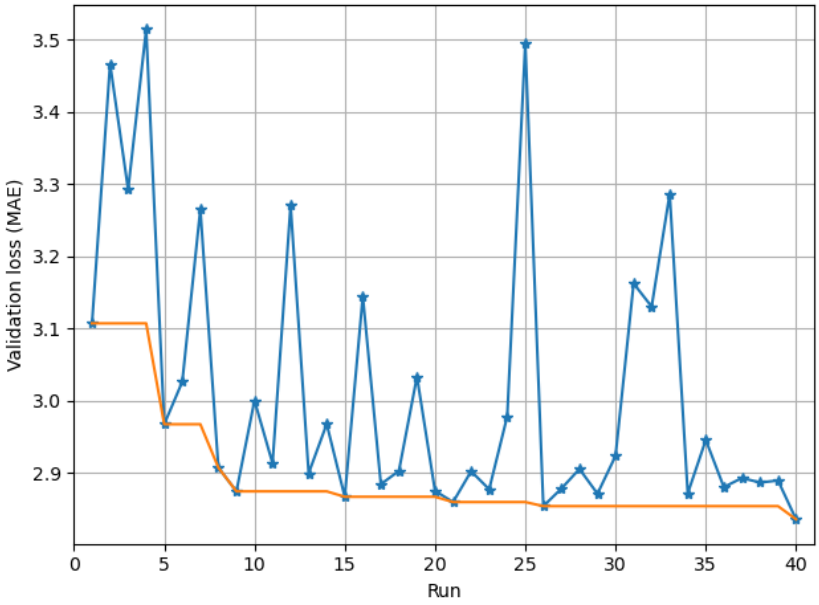


Figure 6.7 – Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training on METR-LA dataset. Each data point represents the best validation loss achieved for the run. The orange line how shows the best validation loss improves as more runs are completed.

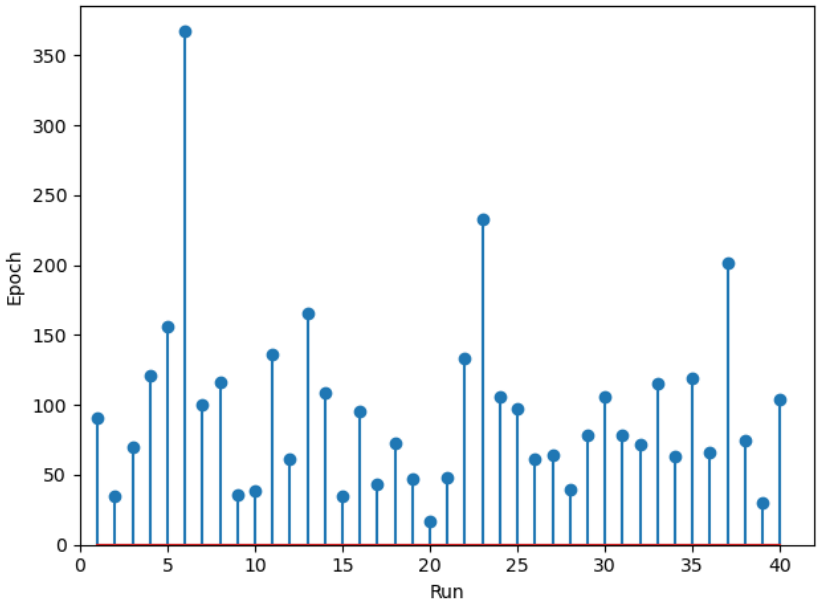


Figure 6.8 – Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training on METR-LA dataset. Each data point represents the epoch on which the best validation loss was achieved for the run.

6.4.5 Dynamically extended training: reduced parameters

The same technique is implemented as in Section 6.4.4 but, this time using parameter ranges as seen in Table 6.2.

By evaluating Figure 6.9 we can see that the optimisation algorithm struggles to converge. By considering the information represented by Figure 6.10 we determine that the best validation loss of each run is achieved on average at epoch 95, and on epoch 95 for the best run.

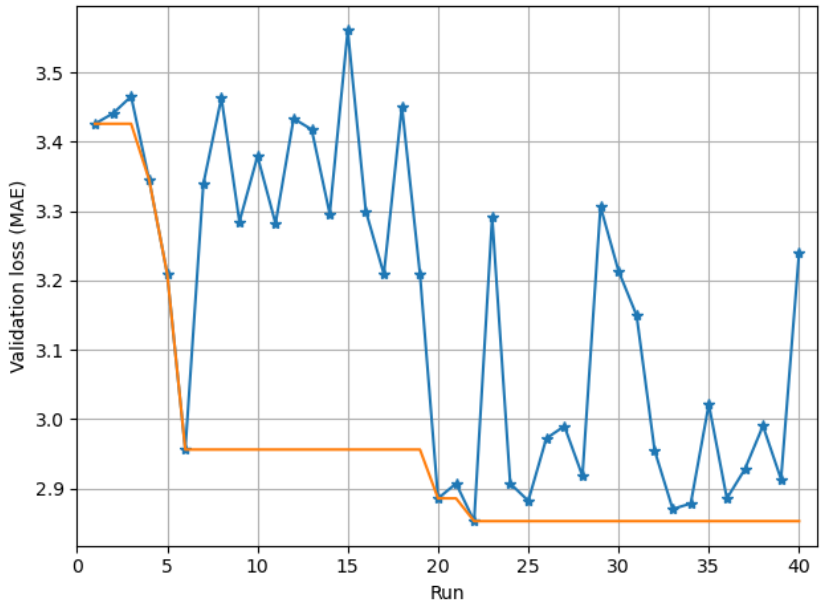


Figure 6.9 – Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training and a reduced number of parameter values to search over on METR-LA dataset. Each data point represents the best validation loss achieved for the run. The orange line shows the how best validation loss improves as more runs are completed.

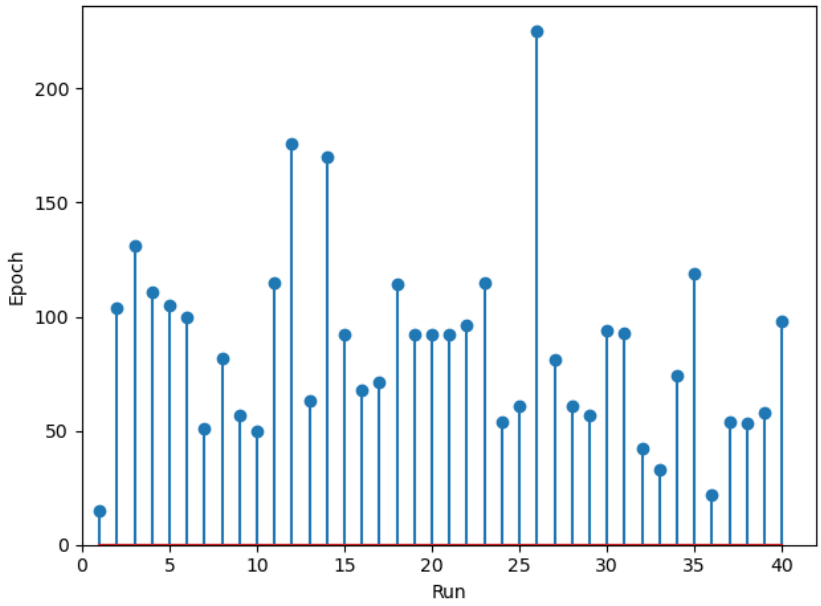


Figure 6.10 – Graph WaveNet Bayesian hyperparameter optimisation with dynamically increased epochs during training and a reduced number of parameter values to search over on METR-LA dataset. Each data point represents the epoch on which the best validation loss was achieved for the run.

6.4.6 Results comparison

The results achieved using each of the optimisation approaches are considered, summarised in Table 6.3, to make an informed decision when choosing which method to use during experiments.

- The mean best epoch when allowing runs to train until no improvement is observed in the last 20% of epochs is 95 epochs for the first method we used and 86 for the reduced parameters version.
- When doubling the number of epochs of each run we only see a 0.002 MAE improvement. This improvement becomes much smaller after 100 epochs while drastically increasing training time.
- The best performance was achieved when only implementing Bayesian optimisation with no additional complexities.

Table 6.3 – Hyperparameter optimisation approaches results and compute time comparison. MAE and RMSE given in miles per hour.

Protocol	**Best mean validation loss (MAE)**	**Compute time (hh:mm:ss)**	**Mean best epoch**
50 epochs	2.762	40:22:10	38
50 epochs reduced parameters	2.772	54:44:58	37
100 epochs	2.760	70:47:36	65
dynamic extend 1	2.835	97:35:45	95
dynamic extend 2	2.853	106:44:29	86

Based on these observations we choose to implement the optimisation method used in Section 6.4.1 when conducting experiments. In addition, after the optimisation is concluded for 40 runs, the hyperparameters of the best run are used to train the model for an additional 100 epochs to ensure convergence. Results using this approach are shown in Table 6.4. From these results, we can see that the performance achieved is very similar to the performance achieved using the recreated Graph WaveNet model.

Table 6.4 – Performance achieved with Graph WaveNet when hyperparameter optimisation is done compared to the performance achieved by the model trained using published hyperparameters from Section 4.3.3. Results are given for the METR-LA validation and training data partitions. MAE and RMSE given in miles per hour.

	15 min			**30 min**			**60 min**		
	MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Recreated train	2.50	6.09%	4.76	2.75	7.15%	5.48	3.08	8.47%	6.35
Recreated val	2.47	6.26%	4.83	2.8	7.72%	5.79	3.15	9.28%	6.76
Optimized train	2.45	6.12%	4.62	2.69	7.19%	5.34	3.00	8.47%	6.16
Optimized val	2.47	6.43%	4.81	2.81	7.89%	5.78	3.17	9.44%	6.72

6.5 Missing data

Graph WaveNet assigns 0 loss to predictions for which the sensor reading is 0. This is done because a 0 sensor reading indicates that no vehicles passed the sensor during the 5-minute interval. But the METR-LA dataset contains instances where all sensors have a 0 reading at the same time which is most likely due to missing data. Zeroes in the data also contribute to the error of predictions of nearby sensors.

The model must learn that lower speed readings indicate more traffic and higher speed readings indicate less traffic. Zero values indicating no traffic add extra difficulty to the learning process. To overcome this, zero values in the input data are replaced by using linear interpolation.

Three models are optimised on interpolated data using the optimisation method described in Section 6.4.1. The performance achieved from the three optimised models on interpolated data from the METR-LA dataset is given in Table 6.5. In Table 6.6 the mean performance of the models trained on the interpolated data is given along with the mean performance of the recreated Graph WaveNet results and the results achieved when optimising the Graph WaveNet model using the method described in Section 6.4.1

By interpolating missing values the performance of Graph WaveNet is only improved slightly. Interpolation does not improve the possible achievable performance sufficiently to incorporate it when constructing new models.

6.6 Conclusion

The performance metrics used in this study are given, followed by a description of validation loss calculation problems found in the Graph WaveNet model. These had to be solved before a hyperparameter optimisation protocol could be chosen. While alternative protocols were also experimented with here, the protocol selected for use in the remainder of this study is as described in Section 6.4.1. A possible solution to the missing data issue

Table 6.5 – Performance of Graph WaveNet models trained on data for which missing values are interpolated to reduce training difficulty (the mean performance of the three models is also given). Results are given for METR-LA training and validation data partitions. The performance of the original vanilla Graph WaveNet model is also given. Original model: OM. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
OM	train	2.50	6.09%	4.76	2.75	7.15%	5.48	3.08	8.47%	6.35
	val	2.47	6.26%	4.83	2.80	7.72%	5.79	3.15	9.28%	6.76
Seed 1 (99)	train	2.48	6.11%	4.68	2.73	7.15%	5.40	3.04	8.40%	6.20
	val	2.46	6.27%	4.77	2.80	7.66%	5.74	3.17	9.26%	6.69
Seed 2 (100)	train	2.47	6.10%	4.69	2.73	7.11%	5.46	3.06	8.30%	6.33
	val	2.46	6.26%	4.79	2.80	7.56%	5.77	3.16	8.98 %	6.72
Seed 3 (101)	train	2.49	6.16%	4.72	2.73	7.24%	5.46	3.02	8.49%	6.25
	val	2.46	6.33%	4.80	2.79	7.81%	5.81	3.15	9.44%	6.77
Mean	train	2.48	6.12%	4.70	2.73	7.17%	5.44	3.04	8.40%	6.26
	val	2.46	6.29%	4.79	2.80	7.68%	5.77	3.16	9.23 %	6.73

Table 6.6 – Mean performance for all twelve prediction horizons of Graph WaveNet models trained on data for which missing values are interpolated to reduce training difficulty to Graph WaveNet (the mean performance of the three models is also given). Results are given for METR-LA training and validation data partitions. The performance of the original vanilla Graph WaveNet model is also given. Original model: OM. MAE and RMSE given in miles per hour.

		Mean 5min – 60 min		
		MAE	**MAPE**	**RMSE**
OM	Train	2.72	7.05%	5.38
	Val	2.76	7.57%	5.65
Seed 1 (99)	Train	2.70	7.03%	5.29
	Val	2.76	7.53%	5.60
Seed 2 (100)	Train	2.70	6.99%	5.34
	Val	2.75	7.43%	5.61
Seed 3 (101)	Train	2.69	7.11%	5.34
	Val	2.75	7.67%	5.66
Mean	Train	2.70	7.40%	5.32
	Val	2.75	7.54%	5.62

previously described in Chapter 3 is explored. The performance improvements achieved using interpolation were not significant enough to warrant further use, given the additional complexities that missing data interpolation introduces.

Chapter 7

Impact of congestion on traffic speed prediction

7.1 Overview

Prediction accuracy is of utmost importance during times of congestion, as little action is required from either road users or the traffic management system when traffic is flowing normally. This is an important aspect of traffic prediction methods to evaluate. In this chapter, Graph WaveNet is evaluated during both congested and uncongested periods. Note that aspects of this chapter have been published in Oosthuizen et al. [4] as described in Section 1.8.

First, a binned traffic speed analysis is performed to evaluate model performance during scenarios which have different levels of congestion (Section 7.2). The effect of different congestion thresholds on the performance of Graph WaveNet is analysed in Section 7.3. We then investigate model performance during times of congestion when models are trained using oversampling in Section 7.4 and undersampling in Section 7.5.

Zero values are disregarded when calculations are done regarding thresholds and mean or median traffic speeds for the analysis performed in this chapter.

7.2 Binned traffic speed analysis

The same Graph WaveNet and STAWnet models evaluated in Section 4.3.3 (when recreating published results) are used to evaluate model performance during different congestion scenarios. The MAE of the models' predictions is calculated for different ranges of traffic speeds. This is analysed either by considering the individual sensor speeds, or from overall network speed, which are two different ways in which to bin traffic speed predictions.

In the first case, *binned sensor speed* analysis, the MAE for all individual sensor predictions that fall within a bin for a given horizon are averaged. In the second case, *binned average network speed* analysis, if the average network speed falls within the bin, the MAE of all sensor predictions at that time step are averaged, irrespective of the individual sensor speeds. In both cases, each prediction horizon is kept separate.

The performance of the two analysed models on the METR-LA and PEMS-BAY datasets are shown in Figures 7.1 and 7.2 for the binned average network speed and in Figures 7.3 and 7.4 for the binned sensor speed. The two models perform better for faster traffic speeds on both datasets while performing worse for slower traffic speeds. This is an indication of the extent to which the performance of the traffic prediction models deteriorates during times of congestion in the road network. The noticeable uptick for higher traffic speeds is not discussed since traffic speed prediction is of little use when there are few people using the road. In both cases, the x ticks used are the centre speed value in the bins, and bins all have the same sizes.

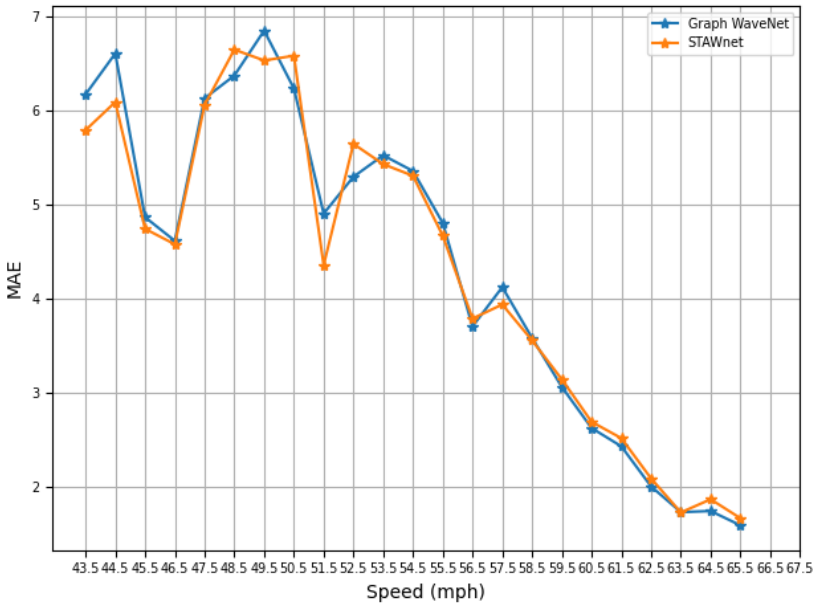


Figure 7.1 – Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon, binned per average network speed: METR-LA, validation data partition.

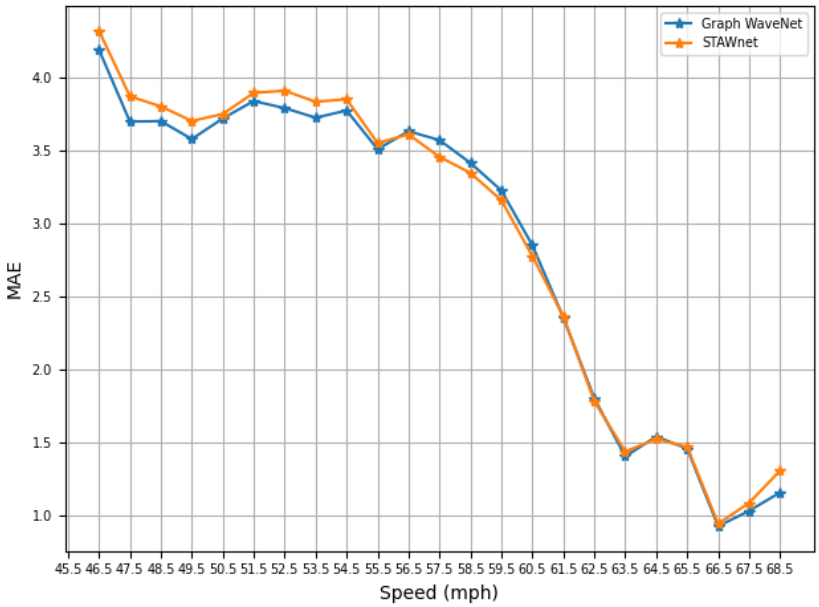


Figure 7.2 – Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon, binned per average network speed: PeMS-Bay, validation data partition.

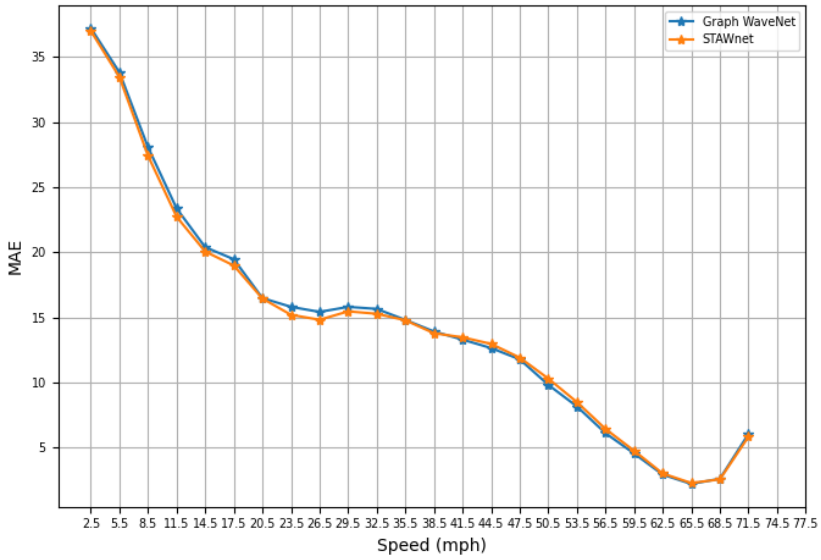


Figure 7.3 – Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon, binned per sensor speeds for METR-LA, validation data partition.

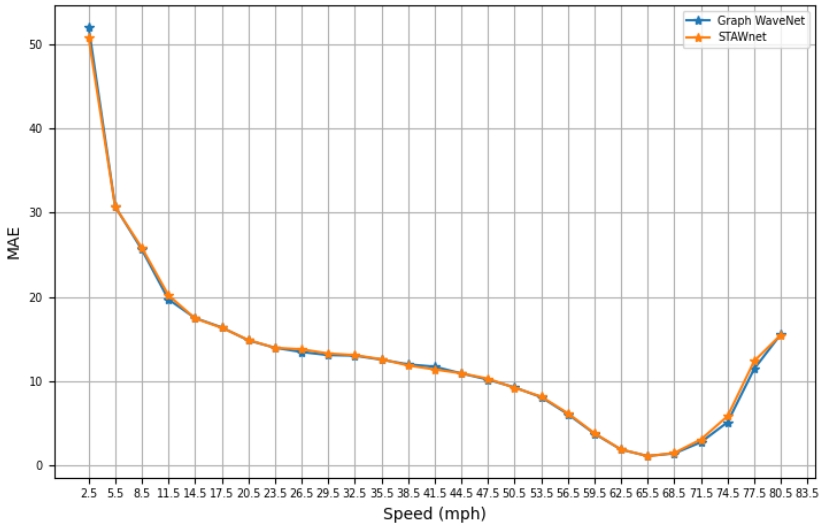


Figure 7.4 – Graph WaveNet and STAWnet performance (MAE mph) for 60-minute prediction horizon binned per sensor speed for PEMS-BAY, validation data partition.

7.3 Congestion analysis

In Figure 7.5 the recurrent congestion observed in the METR-LA training dataset is shown by plotting the average network speed per time of day and day of the week. The first 120 hours represent the average network speed from Monday to Friday and the remaining period represents the weekend. There is a clear drop in average network speed during certain intervals of a typical day. These intervals are also more prominent during weekdays (referred to as 'workdays' for clarity) and less so during weekends. These intervals can be described as periods of congestion and are associated with the peak traffic periods in the morning and afternoon. To select periods of congestion for analysis, a threshold can be used, such as the mean or median traffic speed as seen in Figure 7.5, to separate the congested and not congested data.

In this section, the effect of congestion using different congestion thresholds is analysed in three different ways as well as the effect congestion has on the prediction horizon. The thresholds used for the analysis are first discussed below in Section 7.3.1.

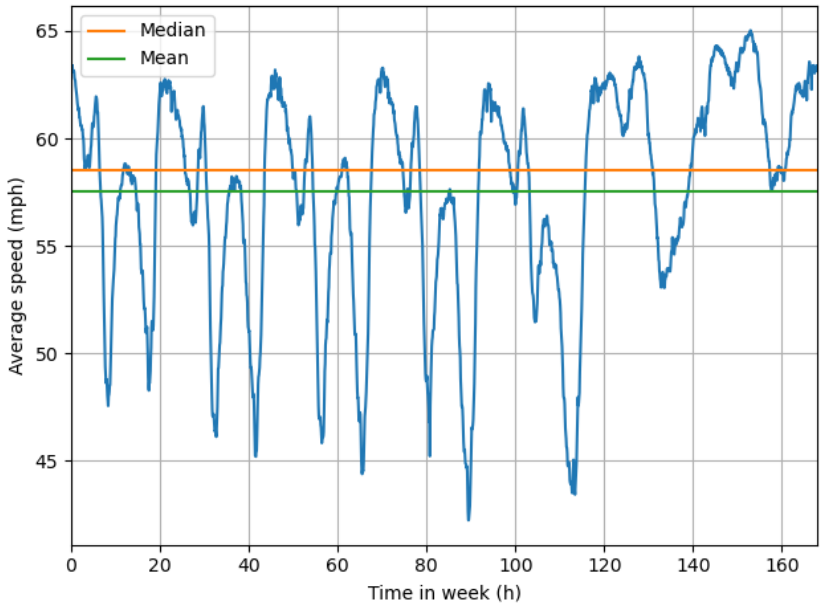


Figure 7.5 – METR-LA training set average network speed per time of day and day of the week.

7.3.1 Thresholds

To compare model performance during times of congestion with performance during the absence of congestion the different congestion thresholds are set around the median speed measured across all sensors, computed using the training partition of the METR-LA and PEMS-BAY datasets. We start with a threshold value that equals the median speed over all sensors and then vary this threshold, to obtain different divisions of the dataset (between congested and not congested), by adding or subtracting a threshold interval value.

Two different size intervals are used for speeds faster and slower than the median threshold, because the data distribution is skewed as can be seen in Figure 7.6. The threshold values that are smaller than the median speed are determined by subtracting multiples of a threshold interval value computed using Equation 7.1 from the median speed, while the thresholds that are larger than the median speed are calculated by adding multiples of the interval value calculated using Equation 7.2 to the median speed value.

$$s_s = \frac{s_c - min(s)}{10} \tag{7.1}$$

$$s_f = \frac{max(s) - s_c}{10} \tag{7.2}$$

In Eq. 7.1 and 7.2 s is the average network speed (across all sensors, per time step), s_c is the centre threshold speed (in this case the median), and s_s and s_f are the speed intervals, slower and faster than the median, respectively.

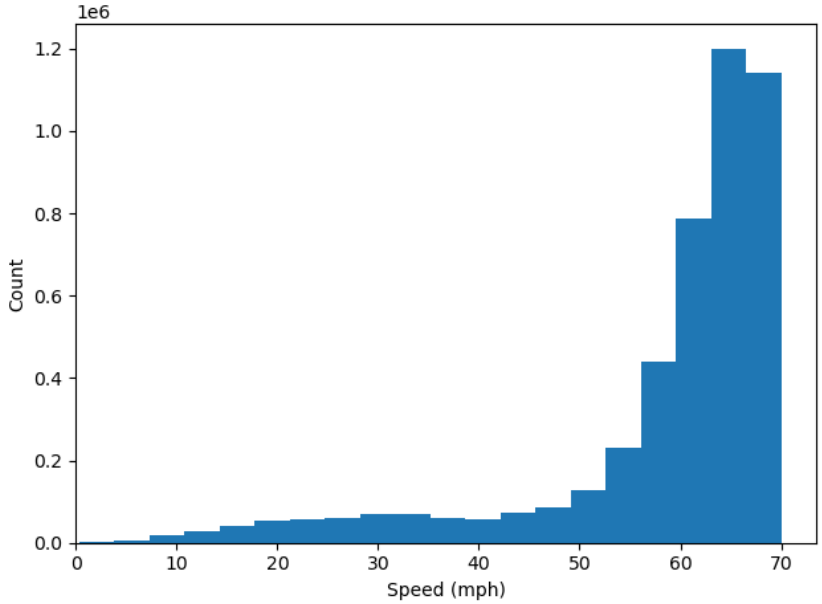


Figure 7.6 – Histogram of METR-LA training partition speed readings of all sensors.

7.3.2 Per time step analysis

Here, a per time step analysis is performed. The average network speed at each time step in the validation set is compared to the threshold speed. If the average network speed is greater than the threshold speed, the network at this time step is not congested, otherwise, the network is congested. The prediction MAE during congested times is then compared to the prediction MAE during times that are not congested across the entire validation set and the complete traffic network.

Figures 7.8 and 7.9 depict the following for different threshold values: the number of observations that form part of the congested and not congested data; the average MAE averaged over all 12 prediction horizons and the MAE for the 60-minute prediction horizon for congested and not congested data; and the difference in MAE between the predictions for congested values and not congested data. From these results the difference in the performance of the models for congested data and not congested data is clear.

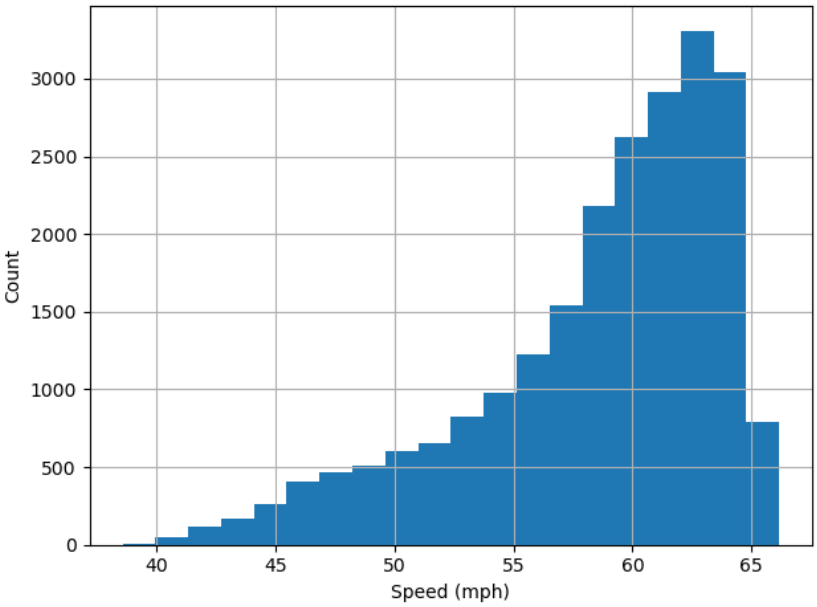


Figure 7.7 – Histogram of METR-LA validation partition average network speed readings of all sensors.

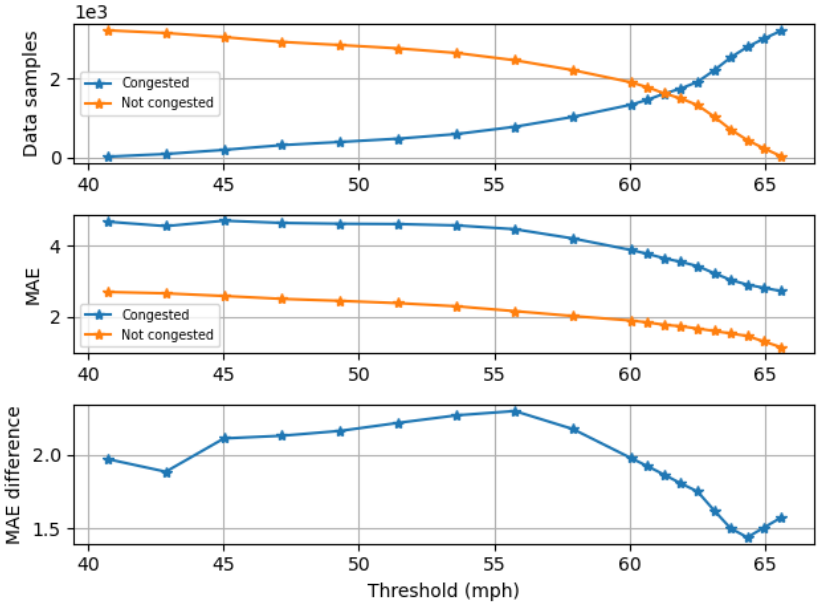


Figure 7.8 – Graph WaveNet per time step congestion analysis: METR-LA, validation set. Mean results over all 12 prediction horizons.

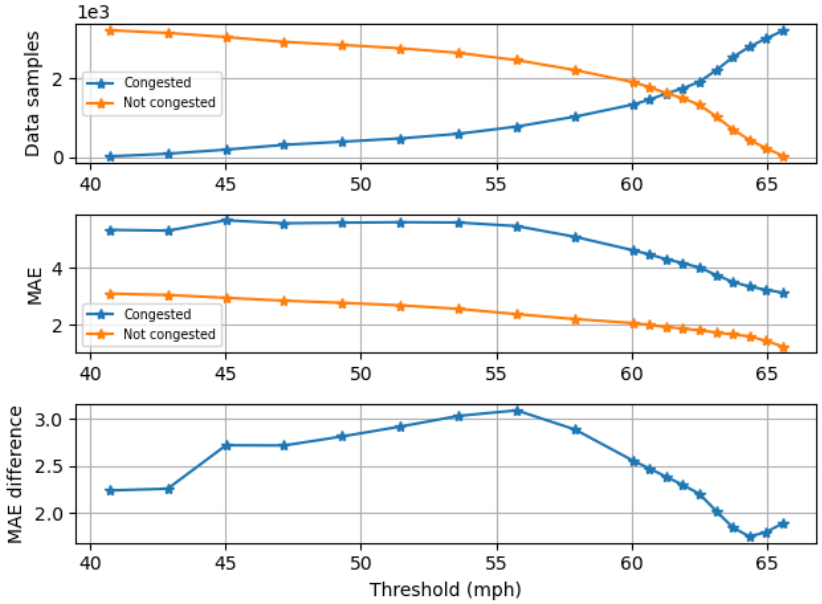


Figure 7.9 – Graph WaveNet per time-step congestion analysis: METR-LA, validation set. Results for the 12th prediction horizon.

7.3.3 Recurrent congestion analysis all days

For the recurrent congestion analysis, a fixed daily division between congested and not congested data is used by calculating the average daily speed over all days for each time of day. The threshold value is then compared with this fixed daily average speed pattern to differentiate between congested and not congested observations. As each prediction uses the past hour of measured speeds, the daily time interval that represents times of recurrent congestion is extracted as follows:

- The time interval is chosen from the hour before the average daily speed drops below the threshold until the hour before the average daily speed goes above the threshold for the last time in the training dataset.
- This time interval of data is then extracted from the dataset as the congested data and the rest of the data is the not congested data.

The performance of the model during the congested time interval is then compared to the performance of the model outside of this interval.

Figure 7.11 depicts the average results over all twelve prediction horizons while Figure 7.12 depicts the results when considering only the 60-minute predictions.

By comparing Figures 7.11 and 7.8 we can also see that by averaging the speed data to extract recurrent congestion data, a smaller difference is obtained in performance between congested and not congested data. This is because by averaging the network speed readings to obtain a daily average we significantly reduce the difference in the tail of the data distribution seen when comparing Figures 7.7 and 7.10.

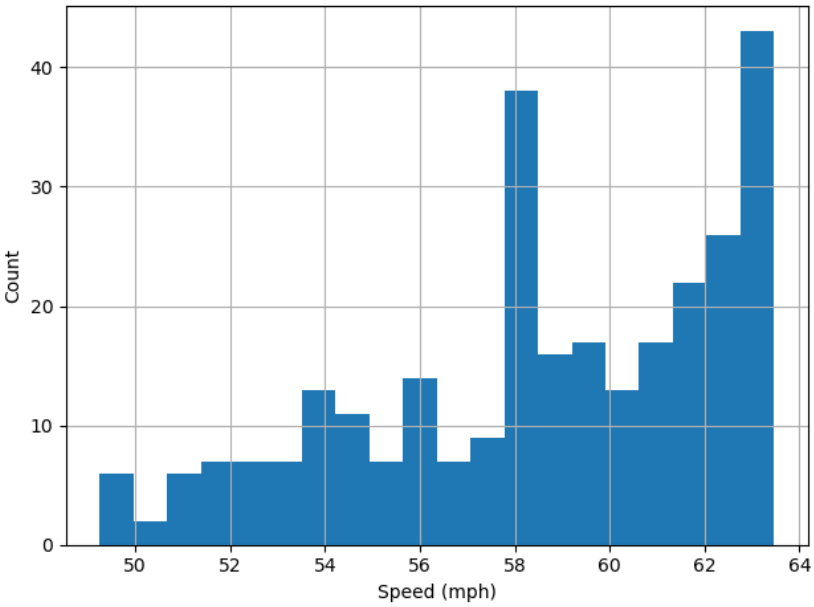


Figure 7.10 – Histogram of METR-LA validation partition average daily network speed readings of all sensors

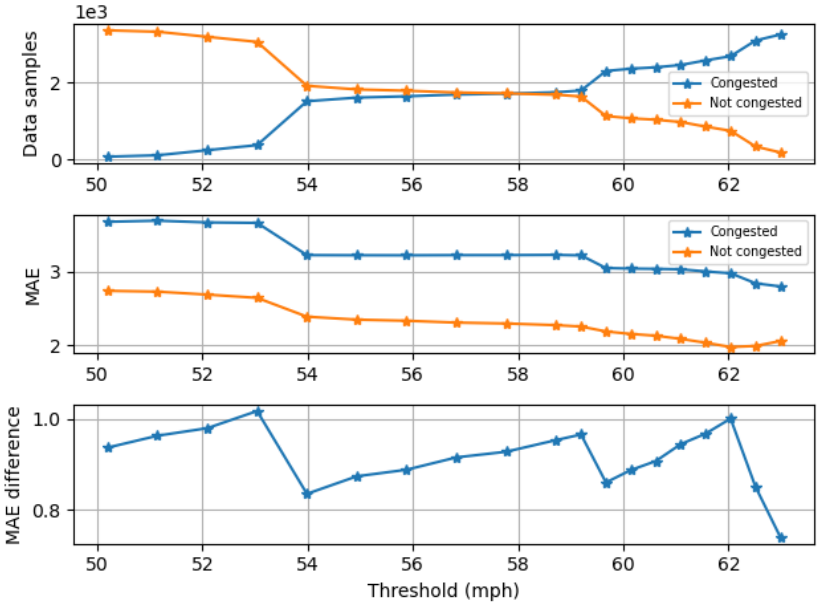


Figure 7.11 – Graph WaveNet recurrent congestion analysis: METR-LA, validation set. Mean results over all 12 prediction horizons.

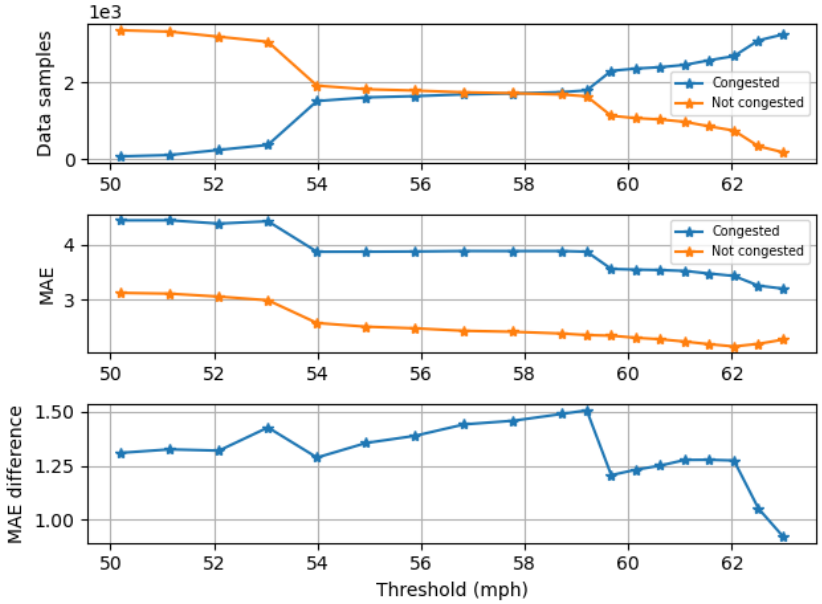


Figure 7.12 – Graph WaveNet recurrent congestion analysis: METR-LA, validation set. Results for the 60-minute prediction horizons.

7.3.4 Recurrent congestion analysis workdays

For the recurrent congestion analysis during workdays, the same method as for the recurrent congestion analysis during all days in Section 7.3.3 is used, to determine recurrent daily congestion patterns, but excluding weekend data.

Figure 7.11 depicts the average results over all twelve prediction horizons while Figure 7.12 depicts the results when only considering the 60-minute predictions.

By comparing Figures 7.11 and 7.14 it is observed that, by excluding weekend data from the recurrent congestion analysis, a bigger difference in performance is obtained between congested and not congested data. This is because, as can be seen in Figure 7.5, weekend data has a smaller interval of congestion during the day and the same recurrent congestion interval can therefore not be used effectively for weekends.

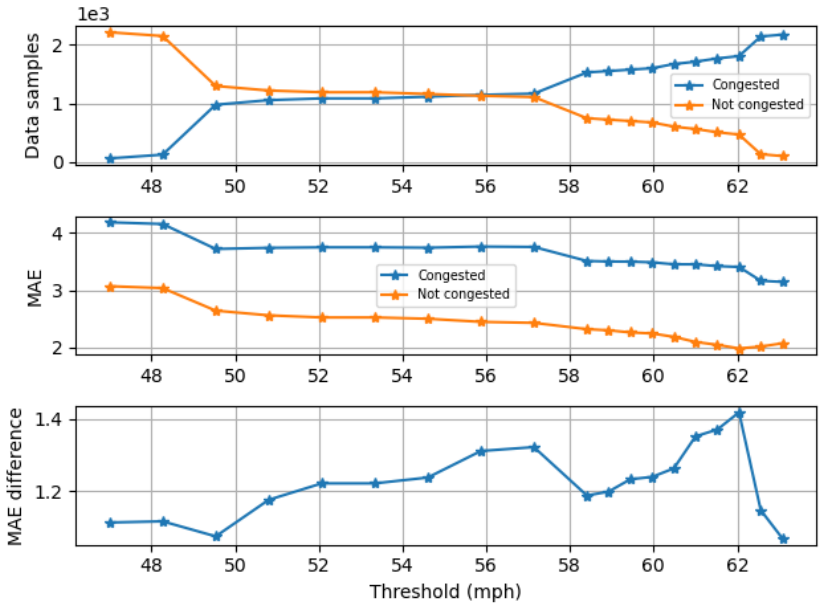


Figure 7.13 – Graph WaveNet workday recurrent congestion analysis: METR-LA, validation set. Mean results over all 12 prediction horizons.

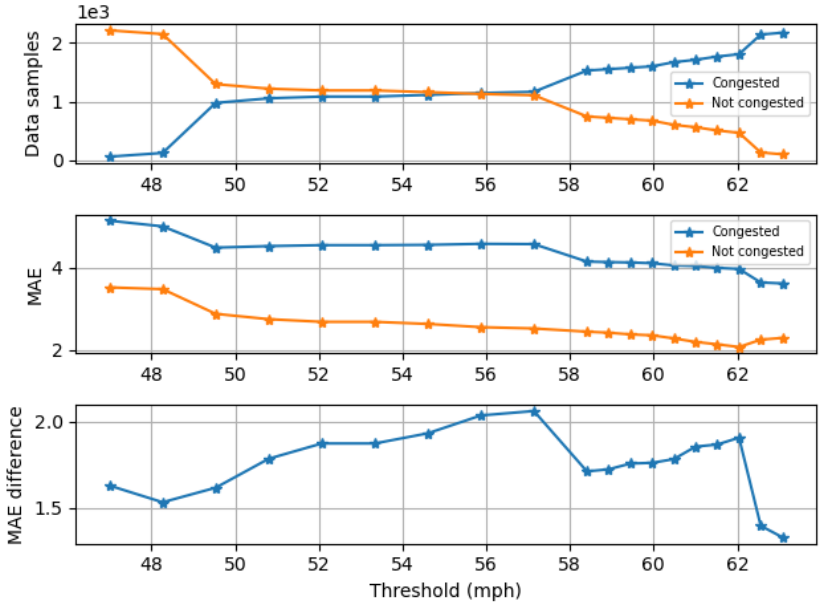


Figure 7.14 – Graph WaveNet workday recurrent congestion analysis: METR-LA, validation set. Results for the 60-minute prediction horizons.

7.4 Congested data boosting

The aim of this section is to train a Graph WaveNet model on the METR-LA dataset to perform better during times of recurrent congestion on workdays by oversampling recurrent congestion data.

7.4.1 Experiment design

A fixed daily time interval is used to classify data as congested or not congested. This time interval is determined using the threshold analysis for recurrent congestion on workdays as described in Section 7.3.4. The data inside of the threshold is extracted from workdays. The extracted congestion data is then added once to the training set.

The results are used when applying the recurrent workday analysis to the training data partition of the METR-LA dataset. This is done since the congested data is extracted from the training data partition. The threshold is the speed value for which the difference in performance between congested and not congested data in the training set is the largest while the number of congested samples is not much less than the not congested data samples. The threshold value is thus 57.14 mph as per the results seen in Table 7.1 for the METR-LA dataset.

Hyperparameter optimisation is done as described in Section 6.4.1 for three different seeds.

7.4.2 Results

The performance of the three Graph WaveNet models, optimised on a congested data boosted training set, on the entire training and validation data partitions is given in Table 7.2. The performance of the three models on congested data only extracted as described in Section 7.4 is given in Table 7.3. The performance of the three models on congested data and not congested data determined using the thresholds seen in Figure 7.13 is visualised in Figures 7.15 and 7.16 for the training and validation data partitions respectively. The

Table 7.1 – Performance of Graph WaveNet when evaluated on recurrent congestion during workdays. Results are given for the METR-LA training partition. MAE is given for mean performance on all twelve prediction horizons and for the 60-minute prediction horizon.

Threshold	Data samples		MAE mean all 12 prediction horizons			MAE 60 min predictions		
(mph)	Congested	Uncongested	Congested	Uncongested	Difference	Congested	Uncongested	Difference
47.00	413	16649	4.25	2.88	1.37	5.27	3.25	2.02
48.26	885	16177	4.22	2.84	1.38	5.11	3.20	1.92
49.53	7198	9864	3.41	2.56	0.85	4.04	2.77	1.27
50.80	7790	9272	3.43	2.49	0.95	4.09	2.64	1.45
52.07	8028	9034	3.44	2.45	0.99	4.11	2.59	1.52
53.33	8028	9034	3.44	2.45	0.99	4.11	2.59	1.52
54.60	8267	8795	3.44	2.43	1.00	4.11	2.55	1.57
55.87	8504	8558	3.45	2.39	1.06	4.13	2.48	1.65
57.14	**8683**	**8379**	**3.44**	**2.38**	**1.06**	**4.13**	**2.45**	**1.68**
58.41	11560	5502	3.23	2.27	0.96	3.75	2.36	1.39
58.93	11740	5322	3.22	2.24	0.98	3.74	2.33	1.41
59.44	11919	5143	3.23	2.21	1.02	3.74	2.30	1.44
59.96	12099	4963	3.22	2.18	1.04	3.73	2.27	1.46
60.48	12637	4425	3.20	2.11	1.10	3.70	2.18	1.52
61.00	12932	4130	3.20	2.03	1.17	3.68	2.11	1.57
61.52	13345	3717	3.18	1.98	1.20	3.65	2.05	1.60
62.04	13640	3422	3.17	1.93	1.24	3.63	2.01	1.62
62.56	16118	944	2.96	2.12	0.84	3.36	2.31	1.05
63.08	16354	708	2.95	2.16	0.79	3.34	2.33	1.01

performance of the recreated Graph WaveNet model used for the analysis is also given. As can be seen from these results, by increasing the congested data in the training data partition the performance that can be obtained for congested data does not improve. Model performance decreases when applied to congested data from the validation data partition. This difference in performance is statistically significant (α=0.05) as evaluated in App A.1.

Table 7.2 – Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds and recreated Graph WaveNet performance. Results are given for the METR-LA training and validation data partitions. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Recreated	train	2.50	6.09%	4.76	2.75	7.15%	5.48	3.08	8.47%	6.35
	val	2.47	6.26%	4.83	2.80	7.72%	5.79	3.15	9.28%	6.76
Seed 1 (99)	train	2.49	6.16%	4.71	2.74	7.21%	5.45	3.06	8.52%	6.28
	val	2.47	6.41%	4.81	2.82	7.85%	5.81	3.19	9.46%	6.79
Seed 2 (100)	train	2.46	6.08%	4.65	2.71	7.20%	5.37	3.04	8.52%	6.23
	val	2.47	6.38%	4.81	2.82	7.91%	5.78	3.21	9.59%	6.80
Seed 3 (101)	train	2.44	5.96%	4.59	2.66	6.97%	5.26	2.98	8.22%	6.11
	val	2.47	6.34%	4.81	2.82	7.83%	5.82	3.18	9.34%	6.78
Mean	train	2.46	6.07%	4.65	2.70	7.13%	5.36	3.03	8.42%	6.21
	val	2.47	6.38%	4.81	2.82	7.86%	5.80	3.19	9.46%	6.79

Table 7.3 – Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds and recreated Graph WaveNet performance. Results are given for congested data only extracted from the METR-LA training and validation data partitions using a fixed daily threshold for work days as described in Section 7.4. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Recreated	train	3.12	8.90%	5.95	3.56	10.82%	7.00	4.08	13.02%	8.21
	val	3.28	10.02%	6.29	3.90	12.89%	7.70	4.56	15.96%	9.15
Seed 1 (99)	train	3.05	8.89%	5.79	3.47	10.65%	6.80	3.91	12.52%	7.84
	val	3.27	10.31%	6.24	3.93	13.16%	7.73	4.59	16.11%	9.15
Seed 2 (100)	train	2.99	8.74%	5.70	3.41	10.69%	6.70	3.89	12.69%	7.80
	val	3.26	10.17%	6.21	3.93	13.26%	7.67	4.61	16.45%	9.11
Seed 3 (101)	train	2.94	8.50%	5.59	3.29	10.17%	6.49	3.75	12.07%	7.59
	val	3.28	10.19%	6.25	3.96	13.14%	7.77	4.61	15.99%	9.18
Mean	train	2.99	8.71%	5.69	3.39	10.50%	6.66	3.85	12.43%	7.74
	val	3.27	10.22%	6.23	3.94	13.19%	7.72	4.60	16.18%	9.15

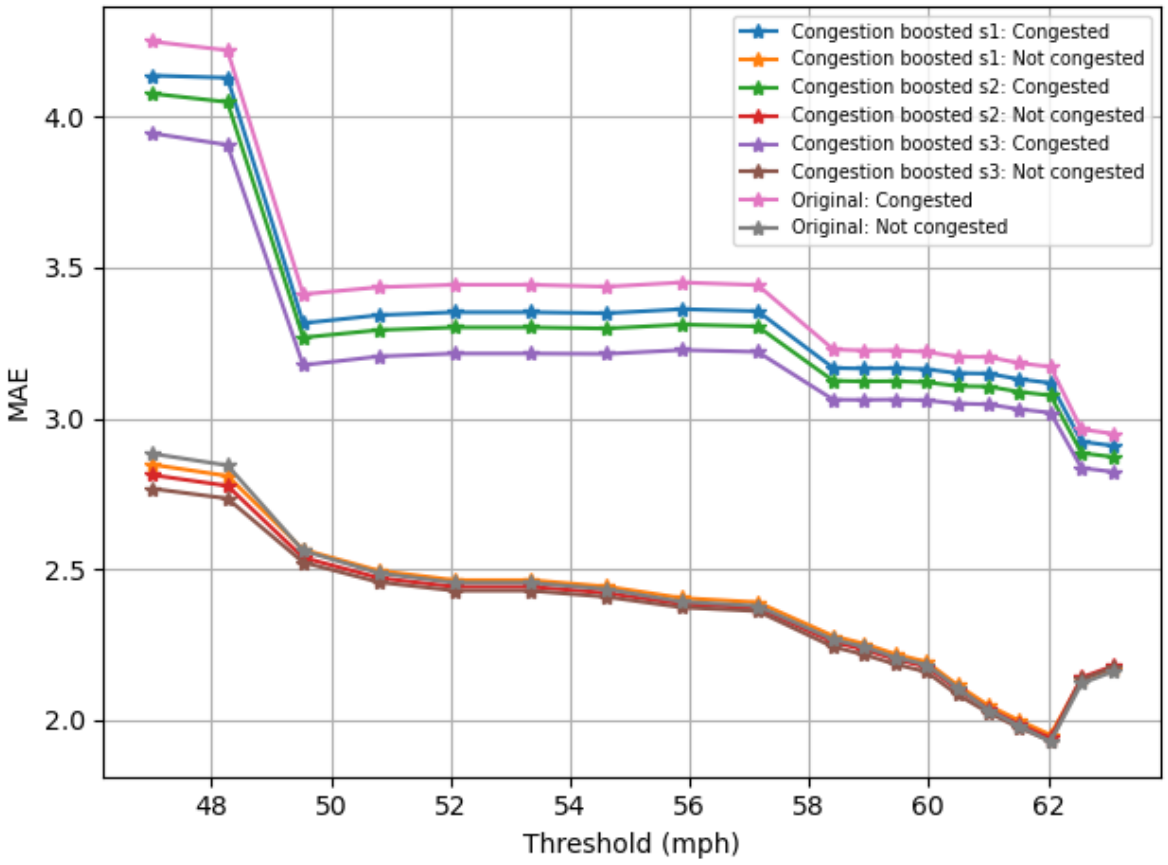


Figure 7.15 – Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA training data partition.

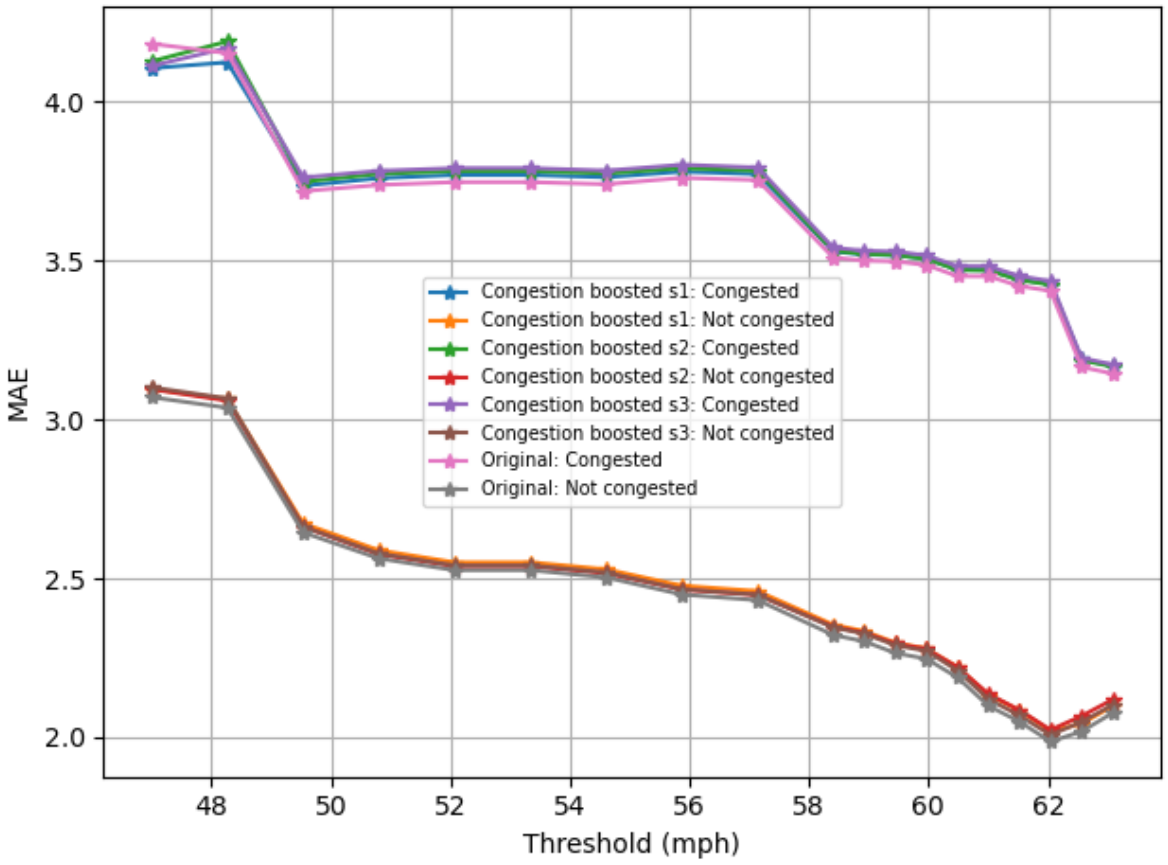


Figure 7.16 – Performance of Graph WaveNet optimised on congested data boosted training data for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA validation data partition.

7.5 Congested data only

Here, the aim is to improve Graph-WaveNet's performance on congested data by only training the model on congested data.

7.5.1 Experiment design

A fixed daily time interval is used to classify data as congested or not congested. This time interval is determined using the threshold analysis for recurrent congestion on workdays in Section 7.3.4. The data inside of the threshold is extracted from workdays only. The congested data is extracted based on the same criteria used in Section 7.4. The extracted congestion data is then used for optimising three Graph WaveNet models each with a different seed.

Hyperparameter optimisation is done as described in Section 6.4.1 for three different seeds.

7.5.2 Results

The performance of the three Graph WaveNet models optimised only on congested data from the training data partition when implemented on the entire validation and training data partitions is given in Table 7.4 and in Table 7.5 for only the congested data. The performance of the three models on congested data and not congested data determined using the thresholds as described in Section 7.3.1 is visualised in Figure 7.17 for the training data partition and in Figure 7.18 for the validation data partition along with the performance of the original Graph WaveNet model used for the analysis.

As can be seen, the model trained only on congested data does not perform better on congested data than the original Graph WaveNet model. This difference in performance is statistically significant (α=0.05) as evaluated in App A.1.

Table 7.4 – Performance of Graph WaveNet optimised only on congested data from the training data partition for three different seeds (the mean performance of the three models is also given) along with recreated Graph WaveNet performance. Results are given for the METR-LA training and validation data partitions. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Recreated	train	2.50	6.09%	4.76	2.75	7.15%	5.48	3.08	8.47%	6.35
	val	2.47	6.26%	4.83	2.80	7.72%	5.79	3.15	9.28%	6.76
Seed 1 (99)	train	2.95	7.19%	5.43	3.41	8.78%	6.54	3.98	10.82%	7.73
	val	2.81	7.04%	5.29	3.23	8.63%	6.33	3.75	10.66%	7.46
Seed 2 (100)	train	2.78	6.76%	5.13	3.18	8.19%	6.11	3.70	10.28%	7.24
	val	2.73	6.84%	5.15	3.21	8.60%	6.32	3.78	10.96%	7.58
Seed 3 (101)	train	2.83	6.95%	5.25	3.29	8.65%	6.36	3.90	10.83%	7.66
	val	2.76	6.95%	5.20	3.24	8.83%	6.42	3.85	11.10%	7.69
Mean	train	2.85	6.97%	5.27	3.29	8.54%	6.34	3.86	10.64%	7.54
	val	2.77	6.94%	5.21	3.23	8.69%	6.36	3.79	10.91%	7.58

Table 7.5 – Performance of Graph WaveNet optimised on congested data only from the training data partition for three different seeds (the mean performance of the three models is also given) along with recreated Graph WaveNet performance. Results are given for congested data from the METR-LA training and validation data partitions. The congested data used is described in Section 7.5. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Recreated	train	3.12	8.90%	5.95	3.56	10.82%	7.00	4.08	13.02%	8.21
	val	3.28	10.02%	6.29	3.90	12.89%	7.70	4.56	15.96%	9.15
Seed 1 (99)	train	3.42	9.88%	6.46	4.06	12.17%	7.82	4.68	14.34%	9.07
	val	3.48	10.65%	6.56	4.12	13.21%	7.93	4.78	16.07%	9.28
Seed 2 (100)	train	3.10	8.92%	5.86	3.46	10.51%	6.77	3.85	12.41%	7.72
	val	3.35	10.23%	6.34	4.01	13.02%	7.86	4.74	16.37%	9.36
Seed 3 (101)	train	3.21	9.28%	6.07	3.74	11.58%	7.34	4.35	13.91%	8.70
	val	3.37	10.29%	6.37	4.07	13.36%	7.94	4.86	16.69%	9.57
Mean	train	3.24	9.36%	6.13	3.75	11.42%	7.31	4.29	13.55%	8.50
	val	3.40	10.39%	6.42	4.07	13.20%	7.91	4.79	16.38%	9.40

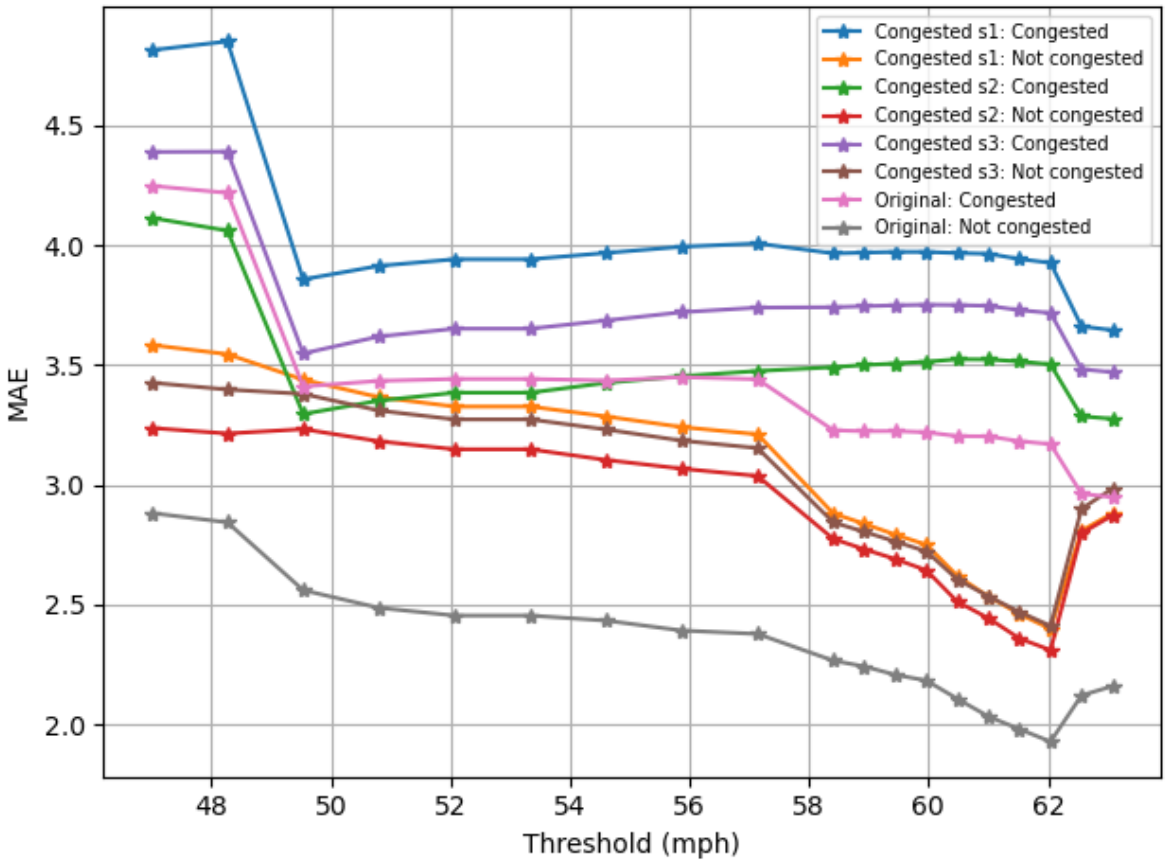


Figure 7.17 – Mean performance of Graph WaveNet optimised on congested data only from the training data partition for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA training data partition.

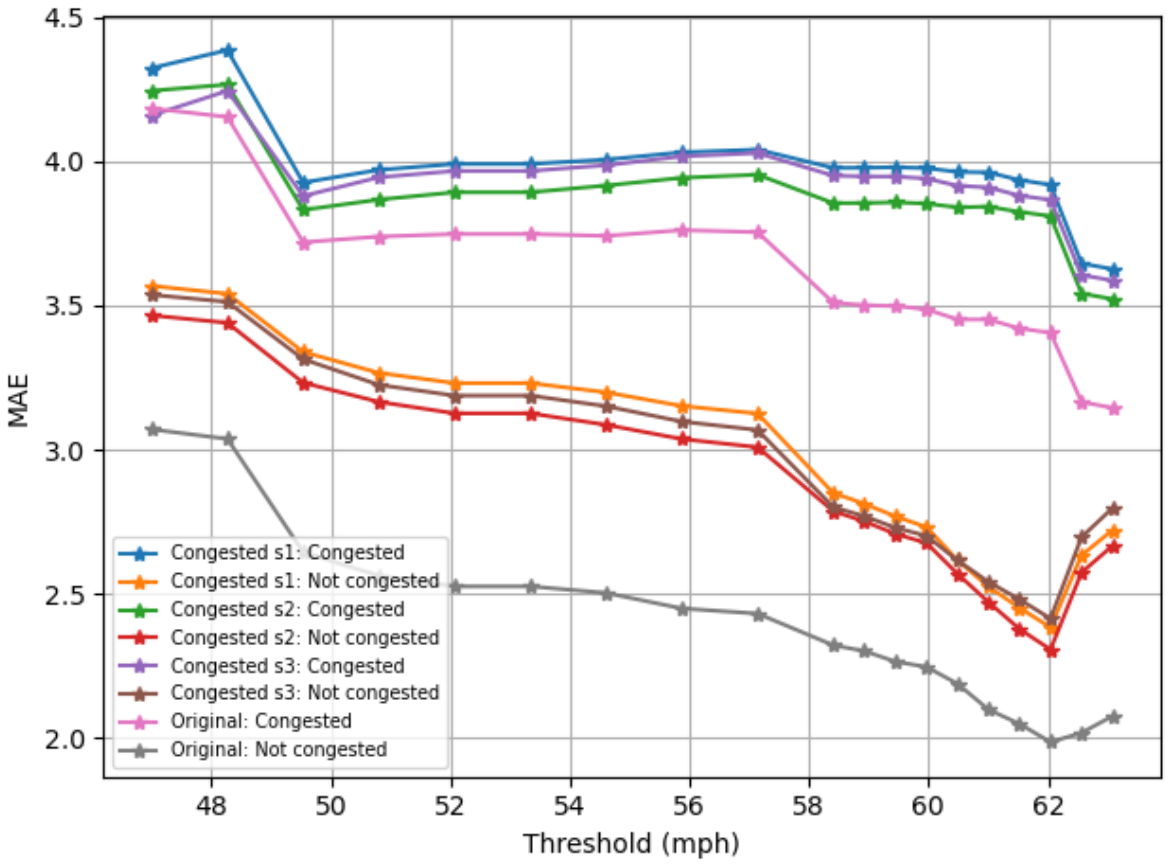


Figure 7.18 – Mean performance of Graph WaveNet optimised on congested data only from the training data partition for three different seeds along with recreated Graph WaveNet performance on congested and not congested data using different thresholds. Results are given for the METR-LA validation data partition.

7.6 Conclusion

First, a binned traffic speed analysis was performed to evaluate model performance during different congestion scenarios. The performance of Graph WaveNet for traffic speed prediction during periods of congestion per time step in the PEMS-BAY and METR-LA datasets was then analysed as well as performance during recurrent congestion. We found that model performance deteriorates during congestion and that a longer prediction horizon has a large effect on performance during periods of congestion, and a surprisingly limited effect otherwise.

We investigated whether it is possible to improve model performance during times of congestion using oversampling in Section 7.4 and undersampling in Section 7.5. These simple methods did not improve model performance during periods of congestion.

Chapter 8

Additional analysis

8.1 Overview

In this chapter, we analyse additional aspects of the Graph WaveNet architecture. Specifically, we perform a trend analysis to determine if Graph WaveNet would benefit from additional historical data (Section 8.2). We investigate the effect of shared vs individual kernels (Section 8.3) and consider how time is encoded by the model (Section 8.4).

8.2 Trend analysis

A trend analysis is performed using the Graph WaveNet model's performance on the PEMS-BAY dataset. The weekly trend of the data per sensor is used for the analysis. The PEMS-BAY dataset is used since there are fewer missing data values in the dataset. The question this analysis aims to answer is whether or not the Graph WaveNet model would benefit from receiving additional trend data during training. To answer this question the trend for each node in the network is computed. The trend of a node is taken as the average speed for a specific time of day for each day of the week. The trend is calculated using the training data partition.

Figures 8.1 and 8.2 visualise the trend for two nodes, node 400073 and node 407339, in the PEMS-BAY dataset in red. The true speed value measured at each node is given in blue, and the predictions made for each time step 5-minutes and 60-minutes into the future are given in orange and green. Predictions made using the PEMS-BAY validation data partition are used for the analysis while the trend is calculated using the training data partition. The node considered in Figure 8.1 is the node for which prediction errors in the network are the largest. Other nodes were also considered but the nodes visualised in Figures 8.1 and 8.2 are good representations of what was observed in the rest of the network.

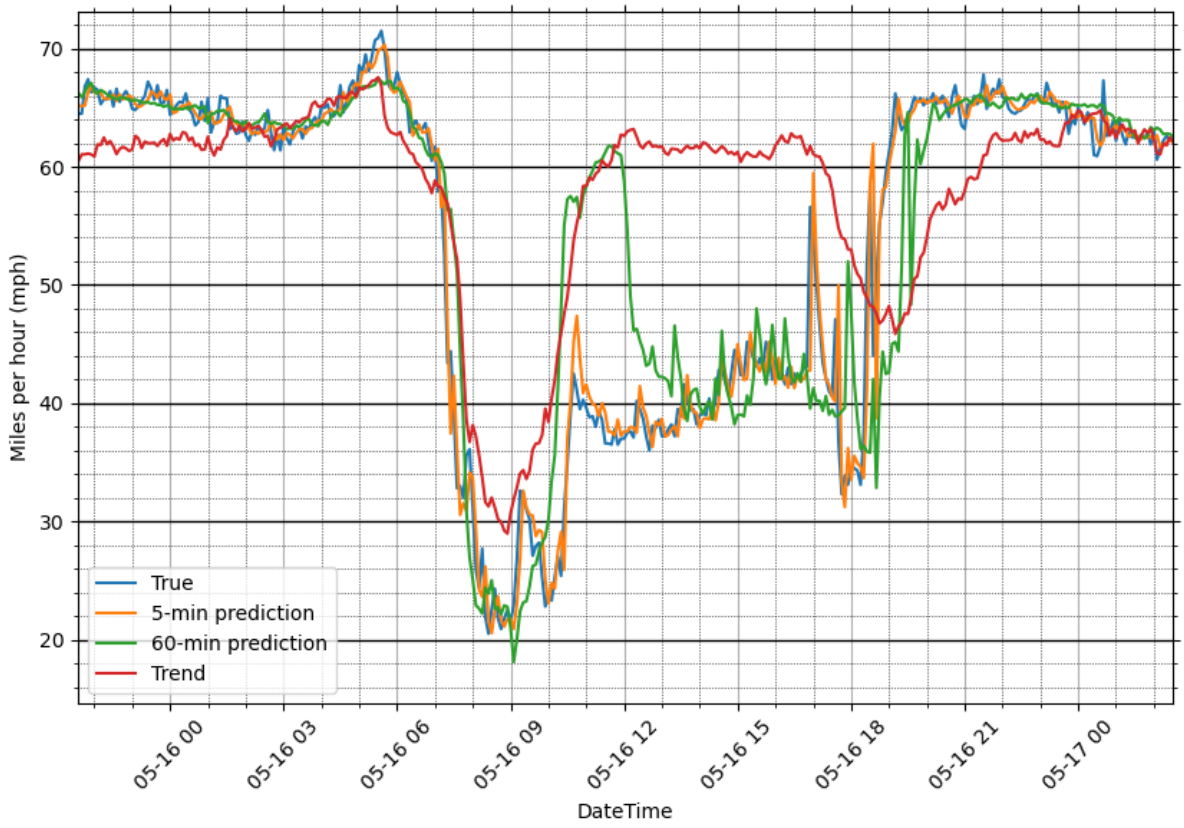


Figure 8.1 – Trend analysis for sensor 407339 in the PEMS-BAY dataset. Sensor 407339 is the sensor for which prediction errors are the largest. Predicted and true traffic speed values are based on the PEMS-BAY validation data partition while the trend is calculated using the training data partition.

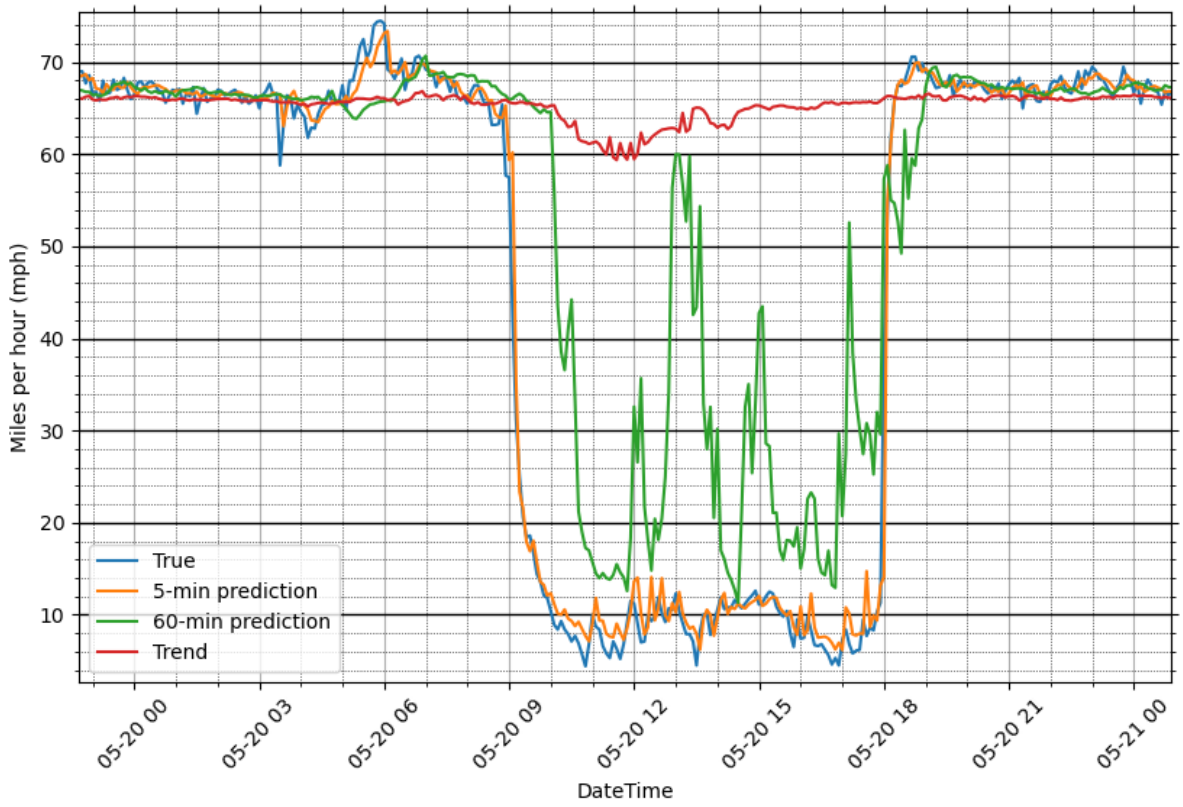


Figure 8.2 – Trend analysis on sensor 400073 in the PEMS-BAY dataset. Sensor 400073 is randomly selected. Predicted and true traffic speed values are based on the PEMS-BAY validation data partition while the trend is calculated using the training data partition.

8.2.1 Discussion

As seen in Figures 8.1 and 8.2 the Graph WaveNet model does well in utilising input data when predicting the future traffic speed and does not merely follow the trend, even though the system is clearly aware of the expected trend. The 60-minute predictions follow the trend of the data until the available data suggests otherwise. This suggests that the Graph WaveNet model would not benefit from receiving additional trend data during predictions; receiving additional trend data might even have a negative impact on model training. The 60-minute prediction lies between the trend line and the true values, not on the opposite side of the true values. Thus, the model is already utilising trend information effectively and further pushing predictions towards the trend would not be beneficial.

8.3 Individual kernels

Even though it does not seem likely that Graph WaveNet would benefit from trend data, as established in Section 8.2, the difference in error per sensor varies, especially for the METR-LA dataset as seen in Table 8.1, meaning that the Graph WaveNet models could benefit from a more individual approach per sensor.

Table 8.1 – Minimum, maximum, mean, and variance of MAE when considering Graph WaveNet model performance on individual sensors in the PEMS-BAY en METR-LA traffic networks using the validation data partition from each of these datasets. MAE given in miles per hour.

	PEMS-BAY	**METR-LA**
min MAE	0.18	1.01
max MAE	4.55	5.50
mean MAE	1.58	2.76
var MAE	0.36	1.01

8.3.1 Experiment design

To extract more individual information from each sensor in the network we add an additional block to the Graph WaveNet model as seen in Figure 8.3. The GCN block used is the same as the one used in the rest of the network. A TCN is created for each of the sensors in the network (or in other words the TCN has a unique kernel for each of the sensors in the traffic network); a new hyperparameter is created that determines the number of output channels from each of these small TCNs. The number of output channels is the same for each of the small TCNs. The new model architecture can be seen in Figure 8.3.

The hyperparameter tuning technique used is described in Section 6.4.1 except for the "nhid" parameter and the new hyperparameter, "nhid_ik", which determines the number of output channels from the individual kernels layer. The range of "nhid_ik" is chosen as 2 to 6 since making this hyperparameter larger significantly increases the training time of the network. The individual kernels TCN increase the training time to such an extent that the range of "nhid" is reduced to 10 to 40.

8.3.2 Results

The performance obtained from the best models can be seen in Tables 8.2 and 8.3 where it is given along with the performance of the original Graph WaveNet model. The inter-node performance comparison is given in Table 8.4. The additional individual kernels block added to Graph WaveNet does not increase model performance nor does it decrease the inter-node performance variance of the models.

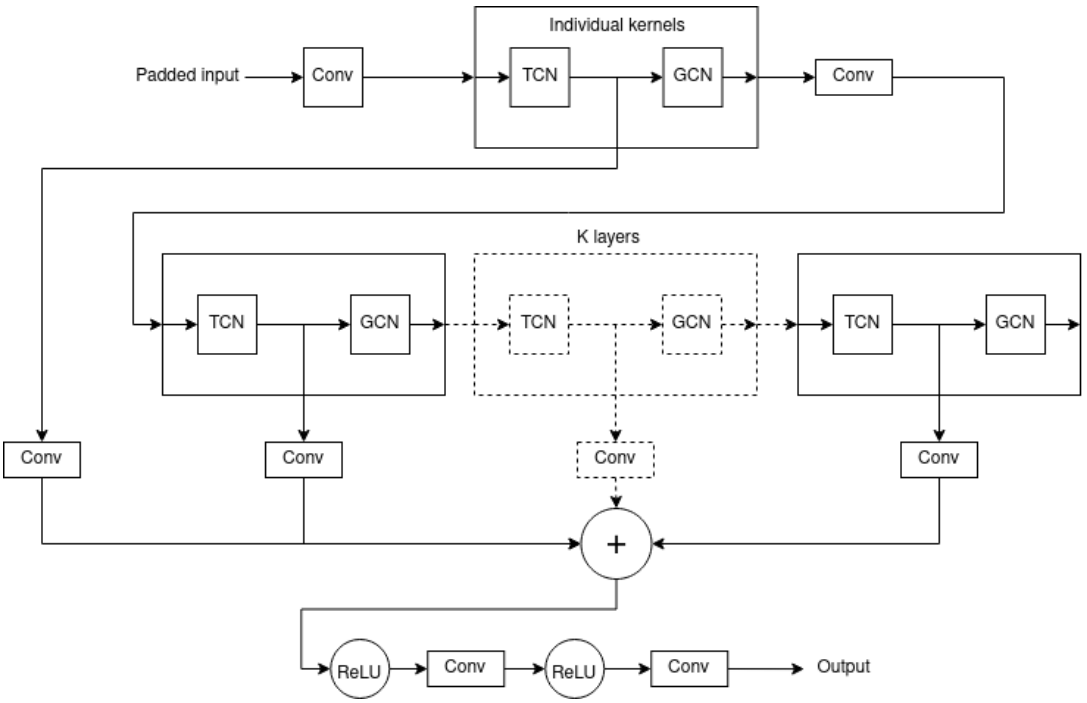


Figure 8.3 – Architecture of the Graph WaveNet model with an additional individual kernels layer inserted. The TCN used in the individual kernels block applies a unique kernel to the input from each of the individual kernels in a traffic network.

Table 8.2 – Performance of individual kernels models for 15 minutes, 30 minutes, and 60 minutes predictions on METR-LA validation and training partitions along with mean performance of three individual kernels models and original Graph WaveNet model performance. Original model: OM. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
OM	train	2.50	6.09%	4.76	2.75	7.15%	5.48	3.08	8.47%	6.35
	val	2.47	6.26%	4.83	2.80	7.72%	5.79	3.15	9.28%	6.76
Seed 1 (99)	train	2.54	6.35%	4.87	2.81	7.43%	5.64	3.10	8.54%	6.43
	val	2.52	6.55%	4.95	2.87	8.07%	6.00	3.24	9.65%	5.84
Seed 2 (100)	train	2.51	6.29%	4.77	2.78	7.34%	5.53	3.11	8.63%	6.39
	val	2.49	6.43%	4.83	2.83	7.80%	5.83	3.23	9.46%	6.87
Seed 3 (101)	train	2.51	6.19%	4.76	2.79	7.18%	5.55	3.14	8.42%	6.45
	val	2.48	6.37%	4.83	2.82	7.67%	5.81	3.22	9.21%	6.87
Mean	train	2.52	6.28%	4.80	2.79	7.32%	5.57	3.12	8.53%	6.42
	val	2.50	6.45%	4.87	2.84	7.85%	5.88	3.23	9.44%	6.53

Table 8.3 – Mean performance of individual kernels models for 5 minutes to 60 minutes predictions on METR-LA validation and training partitions along with mean performance of three individual kernels models and original Graph WaveNet model performance. Original model: OM. MAE and RMSE given in miles per hour.

		Mean 5 min – 60 min		
		MAE	**MAPE**	**RMSE**
OM	train	2.72	7.05%	5.38
	val	2.76	7.57%	5.65
Seed 1 (99)	train	2.76	7.25%	5.50
	val	2.83	7.90%	5.84
Seed 2 (100)	train	2.75	7.24%	5.42
	val	2.80	7.72%	5.70
Seed 3 (101)	train	2.76	7.09%	5.44
	val	2.79	7.57%	5.69
Mean	train	2.76	7.19%	5.45
	val	2.81	7.73%	5.74

Table 8.4 – Minimum, maximum, mean, and variance of MAE for individual sensors in the METR-LA valisation data partition. Results are given for the individual kernels models and original Graph WaveNet model. Original model: OM, individual kernels model: IK. MAE given in miles per hour.

	OM	**IK seed 1**	**IK seed 2**	**IK seed 3**
min MAE	1.01	1.01	1.01	1.00
max MAE	5.50	5.67	5.55	5.48
mean MAE	2.76	2.83	2.80	2.79
var MAE	1.01	1.08	1.04	1.02

8.4 Encoding time as cyclical continuous feature

The time of day is encoded as a value between 0 and 1 for the Graph WaveNet model. This causes an unnecessary discontinuity in the data. Since time is cyclical it can be better interpreted by a deep learning model when it is encoded as a cyclical feature. One way of encoding time as a cyclical feature is to perform a sine and cosine transformation of the feature as described by Equation 8.1 and 8.2. If we visualise the two features determined using Equation 8.1 and 8.2 we get Figure 8.4.

$$t_{\mathsf{sin}} = \mathsf{sin}(\frac{2\pi t}{\mathsf{max}(t)}) \tag{8.1}$$

$$t_{\mathsf{cos}} = \mathsf{cos}(\frac{2\pi t}{\mathsf{max}(t)}) \tag{8.2}$$

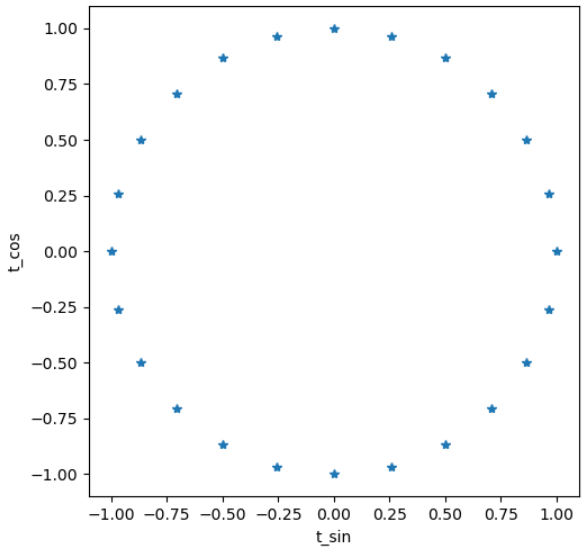


Figure 8.4 – Encoding time in a cyclical way.

8.4.1 Experiment design

Hyperparameter optimisation is done for three models, each initialised with a different seed. The hyperparameter optimisation method used is described in Section 6.4.1. The time column of the METR-LA dataset is encoded as a cyclical feature using Equation 8.1 and 8.2, meaning that time is now represented by two input features.

8.4.2 Results

The performance of the Graph WaveNet models, on the validation and training data partitions of the METR-LA dataset, is given in Tables 8.5 and 8.6. These are the models for which the best performance was achieved during optimisation for each of the three seeds.

As can be seen, encoding time as a cyclical feature does not improve the Graph WaveNet model's performance. This could be because the discontinuity caused by encoding time as a value between 0 and 1 happens in the middle of the night when traffic data is very predictable due to the low traffic flow.

Table 8.5 – Graph WaveNet models trained using time encoded as cyclical feature performance on METR-LA validation and training data partitions. The performance of the original vanilla Graph WaveNet model is given for reference. Original model: OM. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
OM	train	2.50	6.09%	4.76	2.75	7.15%	5.48	3.08	8.47%	6.35
	val	2.47	6.26%	4.83	2.80	7.72%	5.79	3.15	9.28%	6.76
Seed 1 (99)	train	2.48	6.18%	4.68	2.73	7.29%	5.43	3.05	8.72%	6.24
	val	2.50	6.51%	4.88	2.85	8.02%	5.93	3.25	9.85%	6.94
Seed 2 (100)	train	2.41	6.01%	4.55	2.60	6.84%	5.12	2.88	8.04%	5.90
	val	2.48	6.55%	4.84	2.82	7.96%	5.85	3.18	9.52%	6.82
Seed 3 (101)	train	2.49	6.09%	4.71	2.73	7.11%	5.45	3.05	8.36%	6.28
	val	2.49	6.32%	4.86	2.82	7.66%	5.86	3.22	9.21%	6.91
Mean	train	2.46	6.09%	4.65	2.69	7.08%	5.33	2.99	8.37%	6.14
	val	2.49	6.46%	4.86	2.83	7.88%	5.88	3.22	9.53%	6.89

Table 8.6 – Graph WaveNet models trained using time encoded as cyclical feature mean performance for 12 prediction horizons on METR-LA validation and training data partitions. The performance of the original vanilla Graph WaveNet model is given for reference. Original model: OM. MAE and RMSE given in miles per hour.

		Mean 5min – 60 min		
		MAE	**MAPE**	**RMSE**
OM	train	2.72	7.05%	5.38
	val	2.76	7.57%	5.65
Seed 1 (99)	train	2.70	7.21%	5.32
	val	2.82	7.93%	5.78
Seed 2 (100)	train	2.58	6.76%	5.04
	val	2.78	7.80%	5.70
Seed 3 (101)	train	2.70	7.00%	5.34
	val	2.78	7.53%	5.73
Mean	train	2.66	6.99%	5.23
	val	2.79	7.75%	5.74

8.5 Conclusion

We evaluated different ways in which the Graph WaveNet model could be adjusted and found that the baseline model already deals well with the considered complexities. No further improvement was observed in either of the following three cases:

- Additional historical data. (This was not evaluated empirically, but a trend analysis performed suggested that this would not be a fruitful avenue to explore.)
- Additional kernels for each node in the network.
- Encoding time as a cyclical feature.

In Chapter 9 we use the baseline model when developing a model for the South African data since the baseline model already deals well with the complexities observed.

Chapter 9

Application to SA data

9.1 Overview

In this chapter, two new South African datasets are introduced and Graph WaveNet is implemented on them. As they were recorded in Johannesburg and Cape Town, these datasets are referred to from here onward as the JHB and CPT datasets. First, the JHB and CPT datasets are analysed. Graph WaveNet is then implemented on the datasets and the results are discussed followed by a binned traffic speed and trend analysis.

9.2 INRIX data

INRIX is a company that pioneered the use of data collected directly from vehicles along with sensor data for traffic management, instead of using only sensor data [32]. In this section, we first explain how data is recorded by INRIX through the use of road segments and Global Positioning System (GPS) devices in vehicles. The JHB and CPT datasets are then described.

9.2.1 Road segments

INRIX makes use of road segments and GPS devices in vehicles instead of sensors in fixed positions to record traffic data. Segments are the basic elements that a road network is divided into [33]. Segments are used to communicate the traffic speeds at specific locations and incidents that occur on roads. There are two types of INRIX road segments: XDSegments and traffic message channels (TMCs):

- XDSegments are created using an INRIX-proprietary road segmentation system [33]. XDSegments break at certain intersections that are determined using rules that are proprietary to INRIX.
- TMC segments are an encoding of TMC location tables [33]. TMC location tables are created by third-party entities and reviewed and certified by the international standards body, Traveller Information Services Association (TISA). Once certified, TMC tables are encoded onto map links by map vendors.

9.2.2 Data sampling

INRIX traffic data is recorded by using the INRIX Application Programming Interface (API) and requesting the data every 5 minutes. This data is then stored in the cloud. Due to the nature of the recording of the data, data entries are sometimes not recorded in exactly 5-minute intervals. Sometimes, due to network errors, data is recorded later than needed. Where necessary the data must then be moved to the closest 5-minute time-step before being used. Thus, if data is recorded a couple of seconds later than required, as indicated by the timestamp of the data, it is shifted back to the previous 5-minute time-step.

9.2.3 South African datasets

When constructing the JHB and CPT datasets only data from first-class roads, such as national principal arterial roads, and second-class roads, such as provincial arterial roads, are used. This is done to make the datasets comparable to the METR-LA and PEMS-BAY datasets, which also consist only of highway data. Data collection was performed in a parallel project. (The South African road network data was recorded by Schalk Rabe and the JHB and CPT datasets were then constructed by Jan-Hendrik Visagie.)

Both of the datasets are summarised in Table 9.1. Each of the SA datasets consists of seven-and-a-half months' worth of traffic speed data aggregated into 5-minute windows. The floating car data recorded on fixed TMC segments represents traffic speed sensors, 357 for Johannesburg and 170 for Cape Town. The date and time of each 'sensor reading' is also part of the datasets.

Table 9.1 – Basic characteristics of the JHB and CPT datasets constructed using car vehicle data from INRIX.

	JHB	**CPT**
Sensors	357	170
Period	7.5 months	7.5 months
Resolution	5 minutes	5 minutes
Speed unit	kilometres per hour	kilometres per hour
Number of time steps	48 874	48 873
Number of data points	13 878 018	8 308 410

9.3 Data analysis

In this section, the JHB and CPT datasets are analysed. Only the training and validation data partitions of these datasets are used during analysis. The datasets are sorted in chronological order and then partitioned: 70% training, 10% validation, and 20% testing.

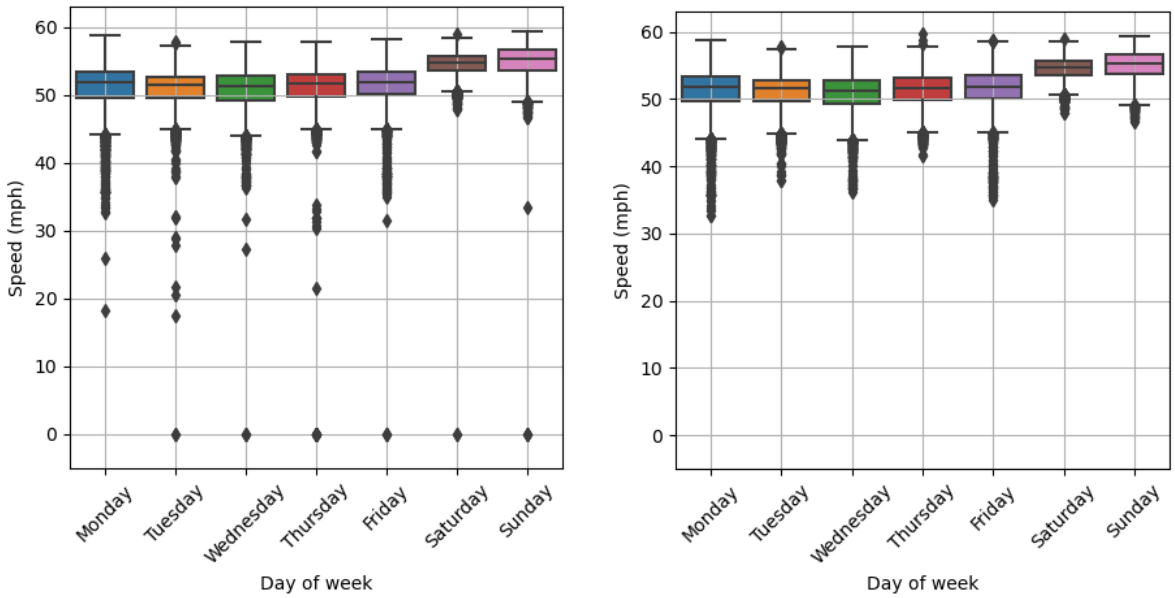
9.3.1 Missing data

The JHB dataset contains 17 864 zero values (0.13%). There are 24 instances in the data where all sensors have a zero value. The CPT dataset (training and validation data partitions) contains 244 102 zero values (3.67%). There are also 28 instances in the data where all sensors have a zero value.

9.3.2 Average speed

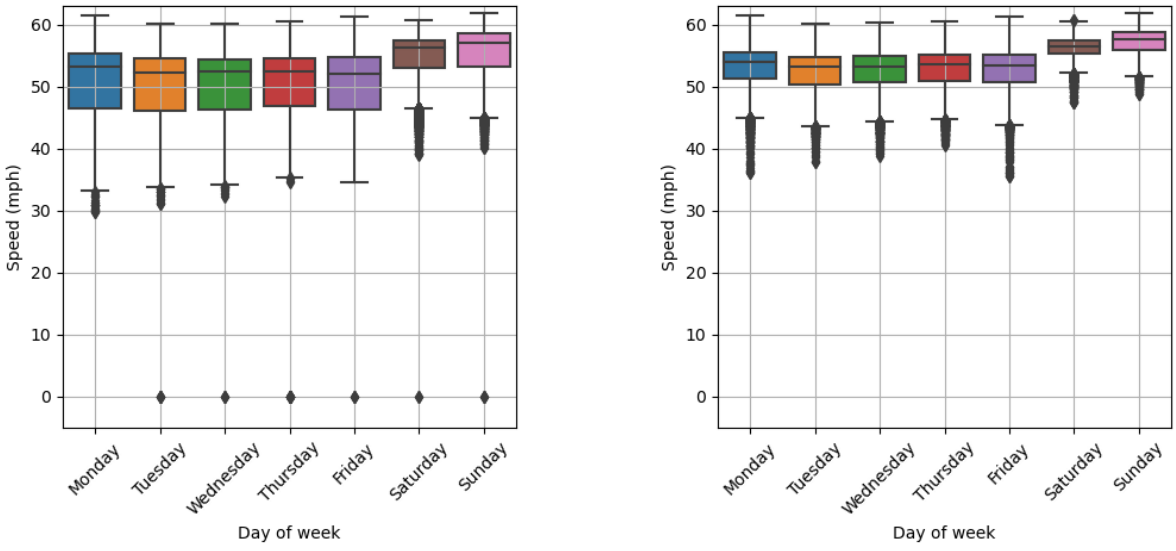
Here we obtain deeper insight into the network behaviour of each dataset. To do this we first look at the boxplot for each day of the week and then for each hour of the day. We then finally take a look at the network variance of each dataset. The unit of measurement used is converted to miles per hour from kilometres per hour in order for the results to be more easily comparable with the results of the American datasets, (PEMS-BAY and METR-LA).

Figures 9.1 and 9.2 show box plots of the average network speed for each day of the week, including zero values on the left and excluding zero values on the right, for JHB and CPT. From the boxplots, we see that for the JHB and CPT datasets the average traffic speed is higher and varies less during weekends. From Figure 9.2 we can see that the zero values have a more significant impact on the CPT data than on the JHB data. When the zero values are removed the standard deviation is much less and it is much easier to see the difference between the weekend data and the workday data.



(a) With zero values **(b)** No zero values

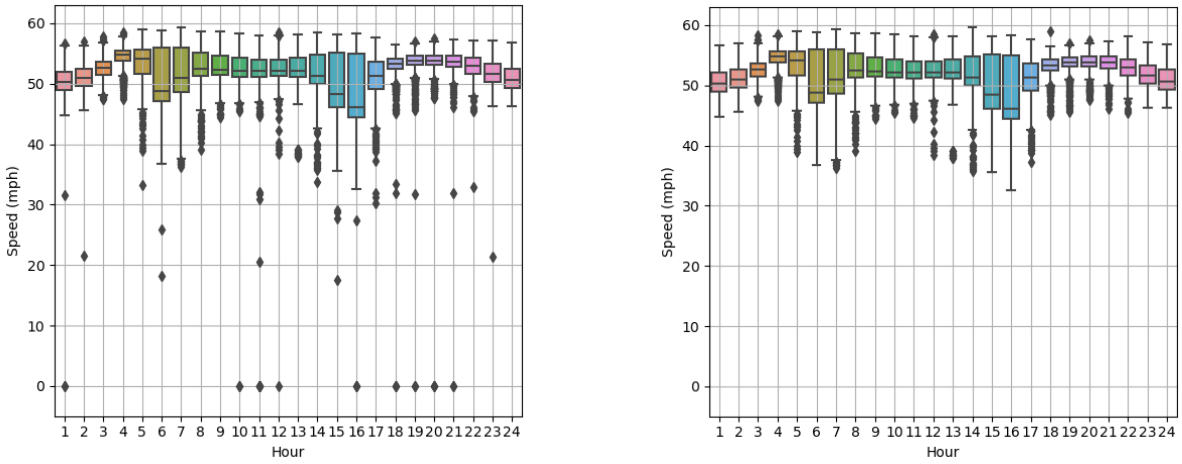
Figure 9.1 – Boxplot of the average network speed per day for the JHB dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right.



(a) With zero values **(b)** No zero values

Figure 9.2 – Boxplot of the average network speed per day for the CPT dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right.

In Figures 9.3 and 9.4 boxplots are given for the average network speed of the JHB and CPT datasets for each hour of the day. We can again see that the missing values (zero values) have a significant impact on the data distribution by comparing the left and right boxplots for each figure. The variance of the data with and without zero values is also given in Figures 9.5 and 9.6, which further visualise the impact missing values have on the dataset.

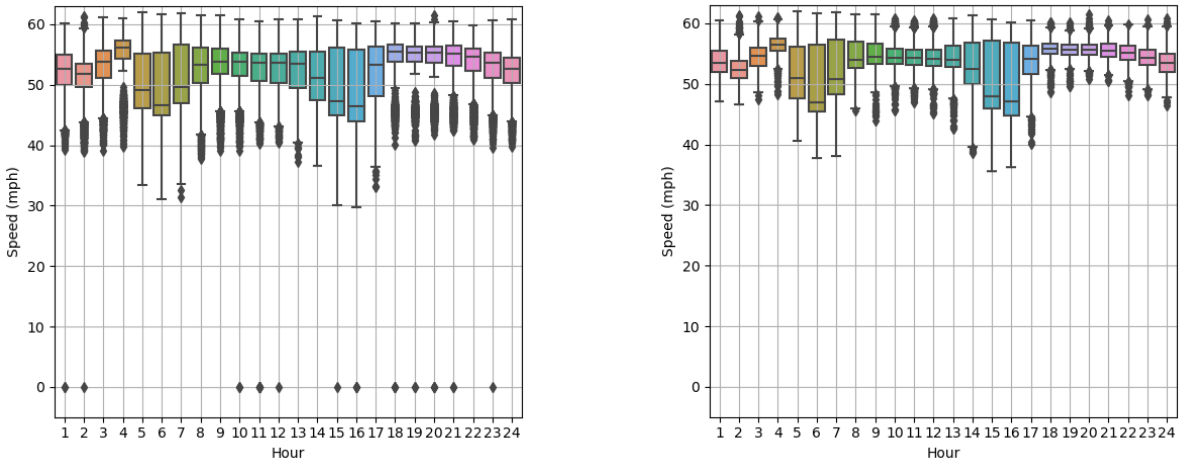


(a) With zero values **(b)** No zero values

Figure 9.3 – Boxplot of the average network speed per hour for the JHB dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right.

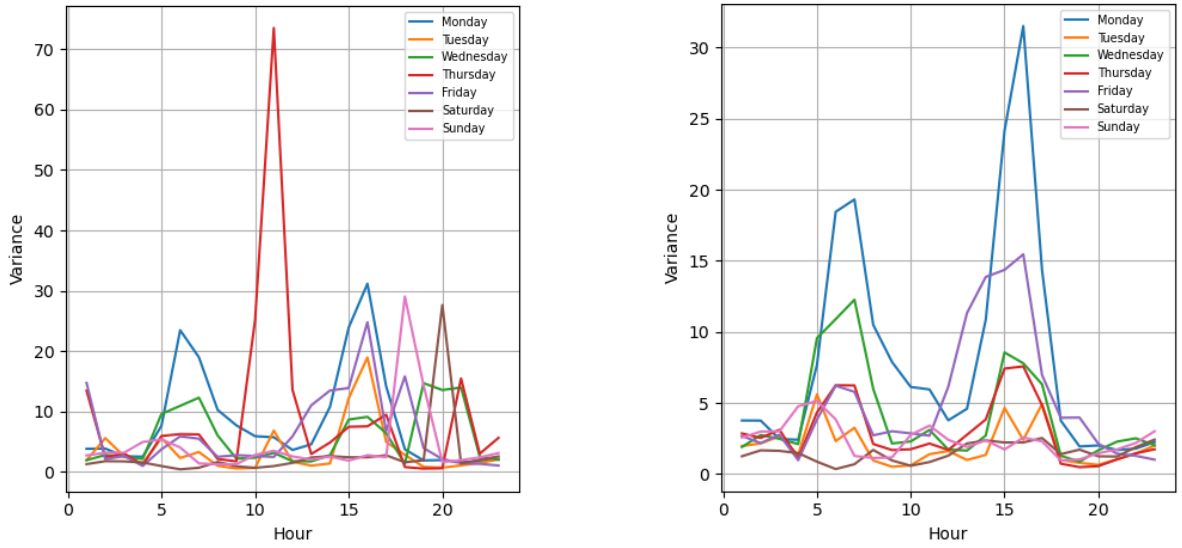
The main insights obtained in this section are:

- Average traffic speed is higher and varies less during weekends, as was similarly observed in the METR-LA and PEMS-BAY datasets.
- Zero values have a significant impact on the variance of the data, as was similarly observed in the METR-LA and PEMS-BAY datasets.
- There is much less traffic speed variance in the JHB dataset than in the CPT dataset.



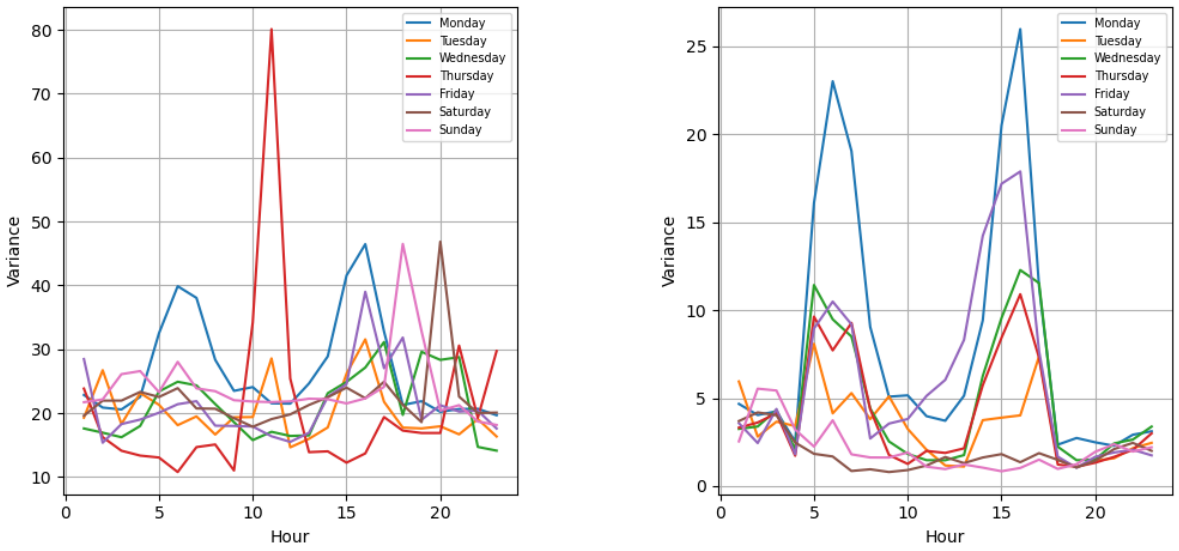
(a) With zero values **(b)** No zero values

Figure 9.4 – Boxplot of the average network speed per hour for the CPT dataset (Training and validation data partitions only); including zero values on the left and excluding zero values on the right.



(a) With zero values **(b)** No zero values

Figure 9.5 – JHB average network speed variance per hour for each day of the week (Training and validation data partitions only).



(a) With zero values

(b) No zero values

Figure 9.6 – CPT average network speed variance per hour for each day of the week (Training and validation data partitions only).

9.4 South African data models

In this section, Graph WaveNet is implemented on the JHB and CPT datasets.

9.4.1 Experiment design

Hyperparameter optimisation is done as described in Section 6.4.1 for three different seeds on the CPT and JHB datasets. The models are trained without using fixed adjacency matrices that are constructed using road-wise distances. Only dynamic adjacency matrices learned through gradient descent are used in the models. To construct fixed adjacency matrices, road-wise distances are necessary but these are not readily available from INRIX. Construction of adjacency matrices is left for future work (Section 10.4).

9.4.2 Results

The performance achieved by the best models on the training and validation data partitions is given in Tables 9.2 and 9.3 for the JHB dataset and in Tables 9.4 and 9.5 for the CPT dataset. It is important to note that the speed readings in the JHB and CPT datasets are measured in kilometres per hour while the METR-LA datasets readings are in miles per hour. The unit of measurement does not affect training since the data is normalised. To make the given results comparable to those of the METR-LA and PEMS-BAY datasets, all MAE and RMSE are given in miles per hour in this section. MAPE is not affected by the unit of measurement. After model selection, the performance achieved on the test data partition is also given in Tables 9.6 and 9.7 for the JHB and CPT datasets respectively.

As can be seen, Graph WaveNet does not perform as well on the South African data as it does on the METR-LA data. What is interesting is that the performance of Graph WaveNet on both the JHB and CPT datasets stays relatively constant for all prediction horizons. The JHB and CPT models also perform better on the testing data partition than on the validation data partition, achieving almost the same performance as for the training data partition. As this is an unexpected result, we review the evaluation data after our best model had been selected, and find that the evaluation data partitions of the JHB and CPT are less congested than the validation partitions (average network speed

Table 9.2 – Graph WaveNet performance on training and validation data partitions of JHB dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Model 1	train	4.83	11.12%	6.65	4.94	11.68%	6.86	5.06	12.44%	7.08
	val	4.94	11.51%	6.78	5.09	12.42%	7.09	5.21	13.36%	7.37
Model 2	train	4.86	11.16%	6.69	4.98	11.85%	6.92	5.10	12.63%	7.14
	val	4.92	11.32%	6.77	5.07	12.18%	7.05	5.20	13.13%	7.31
Model 3	train	4.85	11.15%	6.68	4.95	11.76%	6.89	5.06	12.45%	7.09
	val	4.94	11.41%	6.79	5.09	12.39%	7.11	5.22	13.32%	7.39
Mean	train	4.85	11.14%	6.67	4.96	11.76%	6.89	5.07	12.51%	7.10
	val	4.93	11.41%	6.78	5.08	12.33%	7.08	5.21	13.27%	7.36

Table 9.3 – Graph WaveNet mean performance on training and validation data partitions of JHB dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour.

		Mean 5 min – 60 min		
		MAE	**MAPE**	**RMSE**
Model 1	train	4.88	11.54%	6.79
	val	5.02	12.21%	7.00
Model 2	train	4.92	11.68%	6.85
	val	5.01	12.01%	6.96
Model 3	train	4.89	11.59%	6.82
	val	5.02	12.16%	7.02
Mean	train	4.90	11.60%	6.82
	val	5.02	12.13%	6.99

Table 9.4 – Graph WaveNet performance on training and validation data partitions of CPT dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour.

		15 min			**30 min**			**60 min**		
		MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Seed 1 (99)	train	5.04	10.55%	6.78	5.16	11.19%	7.00	5.30	12.02%	7.29
	val	5.09	10.88%	6.87	5.25	11.74%	7.17	5.45	12.91%	7.60
Seed 2 (100)	train	5.05	10.63%	6.78	5.17	11.40%	7.02	5.30	12.32%	7.30
	val	5.10	10.99%	6.89	5.27	12.02%	7.20	5.46	13.44%	7.61
Seed 3 (101)	train	5.05	10.51%	6.75	5.16	11.08%	6.99	5.30	11.83%	7.30
	val	5.09	10.94%	6.84	5.25	11.82%	7.17	5.43	12.89%	7.57
Mean	train	5.05	10.56%	6.77	5.16	11.22%	7.00	5.30	12.06%	7.30
	val	5.09	10.94%	6.87	5.26	11.86%	7.18	5.45	13.08%	7.59

Table 9.5 – Graph WaveNet mean performance on training and validation data partitions of CPT dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour.

		Mean 5 min – 60 min		
		MAE	**MAPE**	**RMSE**
Seed 1 (99)	train	5.11	11.07	6.95
	val	5.19	11.62	7.12
Seed 2 (100)	train	5.12	11.27	6.96
	val	5.21	11.93	7.14
Seed 3 (101)	train	5.11	10.97	6.94
	val	5.19	11.67	7.1
Mean	train	5.11	11.10	6.95
	val	5.20	11.74	7.12

Table 9.6 – Graph WaveNet performance on the testing data partition of JHB dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour.

	MAE	**MAPE**	**RMSE**
15 min			
Model 1	4.84	11.18%	6.65
Model 2	4.84	11.05%	6.64
Model 3	4.84	11.15%	6.67
Mean	4.84	11.13%	6.65
30 min			
Model 1	4.98	11.98%	6.96
Model 2	4.98	11.89%	6.93
Model 3	5.00	12.06%	6.99
Mean	4.99	11.98%	6.96
60 min			
Model 1	5.09	12.75%	7.20
Model 2	5.10	12.71%	7.19
Model 3	5.12	12.83%	7.26
Mean	5.10	12.76%	7.22
Mean 5min – 60 min			
Model 1	4.92	11.78%	6.87
Model 2	4.92	11.70%	6.85
Model 3	4.94	11.83%	6.90
Mean	4.93	11.77%	6.87

Table 9.7 – Graph WaveNet performance on the testing data partition of CPT dataset for three models optimised and trained with different seeds using only dynamic adjacency matrices. MAE and RMSE given in miles per hour.

	MAE	**MAPE**	**RMSE**
	15 min		
Seed 1 (99)	5.06	10.49%	6.78
Seed 2 (100)	5.06	10.55%	6.78
Seed 3 (101)	5.06	10.50%	6.76
Mean	5.06	10.51%	6.77
	30 min		
Seed 1 (99)	5.19	11.15%	7.04
Seed 2 (100)	5.19	11.27%	7.04
Seed 3 (101)	5.20	11.11%	7.05
Mean	5.19	11.18%	7.04
	60 min		
Seed 1 (99)	5.34	11.81%	7.35
Seed 2 (100)	5.32	11.97%	7.32
Seed 3 (101)	5.34	11.72%	7.38
Mean	5.33	11.83%	7.35
	Mean 5min – 60 min		
Seed 1 (99)	5.14	10.99%	6.98
Seed 2 (100)	5.13	11.10%	6.97
Seed 3 (101)	5.14	10.95%	6.99
Mean	5.14	11.01%	6.98

of 86.42km/h, compared to an average network speed of 86.63km/h and 85.66km/h on the train and validation partitions, respectively for the CPT dataset and average network speed of 84.24km/h, compared to an average network speed of 83.94km/h and 84.21km/h on the train and validation partitions, respectively for the JHB dataset).

9.5 SA data models performance analysis

In this section, the Graph WaveNet models are analysed as was done in Section 7.2 to gauge the performance of the models on congested traffic data. The models are also analysed as was done in Section 8.2 to see whether or not the models trained using only dynamic adjacency matrices would benefit from receiving additional historic data as input.

9.5.1 Binned traffic speed analysis

Here, as in Section 7.2 the traffic speed is first binned based on the individual sensor speeds (the individual road segments), and then based on the average network speed. Based on the binned individual sensor speeds, results seen in Figure 9.7 for the JHB dataset and in Figure A.1, for the CPT dataset, we can see in Appendix A.2 that there is a strong inverse relationship between the traffic speeds recorded on each road segment and the performance achieved. This is also true when considering the binned network speed analysis results for the JHB and CPT datasets. These additional results are given in Figures A.2 and A.3 in Appendix A.2.

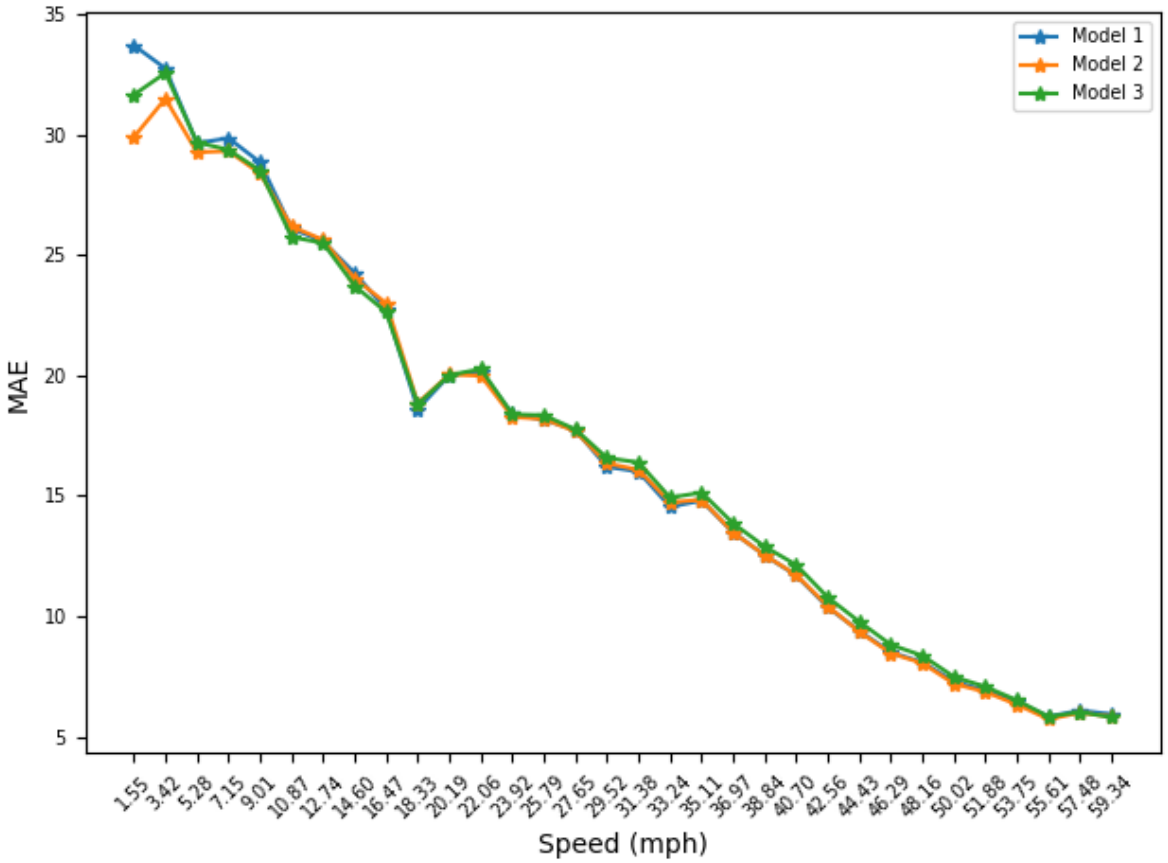


Figure 9.7 – Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per sensor speed: JHB, validation data partition.

9.5.2 Trend analysis

For the trend analysis, as was done in Section 8.2, we compare the trend of the data determined using the training data partition to the actual traffic speed, and the 5-minute and 60-minute predictions to determine whether or not the models would benefit from receiving additional historic data as input. As in Section 8.2 we again notice that the predictions follow the trend of the data until the available data suggests otherwise. In Figures 9.8 and 9.10 the trend, true traffic speed value, 5-minute predictions, and 60-minute predictions are visualised for the road segment in the JHB and CPT datasets on which the models perform the worst, respectively. In Figures 9.9 and 9.11 the same is visualised but for the road segments on which the models perform best. We can see from these figures that the traffic speed curves of the CPT and JHB datasets are much less smooth than those of the METR-LA or PEMS-BAY datasets.

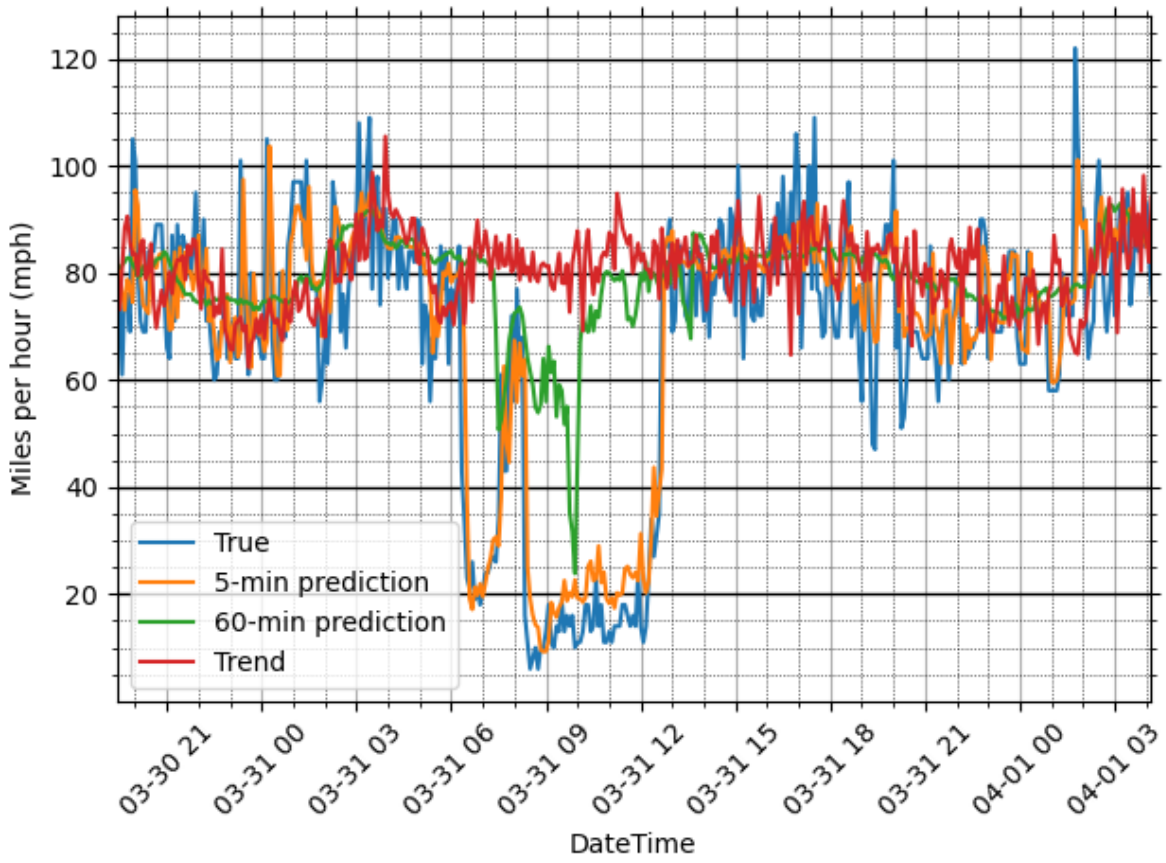


Figure 9.8 – Trend analysis for the sensor in the JHB dataset for which model 1 trained in Section 9.4 performs the worst. Predicted and true traffic speed values are based on the JHB validation data partition while the trend is calculated using the training data partition.

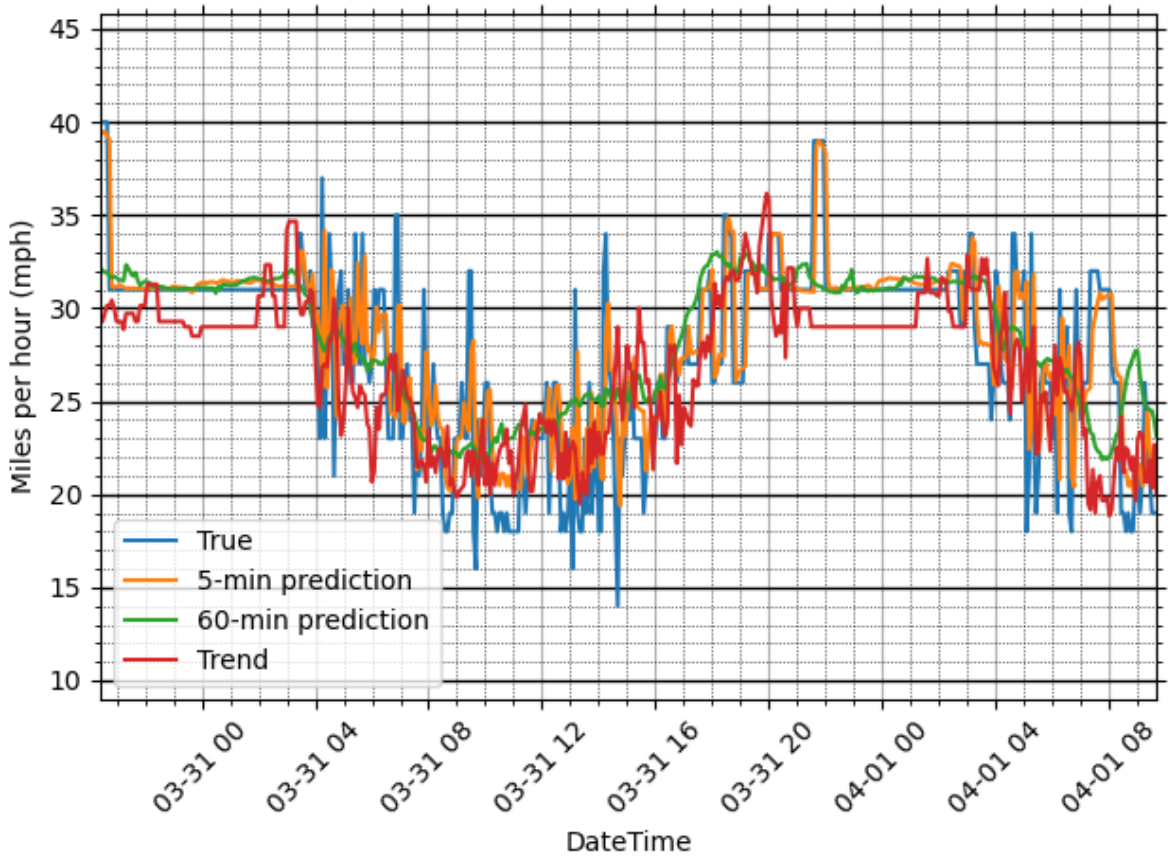


Figure 9.9 – Trend analysis for the sensor in the JHB dataset for which model 1 trained in Section 9.4 performs the best. Predicted and true traffic speed values are based on the JHB validation data partition while the trend is calculated using the training data partition.

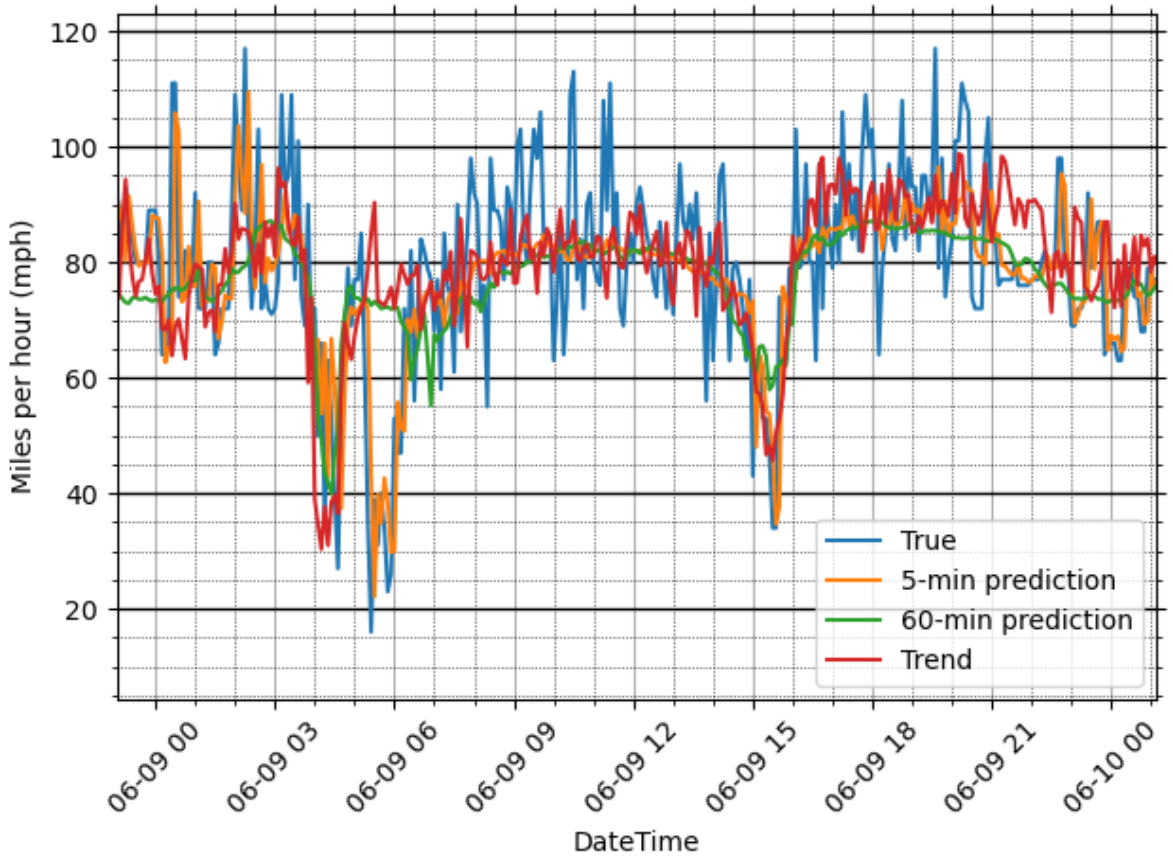


Figure 9.10 – Trend analysis for the sensor in the CPT dataset for which model 1 trained in Section 9.4 performs the worst. Predicted and true traffic speed values are based on the CPT validation data partition while the trend is calculated using the training data partition.

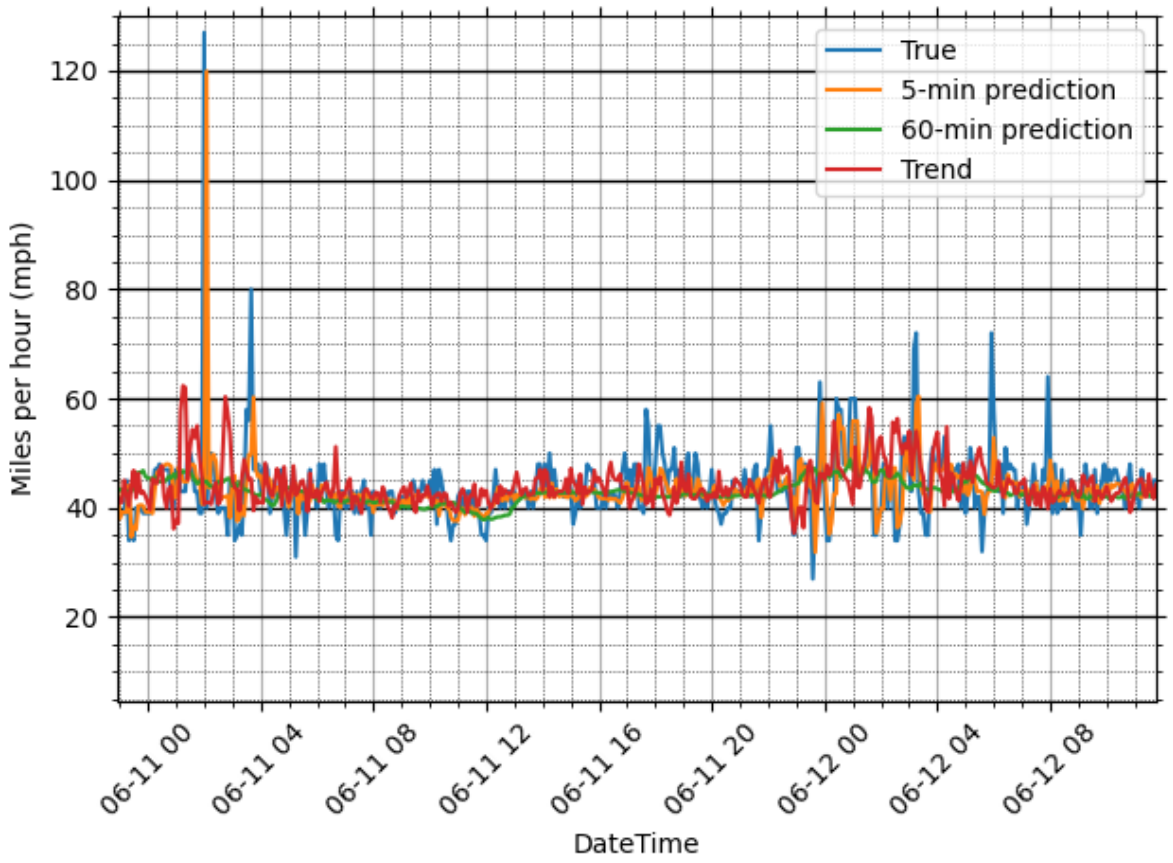


Figure 9.11 – Trend analysis for the sensor in the CPT dataset for which model 1 trained in Section 9.4 performs the best. Predicted and true traffic speed values are based on the CPT validation data partition while the trend is calculated using the training data partition.

9.6 Conclusion

The JHB and CPT datasets were first prepared by means of resampling to overcome data-logging faults. The datasets were then analysed and finally, we implemented Graph WaveNet on the South African datasets and discussed the results. From these results we found that:

- Model performance stays relatively constant for all prediction horizons.
- The performance achieved on the CPT and JHB datasets (5.02 and 5.20 mean MAE validation loss for all 12 prediction horizons, respectively), is not as good as for the METR-LA and PEMS-BAY datasets (2.76 and 1.58 mean MAE validation loss for all 12 prediction horizons).

The performance of the SA data models was analysed during different degrees of congestion showing a clear deterioration of model performance during slower traffic speeds. A trend analysis was performed which gave further insight into why the SA models do not perform as well as the models trained on the METR-LA and PEMS-BAY datasets. The traffic speed curves of data recorded on SA road segments using FCD display rapid small variations that increase the complexity of the data.

Chapter 10

Conclusion

10.1 Overview

The goal of this study, as discussed in Chapter 1, was to implement a SOTA traffic prediction model on South African road networks. In this chapter conclusions reached regarding SOTA traffic prediction models are discussed as well as the degree to which our original goals were achieved. The key findings of the study and their implications are discussed along with potential avenues for future research.

10.2 Key findings

First, findings regarding the datasets used are discussed:

(i) Zero values in the METR-LA dataset have a significant effect on the data variance of these traffic networks. We therefore treated these values as missing data rather than as periods of either no or stationary traffic (as proposed by [22]), as discussed in Section 6.5.

(ii) There is a noticeable difference between traffic data recorded on weekends and traffic

data recorded during the rest of the week. Weekend traffic speed data varies less than workday data and is faster than workday data. This is to be expected since there are far fewer people travelling to and from work during the weekend.

With regard to available SOTA traffic prediction models at the time of this study:

(i) STTN, Graph WaveNet, STAWnet, GMAN and STNN were studied. Graph WaveNet and STAWnet were the best performing models for which results could be successfully recreated.

(ii) Graph WaveNet was selected as the best available tool for implementation and experimentation at the time the analysis was conducted.

During the experimental setup phase of the study the following findings were made:

(i) Validation loss calculation problems were found in the Graph WaveNet model training script. These had to be resolved before a hyperparameter optimisation protocol could be chosen. These validation loss calculation errors are also present in other codebases such as STAWnet [20] and DCRNN [7].

(ii) A possible solution to the missing data issue described in Chapter 3, namely linear interpolation, was explored but the performance improvements achieved using linear interpolation were not significant.

A congestion analysis was performed that led to the following key findings:

(i) As expected, model performance deteriorates during congestion. Unexpectedly, a longer prediction horizon has a large effect on performance during periods of congestion, and a surprisingly limited effect otherwise.

(ii) We attempted to improve model performance during times of congestion using oversampling in Section 7.4 and undersampling in Section 7.5. These simple methods did not increase model performance.

Extensions to Graph WaveNet were evaluated but none produced an improvement over the initial model. The following extensions were analysed:

(i) To determine if Graph WaveNet would benefit from additional historical data a trend analysis was performed. By comparing the trend of the data to anticipated traffic speeds and predictions made for the validation data partition we determined that the additional historical data would not benefit Graph WaveNet models.

(ii) To attempt to reduce internode performance variance a TCN layer was implemented in the Graph WaveNet model that applies a separate kernel to the input from each of the nodes in the METR-LA network. The additional parameters did not increase the performance of Graph WaveNet.

(iii) Time was encoded as a cyclical feature to try to improve model performance but by representing time as two separate input features complexity was increased and the model's performance suffered as a result.

Graph WaveNet was implemented on South African datasets recorded in Johannesburg and Cape Town leading to the following findings:

(i) Graph WaveNet was successfully implemented on the JHB and CPT datasets. The performance achieved on the JHB and CPT datasets (5.02 and 5.20 mean MAE mph validation loss for all 12 prediction horizons respectively) is not as good as for the METR-LA dataset (2.76 mean MAE mph validation loss for all 12 prediction horizons).

(ii) The Johannesburg and Cape Town datasets had to be resampled to overcome data-logging faults.

(iii) The performance of Graph WaveNet on both the JHB and CPT datasets stayed relatively constant for all prediction horizons.

10.3 Outcomes

The following main contributions were made during this study:

(i) Graph WaveNet was analysed and we found that validation loss calculations can be adjusted to improve optimisation capabilities; prediction capabilities deteriorate during times of congestion in the road network, especially over longer prediction horizons.

(ii) Two novel South African datasets were analysed and prepared for experimentation.

(iii) Graph WaveNet was successfully implemented for a South African road network. First models were trained on the new South African datasets and analysed, creating new possibilities for future research.

10.4 Future work

Here, future work is suggested to further the effectiveness of traffic prediction for South African road networks.

(i) While analysing the data, we were unable to find a good reason in the available literature for the use of a threshold of 0.1 for the Gaussian kernel used to construct the adjacency matrices of the PEMS-BAY and METR-LA datasets. This offers an opportunity for further research.

(ii) An increased understanding of model behaviour during congestion can assist future efforts to optimise a model for periods of recurrent congestion, which will improve route-finding algorithms during times of the day when the fastest path-finding technologies are most necessary. An increase in a model's ability to accurately predict traffic during periods of recurrent congestion will likely also increase its performance during non-recurrent congestion events such as traffic accidents. An ensemble learning technique can be explored in the future to attempt to overcome the accuracy

paradox encountered where model performance is exaggerated due to times of day when traffic flow is easily predicted.

(iii) GCNs were the SOTA at the time of this study. By the completion of this study prediction models implementing transformer models, first proposed by Vaswani et al. [34], have become the new state of the art. Bidirectional Encoder Representations from Transformers (BERT), developed by Devlin et al. [35] is a prediction model that can also possibly be explored for the traffic prediction task. BERT allows for pretraining on large datasets and can then be fine-tuned using a smaller dataset.

(iv) Adjacency matrices can be constructed in future work for the South African datasets to allow for further experimentation and analysis.

(v) A filter can be implemented on FCD recorded on road segments to reduce data complexity and consequently produce more accurate models.

10.5 Conclusion

Due to the obscure nature of deep learning models, a thorough analysis is required before their implementation in a traffic network. To this end, various traffic prediction models were considered and one was analysed in depth using available well-known datasets while the South African datasets were being constructed. We analysed the inner workings of Graph WaveNet and showed that the models produced perform well while there is still room for improvement during times of congestion in road networks. Graph WaveNet was successfully implemented for two novel South African datasets and the models produced were analysed, revealing many similarities to international benchmarked datasets. We hope the results of this study lead to future implementations of deep learning models on South African road networks with a view to increasing productivity and reducing pollution.

Bibliography

[1] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, "Enhancing transportation systems via deep learning: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 144–163, 2019, ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2018.12.004.

[2] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis, "Short-term traffic forecasting: Overview of objectives and methods," *Transport Reviews*, vol. 24, no. 5, pp. 533–557, 2004. DOI: 10.1080/0144164042000195072.

[3] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. Choudhury, and A. K. Qin, "A survey on modern deep neural network for traffic prediction: Trends, methods and challenges," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020. DOI: 10.1109/TKDE.2020.3001195.

[4] M. Oosthuizen, A. Hoffman, and M. Davel, "A comparative study of graph neural network speed prediction during periods of congestion," in *Proceedings of the 14th International Joint Conference on Computational Intelligence*, 2022, pp. 331–338. DOI: 10.5220/0011374100003332.

[5] M. Lv, Z. Hong, L. Chen, T. Chen, T. Zhu, and S. Ji, "Temporal multi-graph convolutional network for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3337–3348, 2021. DOI: 10.1109/TITS.2020.2983763.

[6] M. Veres and M. Moussa, "Deep learning for intelligent transportation systems: A survey of emerging trends," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3152–3168, 2020. DOI: 10.1109/TITS.2019.2929020.

[7] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Int. Conf. on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SJiHXGWAZ.

[8] K. Lee, M. Eo, E. Jung, Y. Yoon, and W. Rhee, "Short-term traffic prediction with deep neural networks: A survey," *IEEE Access*, vol. 9, pp. 54 739–54 756, 2021. DOI: 10.1109/ACCESS.2021.3071174.

[9] A. Burkov, *The Hundred-Page Machine Learning Book*, illustrated. Andriy Burkov, 2019.

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," English (US), in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[12] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[13] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial-temporal graph modeling," in *Proc. of the 28th Int. Joint Conf. on AI, IJCAI-19*, Jul. 2019, pp. 1907–1913. DOI: 10.24963/ijcai.2019/264.

[14] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018. arXiv: 1803.01271. [Online]. Available: http://arxiv.org/abs/1803.01271.

[15] M. Xu, W. Dai, C. Liu, *et al.*, "Spatial-temporal transformer networks for traffic flow forecasting," *ArXiv*, vol. abs/2001.02908, 2020.

[16] S. Yang, J. Liu, and K. Zhao, "Space meets time: Local spacetime neural network for traffic flow forecasting," in *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, 2021, pp. 817–826. DOI: 10.1109/ICDM51629.2021.00093.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, pp. 305–309, http://www.deeplearningbook.org.

[18] *Caltrans pems*. [Online]. Available: https://pems.dot.ca.gov/.

[19] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proc. AAAI Conf. on AI*, vol. 34, Apr. 2020, pp. 1234–1241. DOI: 10.1609/aaai.v34i01.5477.

[20] C. Tian and W. K. V. Chan, "Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies," *IET Intelligent Transport Systems*, vol. 15, pp. 549–561, 2021. DOI: 10.1049/itr2.12044.

[21] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul. 2018. DOI: 10.24963/ijcai.2018/505. [Online]. Available: http://dx.doi.org/10.24963/ijcai.2018/505.

[22] S. Shleifer, C. McCreery, and V. Chitters, *Incrementally improving graph wavenet performance on traffic prediction*, 2019. DOI: 10.48550/ARXIV.1912.07390.

[23] T. J. Rivlin, *The Chebyshev Polynomials*. Wiley, 1974, pp. 56–123, http://www.deeplearningbook.org, ISBN: 047172470X, 9780471724704.

[24] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*. Springer US, 2011, pp. 652–653. DOI: 10.1007/978-0-387-30164-8_528.

[25] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[26] A. van den Oord, S. Dieleman, H. Zen, *et al.*, "WaveNet: A Generative Model for Raw Audio," in *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016, p. 125.

[27] *Conv1d*, Accessed: 2022-05-03. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html.

[28] *Conv2d*, Accessed: 2022-05-03. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.htmll.

[29] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[30] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*, PMLR, 2019, pp. 6861–6871.

[31] K. Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969. DOI: 10.1109/TSSC.1969.300225.

[32] *Inrix*, Accessed: 2022-10-11. [Online]. Available: https://inrix.com/about/.

[33] *Inrix segments*, Accessed: 2022-10-11. [Online]. Available: https://docs.inrix.com/reference/getsegments/.

[34] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[35] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

Appendix A

Supplemental content

A.1 Appendix: Chapter 7

Here the statistical significance of the difference between results achieved by models optimised and trained in Chapter 7 is determined. The analysis is performed using the data given in Tables A.1 and A.2 for the congestion data boosted models from Section 7.4 and the models trained on congested data only from Section 7.5 respectively.

To compare performance using all prediction horizons at the same time the performance from each prediction horizon is first normalised. Z-score normalisation rescales values in order for a population to have a mean of 0 and a standard deviation of 1 [9].

$$z = \frac{x - \mu_i}{\sigma_i} \tag{A.1}$$

In Equation A.1: z is the normalised value, x is the value being normalised, μ is the mean MAE of prediction horizon i, and σ_i is the standard deviation of MAE for prediction horizon i.

After normalisation, the significance of the difference in performance is determined using analysis of variance (ANOVA). For a significant difference to exist in performance the

f-statistic must be large (typically larger than 10), and the p-value must be less than the threshold ($\alpha = 0.05$). The results of the ANOVA analysis are given in Table A.3. From these results, we can say that there is a significant difference in performance on congested data and not congested data for both the congestion-boosted models and the congested-only models. We can also say that there is a significant difference in performance between the congestion-boosted models and the congested-only models.

A Welch's t-test is also performed on the normalised data of Tables A.1 and A.2. For the t-test the p-value must also be less than the threshold ($\alpha = 0.05$) for there to be a significant difference in the results. Based on the results in Table A.3 we can reject the null hypothesis and say that there is a significant difference in model performance on congested data and not congested data.

Table A.1 – Performance of each model trained on congested boosted data in Section 7.4. The MAE is given for each prediction horizon on congested data only and not congested data data. Congested data is extracted as described in Section 7.4. Results are given for the METR-LA validation data partition.

	MAE							
	Congested data				Not congested data			
Prediction (min)	s1	s2	s3	**Mean**	s1	s2	s3	**Mean**
1	2.48	2.47	2.48	**2.48**	2.08	2.07	2.07	**2.07**
2	2.94	2.94	2.95	**2.94**	2.18	2.18	2.18	**2.18**
3	3.27	3.26	3.28	**3.27**	2.25	2.24	2.24	**2.24**
4	3.53	3.52	3.54	**3.53**	2.30	2.29	2.30	**2.30**
5	3.74	3.74	3.77	**3.75**	2.34	2.34	2.33	**2.34**
6	3.93	3.93	3.96	**3.94**	2.38	2.37	2.37	**2.37**
7	4.09	4.09	4.12	**4.10**	2.41	2.41	2.39	**2.40**
8	4.23	4.23	4.26	**4.24**	2.43	2.44	2.42	**2.43**
9	4.34	4.34	4.37	**4.35**	2.46	2.46	2.44	**2.45**
10	4.44	4.44	4.46	**4.45**	2.48	2.49	2.47	**2.48**
11	4.51	4.52	4.54	**4.52**	2.51	2.51	2.49	**2.51**
12	4.59	4.61	4.62	**4.61**	2.54	2.55	2.52	**2.54**

Table A.2 – Performance of each model trained on congested data only in Section 7.5. The MAE is given for each prediction horizon on congested data only and not congested data. Congested data is extracted as described in Section 7.5. Results are given for the METR-LA validation data partition.

	MAE							
	Congested data				Not congested data			
Prediction (min)	s1	s2	s3	**Mean**	s1	s2	s3	**Mean**
1	2.64	2.55	2.58	**2.59**	2.55	2.36	2.41	**2.44**
2	3.14	3.02	3.04	**3.07**	2.73	2.53	2.57	**2.61**
3	3.48	3.35	3.37	**3.40**	2.82	2.67	2.72	**2.74**
4	3.75	3.62	3.64	**3.67**	2.96	2.81	2.84	**2.87**
5	3.95	3.83	3.87	**3.88**	3.03	2.93	2.98	**2.98**
6	4.12	4.02	4.07	**4.07**	3.10	3.03	3.09	**3.07**
7	4.26	4.18	4.25	**4.23**	3.19	3.12	3.19	**3.17**
8	4.38	4.32	4.41	**4.37**	3.27	3.20	3.30	**3.26**
9	4.50	4.44	4.54	**4.49**	3.37	3.30	3.39	**3.36**
10	4.59	4.55	4.65	**4.60**	3.43	3.38	3.48	**3.43**
11	4.68	4.64	4.75	**4.69**	3.51	3.46	3.56	**3.51**
12	4.78	4.74	4.86	**4.79**	3.56	3.54	3.64	**3.58**

Table A.3 – ANOVA analyses and Welch's t-test performed on normalised data from Tables A.1 and A.2 results.

ANOVA		**f-statistic**	**p-value**
Congestion boosted (congested data)	Congestion boosted (not congested data)	3 510.7	9.22×10^{-26}
Congested only (congested data)	Congested only (not congested data)	670.11	5.71×10^{-18}
Welch's t-test		**t-statistic**	**p-value**
Congestion boosted (congested data)	Congestion boosted (not congested data)	59.25	1.00×10^{-18}
Congested only (congested data)	Congested only (not congested data)	25.89	1.81×10^{-11}

A.2 Appendix: Chapter 9

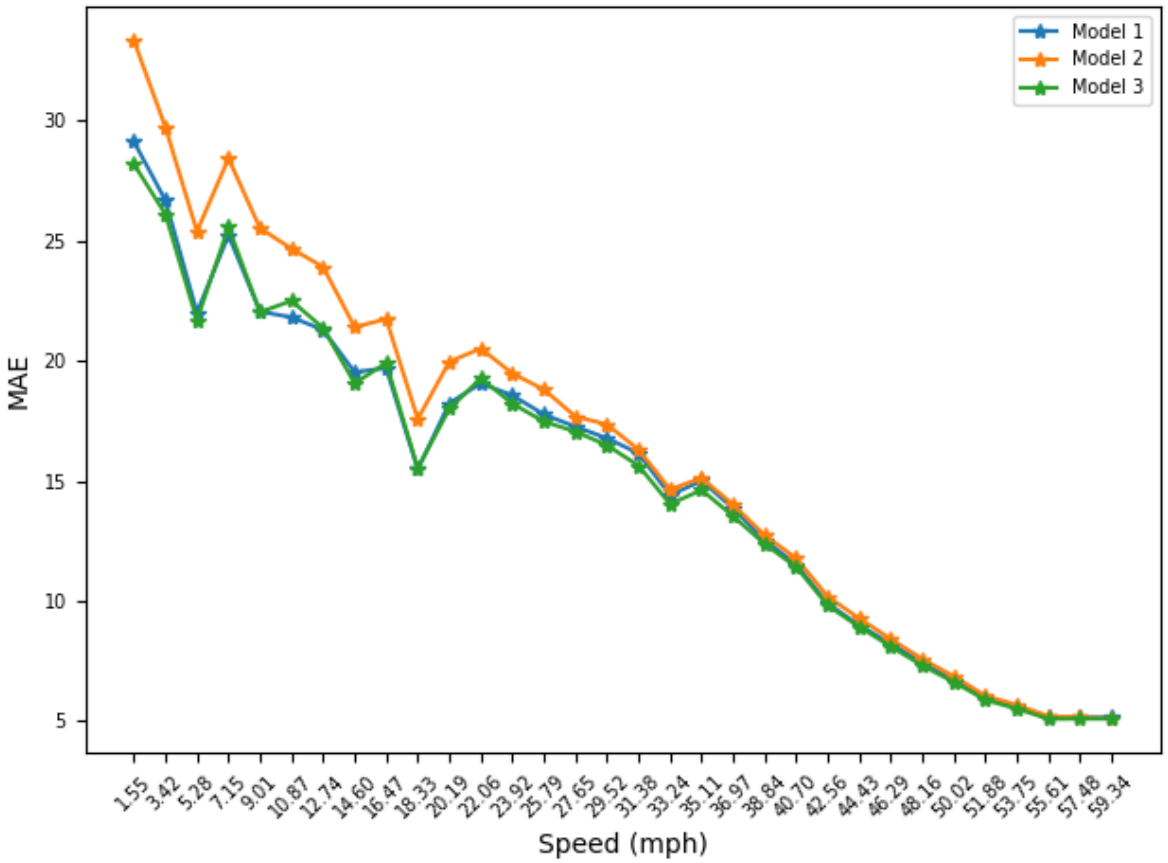


Figure A.1 – Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per sensor speed: CPT, validation data partition.

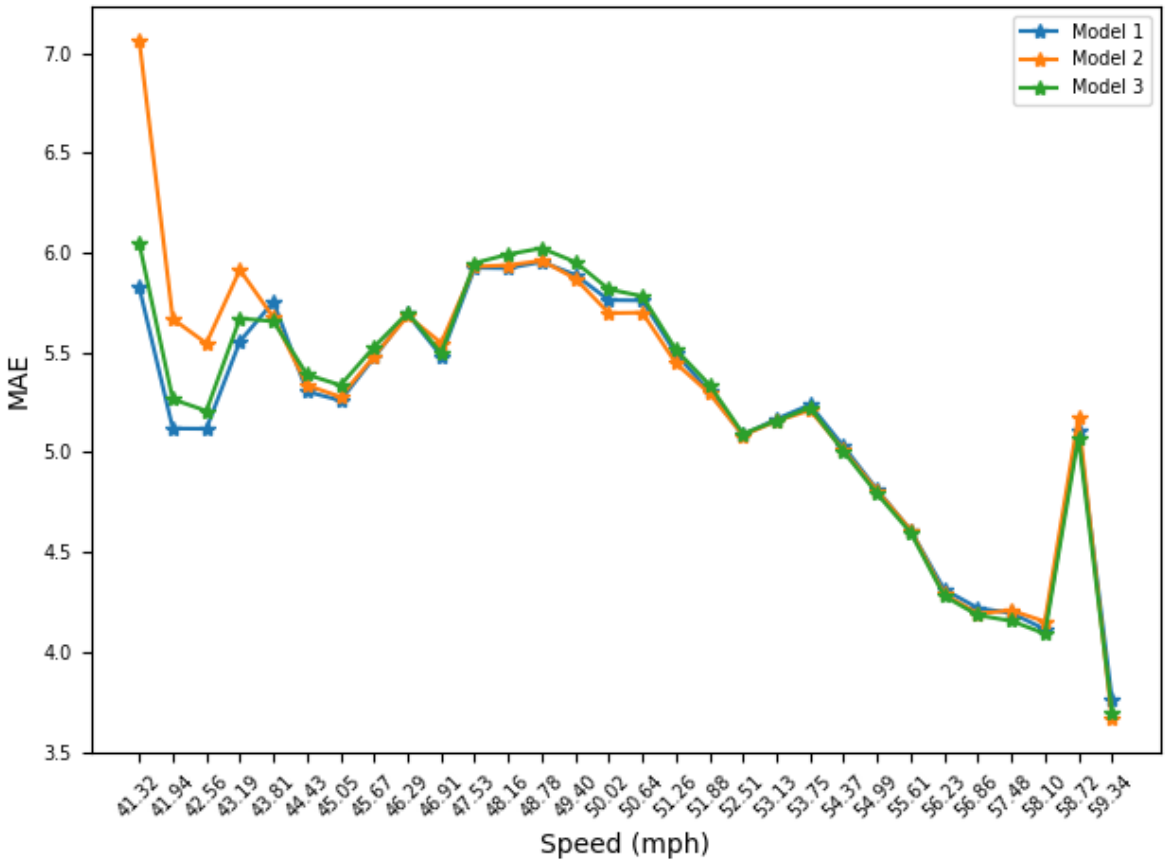


Figure A.2 – Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per average network speed: JHB, validation data partition.

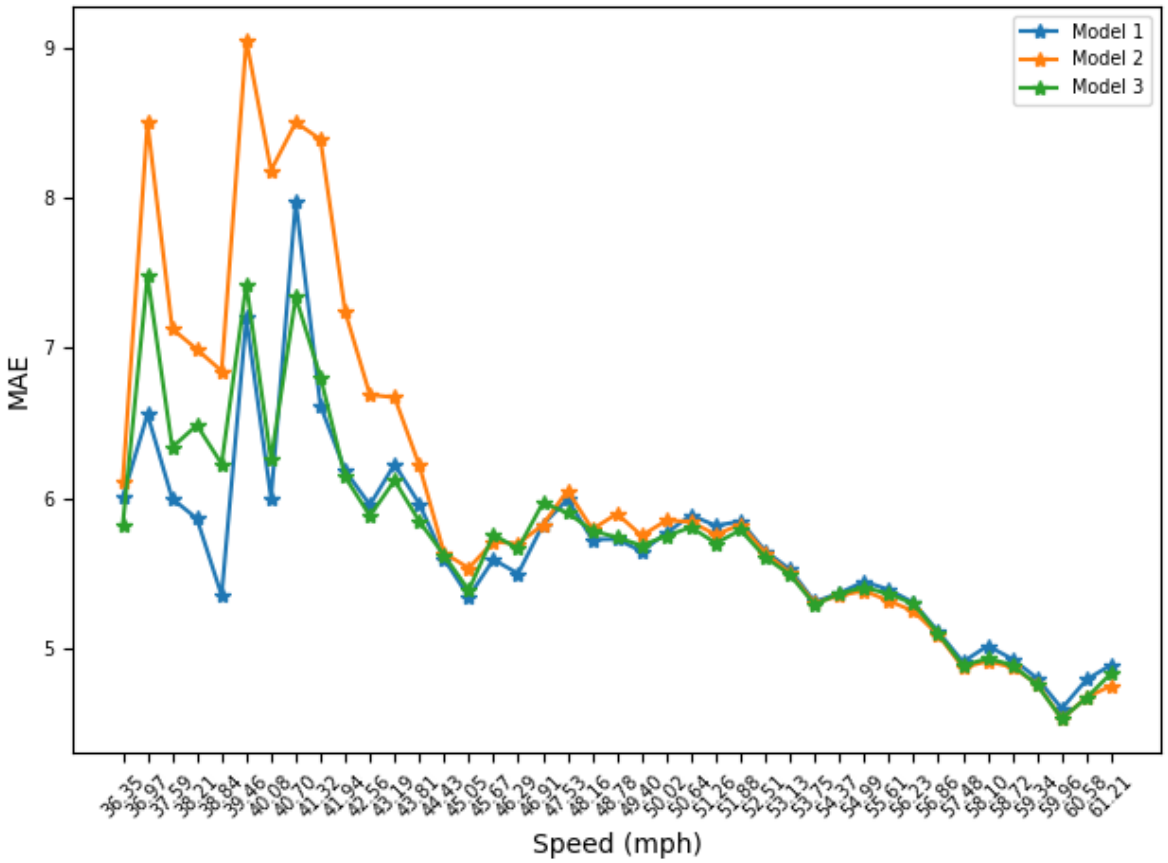


Figure A.3 – Graph WaveNet performance (MAE) of three models trained in Section 9.4 for 60-minute prediction horizon, binned per average network speed: CPT, validation data partition.

A.3 Appendix: Publication

A Comparative Study of Graph Neural Network Speed Prediction during Periods of Congestion

Marko C. Oosthuizen[1,2][a], Alwyn J. Hoffman[1][b] and Marelie H. Davel[1,2,3][c]
[1]*Faculty of Engineering, North-West University, South Africa.*
[2]*Centre for Artificial Intelligence Research (CAIR), South Africa.*
[3]*National Institute for Theoretical and Computational Sciences (NITheCS)*
marko@mco-eng.co.za, {alwyn.hoffman, marelie.davel}@nwu.ac.za

Keywords: Traffic Prediction, Congestion, Graph Neural Network

Abstract: Traffic speed prediction using deep learning has been the topic of many studies. In this paper, we analyse the performance of Graph Neural Network-based techniques during periods of traffic congestion. We first compare a selection of recently proposed techniques that claim to achieve good results using the METR-LA and PeMS-BAY data sets. We then investigate the performance of three of these approaches – Graph WaveNet, Spacetime Neural Network (STNN) and Spatio-Temporal Attention Wavenet (STAWnet) – during congested periods, using recurrent congestion patterns to set a threshold for general congestion through the entire traffic network. Our results show that performance deteriorates significantly during congested time periods, which is concerning, as traffic speed prediction is usually of most value during times of congestion. We also found that, while the above approaches perform almost equally in the absence of congestion, there are much bigger differences in performance during periods of congestion.

1 INTRODUCTION

Traffic speed prediction forms an important element of the management of metropolitan traffic networks. Deep neural networks are one of the most successful techniques used for this purpose, and many different approaches have been published in recent literature (Mena-Oreja and Gozalvez, 2020). The performance measures used to evaluate speed prediction methods include prediction accuracy, robustness under various conditions, the computational intensity of the learning process and others (Mallick et al., 2020). The results of traffic speed prediction are used to advise both traffic authorities and road users about the best course of action to limit or avoid congestion. As a result, prediction accuracy is of most importance during times of congestion, as little action is required from either road users or the traffic management system when traffic is flowing normally.

When evaluating speed prediction methods during both congested and not congested periods, it is observed that prediction accuracy tends to deteriorate under congestion (Polson and Sokolov, 2017). The high prediction accuracies claimed by many researchers may therefore be somewhat misleading, as the published performance levels are achieved when averaging the performance of the model for congested and not congested times, while much worse performance is observed during periods when the prediction algorithms are mostly needed.

In this paper, we evaluate state-of-the-art (SOTA) deep learning traffic speed prediction methods based on their performance during periods of congestion. The rest of the paper is organized as follows: Section 2 contains a survey of related work. In Section 3 we describe the methodology and data used for the study. Section 4 contains the results of the congestion analysis, and in Section 5 we conclude and make recommendations for future work.

2 BACKGROUND

Traffic speed prediction attracts much attention because of the high costs associated with congestion in big cities (Polson and Sokolov, 2017; Mena-Oreja and Gozalvez, 2020). Due to lack of space and funds, it is

[a] https://orcid.org/0000-0003-2514-8135
[b] https://orcid.org/0000-0003-4909-1073
[c] https://orcid.org/0000-0003-3103-5858

in most cases not possible to significantly expand existing road networks in densely populated urban areas (Shi et al., 2019). Other methods to address traffic congestion must therefore be considered.

Traffic congestion can be alleviated through more effective traffic management, as well as by providing road users with more intelligent advice regarding the routes to select to move from a specific origin to a specific destination (Nagy and Simon, 2018). One of the primary information elements that enables more accurate decision-making in such circumstances is knowledge about future expected traffic speeds on the different road sections forming part of the urban road network (Xu et al., 2018). Such information can for instance be used to modify the current cycle lengths of traffic lights (Gunawan and Chandra, 2014), or increase the accuracy with which the expected travel time along a specific route can be estimated (Gandhi et al., 2020).

Deep neural networks have been proven to outperform older techniques such as Autoregressive Integrated Moving Average (ARIMA), vector ARIMA, Support Vector Machines (SVMs) and others (Žliobaitė and Khokhlov, 2016). Different neural network architectures have been developed for this purpose, based on a variety of approaches including Long-Short Term Memory (LSTM), adversarial networks and graph-based networks (Žliobaitė and Khokhlov, 2016). A common requirement for these techniques is the ability to simultaneously model both temporal and spatial aspects of a network's behavior (Akhtar and Moridpour, 2021). Graph Neural Network (GNN)-based techniques are particularly successful, as discussed in more detail in Section 3.2.

Various authors have focused on traffic speed prediction during times of congestion (Zhou et al., 2020; Chikaraishi et al., 2020). Mohanty (Mohanty, 2018) found that traffic congestion should be analysed as a network-wide phenomenon. Large-scale spatial correlation and long-term temporal correlation that govern traffic congestion propagation across the regional traffic network may be exploited to develop congestion prediction algorithms that are more effective than local predictions. A method for the selection of alternative routes under traffic congestion was proposed by Xu et al. (Xu et al., 2018). They developed a deep learning classifier based on stacked Restricted Boltzman Machine layers followed by a backpropagation layer. A method to predict decongestion time at rail crossings was proposed by Jiang et al. (Jiang et al., 2021). They used computer vision techniques to estimate relevant features (such as the number of vehicles waiting) in to measure and then model decongestion.

None of these works however specifically compared different prediction approaches during periods of congestion. This is recognised as a gap in existing knowledge about traffic prediction methods, which we partially address here. Literature differentiates between two types of congestion: recurrent and non-recurrent (Polson and Sokolov, 2017). In this paper, we compare model performance during periods of both recurrent and non-recurrent congestion.

3 METHODOLOGY

The analysis is conducted using the Los Angeles Metropolitan Transportation Authority (METR-LA) and Performance Measurement System (PeMS)-Bay data sets, described in Section 3.1. We review SOTA traffic prediction systems (Section 3.2), and use this to select an approach and establish a benchmark for the study (Section 3.3). For the selected systems, we perform a congestion analysis: demonstrating how model performance differs during congested and not congested periods, as well as how this changes as the definition of network congestion changes.

3.1 Data set and task

The PeMS-Bay dataset consists of six months of traffic speed data aggregated into 5-minute windows recorded using 326 sensors in the Bay area of Los Angeles by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). The METR-LA dataset consists of four months of traffic speed data aggregated into 5-minute windows recorded using 207 sensors on the highways of Los Angeles. Both the PeMS-Bay and METR-LA datasets are used to train traffic forecasting models for traffic speed prediction. Both of the datasets, as summarised in Table 1, were released by Li et al. (Li et al., 2018) and are popular when evaluating traffic prediction models (Yang et al., 2021).

Table 1: Dataset characteristics.

	METR-LA	**PeMS-Bay**
Sensors	207	326
Period	4 months	6 months
Resolution	5 minutes	5 minutes
Speed unit	miles per hour	miles per hour
Time-steps	34 272	52 116

Along with the sensor readings, an adjacency matrix, constructed from the pairwise road network distances between sensors adjusted using a thresholded Gaussian kernel (Li et al., 2018), is also available for each dataset.

3.2 System selection

To select a traffic prediction system, we first review the published performance of prominent systems for which implementations are publicly available. The methods selected for evaluation are briefly described below.

The **Spatial-Temporal Transformer Network (STTN)** consists of stacked spatial-temporal blocks and a prediction layer (Xu et al., 2020). Each spatial-temporal block consists of a spatial transformer and a temporal transformer. In each spatial-temporal block, the spatial transformer extracts spatial features from the input node as well as the graph adjacency matrix. The spatial transformer consists of a spatial-temporal position embedding layer, a fixed graph convolution layer, a dynamic graph convolution layer and a gate mechanism for information fusion.

Graph WaveNet implements a GNN for spatial-temporal graph modelling by implementing an adaptive dependency matrix that does not require any prior knowledge of the road network. By using stochastic gradient descent the model discovers hidden spatial dependencies by itself (Wu et al., 2019). Dilated causal convolution is used in the temporal convolution layer to capture temporal trends of nodes (Wu et al., 2019).

STAWnet implements multiple stacked spatial-temporal blocks and output layers. A spatial-temporal block consists of a gated temporal convolution network and a dynamic attention network. Dilated causal convolution with a gate mechanism is used for extracting temporal dependencies as for Graph WaveNet. To dynamically model spatial dependencies, the self-attention network is used on graph-structured data to extract patterns in the model (Tian and Chan, 2021).

Graph Multi-Attention Network (GMAN) follows the encoder-decoder architecture with a transform attention layer added between the encoder and decoder to convert the encoded historical traffic features to generate future representations. The encoder and decoder are composed of spatial-temporal attention blocks (ST-attention blocks). The ST-attention blocks are composed of a spatial attention mechanism to model dynamic spatial correlations, a temporal attention mechanism to model non-linear temporal correlations, and a gated fusion mechanism to adaptively fuse spatial and temporal representations (Zheng et al., 2020).

STNN implements spacetime interval learning to explicitly capture intrinsic and latent spatio-temporal correlations through a unified analysis of both spatial and temporal features. To learn the spatial-temporal correlations STNN combines novel spacetime attention blocks and spacetime convolution blocks (Yang et al., 2021). The spacetime attention block highlights the interval between events capturing pair-wise influences, and the spacetime convolution block aggregates the learned features from spatial, temporal, and spatial-temporal aspects to capture many-to-one influences (Yang et al., 2021).

The published performance of these systems is compared in Tables 2 and 3 for the METR-LA and PeMS-Bay data sets, respectively. Performance is obtained from the referenced papers as indicated in each table.

3.3 Establishing a benchmark

The official source code is available for STNN[1], GMAN[2], STAWnet[3], and Graph WaveNet[4]. An implementation is available for STTN[5] as well but this is not referenced by the original proposers of STTN. Using the available software and settings as specified by the authors, each of these systems was retrained on the PeMS-Bay and METR-LA data sets, and the recreated models evaluated on the official evaluation sets. It was possible to recreate the results for Graph WaveNet, STAWnet, and STNN. It was not possible to recreate the published results for either STTN or GMAN without altering the available code. For GMAN we experienced similar issues as were observed by other forum users on Github, while the STTN code was not being developed further at the time of writing.

Only Graph WaveNet, STAWnet and STNN were considered further. For these three models, the performance of the published and recreated results is compared in Tables 2 and 3 for the METR-LA and PeMS-Bay tasks, respectively. Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) are reported, as defined in Eq. 1 to 3: x_i is the predicted speed, y_i is the true speed over n measurements.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{1}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}} \tag{2}$$

$$MAPE = \frac{\sum_{i=1}^{n} |\frac{y_i - x_i}{y_i}|}{n} \tag{3}$$

[1] https://github.com/libingixn/STNN
[2] https://github.com/zhengchuanpan/GMAN
[3] https://github.com/CYBruce/STAWnet
[4] https://github.com/nnzhan/Graph-WaveNet
[5] https://github.com/wubin5/STTN

A negative percentage indicates that the published results are better than the recreated results and vice versa. Comparable results were achieved, verifying the correctness of the implementations. However, it was observed that the published results for STNN were obtained using a different validation and test partition than for the other techniques. In addition, performance was determined by averaging over *all* the prediction horizons up to the one reported on, whereas the prediction errors for the other techniques are determined at an individual prediction horizon only (a single prediction, a specific number of steps into the future). The STNN model was thus re-evaluated using the same test partition as Graph WaveNet and STAWnet and the same performance measure as for the other systems. (Note that neither the training data nor the matching hyperparameters were changed but that the correct validation set was used to select the best-performing model from the training sequence.) The change in the model itself had minimal effect, but the change in evaluation measure had a significant effect. This resulted in a decrease in performance, as indicated by the STNN adjusted results in Tables 2 and 3. Graph WaveNet was therefore found to be the best performing model with STAWnet producing very similar results and STNN performing the worst of the three.

4 ANALYSIS AND RESULTS

In this section, we analyse the performance of the three methods for which the published results could be recreated during periods of congestion.

4.1 Binned traffic speed analysis

To evaluate the performance of the different models during different congestion scenarios, we calculate the MAE of the models' predictions on different ranges of traffic speeds. We analyse this in two ways, by considering the individual sensor speed and overall network speed as two different ways in which to bin predictions. In the first case, *binned sensor speed* analysis, the MAE for all individual sensor predictions that fall within a bin for a given horizon are averaged. In the second case, *binned average network speed* analysis, if the average network speed falls within the bin, the MAE of all sensor predictions at that time step are averaged, irrespective of individual sensor speeds. In both cases, each prediction horizon is kept separate.

The performance of the three analysed models on the METR-LA dataset is shown in Fig. 1 for the binned average network speed and in Fig. 2 for the binned sensor speed. As expected, the three models perform better for faster traffic speeds and worse for slower traffic speeds. This is an indication of the extent to which the performance of the traffic forecasting models deteriorates during times of congestion in the road network. Similar results are observed for the PeMS-Bay dataset. In both cases, the x ticks indicate the centre speed value in the bins, and the bins all have the same sizes.

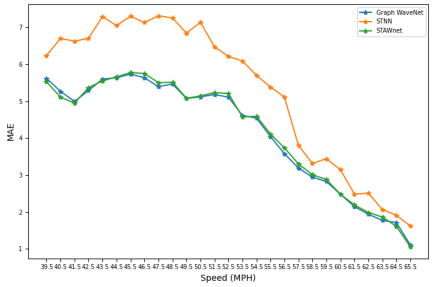


Figure 1: Model performance (MAE) for 60 minute prediction horizon, binned per average network speed for METR-LA test set.

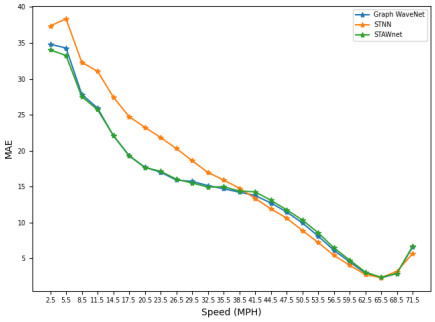


Figure 2: Model performance (MAE) for 60 minute prediction horizon, binned per sensor speeds for METR-LA test set.

4.2 Congestion analysis

All results in this section are reported on the *validation set* as we aim to use some of this information during later congestion modelling. We first analyse network speed averaged across time and day of the week, to determine the significance of recurrent congestion. We then analyse the effect of congestion using different congestion thresholds and three ways to

Table 2: Published and recreated performance of selected SOTA traffic speed prediction models: METR-LA, test set.

	15 min			30 min			60 min		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
Published results									
STNN	2.27	4.46	5.80%	2.56	5.29	6.84%	3.01	6.23	8.50%
STAWnet	2.70	5.22	6.98%	3.04	6.14	8.22%	3.44	7.16	9.82%
Graph WaveNet	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
Recreated results									
STNN	2.29	4.45	5.80%	2.59	5.22	6.84%	3.03	6.26	8.94%
STAWnet	2.72	5.26	6.97%	3.08	6.22	8.30%	3.50	7.27	9.96%
Graph WaveNet	2.69	5.13	6.76%	3.05	6.12	8.17%	3.49	7.21	9.82%
STNN adjusted	2.67	5.28	7.31%	3.19	6.47	9.21%	3.96	8.01	12.34%
Difference between published and recreated results									
STNN	-0.88%	0.22%	0.00%	-1.17%	1.32%	0.00%	-0.66%	-0.48%	0.12%
STAWnet	-0.74%	-0.77%	0.14%	-1.32%	-1.30%	-0.97%	-1.74%	-1.54%	-1.43%
Graph WaveNet	0.00%	0.38%	2.03%	0.65%	1.61%	2.39%	1.13%	2.17%	1.89%

Table 3: Published and recreated performance of selected SOTA traffic speed prediction models: PeMS-Bay, test set.

	15 min			30 min			60 min		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
Published results									
STNN	1.20	2.41	2.53%	1.50	3.26	3.33%	1.86	4.22	4.30%
GMAN	1.34	2.82	2.81%	1.62	3.72	3.63%	1.86	4.32	4.31%
STAWnet	1.31	2.78	2.76%	1.62	3.70	3.67%	1.89	4.36	4.47%
Graph WaveNet	1.30	2.74	2.76%	1.63	3.70	3.67%	1.95	4.52	4.63%
STTN	1.36	2.87	2.89%	1.67	3.79	3.78%	1.95	4.50	4.58%
Recreated results									
STNN	1.21	2.43	2.51%	1.49	3.23	3.23%	1.87	4.18	4.32%
STAWnet	1.32	2.81	2.77%	1.64	3.75	3.69%	1.95	4.46	4.52%
Graph WaveNet	1.30	2.72	2.71%	1.62	3.66	3.64%	1.93	4.43	4.52%
STNN adjusted	1.41	2.94	2.96%	1.83	4.09	4.25%	2.35	5.27	5.85%
Difference between published and recreated results									
STNN	-0.83%	-0.83%	0.79%	0.67%	0.92%	3.00%	-0.54%	0.95%	-0.47%
STAWnet	-0.76%	-1.08%	-0.36%	-1.24%	-1.35%	-0.55%	-3.18%	-2.29%	-1.12%
Graph WaveNet	0.00%	0.73%	1.81%	0.61%	1.08%	0.82%	1.03%	1.99%	2.38%

combine predictions: average network speed, recurrent congestion across all days, and recurrent congestion excluding weekends. Finally, we demonstrate the effect of congestion as the prediction horizon is varied.

In Fig. 3 we show the recurrent congestion observed in the METR-LA data set. The first 120 hours represent the average network speed Monday to Friday and the remaining period represents the weekend. There is a clear drop in average network speed during certain intervals of a typical day. These intervals are also more prominent during weekdays (referred to as 'work days' for clarity). These intervals can be described as periods of congestion. To select periods of congestion for analysis, we can use a threshold, such as the median traffic speed used in Fig. 3, to separate the congested and not congested data.

To compare model performance during times of congestion with performance during the absence of congestion, we set different congestion thresholds around the median speed measured across all sensors, computed using the training partition of the METR-LA and PeMS-Bay datasets. We start with a threshold value that equals the median speed over all sensors

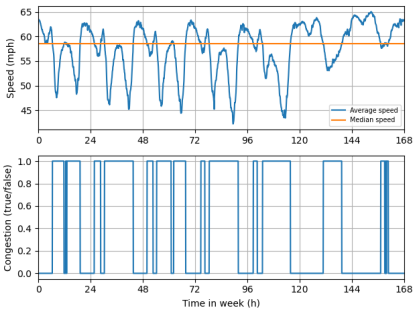


Figure 3: METR-LA training set average network speed per time of day and day of week, and matching congested intervals.

and then vary this threshold, to obtain different divisions of the data set, by adding or subtracting a threshold interval value. We use two different size intervals for speeds faster and slower than the median threshold, as the data distribution is skewed. The threshold values that are smaller than the median speed are determined by subtracting multiples of a threshold interval value computed using Eq. 4 from the median speed, while the thresholds that are larger than the median speed are calculated by adding multiples of the interval value calculated using Eq. 5 to the median speed value.

$$s_s = \frac{s_c - min(s)}{10} \tag{4}$$

$$s_f = \frac{max(s) - s_c}{10} \tag{5}$$

In Eq. 4 and 5 s is the average network speed (across all sensors, per time step), s_c is the centre threshold speed (in this case the median), and s_s and s_f are the speed intervals, slower and faster than the median, respectively.

For our first analysis, we compare the average network speed at each time step in the validation set to the threshold speed. If the average network speed is greater than the threshold speed, the network at this time step is not congested, otherwise, the network is congested. The prediction MAE during congested times is then compared to the prediction MAE during not congested times across the entire validation set and the complete traffic network. Fig. 4 depicts the following for Graph WaveNet for different threshold values: the number of observations that form part of the congested and not congested data, the average MAE for the 60-minute prediction horizon for congested and not congested data, and the difference in MAE between the predictions for congested values and not congested data. From these results the difference in the performance of the models for congested data and not congested data is clear.

For our second analysis, we use a fixed daily division between congested and not congested data by calculating the average daily speed over all days for each time of day. The threshold value is then compared with this fixed daily average speed pattern to differentiate between congested and not congested observations. As each prediction uses the past hour of measured speeds, the daily time interval that represents times of recurrent congestion is chosen as from the hour before the average daily speed drops below the threshold until the hour before the average daily speed goes above the threshold for the last time in the training dataset. This time interval of data is then extracted from the dataset as the congested data and the

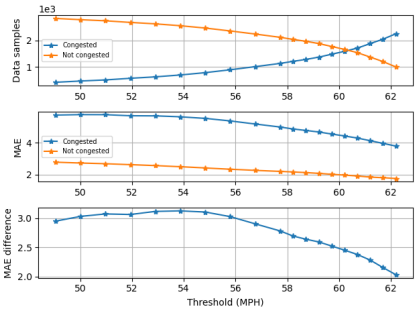


Figure 4: Graph WaveNet per time-step congestion analysis: METR-LA, validation set.

rest of the data is the not congested data. The performance of the model during the congested time interval is then compared to the performance of the model outside of this interval. For the third analysis we use the same method as for the second analysis to determine recurrent daily congestion patterns, but exclude weekend data.

The result for the three described analyses are given in Table 4 for the METR-LA dataset and in Table 5 for the PeMS-Bay dataset. The median threshold is indicated in bold along with the maximum MAE differences. By examining the results in Tables 4 and 5 we can see that STNN does not generalise as well as STAWnet and Graph WaveNet on congested data since the difference in performance on congested and not congested data is much larger for STNN than it is for the other two models. We can also see that, by excluding weekend data from the recurrent congestion analysis, a bigger difference is obtained in performance on congested and not congested data. This is because, as can be seen in Fig. 3, weekend data has a smaller interval of congestion during the day and the same recurrent congestion interval can therefore not effectively be used for weekends.

Finally, we consider the interplay between congestion and the prediction horizon. In Fig. 5 we show the performance of the METR-LA Graph WaveNet model when categorising congestion data based on average network speed per time step and using the median as a threshold. Not only is the model's performance worse for data above the threshold, but the performance of the model also becomes worse more rapidly with increasing prediction horizons. This is a clear indication of how improving the model's performance during times of congestion would benefit the model's traffic prediction capabilities.

Table 4: Performance of the selected systems when evaluated using the three congestion scenarios: The MAE on congested data, and the MAE difference when evaluated on congested and not congested data as the congestion threshold is varied. METR-LA, validation set, 60-min horizon.

Threshold (MPH)	50.97	51.94	52.90	53.86	54.83	**57.72**	58.22	58.72	59.21	59.71
MAE per time step										
GWN congested	5.75	5.68	5.68	5.62	5.52	4.98	4.85	4.76	4.66	4.54
GWN difference	3.07	3.06	3.12	**3.13**	3.11	2.78	2.69	2.64	2.59	2.52
STNN congested	7.57	7.52	7.46	7.33	7.11	5.99	5.88	5.80	5.70	5.54
STNN difference	4.46	4.47	**4.48**	4.44	4.31	3.41	3.35	3.31	3.24	3.13
STAWnet congested	5.72	5.66	5.67	5.63	5.52	5.00	4.89	4.80	4.70	4.57
STAWnet difference	2.99	2.98	3.07	**3.09**	3.06	2.76	2.69	2.63	2.58	2.50
MAE recurrent congestion: all days										
GWN congested	4.38	4.40	3.87	3.87	3.87	3.88	3.88	3.55	3.54	3.53
GWN difference	1.34	1.43	1.29	1.36	1.39	1.50	**1.51**	1.21	1.24	1.25
STNN congested	5.44	5.47	4.45	4.49	4.51	4.55	4.55	4.15	4.13	4.12
STNN difference	**1.92**	2.04	1.39	1.55	1.61	1.80	1.82	1.45	1.48	1.50
STAWnet congested	4.43	4.43	3.90	3.89	3.89	3.91	3.91	3.59	3.57	3.57
STAWnet difference	1.35	1.41	1.27	1.32	1.35	1.49	**1.50**	1.18	1.21	1.23
MAE recurrent congestion: work days only										
GWN congested	5.05	5.16	4.54	4.55	4.57	4.60	4.60	4.15	4.14	4.13
GWN difference	1.62	1.83	1.74	1.85	1.91	2.11	**2.13**	1.71	1.76	1.79
STNN congested	6.41	6.51	5.21	5.31	5.35	5.43	5.44	4.88	4.86	4.85
STNN difference	2.39	**2.62**	1.81	2.08	2.19	2.50	2.53	2.05	2.10	2.14
STAWnet congested	5.10	5.18	4.58	4.60	4.61	4.65	4.65	4.19	4.18	4.17
STAWnet difference	1.62	1.79	1.73	1.84	1.89	2.11	**2.13**	1.69	1.73	1.76

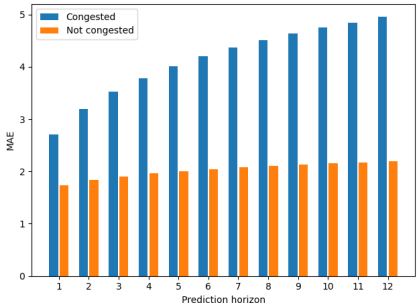


Figure 5: Graph WaveNet per time-step performance on METR-LA dataset for data above and below the median congestion threshold for twelve prediction horizons.

5 CONCLUSION

In this paper, we investigated the performance of deep learning models for traffic speed prediction during periods of congestion. We found that, while all the techniques investigated performed similarly in the absence of congestion, Graph WaveNet and STAWnet outperformed STNN during periods of congestion. It was also found that a longer prediction horizon had a large effect on performance during periods of congestion, and a surprisingly limited effect otherwise. An increased understanding of model behaviour during congestion should assist future efforts to optimize a model for periods of recurrent congestion, which will improve route-finding algorithms during times of the day when fastest path-finding technologies are most necessary. An increase in a model's ability to accurately predict traffic during recurrent congestion will most likely also increase its performance during non-recurrent congestion events such as traffic accidents.

Future work will focus on improving prediction performance during periods of congestion. To optimize a traffic speed prediction model's performance during times of congestion, segments of the day that represent recurrent congestion can be extracted and used to train the model and congestion data can also be repeated within the training set. The loss function of the model can also be modified to give preference to congestion data during training. This will be explored in future work.

REFERENCES

Akhtar, M. and Moridpour, S. (2021). A review of traffic congestion prediction using artificial intelligence. *Journal of Advanced Transportation*, 2021:1–18.

Chikaraishi, M., Garg, P., Varghese, V., Yoshizoe, K., Urata, J., Shiomi, Y., and Watanabe, R. (2020). On the possibility of short-term traffic prediction during disaster with machine learning approaches: An exploratory analysis. *Transport Policy*, 98:91–104.

Gandhi, M. M., Solanki, D. S., Daptardar, R. S., and Baloorkar, N. S. (2020). Smart control of traffic light

Table 5: Same analysis as in Table 4 but for the PeMS-Bay task.

Threshold (MPH)	57.42	58.44	59.47	60.49	61.51	62.53	**63.55**	63.92	64.28	64.64
MAE per time stem										
GWN congested	3.70	3.68	3.66	3.62	3.52	3.25	2.90	2.84	2.80	2.76
GWN difference	2.28	2.31	2.34	**2.35**	2.30	2.12	1.83	1.78	1.75	1.73
STNN congested	5.41	5.39	5.35	5.29	5.12	4.64	4.09	3.99	3.93	3.88
STNN difference	3.64	3.70	3.74	**3.75**	3.66	3.28	2.81	2.74	2.70	2.67
STAWnet congested	3.76	3.74	3.70	3.66	3.55	3.28	2.93	2.86	2.82	2.78
STAWnet difference	2.34	2.36	**2.38**	2.37	2.32	2.12	1.83	1.79	1.75	1.73
MAE recurrent congestion all days										
GWN congested	2.63	2.66	2.67	2.68	2.68	2.67	2.64	2.63	2.61	2.60
GWN difference	1.24	1.37	1.42	1.49	1.54	1.59	1.60	**1.61**	1.61	1.61
STNN congested	3.49	3.61	3.63	3.67	3.70	3.71	3.69	3.68	3.65	3.63
STNN difference	1.60	1.90	1.99	2.13	2.28	2.41	2.47	**2.48**	2.48	2.48
STAWnet congested	2.66	2.70	2.70	2.71	2.71	2.69	2.67	2.66	2.63	2.62
STAWnet difference	1.27	1.40	1.44	1.51	1.56	1.60	1.61	**1.62**	1.61	1.61
MAE recurrent congestion work days										
GWN congested	2.96	3.01	3.03	3.05	3.05	3.04	3.01	3.00	2.98	2.96
GWN difference	1.45	1.62	1.68	1.78	1.85	1.91	1.94	**1.95**	1.95	1.95
STNN congested	3.99	4.15	4.19	4.24	4.29	4.31	4.29	4.28	4.24	4.22
STNN difference	1.85	2.25	2.37	2.56	2.75	2.93	3.02	**3.04**	3.04	3.04
STAWnet congested	3.00	3.06	3.07	3.08	3.08	3.07	3.04	3.03	3.00	2.99
STAWnet difference	1.48	1.66	1.71	1.80	1.87	1.93	1.95	**1.96**	1.95	1.95

using artificial intelligence. In *Proc. IEEE Int. Conf. on Recent Advances and Innovations in Engineering (ICRAIE)*, pages 1–6.

Gunawan, F. E. and Chandra, F. Y. (2014). Optimal averaging time for predicting traffic velocity using floating car data technique for advanced traveler information system. *Procedia - Social and Behavioral Sciences*, 138:566–575. The 9th Int. Conf. on Traffic and Transportation Studies (ICTTS).

Jiang, Z., Guo, F., Qian, Y., Wang, Y., and Pan, W. D. (2021). A deep learning-assisted mathematical model for decongestion time prediction at railroad grade crossings. *Neural Computing and Applications*, page 4715–4732.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *Int. Conf. on Learning Representations*.

Mallick, T., Balaprakash, P., Rask, E., and Macfarlane, J. (2020). Transfer learning with graph neural networks for short-term highway traffic forecasting. In *Proc. Int. Conf. on Pattern Recognition (ICPR)*, pages 10367–10374.

Mena-Oreja, J. and Gozalvez, J. (2020). A comprehensive evaluation of deep learning-based techniques for traffic prediction. *IEEE Access*, 8:91188–91212.

Mohanty, S. (2018). *A Deep Learning, Model-Predictive Approach to Neighborhood Congestion Prediction and Control*. PhD thesis, Berkeley University of California.

Nagy, A. M. and Simon, V. (2018). Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing*, 50:148–163.

Polson, N. G. and Sokolov, V. O. (2017). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17.

Shi, G., Shan, J., Ding, L., Ye, P., Li, Y., and Jiang, N. (2019). Urban road network expansion and its driving variables: A case study of Nanjing City. *Int. Journal of Environmental Research and Public Health*, 16:2318–2334.

Tian, C. and Chan, W. K. V. (2021). Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies. *IET Intelligent Transport Systems*, 15:549–561.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. (2019). Graph WaveNet for deep spatial-temporal graph modeling. In *Proc. of the 28th Int. Joint Conf. on AI, IJCAI-19*, pages 1907–1913.

Xu, J., Zhang, Y., and Xing, C. (2018). An effective selection method for vehicle alternative route under traffic congestion. In *Proc. Int. Conf. on Communication Technology (ICCT)*, pages 494–499.

Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.-J., and Xiong, H. (2020). Spatial-temporal transformer networks for traffic flow forecasting. *ArXiv*, abs/2001.02908.

Yang, S., Liu, J., and Zhao, K. (2021). Space meets time: Local spacetime neural network for traffic flow forecasting. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 817–826.

Zheng, C., Fan, X., Wang, C., and Qi, J. (2020). GMAN: A graph multi-attention network for traffic prediction. In *Proc. AAAI Conf. on AI*, volume 34, page 1234–1241.

Zhou, L., Zhang, S., Yu, J., and Chen, X. (2020). Spatial–temporal deep tensor neural networks for large-scale urban network speed prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3718–3729.

Žliobaitė, I. and Khokhlov, M. (2016). Optimal estimates for short horizon travel time prediction in urban areas. *Intelligent Data Analysis*, 20(6):1459–1475.