2019

A deep learning approach to real-time short-term traffic speed prediction with spatial-temporal features

Sindhuja Gutha
University of Windsor

A deep learning approach to real-time short-term traffic speed prediction with spatial-temporal features

By

Sindhuja Gutha

A Thesis

Submitted to the Faculty of Graduate Studies

through the School of Computer Science

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

at the University of Windsor

Windsor, Ontario, Canada

2019

A deep learning approach to real-time short-term traffic speed prediction with spatial-temporal features

By

Sindhuja Gutha

APPROVED BY:

H. Maoh

Department of Civil and Environmental Engineering

X. Yuan

School of Computer Science

M. Kargar, Co-Advisor

School of Computer Science

J. Chen, Advisor

School of Computer Science

May 15, 2019

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and the intellectual content of this thesis is the product of my own work and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and any ideas or techniques and all the assistance received in preparing this thesis and sources have been fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

In the realm of Intelligent Transportation Systems (ITS), accurate traffic speed prediction plays an important role in traffic control and management. The study on the prediction of traffic speed has attracted considerable attention from many researchers in this field in the past three decades. In recent years, deep learning-based methods have demonstrated their competitiveness to the time series analysis which is an essential part of traffic prediction. These methods can efficiently capture the complex spatial dependency on road networks and non-linear traffic conditions. We have adopted the convolutional neural network-based deep learning approach to traffic speed prediction in our setting, based on its capability of handling multi-dimensional data efficiently. In practice, the traffic data may not be recorded with a regular interval, due to many factors, like power failure, transmission errors, etc., that could have an impact on the data collection. Given that some part of our dataset contains a large amount of missing values, we study the effectiveness of a multi-view approach to imputing the missing values so that various prediction models can apply. Experimental results showed that the performance of the traffic speed prediction model improved significantly after imputing the missing values with a multi-view approach, where the missing ratio is up to 50%.

DEDICATION

Dedicated to my parents Shobha and Ravinder Reddy, my husband Shiv and my sister Hindhuja.

ACKNOWLEDGMENT

I would like to sincerely express my most profound gratitude towards my supervisors Dr. Jessica Chen and Dr. Mehdi Kargar for providing invaluable guidance, continuous support and motivation.

I would also like to thank my thesis committee members Dr. Hanna Maoh and Dr. Xiaobu Yuan for their valuable guidance, comments and suggestions that added more value to my thesis work.

I would like to take this opportunity to sincerely thank Dr. Mina Maleki. As my guide and mentor, she has taught me more than I could ever give her credit for here. I would like to thank my mentors Dr. Hanna Maoh and Dr. Mina Maleki from my research at Cross-Border Institute (CBI) for giving me the opportunity to be a part of this institute for my thesis work.

Also I would like to thank all the staff of Graduate Society of Computer Science for their kindness. I am extending my thanks to all my friends and colleagues who supported and helped me during this period.

On a personal note, I would like to express my deepest gratitude to my parents, my husband, my inlaws and my sister for their immense support from the very beginning.

Collectively, all of their support and guidance has enabled me to successfully complete my masters program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	Definition
ANN	Artificial Neural Networks
DNN	Deep Neural Networks
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
LSTM	Long short term memory neural networks
GRU	Gated Recurrent Units
SVR	Support Vector Regression
CF	Collaborative Filtering
SQL	Structured Query Language
kNN	k Nearest Neighbors
SF	Spatial Features
TF	Temporal Features

Chapter 1

INTRODUCTION

1.1 Motivation

In the transportation field, accurate traffic prediction plays an important role in traffic control and management, and it has gained considerable attention from transportation researchers and practitioners. Analyzing, understanding and estimating future traffic conditions can certainly help road users to settle on better travel choices, improve the quality of traffic operations and reduce traffic congestion. And, to support traffic managers in allocating resources systematically and help people with complete traffic information, understanding traffic movement for the whole road network is of great significance [1]. Due to computational complexity caused by road network topology, spatial correlations in traffic data expanding on a two-dimensional plane, long term prediction to reflect congestion propagation, large-scale network traffic speed prediction is challenging [2]. The main aim of our research work is to predict traffic speed at multiple locations simultaneously on highway 401 in Canada. This highway is one of the busiest corridors in North America. This highway extends from Windsor in the west to the Ontario Quebec fringe in the east [3]. Also, this highway connects to the Ambassador bridge which is an essential single freight link in the Canada-US trade relationship. It carries about 2.5 million trucks for each year, accounting for about 20% of Canada-US trade [4].

In the real-time, traffic speed may be low during a certain time of the day (peak hours traffic) and usually, weekday traffic is different from weekend traffic. Features extracted from this type of information which depends on time are called temporal features. Similarly, traffic in one location depends on the traffic in its neighboring places. Features extracted

from this type of information are called spatial features. And, sometimes the planned incidents such as road maintenance works/construction works affect the traffic conditions. Sometimes unplanned incidents and accidents result in atypical traffic conditions. In addition to the spatial and temporal features, our research work also focuses on atypical traffic conditions because the identification of these conditions can help in better traffic forecasting, provide information to transportation engineers for better road network design and can be used to reduce congestion. Atypical traffic speed can be a significant drop in speed on a section of highway.

These days a large amount of real-time traffic information is available due to the development and deployment of the latest technologies in intelligent transportation systems (ITS). Loop detectors, Global Positioning Systems (GPS) devices and Remote Traffic Microwave Sensors (RTMS), etc. can readily collect traffic data. However, the traffic data is often missing regardless of the technology used. Reasons for incomplete traffic data can be power failures, errors in transmitting the data, hardware or software malfunctions. In many traffic-related times series datasets, missing data are prevalent [5]. Efficient data analysis depends on the quality of data, and the missing data affects the performance of data analysis like traffic forecasting. The performance of several forecasting models will reduce if missing data is present. Even in our dataset, most of the locations contains around 40% of missing data and in some locations it is more than 90%; hence it is necessary to impute the missing data with efficient imputation models. In our dataset, the speed information is collected through cutting edge GPS based vehicle tracking devices which are installed in vehicles. If the vehicles with GPS vehicle tracking devices do not go through any location and during any timestamp then the speed will not be recorded during that timestamp and location. Because of this, we have missing information in our dataset. The missing data problem is challenging especially when the missing data percentage is more. Attributing the missing information by utilizing multi-view approach considering the spatial and tem-

poral features and study the effects of missing data on the performance of traffic prediction , profoundly motivated the present research work.

This thesis first focuses on imputing the missing values based on the Multiview approach [5, 6]. For local temporal and local spatial views we considered collaborative filtering techniques and simple average techniques, and for the global temporal and global spatial views we used the historical average data. Once the complete dataset is available (i.e.,all missing values are imputed), we build the traffic speed prediction model based on CNN [2] and predict the network-wide traffic speed for the next two hours.

The data used in this thesis is collected by GEOTAB and provided to us by the Cross-Border Institute at the University of Windsor. The obtained dataset contains traffic speed records across 72 locations on the 401 Highway from November 2017 to October 2018. The traffic speed data is aggregated in 15-minute time intervals. We used the first eleven months of data for training various models and retain one-month data, i.e., October 2018 data for testing.

The network-wide traffic speed prediction in this research work help in travel time prediction on specific locations on highway 401, crossing time prediction over one of the busiest US Canada bridges, the Ambassador Bridge.

1.2 Research Objective & Solution Outline

This research outlines the prediction of traffic speed using spatial and temporal features across highway 401 at specific locations for the next two hours in the future. The accuracy of the traffic speed prediction depends on many factors like the amount of available historical data, how well the patterns are represented in the historical data, which model is suitable for the nature of the dataset, etc. Our dataset contains more than 40% of missing values in most of the locations, and this might affect the performance of the traffic forecast-

ing model. Hence, our first task is to impute the missing values. Once the complete dataset is available, we build the traffic speed prediction model and predict the traffic speed. The problem statement for the missing value imputation in traffic time series data can be defined as follows: Given data = [l_1, l_2, l_i,....,l_p] with p locations, Where each l_i= [t_1, t_2,....,t_q] represents the time series of traffic speed during q time intervals from the sensor l_i. The unavailable values in this data are called missing values. Based on the available information, missing value imputation models predicts the unavailable data. In Figure 1.1, X represents the missing value. Predicting the future traffic speed of the road segment based on the his-

	t_1	t_2	…	t_{j-3}	t_{j-2}	t_{j-1}	t_j	t_{j+1}	t_{j+2}	…	t_q
l1											
l2	**X**	**X**	**X**	**X**			**X**				**X**
l3									**X**		
….		**X**				**X**			**X**		
l_{i-2}						**X**			**X**		
l_{i-1}			**X**			**X**				**X**	
l_i						**X**	**X**				
l_{i+1}	**X**	**X**	**X**			**X**					
l_{i+2}	**X**	**X**	**X**						**X**		
….	**X**	**X**	**X**								**X**
l_p				**X**							

Figure 1.1: Representation of missing data

torical observations is called Traffic speed prediction. Traffic speed prediction comes under supervised learning which means we must train the model with input/output pairs and the trained model would predict the output for the given input. Such problems come under the category of regression problems. The input data for predicting the traffic speed at single location is w(time lags) historical time steps as shown in the below equation

$$S_i = [S_{i,t-1}, S_{i,t-2}, \ldots S_{i,t-w}]$$

Here $S_{i,t-1}$ indicates the traffic speed at i^{th} location and (t-1)th timestamp.

However, real-time traffic speed at one location may be affected by traffic speed at nearby

locations. Sometimes traffic congestion may propagate from far away locations. In this study, we consider these network-wide impacts into account. Hence the input data for the prediction model is the network-wide traffic speed. We have p locations, and we need to predict the traffic speeds at future n time steps by using w time lags. This input speed data is represented in the matrix format as shown in the below equation.

$$X_T = \begin{bmatrix} S_{1,t-1} & S_{1,t-2} & S_{1,t-3} & \dots & S_{1,t-w} \\ S_{2,t-1} & S_{2,t-2} & S_{2,t-3} & \dots & S_{2,t-w} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ S_{P,t-1} & S_{P,t-2} & S_{P,t-3} & \dots & S_{P,t-w} \end{bmatrix}$$

The predicted output is represented in the matrix format as shown in the below equation.

$$Y_T = \begin{bmatrix} S_{1,t} & S_{1,t+1} & S_{1,t+2} & \dots & S_{1,t+n} \\ S_{2,t} & S_{2,t+1} & S_{2,t+2} & \dots & S_{2,t+n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ S_{P,t} & S_{P,t+1} & S_{P,t+2} & \dots & S_{P,t+n} \end{bmatrix}$$

We have adopted the CNN based model for network-wide traffic speed prediction because CNNs can capture the spatiotemporal features of network traffic with a high prediction accuracy [2]. The objective of this thesis is to impute the missing values and predict network-wide traffic speed. The methods for assigning the missing values include collaborative filtering techniques, simple average and historical data.

1.3 Structure of thesis

The remaining part of this thesis is organized as follows:

Chapter 2 presents a brief introduction to the artificial neural network models and collaborative filtering techniques. This chapter also covers the fundamental algorithms used

for regression problems.

Chapter 3 briefly describes previous studies in the field of transportation for traffic prediction. Various machine learning models and their variations, and other similar approaches for finding missing values are presented in this chapter. And works related to atypical traffic patterns are summarized in this chapter. Several works related to predicting traffic speed are included in this chapter.

Chapter 4 presents the primary contribution of this thesis and describes the implementation details of our approach. The significant contributions of this research are :

- Implementing a multi-view approach for imputing missing values, a CNN based model for network-wide traffic speed prediction with analysis on unusual traffic speed patterns.
- Study the effects of missing data on the performance of traffic prediction.
- Implementing dynamic k-NN combined with CNN model.
- Real-time traffic speed prediction with missing data.

Chapter 5 reports the experimental results along with the detailed discussions.

Finally, Chapter 6 presents the thesis conclusion, and it also includes possible future work.

Chapter 2

BACKGROUND STUDY

2.1 Artificial Neural Networks

Artificial neural networks (ANNs) have been used in various disciplines for solving complex real-world problems. These are computing systems which are inspired by the biological neural networks. In traffic prediction problems, they approximate a mapping function from input variables to output variables based on the historical data (training data). Deep neural networks (DNNs) are ANNs with multiple layers between the input and output layers , and the set of mathematical operations are used to turn the input into the output. In recent years, deep learning-based methods (such as LSTM, CNN) have demonstrated its competitiveness to the time series analysis which is an essential part of traffic prediction. These methods can handle the complicated nonlinear spatial, and temporal correlations and the different variants of these methods have been used for traffic speed prediction. The structure of the ANN is analogous to the structure of a biological neural system. It is composed of multiple processing units (called artificial nodes or neurons) connected in consecutive layers to work together and produce the final output. ANNs have a fantastic information processing characteristics such as non-linearity, fault and failure tolerance, robustness, high parallelism, and their capability to generalize [7]. In transportation research, ANN models have a long application history. Compared to the classical statistical models, ANN models can capture the nonlinear relationship between dependent and independent variables without the need for any prior knowledge about the non-linear relationship [8].

A Neuron in the context of transportation

The basic building block of ANN is the neuron. It can be considered as a computational unit which receives input and process that input to produce the output. Each input has some weight. This processing can be the simple summation of products of inputs and their weights or the summation of products of inputs and their weights passed through some activation function. The activation function is to provide the non-linearity to ANNs. The activation function produces the output in the desired range such as 0 to 1 or -1 to 1 . For example, the logistic activation function produces the output in the 0 to 1 range [9].

Suppose the last one-hour traffic speed information is used to predict the speed in the next 15 minutes and the traffic speed data collected is for every 15 minutes. In this scenario, the number of input features for the neuron is four , and the output is one.

In Figure 2.1, x1, x2, x3, x4 are input features and their respective weights are w1, w2, w3, w4. w0 represents the bias value and y represents the output value.

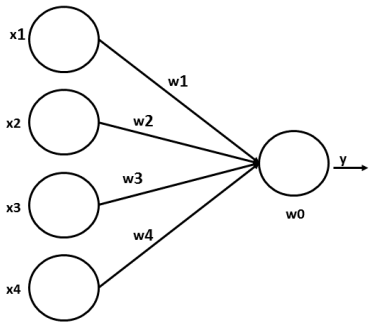

Figure 2.1: A Neuron

Output value y is computed by the below equation which is the function of sum of products of input features and their weights and the bias value.

$$\hat{y} = f(x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4 + w_0) \tag{2.1}$$

$$\hat{y} = f(WX + w_0) \tag{2.2}$$

$$f(x) = \phi(x) \tag{2.3}$$

where ϕ represents the activation function

If $f(x) = x$, the output value of the neuron $y = WX + w_0$ which is the equation for linear regression, here the activation function is identity function.

If $f(x) = sigmoid(x)$, the output value of the neuron $y = \frac{1}{1+e^{-}(WX+w_0)}$ which is the equation for logistic regression, here the sigmoid is the activation function.

The parameters in the above model (single neuron) are weights(W) and the bias values(w_0). These values are learned from the training dataset. Artificial neural networks are just the combination of many such neurons in multiple consecutive layers which are called hidden layers. These neurons are connected in such a way that they can process the information together and solve complex problems.

2.1.1 Feed Forward Neural Networks

In this neural network, information flows in one direction only from the input layer to the output layer and the neurons from one layer are connected to all the neurons in the next following layer as shown in Figure 2.2.

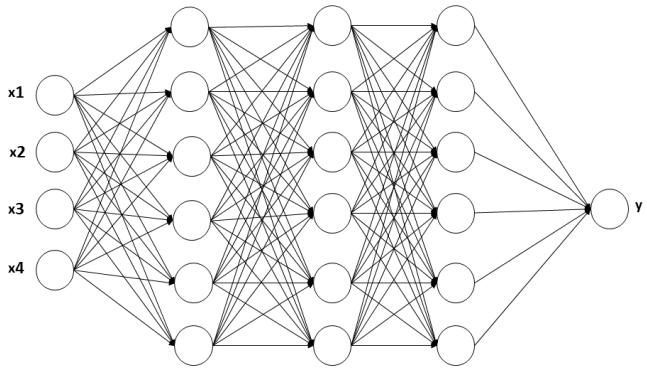

Figure 2.2: Feed Forward Neural Networks

The neurons are connected by weighted connections, which carry the information from one layer to the subsequent layer. These neural networks consist of one or more hidden layers. They can approximate nonlinear functions by adapting nonlinear activation functions. The model can be called a deep neural network if it has two or more hidden layers.

Training of the neural network model A common objective for ANN in regression problems is to minimize the sum of squared errors which means to reduce the error between the actual values and predicted values (Equation 2.4). To achieve this objective, we need to train the neural network model. Training means adjusting the parameters of the model from the training dataset. In other words, it means estimating the best set of parameters i.e., weights and bias of the model such that there will be less error in prediction. A backpropagation approach is typically used to train the neural network model [8] and it has been applied in predicting traffic speed. A number of other optimization algorithms were also developed to achieve this goal like the Genetic algorithm etc.

$$J = 1/n \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \tag{2.4}$$

The backpropagation algorithm is a supervised learning algorithm which is based on a gradient descent technique to train neural networks. The primary goal of this gradient descent method is to minimize the error function (which is also called as a cost function or objective function) by adjusting the parameters in the model. In each training step, error function (Equation 2.4) is calculated, and the parameters(weights and bias) of the neural network are adjusted by back-propagating the error from the output layer to the input layer so that the error in the next iteration will be reduced. In other words, the training process involves running the model in both directions. First, we get the prediction using the current model parameters by forward pass which means we pass the input to the input layer and which is passed through the consecutive hidden layers until the last layer where the output is predicted. In the next step, we compare the predicted value with the actual value and

calculate the error function (Equation 2.4). Then we calculate the partial derivative of the error function (gradient). In the next step we update the parameters of the model along the opposite direction of a gradient to minimize the error function value. Once the model is ready (error rate converged or reached maximum number of epochs), we can use it for prediction.

Gradient descent technique It is a popular optimization method which is used to find maxima/ minima of a differentiable function. It minimizes the error function by iteratively updating the parameters in small steps by using the direction of the gradient of the error function w.r.t. to the parameters.

Let W be the set of parameters

Step 1: Initialize W randomly

Step 2: Update the parameters based on the below equation

While not converged or till the maximum number of epochs :

$W = W - \eta * gradient(costfunction, trainingdata, w)$

Here η is the learning rate which decides how much the parameters are updated in each step and one epoch means all training samples are passed through the neural network only once.

There are three variants of gradient descent. The difference between these three is in how much of the total training data is used to compute the gradient of the error function (also called loss or cost function) [10].

- **Vanilla gradient descent:** The gradient of the cost function w.r.t. to the parameters is computed using the entire training dataset. Here to perform just one update, the gradients for the whole dataset are calculated. It can be very slow when the size of the dataset is large.

- **Stochastic gradient descent(SGD):** Parameters are updated for each training

sample. For large datasets, redundant computations are performed in vanilla gradient descent as gradients are recomputed for similar examples. SGD solves this by performing one update at a time. It is much faster in learning, but the frequent updates can cause the cost function to fluctuate more.

- **Mini-batch gradient descent:** Parameters are updated for every mini-batch of training samples. It requires less memory and leads to more stable convergence. This batch size is one of the hyperparameters in training the neural network model.

There are several gradient-based optimization techniques such as Adagrad, Adadelta, RMSprop, Adam, AdaMax, etc. widely used by Deep Learning community [10] . And also various activation functions exist such as sigmoid, tanh, ReLu, etc. In this study, for all the neural network models we use the Adam optimization algorithm and the ReLu activation function in CNN model, tanh activation function in LSTM model.

2.1.2 Recurrent Neural Networks

The traditional feedforward neural networks do not consider the temporal information for time series inputs. For example, if we consider four previous time lags to predict speed in the next time interval, then these four-time lags are regarded as independent features and the temporal information between these time lags is not considered. Recurrent neural networks can learn the temporal information; hence they are suitable for dealing with time sequences. RNN hidden units receive feedback from the previous state to current state [11]. RNNs use the context information from previous time steps, which is usually referred to as memory. Because of this capability, they can capture flexible temporal dependencies and learn the correlations in sequential data. Fixed number of input features are fed into the conventional neural networks while in RNN, the input features are fed one at a time

and the final output not only depends on the current input but also depends on the previous hidden layer output.

These networks were initially used for language models. RNNs unfold into the very deep feedforward neural networks with more time lags. Because of this, the gradients of the network may vanish or explode. In other words, if the time span becomes longer, the accuracy of the network may reduce due to the vanishing gradient and exploding gradient problems. The neural network models LSTM and GRU (Structures of RNN) were proposed to solve this problem.

LSTM model was proposed [12] to overcome the vanishing gradient problem in traditional RNNs. Vanishing gradient problem prevents the RNNs from capturing the long-term dependencies. A mechanism referred to as a gating mechanism in this model makes the decisions about updating its memory. Typical LSTM unit consists of an input gate, an output gate, a forget gate and a cell. Input gate decides what amount of new information should be stored in the memory; the output gate decides what information in the memory is used to calculate the output of the LSTM unit; the forget gate decides what information should be deleted from the memory. Here the cell represents the memory.

Due to this forget gate, the method can decide when to forget certain information. Output layer of the LSTM cell is the linear regression layer. By maintaining a memory state of the cell c_t, LSTM can learn sequential correlations. The output of the memory cell is controlled by the output gate.

Hidden layers are treated as memory units in LSTM network. They can learn the correlations within time series in both short term and long term. The center of the LSTM unit is the memory cell, and the state of it is denoted by c_t. Input data s_t (traffic speed at time interval t) and output of the LSTM cell in the previous time interval h_{t-1} are the inputs of every gate. The final state of the memory cell and the final output of the memory unit are

calculated using the below equations.

$$i_t = \sigma(W_{i,s}s_t + W_{i,h}h_{t-1} + b_i) \tag{2.5}$$

$$f_t = \sigma(W_{f,s}s_t + W_{f,h}h_{t-1} + b_f) \tag{2.6}$$

$$o_t = \sigma(W_{o,s}s_t + W_{o,h}h_{t-1} + b_o) \tag{2.7}$$

$$\tilde{c_t} = \tanh(W_{c,s}s_t + W_{c,h}h_{t-1} + b_c) \tag{2.8}$$

$$c_t = i_t \odot \tilde{c_t} + f_t \odot c_{t-1} \tag{2.9}$$

$$h_t = o_t \odot \tanh c_t \tag{2.10}$$

The input gate, the forget gate and the output gate at time interval t are denoted as i_t, f_t, o_t. Tanh is a hyperbolic tangent function and o denotes Hadamard product. These both functions are element-wise.

In another variant of LSTM, the input, output, and forget gates also look at the memory state of the cell c_t along with s_t, h_{t-1}.

2.1.3 Convolutional Neural Networks

A CNN is an efficient image processing algorithm and achieved successful results in the computer vision and image recognition fields [13].

Hidden layers in CNN architecture are convolution layers, pooling layers, fully connected layers. Different filters (or kernels) in a convolution layer extracts different features. These filters are set of weights which are learned through the training process to produce the output features. A pooling layer is usually applied after a convolution layer which selects the essential features from the extracted features of the convolution layer , and due to this, the model parameters are reduced tremendously. In traditional feedforward networks, one layer is fully connected to the next layer but in CNN networks, convolution layers are connected locally through sliding filters. With this local connectivity, the CNN model can

capture the local spatial features. The typical fully connected layers are used only in the last stage after the convolution and pooling layers.

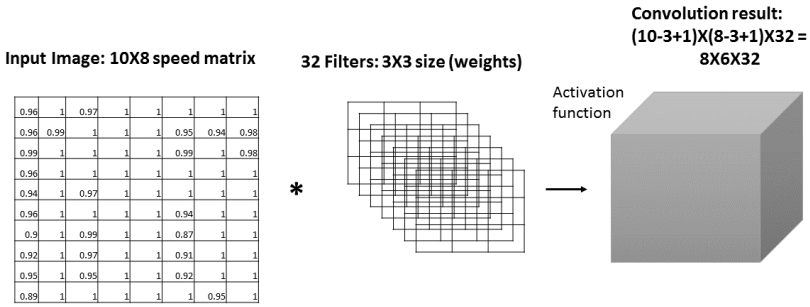

Figure 2.3: a) Convolutional Neural Networks

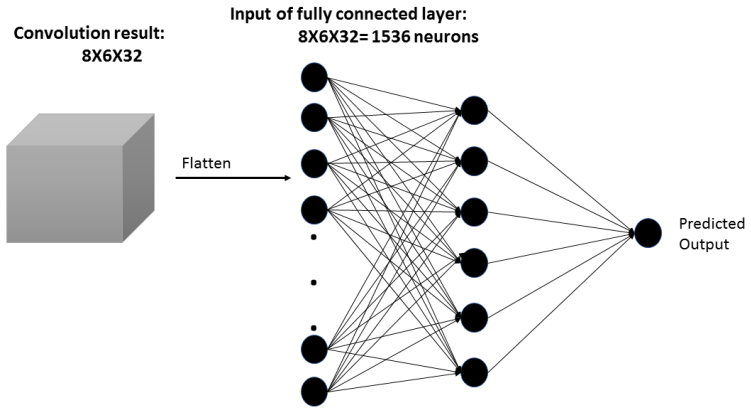

Figure 2.4: b) Convolutional Neural Networks

If we place the convolution filter on top of the input matrix, then the product between the numbers at the same location in the input matrix and the filter are calculated and these products are summed up together to get a single number which is the convolution result of this operation. Then the filter is moved to its right by one element (Here the stride is (1,1) which means filter moves by one element to the right, one element to the down until all values in the matrix are covered) and get convolution result. Likewise, the filter moves

throughout the input matrix to get the final convolution result matrix. In the traffic context, if the size of the input matrix is ten rows which represents the locations, eight columns which indicates time steps and 32 filters with size 3X3, then the convolution result is of size 8 X 6 X 32 as shown in Figure 2.3. In Figure 2.3, input to the CNN model is the traffic speed matrix with size 10 x 8 where we consider traffic speed data from 10 locations and 8 previous time steps(time lags) to predict the speed at 10 locations in the future time step. Extracted features through set of filters from the input speed matrix will be the input to the fully connected layers. The last layer in Figure 2.4 is with the identity activation function and predicts the final output for a given input.

2.2 Multi View Learning Approach

Collaborative filtering techniques are widely used in recommender systems. In recommender system, the similar users make similar rating for similar items [14].

2.2.1 Temporal Collaborative Filtering

In the temporal view for traffic context, each traffic data point at ti is considered as an item. Let p be the number of locations and traffic speed is represented by s; locations are represented by i; timestamps are represented by j. The similarity between the two traffic data points which are at two different timestamps can be computed according to the Equation 2.11.

$$\text{sim}(t_{miss}, t_j) = \frac{1}{\sqrt{\sum_{i=1}^{p} (s_{i,miss} - s_{i,j})^2 / p}} \tag{2.11}$$

$S_{i,miss}$ is the traffic speed at the timestamp where traffic speed value is missing , and it is from the ith location. The local variation in temporal view is computed by the Equation

2.12.

$$\hat{s}(t_{miss}) = \frac{\sum_{j=1}^{w} s_j * sim(t_{miss}, t_j)}{\sum_{j=1}^{w} sim(t_{miss}, t_j)} \tag{2.12}$$

Here w is the window size, it provides the range in which traffic speeds should be considered and t_{miss} represents the timestamp at which traffic speed is missing. Here the missing traffic speed (at t_{miss}) is computed as the weighted average of traffic speeds which are in the same location as t_{miss}. The weights are the similarities between two timestamps (t_{miss}, t_j) at different locations.

	t_1	t_2	…	t_{j-3}	t_{j-2}	t_{j-1}	t_j	t_{j+1}	t_{j+2}	…	t_q
l_1											
l_2	X						X				X
l_3									X		
….		X									
l_{i-2}						X			X		
l_{i-1}			X							X	
l_i							X				
l_{i+1}		X									
l_{i+2}									X		
….											X
l_p		X		X							

Figure 2.5: CF Temporal

To fill the missing value at t_j, we must compute the weighted average of traffic speeds which are in the window range, t_{j-2}(w1) +t_{j-1}(w2) +t_{j+1}(w3) +t_{j+2}(w4)/ (w1+w2+w3+w4) Here *w1* is the similarity of two-time stamps (t_j, t_{j-2}) at different locations, similarly *w2*

is the similarity of two-time stamps (t_j, t_{j-1}) at different locations, *w3* is the similarity of two-time stamps (t_j, t_{j+1}) at different locations, *w4* is the similarity of two-time stamps (t_j, t_{j+2}) at different locations.

2.2.2 Spatial Collaborative Filtering

In this view, the locations are considered as items. The similarity of two locations at different timestamps can be calculated according to Equation 2.13.

$$\text{sim}(l_{miss}, l_i) = \frac{1}{\sqrt{\sum_{j=1}^{w} (s_{miss,j} - s_{i,j})^2 / w}} \tag{2.13}$$

Here w represents the window size and $s_{i,j}$ is the traffic speed at ith location and jth timestamp. The Equation 2.14 computes the local variation in spatial view.

To predict the missing value at location l_i, the weighted average of all available data points at various locations must be computed (same timestamp for location l_i and all other locations).

l_{i-2}(w1) +l_{i-1}(w2) +l_{i+1}(w3) + l_{i+2}(w4)/ (w1+w2+w3+w4)

Here w1 is the similarity between the two locations (l_i, l_{i-2}) at different time stamps which are in the window range. Similarly w2 is the similarity between the two locations (l_i,l_{i-1}) at different time stamps which are in the window range, w3 is the similarity between the two locations (l_i,l_{i+1}) at different time stamps which are in the window range, w4 is the similarity between the two locations (l_i,l_{i+2}) at different time stamps which are in the window range.

$$\hat{s}(l_{miss}) = \frac{\sum_{i=1}^{p} s_i * sim(l_{miss}, l_i)}{\sum_{i=1}^{p} sim(l_{miss}, l_i)} \tag{2.14}$$

	t_1	t_2	...	t_{j-3}	t_{j-2}	t_{j-1}	t_j	t_{j+1}	t_{j+2}	...	t_q
l_1											
l_2	X						X				X
l_3									X		
....		X									
l_{i-2}						X			X		
l_{i-1}			X							X	
l_i							X				
l_{i+1}		X									
l_{i+2}									X		
....											X
l_p		X		X							

Figure 2.6: CF Spatial

2.3 Regression techniques

2.3.1 K- Nearest Neighbours

The k-nearest neighbor's algorithm is a supervised non-parametric technique used for both regression and classification [15]. It is one of the most straightforward and efficient machine learning algorithms. In the k-NN regression, we choose the k nearest neighbors from the training dataset and compute the output as the average of these k nearest neighbors. k-NN regression algorithm:

- Calculate the distance between the query sample and training samples

- Sort the training samples by ascending order
- Choose the first k nearest neighbors from this sorted array
- Compute the output by taking an average of these k nearest neighbors

Here the distance metric can be Euclidean or any other distance metric. There are various types of k-NN regression models in which different weights can be assigned to neighbors or for features in the sample. These algorithms are robust to noisy training data and can be efficient if training data represents almost all patterns. However, it may take more time to compute the distances for each query sample when the training data size is huge.

2.3.2 Support Vector Regression

The support vector machines (SVM) are one of the popular supervised learning models in machine learning which are used for classification and regression tasks. Authors [16] proposed a version of SVM for regression, called as support vector regression (SVR). This method constructs a hyperplane or a set of hyperplanes in a high dimensional space on training data. Later on, this hyperplane would help us in predicting the value. Two boundary lines create a margin, and these are different from the hyperplane. The best hyperplane represents the largest margin. The cost function for building the SVR model depends only on a subset of training data; hence it is memory efficient. In summary, the SVR regression model first maps the input onto a high dimensional feature space using nonlinear mapping and then linear regression is performed in this space.

Chapter 3

LITERATURE REVIEW

This chapter gives a brief overview of the research work carried out for imputing the missing values and predicting the network wide traffic speed prediction. A lot of research work has been done in the both areas. Methods for imputing the missing values can be the simple mean value imputation or the complex spatial-temporal imputation methods. These methods are also highly depend on the nature of dataset, amount of the available data, etc.

In this section, we briefly introduce various methods used to impute missing values. Various types of methods have been developed and different techniques are used as per different application needs.

3.1 Related works on the imputation of missing values

Methods for imputing the missing data can be divided into three categories: interpolation based, statistical learning based, and prediction based [17]. In interpolation-based methods, we fill the missing data with an average or weighted average of historical data observations either from the same sensor and same time in neighboring days or from the same sensor and same time in other days with similar traffic pattern. In this category one of the popular approaches to impute missing traffic data for a road segment is the historical average. In this method, the missing data point is imputed for a road segment based on the average value of historical data corresponding to the same time interval and same location [18]. However, this approach performs poorly in the presence of unusual traffic conditions. The most popular pattern matching interpolation methods are based on k-nearest neighbors (k-NN). In this method, the missing value is calculated from the k nearest neighbors from the

whole dataset [19]. The k nearest neighbors are similar traffic patterns in other days. The performance of these approaches is affected by the availability of neighboring data.

Statistical learning-based methods try to take advantage of the statistical feature of traffic data. These methods assume a special probability distribution from the available data and the values which best fit the assumed probability distribution will be used to impute the missing data. Markov Chain Monte Carlo (MCMC) imputation method [20] and Probabilistic Principal Component Analysis (PPCA) based imputation methods are popular in this category. The PPCA- based model combines two techniques: a) principal component analysis (PCA) which is used to separate the significant parts of the traffic data from the unable to model parts. B) The second technique is maximum likelihood estimation (MLE) which is used to estimate the missing values based on the obtained significant parts. In other words, PPCA extracts the statistical characteristics of the available data and indirectly builds the regression model. This model makes a reasonable trade-off between the local predictability, periodicity and other statistical properties of the traffic data [21]. Authors [17] extended the probabilistic principal component analysis (PPCA) based imputing method by considering multiple sensors measurements. Experimental results show that the temporal-spatial dependence is nonlinear, and it could be better retrieved by kernel probabilistic principal component analysis (KPPCA) based method instead of PPCA method. Overall if temporal- spatial dependence is appropriately considered, the imputing errors can be significantly reduced. The KPPCA model is more powerful than the PPCA model because it is less strict about the linear mapping assumption of PPCA model because it assumes a nonlinear relationship between an observed sample and a latent variable. Authors in this paper compared the performance of these two methods for four different scenarios with missing ratios 5% to 30%. In the first scenario, they collected the data from a single detector for one month. The RMSE error value for these two methods is smaller than the conventional simple average method (collected at the same daily point in the last 1 or 3

months), nearest historical imputing method (collected at the same daily point in the nearest day) and spline interpolation method. However, the difference between the PPCA and KPPCA method is minimal. In the second scenario, the dataset is from a single detector, but it is collected for three months. The error rate is unchanged in this scenario which indicates that the imputing error cannot be reduced by feeding more data. In the third scenario, one-month data is collected from three detectors. Error rates are decreased in this scenario which shows the advantages of spatial dependencies in PPCA and KPPCA methods. The fourth scenario is like the third one, but the influence of time lag is considered. In this one, KPPCA method is better than the PPCA method. Authors stated that the calculation time of KPPCA method is more than PPCA. Hence, this method considering only spatial dependence is highly recommended for online systems. In summary, the performance of PPCA based methods are often better than the conventional methods although these methods have a strong assumption on the data. Another group of approaches are based on regression algorithms which include linear regression model, quadratic regression model, etc. To capture the relationship between neighboring sensors, authors in [22] used a linear regression model to impute missing traffic volumes and occupancies. Authors in this paper modeled each pair of neighbors linearly and the parameters of the model are fixed based on the historical data.

Prediction based methods use existing traffic prediction methods. Some of the famous traffic prediction models are time series models based on the Auto-Regressive Integrated Moving Average (ARIMA) model, Support Vector Regression (SVR), Artificial Neural Networks, etc. In these models value is predicted using the function which is formed from historical data pairs (Input/ Output) [23]. Authors in [24] studied the effects of missing data on ARIMA and neural network model performances. They developed two-hybrid approaches. One is based on the self-organizing map (SOM) and ARIMA, and other is based on SOM and neural network. For a neural network, multilayer perceptron was

used. In SOM/ARIMA hybrid approach, a series of traffic patterns are classified using a Self-Organised Map and then ARIMA model is associated with each SOM cluster. The SOM/MLP model was similar to the SOM/ARIMA model. Three other naive models were also used for forecasting. The first one simply used the current traffic flow as the predicted value. The second naive model assumes the linear relationship between the predicted, present and the last traffic flows. In this study, SOM/MLP model achieved the best forecasting performance compared to all other models. To study the effects of missing data on forecasting performance, missing data is generated randomly until 30%. Three methods were used to deal with the missing data. The first method is the average value of data used for forecasting, second is replacing the missing value with 0, and the third one is the average of all traffic flow data. The first method showed the best results among all. The second method is the worst option among all. And also, ARIMA models are more sensitive to the percentage of missing data when compared to neural networks. Most of the prediction models predict the data by considering a few current time lags before the value to be predicted. In this scenario, they do not use the data after missing timestamp, and if the specified time lag data is missing, these prediction models fail to give the results.

In recent years, several machines learning based approaches have been proposed for imputing missing traffic data. The data usually determine the form of the function in these approaches. They try to take full advantage of the data. One of the popular machine learning based approaches for imputing the missing values are tensor-based methods [25, 26]. In this paper [27] authors proposed a Bayesian probabilistic imputation framework for imputing the spatiotemporal traffic data. In this method, the Bayesian probabilistic matrix factorization model was extended to higher order tensors and applied it for imputation tasks. The performance of the method is evaluated by using a nine-week traffic speed dataset. The spatiotemporal traffic data can be organized into a multidimensional array (road segment * day*time of day) which is called as the sensor. In this study, the data was organized into

three different representations (A) Matrix (road segment * time series), (B) Third-order tensor (road segment * day * time interval), and (C) Fourth-order tensor (road segment * week * day * time interval). The performance of different tensor-based approaches are evaluated using these three representations. Through extensive experiments, this study showed how different data representations affect imputation performance and proved that data representation is an essential factor for model performance. And also showed that a third order tensor structure outperforms the matrix and fourth order tensor representations.

The Kriging method can also be used for imputing the missing data. It was first developed as an interpolation technique for geographical surfaces, it has become a representative geostatistical approach to predict an unknown value at an unobserved location by adopting various statistical assumptions and conditions in the modeling and has further advanced to different kriging methods. Here the interpolation was formulated as a weighted sum of the values of their known neighbors. The multivariate extension of kriging is Cokriging which allows using secondary data sources to complement observed primary data. In this study [28], authors employed ordinary and simple cokriging methods to use a secondary trac dataset for imputing the missing detector data. Using the information from multiple data sources can improve the imputation results of the spatial-temporal cokriging approach. Authors in this paper proposed two cokriging methods that exploit the existence of spatial-temporal dependency in trac data and employ multiple data sources, each with independently missing data, to impute high-resolution traffic speed data under dierent data missing pattern scenarios. Consider secondary data sources only if they are highly correlated with the primary data to improve the prediction performance of cokriging. The cokriging methods may have the test errors if there is a relatively weak correlation between two data sources. When the missing pattern follows not random in time and location pattern, using secondary data sources with the simple cokriging can improve prediction results. The prediction errors for these scenarios decrease gradually as the missing rate increases because

the proportion of high-speed observations in the validation dataset of a higher missing rate scenario increases. The overall imputation error decreases as the size of missing block increases, since these high-speed observations are like the mean of the RTMS data used in this study. Overall, this paper provides a brief guidance of when and how to utilize multi-source data for traffic speed data imputation using the real traffic datasets.

Clustering based approaches have also been proposed for imputing the missing traffic data. Authors in this paper [29] proposed the method in which first the road segments with similar traffic flow patterns are grouped through k-means clustering. In the next step, for each group of road segments, a deep learning model based on stacked denoising autoencoders is used to extract the spatial-temporal relationships between those road segments and use this information for imputing the missing data points. Implemented experiments show that under different missing rates, the imputation accuracy of the proposed method is robust. In the proposed method, missing data is imputed collectively for a group of road segments. An autoencoder is a neural network which has the same number of input and output neurons. The input vector is reconstructed in this model. A denoising autoencoder (DAE) is a variant of autoencoder for which corrupted input is passed during the training process. In this model, the hidden layer learns the robust features underlying the input data. For the training sample, some random entries in the input are masked with zero. This training sample is considered as the incomplete traffic data with some missing values. Each layer in the proposed method is pre-trained using a greedy layer-wise approach and to improve the imputation performance; the whole network is fine-tuned. The proposed method can impute the missing data on any road segment using the traffic flow relationships with other road segments in the same cluster. Authors in this paper [30] proposed a denoising stacked autoencoders (DSAE) model for imputing traffic flow data. This model consists of two blocks, one is autoencoders (AEs) and the other one is denoising autoencoders (DAE). AE can extract features from original input data. A DAE can cap-

ture the statistical dependencies between the inputs. If DAEs are connected to form deep networks, then such models are called stacked denoising autoencoders (SDAE). However, both DAE and SDAE can be used for cleaning or denoising the data. In this paper, authors combined the DAE which is used for denoising the corrupted traffic data, with the stacked AEs which help in extracting the features from high dimension traffic data. This combination of models results in forming a deep learning model named DSAE which has the advantages of both DAE and stacked AEs. Overall this model recovers the data through statistical dependency learning and the feature extraction. The traffic flow data from PeMS aggregated in five-minute intervals is used in this paper. Traffic flow data for each day are represented in the form of vectors. Each vector indicates whether it is a weekday or not. Authors used a k means clustering method to group these vectors into weekdays and non-weekdays. They assumed that vectors of the same date from different VDSs are similar (same weekday property). So, there are two clusters, one with weekday pattern and the other with the non-weekday pattern. A new corrupted vector is assigned to the closest cluster based on the distance. Authors in this paper investigated 18 different scenarios for the model with different spatial (upstream, downstream, all other VDSs) and temporal factors (weekdays and non-weekdays). Experimental results show that the model using all VDSs data of weekdays and non- weekdays is the best. The proposed model is compared with the history model, ARIMA and BP neural network model under the missing rates ranging from 5% to 50%. The experimental results show that the imputation accuracy is better in the proposed model. Also, authors in this paper show the consistency of the imputed data by the proposed model with the observed data.

Authors in this paper [31] proposed two machine learning approaches to impute missing data. One approach is based on the information provided by surrounding sensors, and the other method is a clustering technique that use optimal pattern clusters to impute missing values. Clustering technique includes external data such as days of the week, months or

holiday information. Authors used an Extreme Learning Machine model optimized with a genetic algorithm to capture surrounding sensors information. The dataset used in this paper is collected from the city council of Madrid (Spain) and it is aggregated in 15-minute periods for the years 2014, 2015 and three months of 2016. The model on surrounding sensors depend on neighboring sensors with complete information. So, for training and testing of this model, authors choose sensors with more than 98% available data. This model depends on the relationships between measurements of different sensors at the center of the city. These sensors need not be nearby sensors. In this, the model is like a forecasting model which use the information provided by other sensors and predicts the missing data value. The second method proposed in this paper includes both clustering and classification. The day wise traffic flow vectors are constructed, and a clustering technique is performed on this dataset to obtain groups of days with similar measurements. Clustering on data with a large number of dimensions needs enormous computational resources. And also, it could be biased by localized, high-frequency noise and produce too many numbers of clusters. To overcome this, the dataset is preprocessed by averaging every k samples. Authors considered both DBSCAN and Affinity Propagation clustering techniques and stated that both algorithms produced similar results if their parameters are appropriately chosen. Hence, DBSCAN is used for all the experiments. Missing values are imputed by choosing the closest cluster. This clustering method may not be able to impute the missing values for a day with all measurements missing. To overcome this issue, authors used the classification technique where clusters are the classes, features are the day of the week, the month and a binary feature to indicate a day is a bank holiday or not. The random forest supervised classifier is used for training on the dataset with these three features and C classes (clusters). Through this process, cluster can be assigned to the day with all missing values. The percentage of missing values in this paper range from 1% to 100%. In this paper, the imputed data are analyzed from the view of their ability to get accurate predictions. Authors

stated that if the percentage of missing data is less, the rest of the data is enough to build a good forecasting model. If a machine learning based approach capable of dealing with noisy data is used for prediction and the percentage of imputed data is less than 10%, then these values will have less impact on the final forecasting performance. For large amounts of missing data, the proposed algorithms in this paper produce robust predictions.

Recently, Long Short-term Memory (LSTM) neural network has been applied to time-series forecasting tasks. When a large amount of annotated data is provided, the inference accuracy can be improved. Most of the existing approaches utilize only valid data to train the network model, which dramatically decreases the training set size. Some approaches utilize a temporal smoothness constraint to infer the missing data or take advantage of the mean to study the missing data. But these solutions often cause the compensation process to differ from the prediction models and the missing patterns to be explored inefficiently, thereby resulting in suboptimal analyses and predictions. In this paper [32], authors developed a novel LSTM-based traffic ow forecasting method which not only acquires the long-term and short-term temporal dependencies of time-series observations but also utilizes the missing patterns to improve the prediction results. Authors stated that although LSTM networks have achieved competitive results in traffic ow prediction, there has been little work on handling missing values in the LSTM network structure. When a missing value is imputed via mean or temporal smoothing, it is impossible to distinguish whether the value is an imputed missing value or a true value. Merely concatenating the time interval vectors and the valid masking fails to exploit the temporal structure of the missing values. In this paper, the missing patterns in the data are modeled in the network structure. Most missing observations can be divided into two categories (I) short-period missing values, which can last less than 5 min and these missing values are chiey caused by unsteady equipment or a cluttered environment. II) long-period missing values, which can last hours, or even days and these missing values are principally caused by system closure. Proposed

model LSTM-M manages the missing data from the two categories, in which a long period and a short period mechanism are designed for modeling missing data in the input variables. Masking vector is used to represent the missing flag at time step t. The proposed time-series model for missing data incorporates two temporal prediction scales to obtain the missing data directly from the input values and implicitly in the RNN states. Authors utilized an inuence factor considering a missing observation to represent the weight of the previous observation and an inuence factor to represent the weight of the periodic factor. Authors simulated the residual between the predicted value and the ground truth value in the LSTM unit by introducing a masking vector directly into the model. The proposed model is comprised of 6 LSTM layers and one fully connected (FC) layer. The size of the hidden unit in the LSTM is 32 throughout this paper, and the observation variable (input) has dimension 1. The activation function for the LSTM layer is the tanh function. Adam algorithm is used for optimization because the traffic ow data are noisy, and Adam is appropriate for problems with very noisy and/or sparse gradients. Training is terminated after 10 epochs has been reached. The mini-batch size is set as 32 and the other hyperparameters are optimized via cross-validation. The proposed model can be trained within 3 hrs on a single Titan X GPU. Authors used the mean absolute error (MAE), the mean relative error (MRE), and the root-mean-squared error (RMSE) as the evaluation criteria to calculate the prediction accuracy. Performance of LSTM and the LSTM-M approaches are compared in short-time-interval sequences (5 min intervals) in peak (8.10 am and 17.19 pm) and off-peak times. The MAE of the LSTM model is 16.86, while that of the proposed model LSTM-M model is 14.57 (improved by 2.29). The MRE of the LSTM model is 6.97%, whereas that of the proposed model LSTM-M is 5.12% (improved by 1.85%). And, the RMSE of the LSTM model is 26.24, whereas that of the proposed LSTM-M model is 21.98 (improved by 4.26). The proposed model has a lower error rate in comparison with the LSTM model, which is mainly because the prediction residual is explicitly modeled based on the pattern of the

missing data. Overall, the experimental results on the PeMS data set and the authors own data set show that the proposed approach outperforms several state-of-the- art methods in terms of accuracy.

Authors [5] in this paper proposed a multi-view learning method (MVLM) to impute the missing values for traffic-related time series data. In the proposed method authors considered four views. A) Global variation in temporal view which means in the data, there exists a change regularity over a long time period. LSTM prediction model was used for this view B) Traffic volume fluctuates from non-peak hours to peak hours which is considered as local variation in temporal view. Collaborative Filtering technique was used for this view. C) In the local variation in spatial view, traffic state of a specific sensor is affected by its adjacent sensors traffic state and proportional to the distance. Collaborative Filtering technique was used for this view.D) Data in adjacent locations will be missing in block missing scenarios. In this situation data from the long-distance sensor is considered which is called the global variation in the spatial view. SVR is applied for this view. LSTM is used to capture the global variation in the temporal view. The input of the LSTM is (x1, x2,,xt) which is considered as historical complete traffic data and output is (y1, y2,,yt) which is the estimated missing values. The relationship among all the locations where to obtain traffic flow data is modeled by SVR. In SVR (x1,y1)..(xn,yn), traffic data from the target site is represented by y, x represents the traffic data from all other sites. In the proposed method, block missing values are initialized with GVTV, GVSV. Next missing values estimated from four different views are combined using a linear kernel function based on a multi-view learning algorithm. PeMS dataset is used in this paper for conducting experiments. The data is updated every 5 minutes and collected over one year. First 8 months is used to train the models, and the rest is used to evaluate the models. Stability of the model is tested across different missing ratios 5% to 50%. The proposed method improved the accuracy of the estimated values, especially when the missing ratio is higher than 25%.

3.2 Related works on the traffic prediction during unusual traffic conditions

Online Support vector regression (OL-SVR) model was proposed by the authors [33] for short term traffic prediction under both typical and atypical conditions. In this paper, authors compared the OL-SVR model with three well-known prediction models which are Gaussian maximum likelihood (GML), Holt exponential smoothing and artificial neural network models under two scenarios, 1-typical, and 2-atypical traffic conditions. California Freeway Performance Measurement System (PeMS), version 7.0 data was used in the experiments. This data also includes traffic incident data. In the scenario-1, data related to the special occurrences is not present. Only data with normal traffic behavior is included. Sixteen days of 5-min traffic flow data from 5:00 am to 10:00 am for each of the seven randomly selected freeway locations was collected. The first 15 days of data were used for model training and the 16th day was used for model testing. In the scenario-2, the only difference is that the testing day (16th day) had an unexpected event or was a holiday. For one-step ahead short-term prediction under normal conditions, the GML method is slightly better. OL-SVR method outperformed other methods under atypical conditions at some vehicle detection stations.

Authors [34] tested three machine learning models such as Time Delay and Recurrent Neural Networks and the k-Nearest Neighbour (kNN) algorithms during both normal and incident conditions using three different models with increasing information in explanatory variables. To predict the flow at f(t+h), the first model used four temporal lags, four spatial lags. In addition to the features in the first model, the historical average is used in the second model. The third model used error feedback mechanism in addition to the features in the second model. Each machine learning model used these three models and tested on

both normal and incident conditions. During abnormal conditions, there is a significant advantage of error feedback in the improvement of prediction accuracy. They stated that the information used in the prediction is more important than the machine learning tool used. k-NN method outperformed other models.

In this paper [35], the authors proposed an online boosting non-parametric regression (OBNR) model for traffic flow prediction, which can work effectively under abnormal traffic conditions. This model consists of two parts: the base part and the boosting part. The boosting section adapts the model with abnormal conditions and updates in real time. It is constructed in a gradient boosting way. The base part deals with normal prediction. When the traffic state becomes normal, the boosting part is disabled. Experiments conducted in this paper showed that OBNR is much more effective than traditional online learning models in dealing with abnormal traffic conditions.

Authors [36] implemented k-Nearest Neighbours (k-NN) and Support Vector Regression (SVR) using the model structures in [34] under normal and abnormal conditions. Authors stated that during the normal, non-incident traffic conditions, knn and SVR have similar prediction accuracy. During incidents k-NN method outperformed SVR.

Authors in this paper [37] proposed a method for short term traffic prediction under both typical and atypical conditions. Authors used an automatic incident detection (AID) algorithm based on support vector machines (SVM) for checking atypical events (e.g., traffic accident), and if such an event occurs, a k-NN regression model is used for traffic prediction. Otherwise, the ARIMA model is used for traffic prediction. Due to the presence of atypical conditions (weather conditions, accidents, road maintenance works, etc.), the conventional traffic prediction models fail to predict traffic in real time accurately. They used the Caltrans Performance Measurement System (PeMS) dataset for experiments, and this also includes the incident data. Traffic speed data aggregated in 5-minute intervals was used. For each road of interest, a different SVM- based AID model is built based

on five features. Feature one is at the current time, the difference between the road speed and the average speed of its adjacent roads. Similarly, second, third, fourth features are extracted for three-time intervals before the current one. The fifth feature is the average absolute deviation of the real speed of road at present with the average value of all previous intervals up to the current one. ARIMA (3, 1, 0) used for typical traffic conditions. In k-NN prediction, the state vector is constructed for the road of interest using the speed at the current time interval and previous p time intervals. Similar vectors are created for N other roads of the network. Euclidean distance is used to choose the k nearest neighbors. AID and k-NN models were trained using the total dataset and the ARIMA model with incident free data set.

Authors [38] used the density-based clustering algorithm DBSCAN to identify the traffic patterns under both normal and abnormal conditions. They used different prediction models (k-nearest neighbor, the support vector regression and the ARIMA models) for each cluster that represents a traffic pattern. They performed the prediction by discovering the traffic pattern that is formed when a specific class incident occurs. Abnormal traffic conditions data is also used in training the prediction model. The traffic flow dataset and the incident dataset from the PeMS is used in the experiments. The main focus of this study is to identify the pattern of the traffic flow data when incidents of a specific class occur. They constructed different classes of incidents by applying a set of filters to an incident dataset. 120 time series of size 288 is considered, each time series split into segments where each segment represents 1 hour of traffic flow. These partial time series are transformed into 3 -dimensional feature vectors using Principal Components Analysis (PCA). DBSCAN is used to create clusters of partial time series. For each separate cluster, suitable traffic prediction model which is among k-NN, SVR, ARIMA was constructed. So overall, clusters are created for each segment (total of 24 segments). Each cluster is with a different prediction model. The prediction model is fetched based on the cluster to which the input feature

vector was classified, and the traffic flow is predicted accordingly. In terms of prediction accuracy under normal and abnormal conditions, the proposed model outperforms the typical traffic prediction models from the literature. They also stated that the use of time series segmentation and clustering results in better accuracy.

The k-nearest neighbors (k-NN) is one of the most widely used methods for short-term traffic prediction. Parameters to be tuned for k-NN are k which is the number of nearest neighbors, d which is the search step length used to find the most similar vectors of history days, v which is the window size used to describe the maximum time point shift when searching for neighbors. Authors [39] improved the k-NN prediction accuracy by tuning all parameters at the same time considering dynamic traffic characteristics. For each training time point, predictions are conducted using different values of (k, d, v) and then according to the performance; weights are generated for each configuration (k, d, v). To get the result, selected tuples are used to make a prediction, and a weighted average is calculated. Authors state that it is not guaranteed that the flow situation is the same, if records are separated by time of day or the day of the week, etc. Instead they are separated by flow rate levels. The proposed method adapts the weights of tuples to the flow according to separated flow levels. Here the weights are for tuples, not for neighbors or search steps. The results show that the proposed method performed better than manually tuned k-NN.

The prediction accuracy in this paper [40] is improved by fusing information considering different traffic conditions in a k-nearest neighbor (k-NN) based ensemble method. A day-week decomposition (DWD) method is introduced in addition to conditional information fusion for preprocessing before anomaly detection. Authors in this paper modified the method in this paper [39] to fuse the information according to the flow conditions, i.e., normal vs. abnormal data, by not separating the data for different flow rates. Here the abnormal history time points and normal data points are considered as two different groups. To train parameter tuples, the time points in those two groups are used separately. One

of the two groups of trained parameter tuples weights will be used respectively for the prediction of traffic flow.

Table 3.1: Summary on traffic prediction during normal and unusual traffic conditions

Ref	Paper Title	Normal	Atypical
[33]	Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions	Gaussian maximum likelihood (GML) performed slightly better than Holt exponential smoothing and artificial neural network models	OL-SVR performed better than GML, Holt exponential smoothing and artificial neural network models under non- recurring atypical traffic conditions
[35]	A online boosting approach for traffic flow forecasting under abnormal conditions	The base part in online boosting non-parametric regression (OBNR) model deals with normal traffic prediction	The boosting part in the model deals with abnormal traffic conditions
[36]	Short-term traffic prediction under normal and abnormal traffic conditions on urban roads	During the normal, non-incident traffic conditions, k-NN and SVR have similar prediction accuracy.	During incidents k-NN method outperformed SVR.
[37]	Short-Term Traffic Prediction under Both Typical and Atypical Traffic Conditions using a Pattern Transition Model.	The ARIMA model is used for traffic prediction.	A k-NN regression model is used for traffic prediction.

Table 3.2: Summary on traffic prediction considering various traffic patterns

Ref	Paper Title	Methodology
[38]	Identifying patterns under both normal and abnormal traffic conditions for short-term traffic prediction	Authors used the density-based clustering algorithm DBSCAN to identify the traffic patterns under both normal and abnormal conditions. They used different prediction models (k-nearest neighbor, the support vector regression and the ARIMA models) for each cluster that represents a traffic pattern. They also stated that the use of time series segmentation and clustering results in better accuracy
[39]	Flow-aware WPT k-nearest neighbors regression for short-term traffic prediction	Traffic flow data is separated according to the different flow rate levels in this paper. Authors used a k-NN prediction model by tuning all parameters at the same time considering dynamic traffic characteristics. The results show that the proposed method performed better than manually tuned k-NN as well as benchmark methods such as extreme gradient boosting (XGB) and seasonal autoregressive integrated moving average (SARIMA).
[40]	Anomaly-Aware Traffic Prediction Based on Automated Conditional Information Fusion	Authors used k-nearest neighbors (k-NN) based ensemble method considering different traffic conditions which are normal and abnormal traffic conditions. The prediction accuracy in this paper is improved by fusing information considering different traffic conditions.

Traffic conditions are affected by different factors. Sometimes planned incidents such as road maintenance works/construction works might affect the traffic conditions. And some other times unplanned incidents and accidents may result in abnormal traffic conditions. On the absence of such incidents, the traffic can be considered normal. Most of the studies focus on normal traffic conditions or using a single model for traffic prediction.

From the above literature, we can conclude that the traffic prediction under different traffic conditions requires different techniques or special attention is required when dealing with abnormal or atypical traffic conditions. Most of the studies used incident data while analyzing traffic prediction under atypical traffic conditions. However, if we do not have the incident data for the dataset, we can use the pattern matching techniques or unsupervised clustering to know the various unusual traffic conditions and use this information for prediction.

3.3 Related works on the traffic prediction

In the last few decades, various methods have been developed for traffic prediction in terms of predicting volume, speed, travel time. These all studies aim at improving efficiency, robustness or prediction accuracy. Methods for traffic prediction falls into two major categories: parametric approaches and nonparametric approaches [41]. Parametric approaches are also known as model-based methods. In these approaches, the model structure is predetermined based on certain theoretical assumptions, and the model parameters can be computed with empirical data. In nonparametric approaches, both the model structure and parameters are not fixed. These are data-driven methods.

One of the popular parametric regression models is an Auto-Regressive Integrated Moving Average (ARIMA) model. This model is based on time-series methods, and it is widely used for traffic prediction [42]. This model considers the essential traffic characteristics

and finds the patterns of the temporal evolution formation by moving average (MA) and autoregressive (AR). Many variants of this model such as KARIMA, seasonal ARIMA, ARIMAX were developed to improve the prediction accuracy. This model assumes that the traffic is a stationary process where the mean, variance and auto-correlation are unchanged. Due to the stochastic and nonlinear nature of traffic, parametric models cannot represent the traffic accurately with analysis formulas. Hence, the attention of researchers moved towards the nonparametric approaches [43].

The popular methods in nonparametric models are the k-nearest neighbors (KNN), Support Vector Regression (SVR), Artificial Neural Networks (ANNs), etc. The k-nearest neighbors (k-NN) is one of the most widely used methods for short-term traffic prediction. Authors in this paper [44] adopted k-NN models for speed prediction in an urban traffic network. In this paper, several k-NN models were employed using different sets of features which are the past and current traffic speeds of the target link and the up/down-stream links. All these models are compared by the prediction accuracy and the time taken to make a prediction using different amounts of data. The traffic speed dataset aggregated in five-minute intervals is used for experiments. The state of the traffic shows a cyclic nature over time. By considering this, authors used the data in a time zone which is near to the prediction time. This data is from recent few days or weeks from the prediction time to thirty minutes before and after. Authors stated that the model which uses the neighboring links in addition to the target link shows better performance compared to the model which use only target link information. But the former model has more features and can take more time for a prediction. Here the trade-off should be made between prediction accuracy and time cost.

The Support Vector Regression is another popular approach which is used for traffic prediction problems [45, 46].

The artificial neural networks are another group of methods which are used for traffic prediction, and it receives numerous successes in the domain of transportation. Artifi-

cial Neural Networks (ANNs) are computing systems which are inspired by the biological neural networks. In traffic prediction problems, they approximate a mapping function from input variables to output variables based on the historical data. Deep neural networks (DNNs) are ANNs with multiple layers between the input and output layers, and the set of mathematical operations are used to change the input into the output. In recent years, deep learning-based methods (such as long short-term memory neural networks, gated recurrent neural networks, Convolutional neural networks) have demonstrated its competitiveness to the time series analysis which is an essential part of traffic prediction. These methods can handle the complicated nonlinear spatial, and temporal correlations and the different variants of these methods have been used for traffic speed prediction. Artificial Neural Network (ANN) has the capability of handling multi-dimensional data, flexible model structure, strong generalization, and learning ability as well as adaptability [47]. ANN does not require underlying assumptions regarding data, and is also robust to missing and noisy inputs, unlike the statistical methods [47]. Authors [48] introduced the concept of the neural network into freeway traffic time estimation. Since then many variants of neural network models have been developed for traffic prediction. In this paper [8], authors focused on the analysis of border traffic volume and crossing times at the Ambassador Bridge, which connects Windsor, Ontario in Canada to Detroit, Michigan in the US. They used different Artificial Neural Network (ANN) models. To train the ANN models, a multilayer feedforward neural network with backpropagation approach was used. The number of lags used in experiments was decided based on the correlations between the dependent variable and its sequential time lags. A correlation greater than 0.5 is used as a threshold to limit the number of lags in the model. As such by the conducted correlation analysis, d was determined to be 6. Along with the lags of crossing time, three explanatory variables were used in developing the ANN models for crossing time which are truck volume on the bridge, hours of day and days of the week. These variables were considered since they significantly improved

the accuracy of predictions. For crossing time prediction, the number of time lags was decided like the truck volume models (i.e. based on a correlation test). The different number of lags was used for Canada and US-bound crossing time ANN models, which are the first eight-time lags in the Canada bound case and the first six-time lags in the US-bound case. One of the datasets used in this study is the GPS database. The records in this database summarize the movement of the trucks for a full year (September 2012 to August 2013). A subset of these records was taken to conduct the study which resulted in approximately 11,000 trucks pertaining to 354 carriers that crossed the Ambassador Bridge. Each GPS ping in the database comes with its geographic location (latitude, longitude) along with the truck and its carrier identifiers. The study in this paper also used a total truck volume database that was compiled from a network of RTMS that are operated by the Cross-Border Institute (CBI) at the University of Windsor on the Canadian side of the bridge. The GPS data were used to model the crossing times on the bridge, and the RTMS data were used to model truck volumes on the bridge. To train and validate the ANN models in this paper the Neural Network Toolbox of MATLAB was used. Two ANN models were designed and trained to predict crossing time for traffic crossing the bridge in both directions and two more ANN models were designed and trained to predict truck volumes on the bridge per direction. The prediction accuracy of crossing time for the ANN models are compared with ARIMAX (ARIMA with Exogenous Inputs) and two other structures of ANN models, namely multilayer perceptron network and radial basis function network. Results from the experiments show that the prediction from the multilayer feedforward ANN model has the lowest MAPE among the modeling approaches utilized in this paper.

Recurrent neural network (RNN) is a variant of ANN which can efficiently capture the temporal and spatial features of traffic, due to the dynamic nature of the transportation system. These networks are the traditional ANNs to which temporal component is added means these RNNs can remember the sequence of events. RNNs maintain the memory,

and because of this capability, they can capture the patterns in a sequence of inputs. The Long short-term memory neural networks (LSTM) and the Gated recurrent neural networks (GRU) are the popular RNNs. Many variants of these neural networks have been applied in traffic prediction.

In this paper [49], the authors used stacked long short-term memory model on online open data to learn and predict the patterns of traffic conditions. The proposed model is compared with the multilayer perceptron model, decision tree model and support vector machine model and has shown the superior performance over them from the experimental results. Resources to get online open data are the local events, official websites related to traffic management and operations, weather forecasting websites, Google maps, etc. Authors in this paper also discussed the existing online open source services that offer traffic-related information and the methods to collect data from those resources. The dataset used in this paper is aggregated in the five-minute interval. Performance of the proposed method is evaluated on three performance indices which are precision, recall, and F- measure. In the LSTM layer, the number of output units is chosen from 3, 6, 9, 12, and 36. The learning rate is chosen from 0.1 to 1.2 with a step 0.1. Once the grid search is performed, three layers of LSTM with output units size of each layer is 6, 6, and 6, is selected as the best architecture.

Authors [50] applied LSTM on traffic speed data which was collected from traffic microwave detectors in Beijing to predict traffic speed. The feasibility of LSTM NN for short-term traffic speed prediction is examined by comparing with other AI methods such as three RNN models (Elman NN, Time delay NN and NARX NN), Support Vector Regression, ARIMA, and Kalman Filter approach. All these models aim to predict speed in the next 2 min based on speed and volume in the previous period on the same day. From two separated locations in a major ring road around Beijing, speed data were collected. The data with the updating frequency of 2 min were collected from Jun. 1, 2013 to Jun.

30, 2013. The collected data includes volume, occupancy, and speed. The missing and erroneous records were removed using temporally adjacent records. First, 25 days data is used for training and the remaining five days data was used for testing. To reduce randomness, each algorithm was executed for ten times. All RNN models were trained using the LevenbergMarquardt method and they have the same structure that is one input layer, one hidden layer, one output layer and ten hidden neurons in the hidden layer. Radial Basis Function (RBF) was used in SVM and parameters in this method were calculated using 5-fold cross validation for a fair comparison with neural network-based algorithms. ARIMA models parameters were determined based on the best Akaike Information Criterion (AIC) value. The noise was considered Gaussian for Kalman Filter approach. No predetermined time window size is used for LSTM. The Mean Absolute Percentage Errors (MAPE) and Mean Squared Errors (MSE) are used to evaluate the effectiveness of different travel speed algorithms. Authors stated that LSTM NN is an effective approach for short-term travel speed prediction without prior information of time lag.

The length of the historical input data is predefined and static in most of the existing models and algorithms based on time series prediction and machine learning. Optimal time lags cannot be determined automatically. Authors [51] proposed LSTM RNN method to predict short term traffic flow which takes advantage of determining optimum time lags dynamically and can capture the nonlinearity and randomness of traffic flow more effectively. The Caltrans Performance Measurement System (PeMS) dataset is used in this paper. LSTM RNN is compared with random walk (RW), support vector machine (SVM), single layer feed forward neural network (FFNN) and stacked autoencoder (SAE). The results show that higher accuracy is achieved by the proposed prediction model. Four key hyperparameters must be determined(size of the input layer, the number of hidden layers, the number of hidden units(memory blocks for LSTM RNN) in each hidden layer and the size of output layer) to build the model based on LSTM RNN. Size of the input layer is equal to the input

historical data length, which is defined from 1 to 12 in the proposed model. The number of hidden layers is assigned to 1, and the number of units in it is assigned in the range from 5 to 40 with a step of 5. To indicate the traffic flow of the next time step, the size of the output layer is 1. To obtain the optimal parameters, Grid search method is used. This paper only focuses on the traffic flow prediction on workdays which is 249 work days in 2014. First 200 workdays are used for training and remaining 49 days for the test set. Only traffic flow data was used as input for experiments in this paper and 15-min, 30-min, 45-min, and 60-min prediction intervals are considered. Three aspects (the prediction accuracy, the memory ability of long historical data and the generalization capability with different prediction intervals) of the LSTM RNN are tested through the experiments. Two commonly used metrics, i.e., Mean Absolute Percentage Error (MAPE) which evaluates the relative error and Root Mean Square Error (RMSE) which assesses the absolute error, are used to evaluate prediction accuracy. Both MAPE and RMSE of LSTM RNN are lowest compared to the other four models. The two metrics of LSTM RNN and SVM are close, but the input size of SVM and LSTM are 8 and 1 respectively which states that SVM is more complicated than LSTM and shows that LSTM can memorize long historical data. With the increase of hidden units, MAPE and RMSE fall and remain stable or slightly up after a certain number, in this, its after 20 hidden units. From this experiment, it can be concluded that the prediction performance is related to the complexity of the model. Each experiment is conducted 3 times due to the random initialization of the model. LSTM RNN can also achieve pretty good traffic flow prediction results even when the input data length is 1. From this, it can be inferred that through the recurrently connected memory blocks, the model can memorize the earlier inputs. Therefore, LSTM can determine the optimal time lags dynamically and capture the long-term dependencies, which leads to the desired results of short-term traffic flow prediction. The error rates of LSTM RNN are all lowest among the four-machine learning based models with different prediction intervals. This

demonstrates the excellent generalization capability of LSTM RNN.

In this paper [52] authors used LSTM to predict short term traffic flow and analyzed the effects of various input settings on the LSTM prediction performances. Traffic speed, traffic flow and occupancy at the same detector station are used as inputs to predict traffic flow. The results show that the overall performance of the model may enhance if occupancy/speed information is included. In improving prediction accuracy both downstream and upstream traffic information is included. Adam optimizer with adaptive learning rates is applied for backpropagation through time (BPTT) to minimize training error. The normalization method is used to preprocess the traffic flow data and dropout methods for LSTM to reduce overfitting and apply weighted L1 and L2 regularization methods. In general, with a higher number of layers, the LSTM can have a strong learning ability, but it is also easier to be overfitting. Two LSTM layers and a dense layer is used in this paper to capture the characteristics of traffic flow dynamics and achieve satisfactory results. Three performance indexes, which are the root mean square error (RMSE), mean absolute error (MAE), and the mean absolute percentage error (MAPE) are used to evaluate the effectiveness of the model. The California Performance Measurement System (PeMS) traffic flow dataset is used and the data is aggregated to 5 min intervals.

Long Short-Term Memory Networks (LSTMs) is applied for short-term traffic flow prediction in this paper [53]. The significant feature of capturing long-term dependencies in sequential data makes it a suitable choice in traffic prediction. The author proposed an encoder-decoder model based on LSTM blocks. The encoder converts the sequence of traffic flow x = (x1, x2, ..., xT) into a fixed length vector c. This fixed length vector works as a high-level representation of the input sequence. The decoder is trained to predict the traffic flow at the next timestep, and here the decoder adopted is linear regression. The data collected by Caltrans Performance Measurement System (PeMS) is used. One-month data (from August 1st to August 31th 2014) is used and only weekday traffic flow data is

analyzed. Two commonly used metrics are used, mean absolute percentage error (MAPE) and root mean square error (RMSE). The proposed model is compared with a random walk (RW), support vector regression (SVR), wavelet neural network (WNN), and the stacked autoencoder (SAE). In the proposed model, a one-layer LSTM with 32 neurons in the hidden layer, the input length six is used and obtained lowest in both MAPE and RMSE compared to the other four models. The grid search method is used to select the optimal hyperparameters that will give the best result and carried out some experiments on the effect of different hyperparameters on MAPE and RMSE metrics. The experiments shows that six timesteps is an optimal value to the model. Timesteps are the input size of the model, and they determine the number of LSTM blocks in each level. Both RMSE and MAPE are increased when more LSTM layers are added to the model. One possible reason might be due to the moderate size of the used data set, and overfitting occurs easily when we try to increase the number of layers. Thus, the optimal layer here would be one-layer LSTM. As one of the deep learning approaches, LSTM can discover the latent feature representations hidden in the traffic flow.

In this paper [41], Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) neural network (NN) methods are used to predict short-term traffic flow, and experiments show that these methods perform better than autoregressive integrated moving average (ARIMA) model. In this study first time, the GRU is applied to traffic flow prediction. Adam optimizer with adaptive learning rates is used for LSTM optimization. Performance of ARIMA, LSTM NN and GRU NN model has tested on the Caltrans Performance Management System (PeMS) dataset and found that GRU NNs perform a little better than LSTM NNs and usually converge faster than LSTM. A linear regression layer is applied on the output layer of the LSTM cell, in traffic flow prediction model. 5-minute aggregated traffic data is used in experiments. As the missing data points only occupy a small part of the whole dataset, they are imputed using a historical average value. The traffic flow of

the past 30 minutes (time sequence of 6 data points) is used to predict the coming traffic flow in the next 5 minutes. The first three weeks are used as the training dataset, and data of the last week is used to test the model prediction accuracy. Experiments are conducted on 50 traffic sensors data, and unique model is created for each sensor. Both mean square error (MSE) and mean absolute percentage error (MAPE) are used to test model prediction accuracy.

In this paper [54] authors proposed a novel end-to-end recurrent neural network architecture based on LSTM to predict the completed trips during special events at UBER. The proposed method achieved superior performance compared to their baseline as it leverages autoencoder for feature extraction. Authors stated that de-trending the data, as opposed to de-seasoning, produced better results. In this paper Bootstrap and Bayesian approaches are combined for uncertainty estimation. In the model, the features vectors after the auto feature extraction are aggregated via an ensemble technique. The new input is concatenated with the final vector and fed to LSTM forecaster for prediction. This approach achieved an average 14.09% improvement over the multilayer LSTM model trained over a set of raw inputs. Experiments in this paper show that having a separate auto-encoder module produced better results. SMAPE was used for calculating forecast error. The five years of completed trips data across top US cities were used to provide forecasts across all major US holidays. According to this paper, there are three criteria for selecting a neural network model for time-series: (a) length of time-series (b) number of time series and (c) correlation among the time-series. The neural network can be a excellent choice if all the three are high; otherwise classical time series approach may work best.

Authors in this paper [55] proposed a deep stacked bidirectional and unidirectional LSTM to predict network-wide traffic speed. The spatial features and the bidirectional temporal features from historical data are captured by the bidirectional LSTM layer. The missing values in input data are handled by a masking mechanism in the proposed method.

The two hidden layers are connected to the same output layer in bidirectional LSTMs (BDLSTMs). The unfolded structure of BDLSTM contains a forward LSTM layer and a backward LSTM layer. The forward layer output sequence is calculated using inputs from T-n to T-1 and using the reversed inputs, a backward layer output sequence is calculated. Both the layer outputs are calculated using the LSTM updating equations. In the masking mechanism, a pre-defined value is set to all missing values which is zero or null. In the input time series data Xt, if xt is missing, then the training process at t-th step will be skipped and (t-1) th step cell state will be the input to the (t+1)th step. The proposed model can predict multiple future time steps, but this paper focus is to predict for one future time step. The dataset used in this paper is aggregated in 5-minute intervals. The input to the proposed model is a 3-D vector [N,n,P] where N is the total number of samples, n is the number of time lags, P is the total number of locations. The mean squared error loss function and the RMSProp optimizer is used in the proposed method. The proposed model is compared with SVR, random forest, feed-forward NN, GRU NN. The random forest showed better performance when compared to SVM among the non-neural network algorithms. Overall, the proposed method is better than the other four methods. The proposed model (the combination of BDLSTM and LSTM layers) is compared with the pure deep (N- layers) BDLSTMs and LSTMs, a deep LSTM NN adding a fully connected deep neural network layer. All the compared models achieved the best performance when they have two layers. The proposed model outperformed the others for all the layer numbers. The proposed model achieved the best performance when it has no middle layer. Authors also stated that the order of spatial dimension of input data basically does not affect the model performance and the volume and occupancy have a slight influence on the traffic speed prediction based on the experiment results. The proposed method can deal with multiple types of the traffic network and works even when the size of traffic network changes.

Some researchers applied the convolutional neural networks to time series forecasting

problems and have shown successful results. This paper [56] proposed a method for conditional time series forecasting based on the deep convolutional model. The proposed model contains layers of dilated convolutions which allow accessing a broad range of data when prediction and a ReLU activation function. The conditioning is performed in the proposed method by applying multiple convolutional filters in parallel to separate time series. This technique allows for the fast processing of data and the exploitation of the correlation structure between the multivariate time series. This paper shows that a convolutional network is well-suited for regression-type problems and dependencies in and between the series can be captured efficiently, and it can be alternative to recurrent-type networks. And also, this paper analyzes the performance of the convolutional network both unconditionally as well as conditionally for financial time series forecasting. The proposed method makes use of the dilated convolutions, which are applied with parametrized skip connections from both the input time series as well as the time series on which the method condition on. This way long-term and short-term interdependencies can be learned in an efficient manner. The gated activation function used is a rectified linear unit (ReLU). Authors stated that the WaveNet model is a time-efficient and easy to implement an alternative to recurrent-type networks and outperform the linear and recurrent models. Data sets used are artificial time series as well as the SP500, VIX, CBOE interest rate, and five exchange rates.

Both CNN and LSTM have been successfully applied to time series prediction and have shown excellent results. Motivated by these neural networks, some studies used the combination of CNN and LSTM for traffic prediction problems. Also, these neural networks can handle the multi-dimensional data efficiently; hence they can be used for network wide traffic prediction.

Authors in this paper [57] proposed a novel traffic flow prediction method based on deep learning framework. In the proposed method deep convolutional neural networks were utilized to mine the spatial features of traffic flow data, and recurrent neural networks were

employed to learn temporal features. The proposed model in this paper was evaluated on the prediction of future traffic flow on a long future horizon (45 min, traffic flow is recorded every 5 min). The attention model is used in this study. Each element in the attention matrix can be interpreted as the importance of this space-time point for future forecasting. The attention matrix is point-wise multiplied with the traffic flow matrix to obtain a weighted traffic flow matrix for deep learning procedures. CNN structure is used for mining the spatial features within traffic flow data, and the conventional 1D CNN is exploited to capture the spatial features. In the proposed model pooling layers are not used. The spatial features of traffic flow are signicantly different from its temporal features. Due to the dynamic nature of the transportation system, traffic flow exhibit stronger correlation in a short time period and the long-term temporal dependency also exists within traffic flow data. The long short-term memory (LSTM) network model is used to capture the temporal features of traffic flow. It uses gated neurons to capture both the short-term and the long-term memories within traffic flow and to avoid the gradient vanishing/exploding problem. To mine the spatial-temporal features of inputs of daily periodicity and weekly periodicity, the same CNN and GRU structures on near-term data are used. All the features from the above steps are concatenated together and input into a regression layer to perform forecasting future traffic flow in (t,t+1,...., t+h). To evaluate the performance of the proposed DNN-BTF model, the traffic flow data from PeMS was used in the conducted experiments. The proposed model is compared with several state-of-the-art forecasting methods with deep architectures. Traffic volumes data are aggregated into 5 min interval; one detector preserves 288 data points per day. The time window size n of traffic flow matrix is set as 21, and the prediction horizon h is set as 9, which means that 105 min historical data are used to perform the traffic flow forecasting of the next 45 min. The number of training samples in the experiments is 100,000, and all the prediction models are simultaneously trained to forecast the traffic flow at those 33 detector locations. All the experiments on

neural network approaches are trained by Adamax optimizer with a batch size of 300. 10% of training data is used as a validation set to avoid overtting. Three 1D convolutional layers with SReLU activation were used to extract the spatial features and the number of feature maps of each layer of the CNN is fixed as 30 to balance the computational cost and prediction accuracy. After the experiments, it is found that the CNN's with the filter lengths 4, 3, and 2 for the 1st, the 2nd, and the 3rd layers achieve the lowest error rates. Stacked GRUs with 2 layers were used to extract the temporal features and the dimensions of hidden states of all GRUs were set as 50. To determine the importance of the near-term inputs of traffic flow, A single layer neural network with 600 hidden neurons using RelU activation was used as the attention model. Parameters of the compared methods were chosen according to the results in the literature and compared with the experimental results of the proposed DNN-BTF model. For the BPNN method, a single layer neural network with 1900 hidden neurons and RelU activation is used. A three-depth deep neural network are used for SAE. The experimental results confirms that the proposed approach can learn compact spatial-temporal features within traffic flow.

Authors in this paper [2] proposed a convolutional neural network (CNN)- based method to predict large scale, network-wide traffic speed. For the network-wide traffic prediction, traffic information in time and space dimensions should be considered jointly. In the time-space matrix, x-axis represents time and y-axis represent space. The input images to the CNN have only one channel which is the traffic speeds of all roads in the network and the values of each pixel range from zero to maximum traffic speed. Input data is normalized. Outputs of the model are the predicted traffic speeds on all road sections of the network. Authors stated that in the context of transportation, features extracted by the convolutional and pooling layers represent relations among road sections. First, a two-dimensional time-space matrix is constructed, and a CNN is applied on this matrix. Spatiotemporal traffic features are extracted by the convolutional and pooling layers and these extracted features

are transformed into outputs through a final fully connected layer. To evaluate the effectiveness of the proposed method, two real-world transportation networks were used in this paper. These datasets are the second ring road and northeast transportation network in Beijing. The proposed method outperformed the algorithms k-nearest neighbors, ordinary least squares, random forest, artificial neural network, and three deep learning methods, namely recurrent neural network, stacked autoencoder and long-short-term memory network, by an improvement of 42.91% average accuracy within an acceptable execution time. The authors stated that CNN could train the model in a reasonable time and, thus, is suitable for large-scale transportation networks.

In many fields such as image and textual data analysis, traditional convolution neural network has been used. These methods can capture spatial features from adjacent pixels/grids of a tensor. Adjacent road segments can impact the speed of a road. For learning spatial correlations topology must be embedded into convolution with road network constraints, which can be achieved by the look-up operations. Thus, authors in this paper [58] designed a look-up convolution layer that embeds the topology of road network into convolution to capture more meaning spatial features and further used the LSTM (long short-term memory neural network) model to learn the long-term temporal patterns that can reference surrounding area traffic dynamics on top of the look-up convolution. To learn more meaningful time-series patterns that can adapt to the traffic dynamics of surrounding areas the proposed model takes advantage of both Recurrent Neural Networks (RNN) and Convolution Neural Networks (CNN) models by a rational integration of them. A network embedded convolution structure is proposed to capture topology aware features, since traffic evolution is restricted by the underlying road network. The other information including periodicity and context factors are also used in addition to the spatial-temporal trend. Current speed on the certain road will be almost the same as days/weeks ago which is called periodicity. Fully-connected (FC) layers are fed with speed vectors of correspond-

ing time intervals several days/weeks ago. Context factors such as weather, holiday and so on are extracted by another two-layer FC neural network. Finally, all the extracted features are fused to predict traffic speed. The proposed model is compared with the methods SVR: Support Vector Regression, H-ARIMA is a method combined of ARIMA and HA to predict future values , Stacked Auto-Encoders (SAE) is a deep learning model to learn generic traffic features and predict future values, LSTM, Graph Convolution (GC) in which pooling and fully-connected is used to forecast future speed, Deep Convolution Neural Network (DCNN) with convolution, pooling and fully connected is used for speed prediction, Spatio-Temporal Residual Network (ST-ResNet) uses a residual network to model three temporal properties to do prediction. Since the correlation of traffic speed between time intervals being predicted and the current moment decreases, performance becomes worse with the quantity increases. Two datasets were used in the experiments. On the Beijing dataset, from the experimental results, LC-RNN achieved the best performance compared with the state-of-the-art. On Shanghai dataset, the result about varying the size of the time interval and the number of predicted intervals is also similar with one on the Beijing.

Authors [59] proposed a Traffic Graph Convolutional Long Short-Term Memory Neural Network (TGC-LSTM) method in which the traffic network is represented as a graph. The proposed method can learn the interactions between roadways and forecast the network-wide traffic speed. The traffic graph convolution in this paper is based on the physical network topology. Authors also discussed the relationship between traffic graph convolution and the spectral graph convolution. Authors applied two regularization methods which are L1- norms on traffic graph convolution weights and L2- norms on traffic graph convolution features. These regularization terms help the proposed model to be more stable and interpretable. The relationship between sensor locations and the traffic network is represented by an undirected graph G where G= (V, E, A). Here V represents vertices, E represents edges, A is an adjacency matrix. Total nodes in the network are N, each element

in adjacency matrix Ai,j = 1 if there is a link connecting node i and node j, otherwise Ai,j = 0. The two real-world network-wide traffic speed datasets are used in this study. Due to the spatiotemporal dependencies and the high dimension features in the datasets, the non neural-network methods are less appropriate. Authors compared the proposed method with ARIMA, SVR, Feedforward neural network and LSTM network, SGC+LSTM (spectral graph convolution and LSTM), LSGC+LSTM (localized spectral graph convolution and LSTM). The proposed method outperformed all these baseline methods.

Table 3.3: Summary on methods for network-wide traffic prediction

Ref	ARIMA, SVR, Kalman filter	Neural Networks (LSTM, CNN)
[59]	Non- neural methods such as ARIMA, SVR are less appropriate for network-wide prediction task	Neural networks are more appropriate for network-wide traffic prediction
[55]	Most of classical methods are not suitable for network-wide traffic speed prediction through a single model, they cannot process 3-D spatial temporal data	Neural network models can handle multi-dimensional data efficiently

Table 3.4: Summary on methods using CNN for traffic prediction

Ref	CNN
[2]	Features extracted by the convolutional and pooling layers represent relations among road sections. Spatiotemporal traffic features are extracted by the convolutional and pooling layers.
[57]	In the proposed method deep convolutional neural networks were utilized to mine the spatial features of traffic flow data, and recurrent neural networks were employed to learn temporal features.
[58]	Look-up convolution layer embeds the topology of road network into convolution to capture more meaning spatial features and further used the LSTM (long short-term memory neural network) model to learn the long-term temporal patterns that can reference surrounding area traffic dynamics on top of the look-up convolution.
[59]	Authors proposed a Traffic Graph Convolutional Long Short-Term Memory Neural Network (TGC-LSTM) method in which the traffic network is represented as a graph. The proposed method can learn the interactions between roadways and forecast the network-wide traffic speed. The traffic graph convolution in this paper is based on the physical network topology.

Overall, from the above literature we can conclude that CNNs can capture the spatial features of traffic efficiently. CNNs can capture the spatiotemporal features of network traffic with a high prediction accuracy and also CNNs could train the model in a reasonable time and, thus, is suitable for large-scale transportation networks [2].

Through various experiments, authors in this paper [60] showed that the subsequence

time series (STS) clustering is meaningless. Here the meaningless word means that the output of STS clustering is independent of the input. They also showed that many algorithms which use STS clustering produce results which are consistent with random clusters. In this work, subsequences are extracted from the time series using a sliding window approach. Authors demonstrated that meaninglessness is due to the way that data is obtained by sliding windows. Let A and B be the two sets of cluster centers. Each cluster center in A mapped on to its closest counterpart in B, and Euclidean distance is calculated between them. The sum of all such distances shows the similarity between the two sets of clusters. Authors used this similarity measure to compare two sets of clusters formed from the same dataset and two sets of clusters derived from two different datasets. Authors used k-means and hierarchical clustering on two different datasets (stock market, random walk). They defined the clustering meaningfulness index value as the fraction where the numerator is the similarity between two sets of clusters formed from the same dataset, and the denominator is the similarity between two sets of clusters derived from two different datasets. The numerator should be close to zero if the clustering algorithm returns similar sets of clusters for different initial seeds. The denominator should be large as two sets of clusters are from two different unrelated datasets. Overall, the clustering meaningfulness index value should be close to zero. Authors performed the previously mentioned steps on the same data, but subsequences were randomly extracted instead of the sliding window approach, which is referred to as the whole clustering. They conducted various experiments with the different number of clusters and different sliding window lengths. The results show that cluster centers on one dataset are not significantly more like each other than they are to cluster centers taken from another different dataset. Basically, there is no significant difference between the results. Authors also performed the same experiment using two different time series datasets, which are different from the datasets as mentioned above. Again, there is no significant difference between the results. Here the interesting point to be noted

that if we cannot find the difference between clusters from two extremely different time series, then how one could discover meaningful clusters in any data. Authors performed experiments with different datasets, different clustering algorithms and different distance metrics. Overall, the results show that the sliding window time series clustering is never meaningful. Also, with STS clustering, sine waves appear as cluster centers regardless of the dataset used, clustering algorithm and the number of clusters. Here the interesting point to be noted is that for every dataset if sine waves appear as cluster centers then it will be impossible to differentiate one dataset clusters from another.

Chapter 4

METHODOLOGY

This chapter presents a detailed methodology to achieve the objective of this research. The proposed method starts by imputing the missing values using the multi-view approach. In this approach local temporal and local spatial views are computed using the collaborative filtering techniques [5] and simple average methods. Global temporal and global spatial views are calculated using the historical average data. Values derived from these views are combined to get the final missing value. We use this framework to impute all the missing values. The next task is to build the network-wide traffic speed prediction model based on CNN.

4.1 Data Processing

This thesis utilizes the GEOTAB dataset that was provided to the Cross-Border Institute (CBI), University of Windsor from Geotab Inc. This organization specializes in the area of global positioning system (GPS) fleet management and vehicle tracking, also known as telematics industry [61] and provide open platform fleet management solutions to businesses. It is headquartered in Oakville, Ontario, Canada. They collect rich, accurate real-time data using the vehicle tracking devices based on advanced GPS technology. The information collected by these devices include the vehicle location, time, speed, etc. They process such information which includes using tools to cleanup data, validate the data, etc. and finally provide clean and compiled data with complete information. The data provided to the CBI has the traffic speed records for 72 locations across the 401 highway. This highway is in the Canadian province of Ontario. It extends from Windsor in the west to the

Ontario-Quebec fringe in the east. The postal speed of highway 401 is 100 km/h throughout its length, with the only exceptions in Windsor and in most construction zones where the postal speed is 80 km/h [3]. These 72 locations in the dataset are between Puslinch and Windsor. For the first 55 locations, traffic speed data collected is for every 5 km and for the remaining locations, it is 1 km. This data is collected for every 15 minutes from 2017 November to 2018 October. In Figure 4.1, we can observe the 72 locations across highway 401.

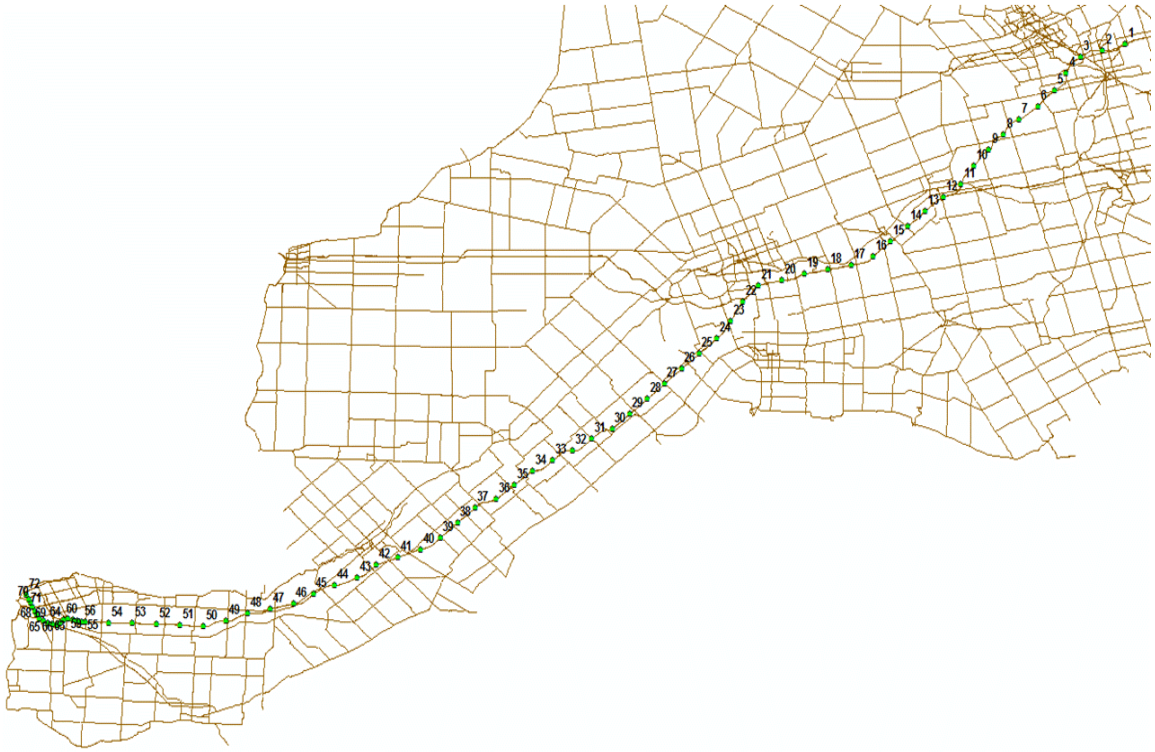

Figure 4.1: 72 Locations across the 401 Highway

In our experiments, we use the data from locations 12 to location 21 because these locations have the maximum percentage of available data compared to other locations as it can be seen in Table 4.1 and Figure 4.2.

The information shown in Table 4.2 represents the format of original raw GEOTAB data containing records that correspond to traffic speed at particular locations and timestamps. Each record in raw GEOTAB data consists of the following fields:

Table 4.1: Percentage of missing values in each location

Location ID	Missing %	Location ID	Missing %	Location ID	Missing %
1	23	25	63	49	64
2	24	26	63	50	64
3	36	27	63	51	64
4	35	28	63	52	64
5	36	29	63	53	65
6	36	30	63	54	65
7	37	31	66	55	67
8	37	32	63	56	67
9	37	33	64	57	67
10	37	34	63	58	67
11	40	35	64	59	75
12	34	36	64	60	82
13	33	37	64	61	80
14	33	38	64	62	80
15	34	39	64	63	77
16	34	40	64	64	79
17	34	41	66	65	76
18	35	42	65	66	78
19	34	43	66	67	77
20	36	44	65	68	76
21	39	45	65	69	70
22	62	46	62	70	70
23	61	47	63	71	71
24	66	48	62	72	73

- Geohash: It is a short alphanumeric string to indicate a location
- DateTime: This field represents the timestamp at which traffic speed was recorded
- Direction: This field represents the direction in which all vehicles are going
- Average Speed: This field shows the average speed recorded in a location during a timestamp in this record
- StdDev Speed: It represents the standard deviation of the speed
- Vehicle Count: Total number of vehicles passing through the location in this record

This raw data is processed using the SQL scripts. In the raw data, speed records for some

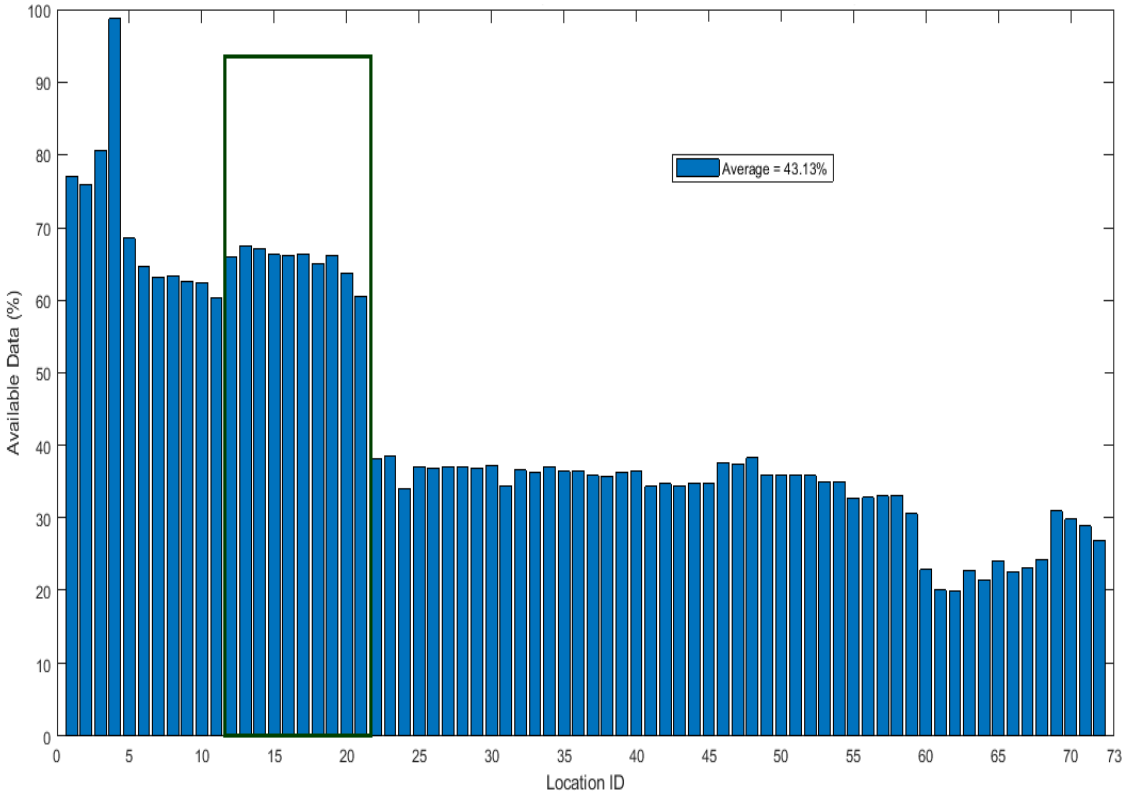

Figure 4.2: More percentage of available data in locations 12 to 21

of the timestamps are unavailable. We process this data to add missing timestamps. Also, in the raw data we observed that there is more than one record for the same timestamp with different directions. The processed data contains only the records which show the direction towards Windsor and we represent the 72 locations by 1 to 72 integers. The fields in the data are location id, year, month, day, hour, time quarter, weekday, and speed. Total number of records in the processed data are 2522880 (72 locations x 365 days x 24 hours x 4 Time Quarters). Figure 4.3, depicts locations 12 to 21 on highway 401.

Table 4.2: Sample records from raw Geotab data

Geohash	DateTime	Direction	AvgSpeed	StdDevSpeed	VehicleCount
dpt07cr	2018-07-12 20:30:06.783 UTC	W	109.08	6.27	8
dpt07cr	2018-06-04 11:00:06.783 UTC	W	110.07	6.21	8
dpt07cr	2018-06-07 17:15:06.783 UTC	W	106.37	5.44	8
dpt07cr	2018-08-14 00:30:06.783 UTC	W	102.24	3.29	8
dpt07cr	2018-10-25 14:45:06.783 UTC	W	106.59	5.96	9
dpt07cr	2018-09-17 17:00:06.783 UTC	W	104.21	3.77	9
dpt07cr	2018-10-10 13:15:06.783 UTC	W	104.97	8.85	9
dpt07cr	2018-10-17 21:30:06.783 UTC	W	105.89	8.7	9
dpt07cr	2018-08-23 10:15:06.783 UTC	W	106.01	9.32	10
dpt07cr	2018-07-19 11:45:06.783 UTC	W	108.46	7.62	10
dpt07cr	2018-05-03 11:30:06.783 UTC	W	105.88	3.21	10
dpt07cr	2018-10-15 14:15:06.783 UTC	W	104.62	6.76	10

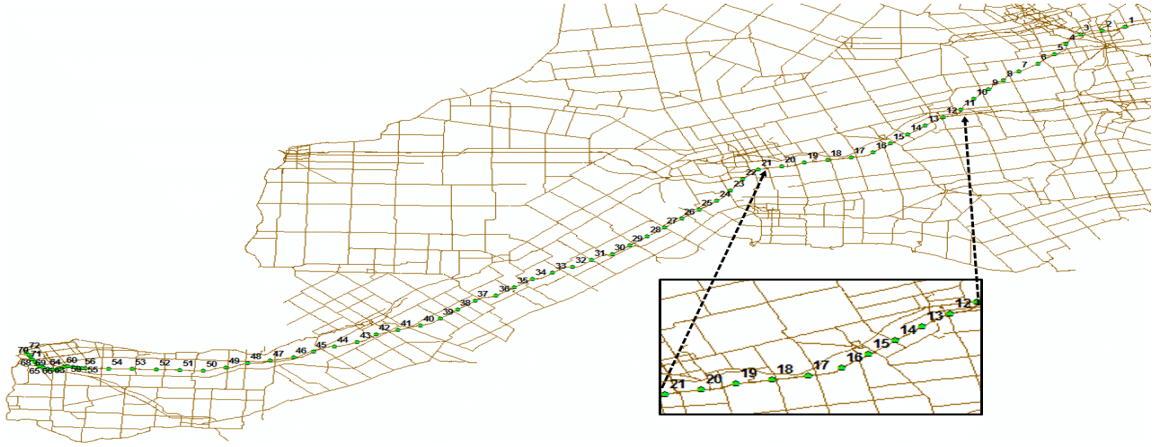

Figure 4.3: Locations 12 to 21 across highway 401

The data shown in Table 4.3 represents the sample processed data. Once the data is processed, we build the historical average table which contains average traffic speed for each location, hour, time quarter and weekday. Total number of records in the historical average table are 48384 (72 Locations x 24 Hours x 4 Time Quarters x 7 Weekdays). The data shown in Table 4.4 represents the sample historical average data.

From locations 12 to 21, October 2018 has more percentage of available data compared to other months. The Figure 4.4 shows the percentage of missing values in each month.

Table 4.3: Sample of processed GEOTAB data

Location ID	Year	Month	Day	Hour	TimeQuarter	Weekday	Speed
13	2018	9	18	4	1	3	NULL
13	2018	9	18	4	2	3	NULL
13	2018	9	18	4	3	3	104.45
13	2018	9	18	4	4	3	102.86
13	2018	9	18	5	1	3	102.45
13	2018	9	18	5	2	3	102.54
13	2018	9	18	5	3	3	104.79
13	2018	9	18	5	4	3	102.22
13	2018	9	18	6	1	3	107.85
13	2018	9	18	6	2	3	101.64
13	2018	9	18	6	3	3	103.32
13	2018	9	18	6	4	3	NULL
13	2018	9	18	7	1	3	103.63
13	2018	9	18	7	1	2	NULL

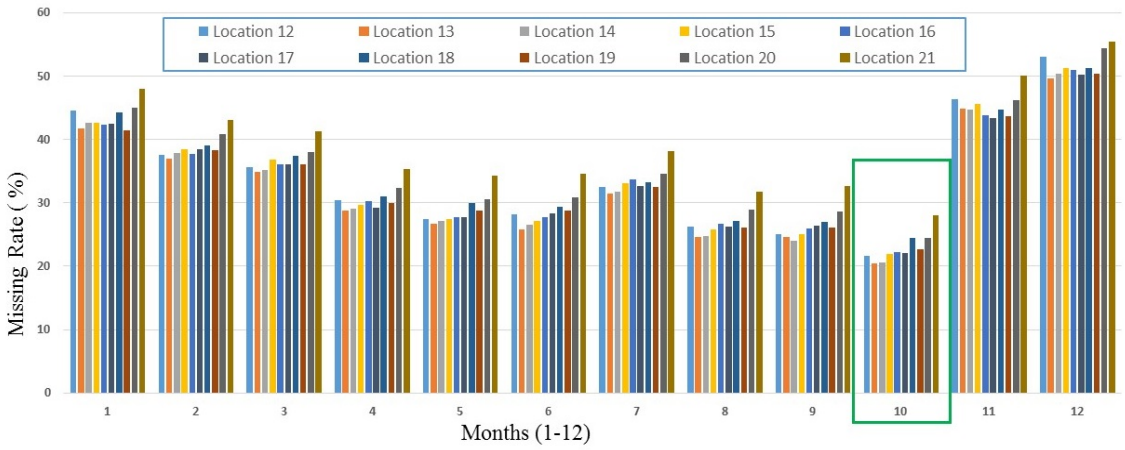

Figure 4.4: Less percentage of missing values in October

Also in our dataset, we have more percentage of available data during the day time when compared to night time. Figure 4.5 shows the percentage of missing data in each hour.

Table 4.4: Sample of historical average data

Location	Hour	TimeQuarter	Weekday	AverageSpeed
17	4	1	3	100.21
17	4	2	3	102.35
17	4	3	3	101.05
17	4	4	3	102.05
17	5	1	3	102.45
17	5	2	3	102.02
17	5	3	3	101.28
17	5	4	3	102.20
17	6	1	3	103.85
17	6	2	3	101.64
17	6	3	3	102.32
17	6	4	3	101.06
17	7	1	3	103.63
17	7	2	3	101.02

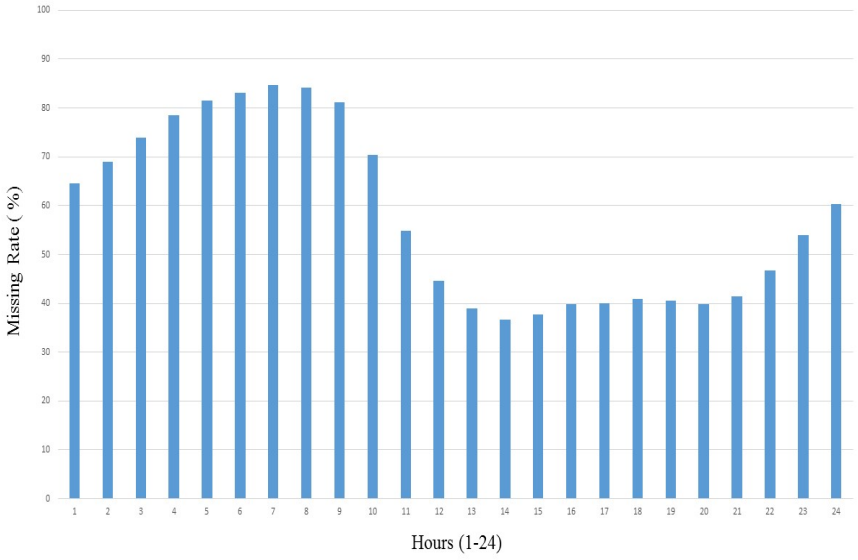

Figure 4.5: Percentage of missing values in each hour

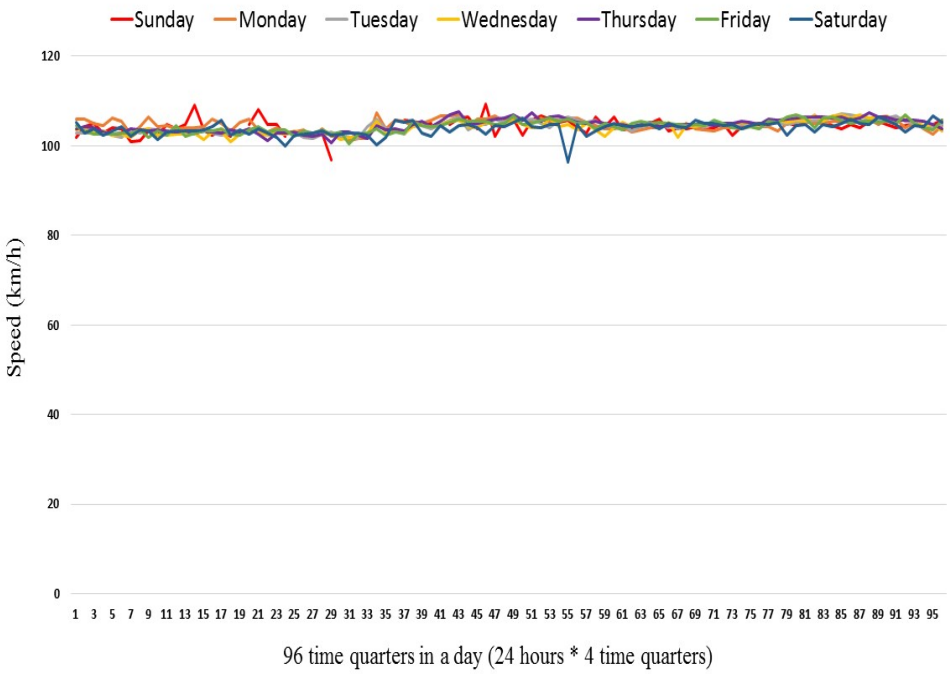

Figure 4.6: Historical Average data for location 15

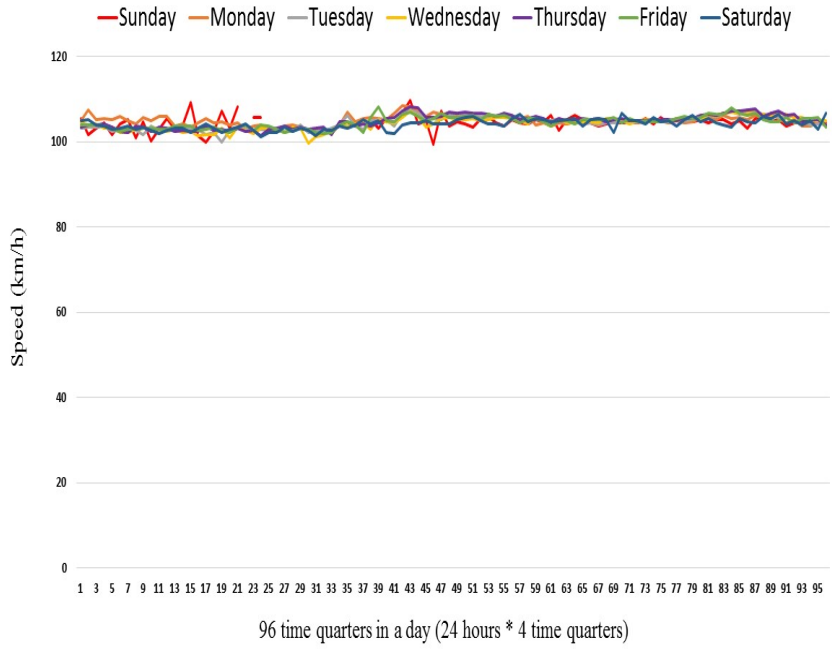

Figure 4.7: Historical Average data for location 19

As shown in Figures 4.6 and 4.7, the historical average values are always around 100 km/h in most of the locations. Figures 4.6 and 4.7 represent the historical average traffic speed data on all weekdays from locations 15 and 19. Within the time quarters in these figures, we can observe a similar variance for all weekdays.

4.1.1 Speed profile extraction

A time series is a sequence of values of the same variable taken at equal intervals in time. In this study, a time series is a sequence of traffic speed values collected once every 15 minutes. Time series analysis contains methods for extracting meaningful information from the time series data. Time series forecasting comes under time series analysis in which we use a model to forecast future values based on the past values. In time series, the lag function shifts back a time series to certain values. The number of lag values represent the number of past values which we use to predict the future values. In most of the traffic speed prediction models, considering an optimal number of lags is very important because the prediction error is based on this parameter. In this study, the different number of lag durations are considered ranging from 45 min to 2.5 hrs. For example, if the lag duration is two and half hours, we generate the subsequences of length eleven so that we can predict the traffic speed in the next 15-minutes using the last 150 minutes.

Subsequences are generated using the sliding window approach. Suppose there is a time series data X of length l, where X = x_1,x_2,..., x_l, a sliding window with length w moving through the entire time series data, X. In every iteration, the sliding window moves q steps and leaves a segment of length w behind. Here, moving q steps means skipping q data points in the time sequence. By using this sliding window approach, time series can be separated into N segments. Suppose if the time series is of length (l) = 20, sliding window (w) = 6 and stride value(q) = 2 then the total number of subsequences generated according to Equation 4.1 will be 8.

$$N = \frac{l - w}{q} + 1 \tag{4.1}$$

Table 4.5: An example of time series with length 20

Time	Value
t1	x1
t2	x2
t3	x3
t4	x4
...	...
t19	x19
t20	x20

The time series data of length (l) = 20 is shown in Table 4.5. The subsequences generated according to the previously mentioned values are as shown below.

$$S_1 = x_1, x_2, ..., x_6$$

$$S_2 = x_3, x_4, ..., x_8$$

$$\vdots$$

$$S_8 = x_{15}, x_{16}, ..., x_{20}$$

In our experiments, we have used the stride value q as 1. It means that sliding window moves one step each time through the sequence. We have data aggregated in 15-minute time intervals and each day represents a time series length of 96 (24 hours x 4-time quarters). If the window size is five, then the total number of subsequences generated according to the Equation 4.1 are 92.

However, we do have some missing data. We removed all the subsequences which have one or more missing values. Only the subsequences without the missing values are considered for training the models in our experiments.

Likewise, all subsequences are generated for the whole year and from locations 12 to 21. In all the experiments we have used the first 11 months data for training, i.e., from November 2017 to September 2018 and the last one-month data for evaluation, i.e., October 2018.

Let the window size be w, which can be 4, 5, 7, 9, 10, 11. The speed profile can be generated by the following steps:

1. Load the traffic speed dataset from locations 12 to 21 and also load the corresponding history dataset for the same period.

2. Wherever the values are greater than 100, replace them with 100.

3. Do the normalization for the data (min-max normalization is used with minimum value as 0 and maximum value as the maximum speed recorded in that particular location).

4. We used the sliding window approach with size w and stride 1 to create the subsequences of available data. Corresponding subsequences of historical data and anomaly data are also generated. Anomaly subsequence is generated based on the values of the actual dataset and history dataset. If the actual value is less than the historical value, then the difference of the actual and historical value is placed in the anomaly subsequence, otherwise 0 is placed. Thus, anomaly subsequences are created.

5. Also, in the subsequences for each sample, total sum of anomalies and mean of anomalies are stored.

6. Along with this information for each sample, we also stored other information such as location, year, month, day, hour, time quarter, and weekday.

Table 4.6 provides us with an example showing speed profile with subsequence length

Table 4.6: An example showing speed profile with subsequence length 4

t-3	t-2	t-1	t	Location	Month	Day	Hour	Time quarter	Weekday	Sum	Mean
1	0.93	0.95	0.98	12	10	1	0	4	2	0.14	0.035
0.93	0.95	0.98	0.97	12	10	1	1	1	2	0.17	0.0425
0.95	0.98	0.97	0.95	12	10	1	1	2	2	0.15	0.0375
0.98	0.97	0.95	0.94	12	10	1	1	3	2	0.16	0.04
1	1	0.99	1	12	10	1	8	1	2	0.01	0.0025
1	0.99	1	1	12	10	1	8	2	2	0.01	0.0025
0.99	1	1	1	12	10	1	8	3	2	0.01	0.0025
1	1	1	0.97	12	10	1	8	4	2	0.03	0.0075

of 4.

4.2 Imputing missing values- Multi-view approach

The accuracy of the traffic speed prediction depends on many factors like the amount of available historical data, how well patterns are represented in the historical data, which model is suitable for the nature of the dataset, etc. As shown in Figure 4.2, our dataset contains more than 40% of missing values in most of the locations, and this might affect the performance of the traffic forecasting model. Hence, our first task is to impute the missing values. Once the complete dataset is available, we build the traffic speed prediction model and predict traffic speed.

In intelligent transportation systems, missing patterns can be classified into three categories: Missing Completely at Random (MCR), Missing at Random (MR) and Missing at Determinate (MD) as shown in Figures 4.8 to 4.11 [62].

1) MCR: In this category, the missing values are randomly scattered (Type 1) as shown in Figure 4.8. They may occur due to a communication failure or temporary power failure.

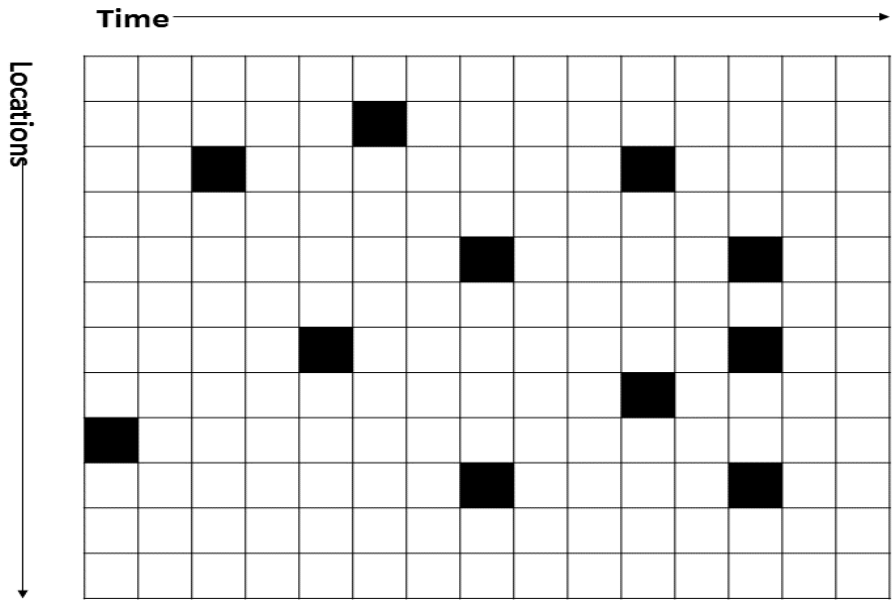

Figure 4.8: Type 1 missing pattern

2) MR: In this category, the missing values occur sequentially at the same time (Type 2) or the same location (Type 3). They may occur due to maintenance backlog or physical damage.

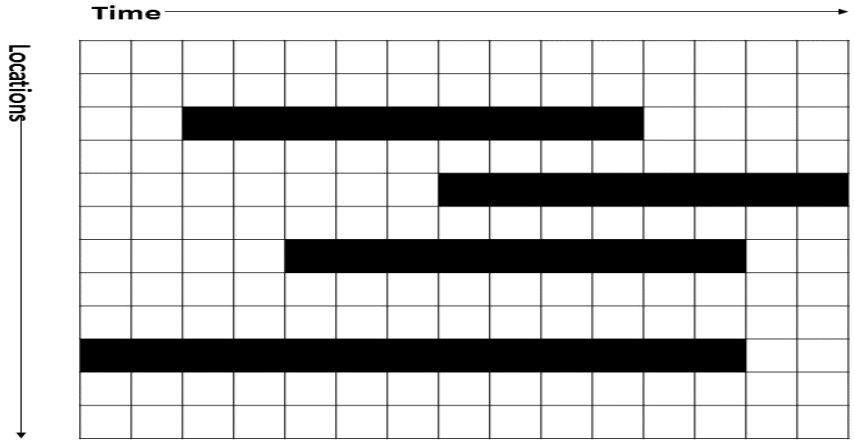

Figure 4.9: Type 2 missing pattern

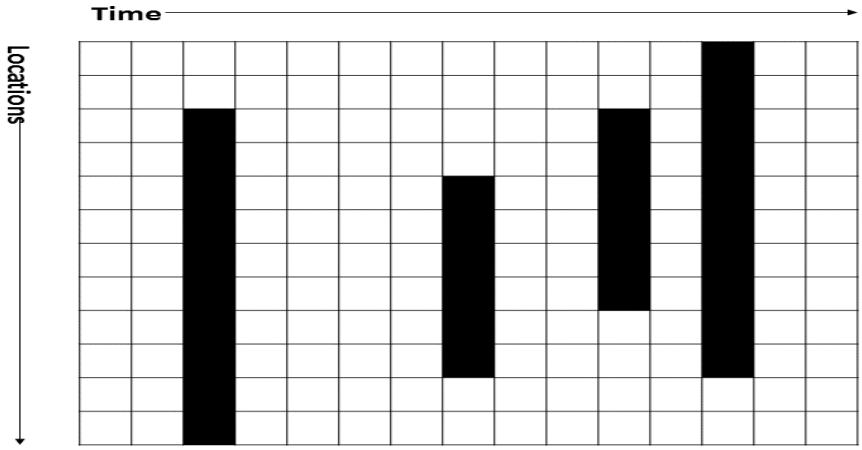

Figure 4.10: Type 3 missing pattern

3) MD: In this category (type 4), the missing values are like blocks. They may occur due to the long time malfunction of the sensors. They have certain patterns.

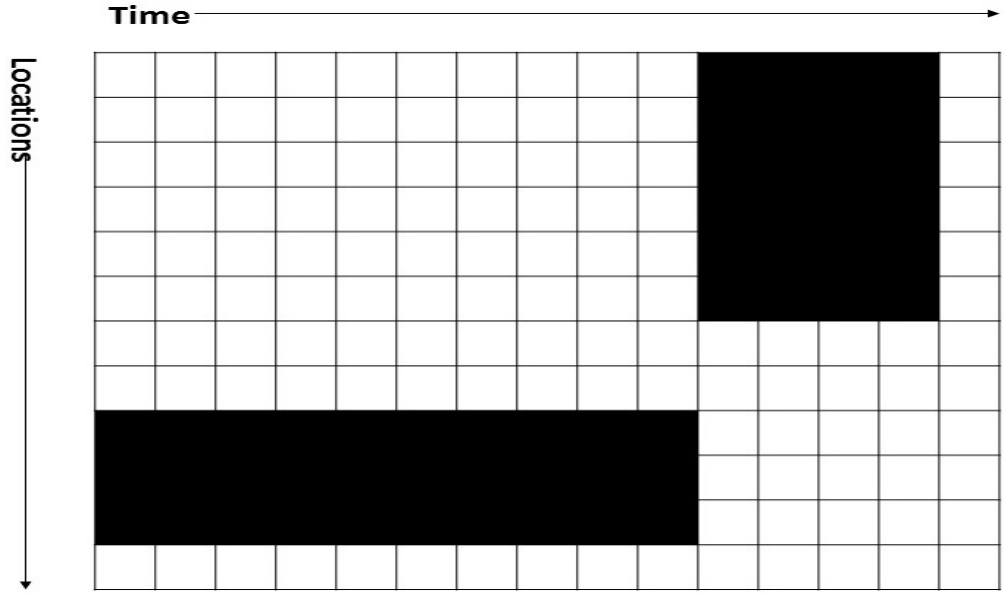

Figure 4.11: Type 4 missing pattern

An ideal imputation method should appropriately combine and use the global information (neighboring historical data in terms of pattern distance), as well as the local information (current day traffic data) [21].

In the proposed approach, we consider the local temporal, local spatial, global temporal and global spatial views as shown in Figure 4.12.

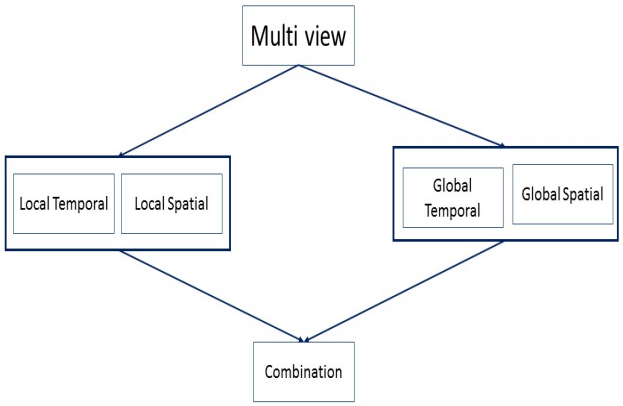

Figure 4.12: Multi View

Local Temporal: In this view we consider the surrounding temporal information within the current day to impute missing values. Suppose if traffic speed at time t is missing, then we use the values from time steps t-1, t-2,..., t-w and t+1, t+2,..., t+w where, w represents the number of time steps. For this view we applied collaborative filtering techniques and simple average techniques. The local temporal view defines current traffic conditions.

- Collaborative filtering temporal: In this method we take the weighted average of available t-1, t-2,...t-w and t+1, t+2,...t+w values. Here the weights are similarities which are calculated based on available traffic speed values at various locations in that particular time step. For example, if we predict the missing value at time t then the weights for t-1, t-2, ..., t-w are calculated based on the similarities between t and t-1, t and t-2, ..., t and t-w in all other locations. In this method similarity depends on the number of available traffic speeds neighbouring the missing datapoint.

- Simple average temporal: In this method we take the simple average of available t-1, t-2,...t-w and t+1, t+2,...t+w values to impute missing value at time

t.

Local spatial: In this view we consider the current day spatial information to impute missing values. We use the neighbouring spatial information to impute missing values. Suppose if traffic speed at location l is missing then we use the traffic speed on other locations l_1, l_2, ..., l_i. For this view we applied collaborative filtering techniques and simple average techniques.

- Collaborative filtering spatial: In this method we take the weighted average of available spatial traffic speed values in all other locations. Here, the weights are similarities which are calculated based on the available traffic speed values at various time steps from t-w to t+w. For example, to predict the missing value at location l we use the weighted average of all other locations. Here, the weights for l_1, l_2, ..., l_i are calculated based on the similarity between l and l_1, l and l_2, ..., l and l_i during time steps t-w to t+w. In this method also similarity depends on the number of available traffic speeds neighbouring the missing datapoint.

- Simple average spatial: In this method, we take the simple average of available spatial traffic speed values. Here, we calculate the missing value in location l using before and after locations neighbouring l.

The local methods are useful to impute random missing values and they show good accuracy when a greater percentage of neighbouring values are available.

Global temporal and Global spatial: In these methods, we considered the historical average values based on the same location and time. Here, time indicates the same hour, time quarter and weekday. For example, if the value is missing at location 12, time 2, time quarter 3, and weekday 4, we take the historical average value calculated from the same location and time.

The global methods are useful to impute sequential missing values.

4.3 Unusual traffic patterns

Unusual traffic pattern - Temporal: In Figure 4.13, we observe an anomaly in time quarters 4 to 10 on one of the Thursdays (25th October, 2018) whereas in other Thursdays, the speed is around 100km/h.

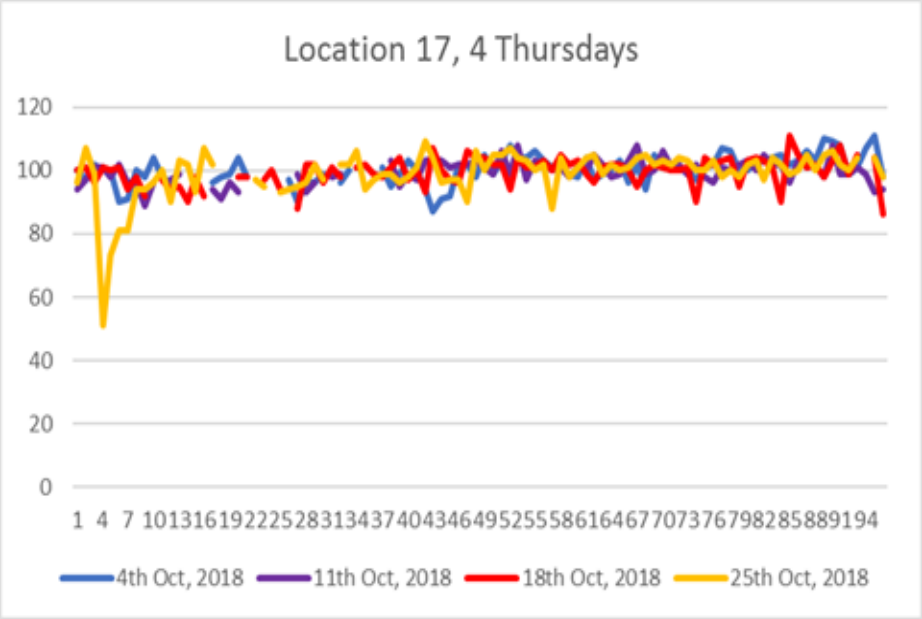

Figure 4.13: Temporal unusual traffic pattern

Atypical traffic pattern - Spatial: In Figures 4.14, 4.15, and 4.16 we show the traffic speed patterns from one Thursday in December 2017. The anomaly is propagating from location 16 to 18 during the time quarters of 1 to 17 and 39 to 55.

Figure 4.14: Unusual traffic pattern in location 16

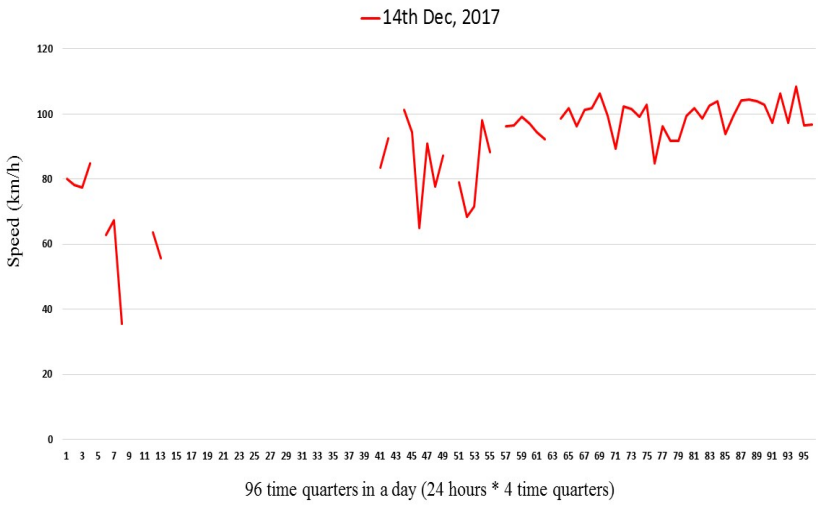

Figure 4.15: Unusual traffic pattern in location 17

Figure 4.16: Unusual traffic pattern in location 18

4.3.1 Traffic patterns in different locations

Usually the traffic shows time-variant characteristics in different days of the week, hours of day, etc. However in real-time, unusual traffic speed can occur anytime of the day or week due to accidents or bad weather conditions, etc. Authors [39] stated that the flow situation is not guaranteed to be the same in any day of the week. Hence, in this study we assume that unusual traffic speed patterns are independent of the temporal features such as weekday and hour of day.

We have selected two different weekdays, Wednesday(10th October 2018) and Thursday(25th October 2018) to show traffic patterns in all ten locations and this can be seen from Figure 4.17. For most of the locations we can see that traffic speed is always around 100 km/h. And also in our dataset we observed that traffic speed did not show significant time-variant characteristics in different hours of day, days of the week which can be seen from Figure 4.17. Also from Figure 4.17, we can see most of the unusual traffic speed patterns at locations 12, 17 and 18. At location 17, we can see the unusual traffic pattern

during 13th to 20th time quarter on 10th October 2018 and from 2nd to 9th time quarter on 25th October 2018. In locations 12 and 18 we can see the unusual traffic patterns from 75th to 81st time quarter on 10th October 2018 and from 68th to 73rd time quarter on 25th October 2018.

In our dataset we observed that there are more unusual traffic speeds at locations 12, 17 and 18. For example, in Table 4.7, we can see the speed count for locations 12 to 21 where the speeds observed to be less than 70 km/h.

Table 4.7: Speed count less than 70

locations	Speed
Location 12	263
Location 13	70
Location 14	61
Location 15	66
Location 16	63
Location 17	276
Location 18	137
Location 19	55
Location 20	76
Location 21	37

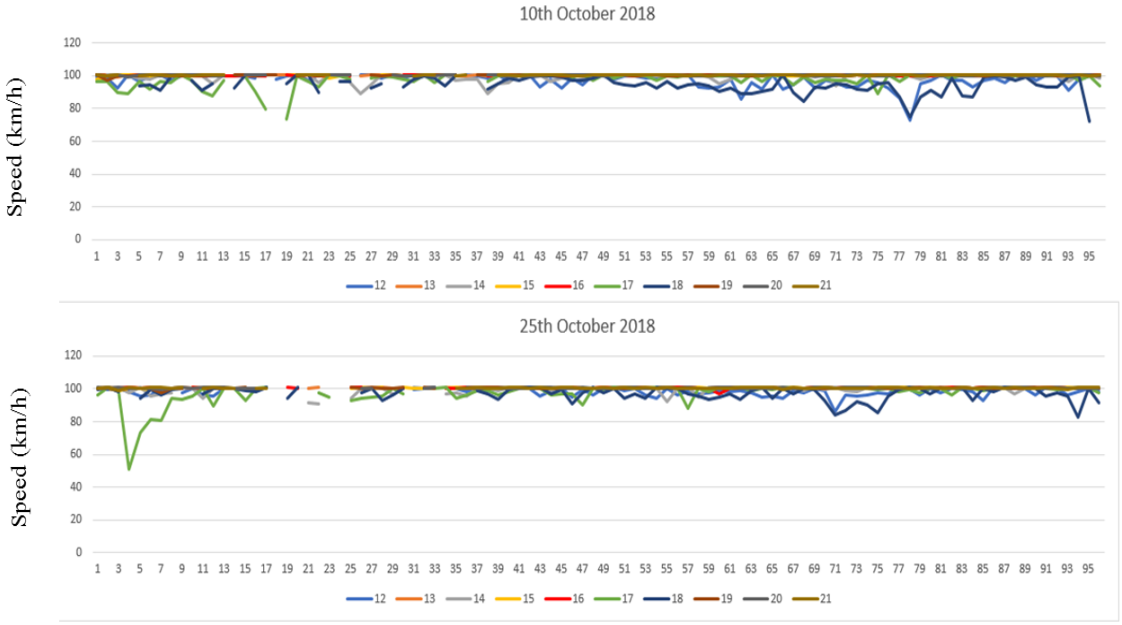

Figure 4.17: Traffic patterns in all locations

We have normalized the traffic speed data for each location. We used the min-max normalization where the minimum and maximum observed speeds in the location are used to normalize the data in that same location. Once this is done, we extracted the subsequences of length five from all the locations to show similar traffic patterns in Table 4.8.

In Table 4.8, we can see similar traffic patterns in different locations, different time and different weekday. In unusual traffic pattern- A, we can see the similar traffic pattern with length five in two different locations 17 and 19. Also from this, we can see the similar traffic pattern occurring in different month, day, hour, time quarter and weekday. Also in unusual traffic patterns B and C, we can see similar traffic patterns in different locations, weekdays and time of the day. Overall, from all these patterns we can see similar traffic patterns occurring in different locations and time. In this study, we assume that there exists similar patterns not related to the location, time, and weekday.

Table 4.8: Unusual traffic patterns

Unusual traffic patterns- A										
t-4	t-3	t-2	t-1	t	Location	Month	Day	Hour	TQ	Weekday
0.65	0.63	0.71	0.68	0.67	17	9	20	2	1	5
0.68	0.59	0.61	0.65	0.67	19	1	8	11	2	2
Unusual traffic patterns- B										
t-4	t-3	t-2	t-1	t	Location	Month	Day	Hour	TQ	Weekday
0.66	0.6	0.68	0.69	0.63	16	1	8	11	2	2
0.7	0.54	0.69	0.61	0.63	17	1	12	19	1	6
Unusual traffic patterns- C										
t-4	t-3	t-2	t-1	t	Location	Month	Day	Hour	TQ	Weekday
0.74	0.62	0.57	0.69	0.59	12	9	28	21	1	6
0.77	0.64	0.75	0.67	0.62	14	1	8	11	1	2

4.4 Recursive prediction employing dynamic kNN method

In recursive prediction, we predict future time step values in all locations as shown in the time step labelled as 'prediction values' in Figure 4.18. In our dataset we observed most of the anomalies in locations 12, 17 and 18, for example, we can see in Table 4.7 that most of the speed counts with speeds less than 70 km/h were observed at the aforementioned locations. Unusual traffic patterns in these locations do not comprise of enough training data compared to normal traffic patterns for the neural network. Hence, for these locations we use k-NN to predict with greater accuracy. This updated time step's values are appended to the previous time steps and used as the new input for the prediction model to predict the next time step. For instance, we can infer from Figure 4.18, the updated time step t+1 is appended to the previous time steps t-8 to t, and forming the new input data with time steps t-7 to t+1 to predict t+2. This procedure continues for eight future time steps as we are

predicting for two hours in the future. While considering the nearest patterns, we ensure that no two patterns are from the same day. We have used Euclidean distance to get the nearest patterns.

Fixing the number of nearest neighbors is one of the challenges in k-NN. For each test sample the number of nearest neighbors may be different based on the actual number of neighbors available in the dataset. For example, if k = 10 is fixed, there may be 6 nearest neighbors with similar traffic pattern as the test sample and the next four nearest neighbors with traffic pattern different from the test sample which are also considered for prediction and it may add error to the prediction. In order to solve this, we use dynamic k-value. Let n be the minimum k-value fixed by the user. The threshold is initially set to 0. Get all the nearest neighbors to which the distance is less than or equal to the threshold and increment the k-value by 1. If the minimum value of k is not obtained or any of the distances to the nearest neighbors is greater than threshold then stop obtaining nearest neighbors and increment the threshold value by 0.01. Based on the new threshold, get all the nearest neighbors. Continue this process until all distances to the nearest neighbors are less than or equal to the threshold.

Algorithm: Dynamic k-value

Input: Distances to the nearest neighbours

Output: k-value

Method: *Initialize threshold value*

Repeat:

1. *Update k-value if distances to nearest neighbours are less than or equal to threshold*
2. *Increment threshold value if any of the distances are less than threshold*

Until:

All distances to the nearest neighbors are less than or equal to the threshold

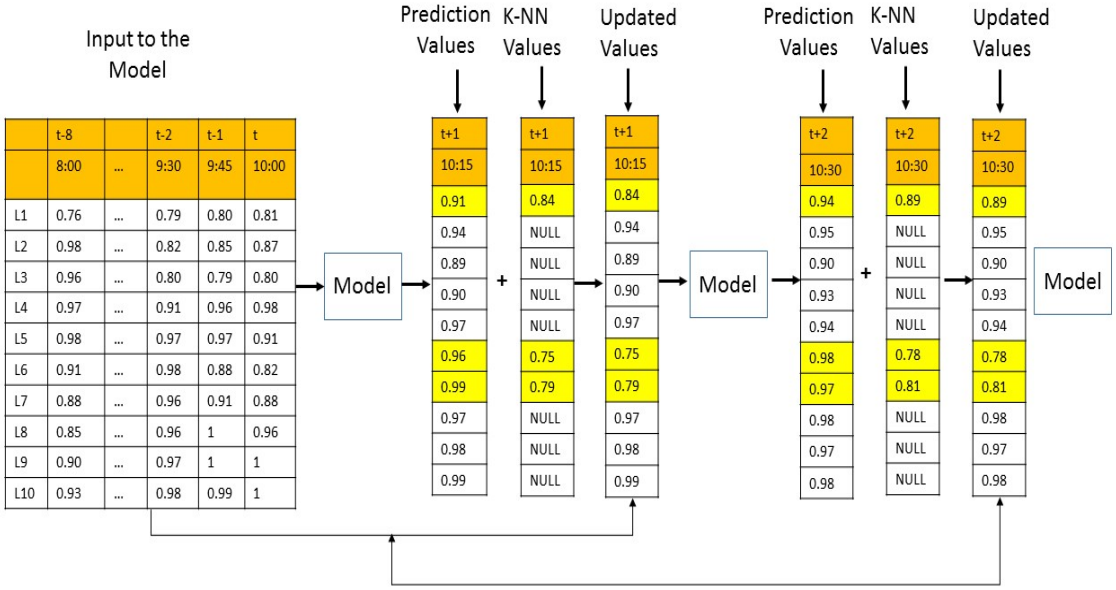

Figure 4.18: Recursive model prediction with dynamic k-value

In Figure 4.18, model refers to the CNN model.

4.5 Real-time speed prediction

In real time prediction, we predict future time step values in all locations as shown in the time step labelled as 'prediction values' in Figure 4.19. We now obtain the real-time values as shown in the time step labelled as 'real-time values' in this figure. These real-time values may contain missing values. Wherever we notice a missing value, we replace it with the model prediction value and generate an updated list of values for that time step. This updated time step is appended to the previous time steps and used as the new input for the prediction model to predict the next time step. For instance, we can infer from Figure 4.19, the updated time step t+1 is appended to the previous time steps t-8 to t, and forming the

new input data with time steps t-7 to t+1 to predict t+2. This procedure continues for all future time steps.

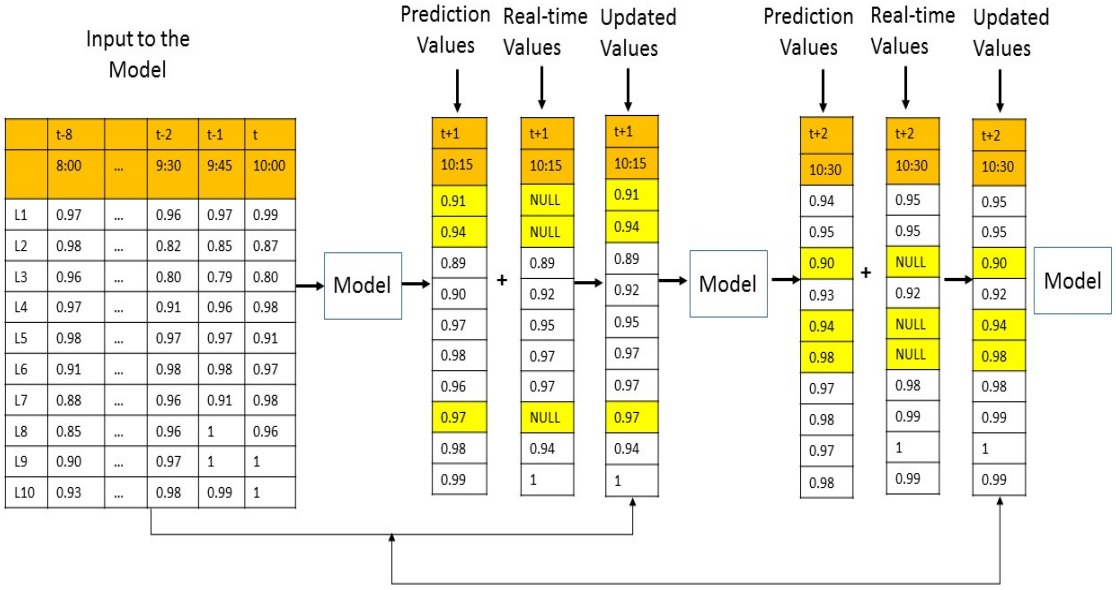

Figure 4.19: Real time model prediction

In Figure 4.19, model refers to the CNN model.

4.6 Experimental settings and evaluation metric

Experimental settings for various prediction models In this study we have used three different neural network prediction models ANN, LSTM, CNN in various experiments. In all experiments we assume N is the total number of samples, P is the number of locations, w is the number of time lags, F is the number of features. Input samples to these models are the complete blocks of spatial-temporal data. Suppose if we use 10 locations, and 10 time lags data to predict the one future time step for all the 10 locations then the size of the input sample is 10x10, and the output is the 1-D vector of size 10.

To predict the network-wide traffic speed, we need to pass the 4-D data to the 2D- CNN model, 3-D data to the LSTM model, and 2-D data to the ANN model.

Input data to the 2D CNN model is of the shape [N, P, w, F]. In all our CNN experiments we used only one feature which is traffic speed. In each sample of the CNN model, the number of rows represents the locations, and the number of columns represents the time lags. Hyperparameters to the model depends on the nature of the dataset. We choose our hyperparameters based on the previous studies using the traffic speed and also from our experiments on the dataset. In our CNN model, we used the Adam optimizer, mean squared error loss function, ten epochs, 64 batch size, filter size 3 and the ReLu activation function. Our CNN Model consists of two convolution layers with 64 and 32 filters and other than the mentioned values, all other hyperparameters are defaults provided by keras 2D CNN. We did not use the max pooling layer in our model architecture because the size of the input data is less.

Input data to the LSTM model is of the shape [N, w, P] where N is the total number of samples, w is the number of time lags, and P is the number of locations. The LSTM model takes each location time series as a different feature. At each timestep, we pass traffic speed values in all locations to the LSTM model. For this model we choose two LSTM layers with 10 neurons, 64 batch size, ten epochs, tanh activation function, Adam Optimizer, mean squared error loss function and other than the mentioned values, all other hyperparameters are defaults provided by keras LSTM.

Input data to the ANN model is of the shape [N, F] where N is the total number of samples, F is the total number of features. Suppose, we use 10 locations, and 10 time lags data to predict the one future time step in all 10 locations then the number of features in our ANN model are 100. In this model, we used the single fully connected hidden layer with 55 neurons and the relu activation function.

In all the models, output layer is the fully connected layer with linear or identity activa-

tion function.

We implemented all the experiments in python and used the keras sequential API and functional API for neural network models [63] with tensorflow as backend [64]. All experiments were run on CPU: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, GPU: NVIDIA GeForce GTX 1060.

In all the experiments we used the MAPE error rate.

$$\text{MAPE} = \frac{\sum_{i=1}^{N} |(x_i - x_{\hat{i}})/x_i|}{N} * 100\% \tag{4.2}$$

In the above equation, x_i represents the actual value and $x_{\hat{i}}$ represents the predicted value, N is the total number of samples. The data to all neural network models was normalized using min-max normalization.

$$\text{Normalized Data} = (ActualData - min)/(max - min) \tag{4.3}$$

In the Equation 5.2, max represents the maximum value observed for that feature and min represents the minimum value observed for that feature. In all our experiments we have normalized the data according to the location.

Chapter 5

RESULTS AND DISCUSSIONS

5.1 Network-wide traffic speed prediction without imputing the missing values

5.1.1 Analysis on the number of time lags

The performance of the prediction model depends on the number of time lags. In this experiment, we have tested the performance of our prediction model using the time lag durations of 45, 60, 75, 90, 105, 120, 135 and 150 minutes. We used the CNN model for prediction. In this experiment, we first generated the time series with length 11 which means we use 10 time lags to predict the traffic speed at 11th time step. To predict this 11th time step we use previous 3 time lags (45minutes), previous 4 time lags (60 minutes) and so on till previous 10 time lags (150 minutes). In this way we are always predicting the same value by considering different input lengths.

As shown in Figure 5.1, after the first one hour, the error rate is almost the same, and it gets smoother as time lags increase. In most of the studies using the deep learning methods for traffic prediction, the minimum number of time lags is 10 [59, 65]. Hence in all our experiments we use 10-time lags i.e., we use the last 2.5 hours traffic speed data to predict the next 15-minutes traffic speed. Also from the Section 5.4.1 we can see most of the atypical patterns within 2.5 hours range. This means we can cover most of the traffic patterns with time lags 10.

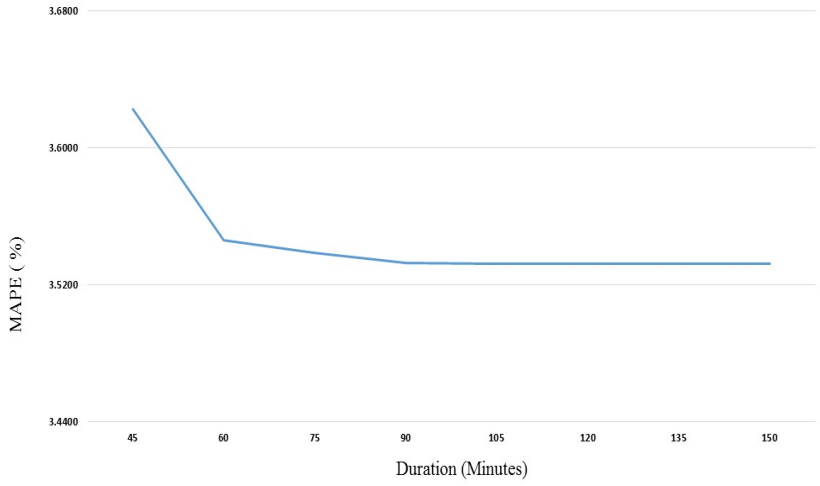

Figure 5.1: Traffic speed prediction using CNN with different number of time lags

5.1.2 Analysis on the prediction models

In the next experiment, we compared the ANN, LSTM, CNN models for multiple step prediction. Here, we predict the traffic recursively for the next two hours. In this experiment we predict next 15 minutes based on previous 150 minutes. we use the predicted values as input and then predict the next 30 minutes, likewise we predict for 2 hours. Table 5.1 shows the average of MAPE error rates in all locations at each time step for three different models. For all the models in this experiment we used the same settings as explained earlier in the section 5.1. As shown in Table 5.1, the CNN model showed better performance for all the time steps. From all the three models, we can see that CNN model is more stable and the error rate is less when compared to LSTM and ANN models. In the LSTM model till 45 minutes the error rate is less when compared to ANN model but after that it increased more when reached 120 minutes. In CNN model, the error rates are less in all the time steps for 120 minutes.

Table 5.1: Traffic speed prediction for 2 hours using CNN, LSTM, ANN models

Model	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
ANN	3.0692	3.2528	3.3196	3.2797	3.1748	3.5486	3.6318	3.4830
LSTM	2.7996	2.7427	2.9040	3.0162	3.2115	3.4678	3.8330	4.3451
CNN	1.5921	1.6596	1.7206	1.7156	1.7354	1.7508	1.7892	1.8256

5.1.3 Analysis on the different missing rates

In the next experiment, we generated the missing values ranging from 5% to 35% in the train data and imputed with zero values. The ANN, LSTM, CNN models are compared for single step prediction with different missing rates. In this experiment we used the same settings as mentioned in the section 5.1 for all models. The MAPE error rates in Figure 5.2 are the average error rates taken for all locations. From the Figure 5.2, we can see that the ANN model is more sensitive to the missing rates compared to LSTM and CNN models. However, for all the missing rates in this experiment CNN model showed better performance than LSTM model. In the Figure 5.2, we can see that with 5% missing values, the error rate for ANN is around 5% and for CNN and LSTM models it is between 1% to 2%. When the missing rate is increasing, the error rate for ANN is increasing significantly and reached to 45%. For LSTM model, the error rate increased slightly till 15% missing rate and remained at around 5% error rate till the 35% missing rate. In CNN model, the error rate increased from 1% to 2% as the missing rate is increasing from 5% to 35%. Hence from this experiment we can conclude that CNN model is more stable with the increasing missing rate.

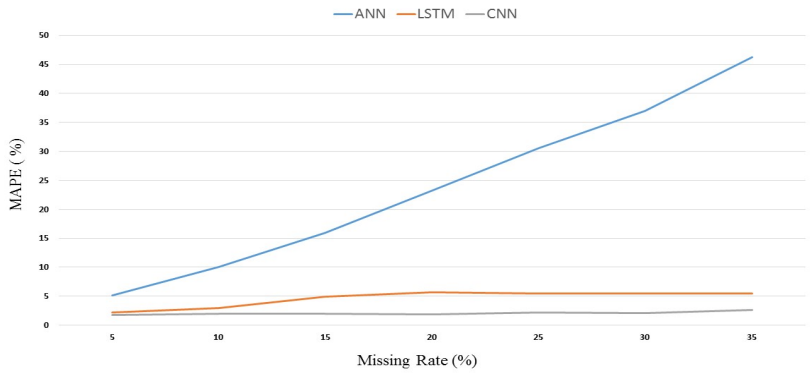

Figure 5.2: Traffic speed prediction using CNN, LSTM and ANN models with different missing rates

5.1.4 Analysis on multi-step speed prediction with CNN model

All the below experiments use the complete samples for training the models. In this experiment, we created the training data set by taking entire blocks of spatial-temporal data from the first eleven months and similarly we created the test data samples from the month of October 2018. These spatial-temporal blocks of data consists of complete sequences without any missing values. There are a total of 5524 samples in train dataset and 646 samples in the test dataset. We used CNN model for this experiment.

In the below experiment, for the first future time step, we used the model prediction and then used this predicted value along with the previous nine time lags to get the next time step prediction and likewise the recursive prediction is made till the eight future time steps. Which means here we are predicting the traffic for the next two hours in the future. The results for this network-wide recursive traffic speed prediction are shown in Table 5.2.

In the other experiment, we used the actual speed values to predict the speed at each time step. This is like a real-time prediction which means we use the real existing data to predict the next value. The results for this experiment are shown in Table 5.3. These results act as

the baseline to see the variations of error rates in recursive prediction for each time step. As shown in Table 5.2, the error rate is less in time step t+1 because it uses the actual data for prediction and in further steps the error rate start increasing and it is more in time step t+8. In all locations we can see that the error rate is less in initial time steps and more in later time steps. Also we can see that the error rate is more in locations 12, 17, and 18. The reason for this is the presence of greater anomaly in these locations which can be seen from the table 4.7. Comparing Tables 5.2 and 5.3, error rates at time step t+8 indicate that there is more error in recursive prediction when compared to actual prediction. Also from these two tables we can see that the error rate increased only in location 12, 17, 18 and the reason for this is again the same. Due to greater anomaly there would be an error in the initial steps, leading to a higher error rate in every subsequent time steps.

Table 5.2: CNN recursive multi-step prediction - MAPE

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.6728	3.9268	4.2048	4.3894	4.6136	4.7912	5.0181	5.1710
Location 13	0.4850	0.4711	0.3818	0.3248	0.3102	0.2938	0.2777	0.3053
Location 14	1.3083	1.2921	1.2821	1.2648	1.2458	1.2520	1.2663	1.2657
Location 15	0.4515	0.4337	0.4297	0.4324	0.4676	0.4748	0.4871	0.5290
Location 16	0.2109	0.2212	0.2364	0.2542	0.2792	0.2802	0.3025	0.3151
Location 17	2.7497	3.0408	3.1419	3.1653	3.1330	3.1364	3.1107	3.0894
Location 18	5.8078	5.9835	6.1827	6.0659	6.1322	6.0816	6.2857	6.4136
Location 19	0.1718	0.1847	0.1950	0.1882	0.1814	0.1769	0.1811	0.1980
Location 20	0.5204	0.5543	0.5671	0.5700	0.5412	0.5845	0.6287	0.6344
Location 21	0.2342	0.2126	0.2133	0.2078	0.2051	0.2218	0.2341	0.2443

Table 5.3: CNN single-step prediction- MAPE

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.6728	3.6743	3.7336	3.6821	3.6892	3.7100	3.7865	3.7721
Location 13	0.4850	0.4893	0.4892	0.4935	0.4934	0.4937	0.4917	0.4914
Location 14	1.3083	1.3012	1.3039	1.2952	1.2814	1.2760	1.2764	1.2940
Location 15	0.4515	0.4593	0.4595	0.4663	0.4658	0.4690	0.4726	0.4769
Location 16	0.2109	0.2124	0.2141	0.2172	0.2219	0.2232	0.2237	0.2236
Location 17	2.7497	2.8364	2.8555	2.8742	2.8456	2.8785	2.8982	2.9259
Location 18	5.8078	5.8372	5.8138	5.5743	5.5175	5.3408	5.4206	5.5160
Location 19	0.1718	0.1739	0.1790	0.1815	0.1849	0.1852	0.1860	0.1873
Location 20	0.5204	0.5252	0.5308	0.5324	0.5330	0.5367	0.5368	0.5379
Location 21	0.2342	0.2300	0.2286	0.2272	0.2192	0.2182	0.2216	0.2245

5.1.5 Analysis on single-step network-wide speed prediction with CNN and LSTM models

In the next experiment, we used LSTM and CNN models for single-step network-wide traffic prediction. As shown in Table 5.4, the results for CNN model are slightly better than that of LSTM model. And also, the number of parameters is lesser in CNN when compared to LSTM model hence, less prone to overfitting problems. Training time is also less for CNN model compared to LSTM model.

Table 5.4: LSTM and CNN for single step prediction - MAPE

locations	CNN	LSTM
Location 12	3.6728	3.7455
Location 13	0.4850	1.0483
Location 14	1.3083	1.3834
Location 15	0.4515	0.9360
Location 16	0.2109	0.7113
Location 17	2.7497	2.3515
Location 18	5.8078	6.7658
Location 19	0.1718	0.8297
Location 20	0.5204	1.1533
Location 21	0.2342	0.8100

5.1.6 Analysis on network-wide speed prediction with CNN mask

In the next experiment, we used the mask as the extra channel to CNN. The values in the mask are zero and one. One represents the available data and zero represents the missing data. Input to the 2D CNN is 4D Tensor, (nSamples, rows(locations), columns(timeLags), channels(features, in this experiment, features= 2). So the CNN model has two channels, one is for traffic speed and the other is for mask values. The total number of samples for training the model in this experiment are 122940. These samples are from the original data with missing values taken from the first 11 months. Comparing results reported in Tables 5.2 and 5.5 demonstrate that using the mask as the channel for CNN model did not show the significant improvement in the prediction.

Table 5.5: CNN single-step prediction with mask - MAPE

locations	t+1
Location 12	3.7109
Location 13	0.4661
Location 14	1.5309
Location 15	0.3741
Location 16	0.4440
Location 17	3.5617
Location 18	7.2971
Location 19	0.2623
Location 20	0.5113
Location 21	0.5428

Overall from the above experiments, we can conclude that CNN is best model for network-wide traffic prediction.

5.1.7 Analysis on speed prediction using weekday and hour

In this experiment we have incorporated auxiliary features such as the days of the week and times of day into our CNN based network-wide prediction model. The model structure of the CNN with weekday and hour is with two convolution layers, each with 64 and 32, filters with size 3 and two fully connected layers with 40 and 10 neurons. We converted the weekday and hour into the binary form. For fair comparison, we ensure that the traffic data is in the same hours for both train data and test data. So, in this experiment, there are a total of 5521 training samples and 859 test samples. However, adding this temporal information does not improve the performance of the model. The results are in accordance with the previous studies that used traffic speed for network-wide prediction [55]. Also, from the graphs 4.6 and 4.7, we can see that traffic speed is the same throughout the day

and speed patterns are almost the same for all weekdays. Hence, adding these features did not improve the results significantly.

Table 5.6: CNN model with weekday and hour - MAPE

Locations	CNN- Model	CNN-Model with weekday and hour
Location 12	3.6728	3.3633
Location 13	0.4850	0.4937
Location 14	1.3083	1.4709
Location 15	0.4515	0.8683
Location 16	0.2109	0.8020
Location 17	2.7497	2.2979
Location 18	5.8078	6.1700
Location 19	0.1718	0.5430
Location 20	0.5204	0.5420
Location 21	0.2342	0.4041
Average	1.5612	1.6955

5.2 Network-wide traffic speed prediction with imputing the missing values

To generate the missing values, we created the mask of True and False values randomly using python Pandas library. This mask is of the same size as the speed matrix (time series x locations). The number of True values in the mask represents the percentage of missing values and we can specify the percentage of missing values to be generated. Once this mask is created, we apply this to the original speed matrix to get the speed matrix of missing values. Thus, missing values can be generated randomly.

In the collaborative filtering techniques, as the window size increases, the number of

available data points surrounding the missing data point will increase. Hence, there will be a more significant number of predictions with bigger window size. In our experiments, if there are no available data points within the window range surrounding the missing point then we replace the missing point with zero as we cannot calculate it using CF technique and with more missing value percentage, there would be a smaller number of available data points in the window range. The error rate can be more if the window size is greater because we will be using data points that are far away to estimate the missing value. In traffic related time series data, the current traffic condition is always impacted by nearby traffic conditions in the temporal range. Therefore, we must make the trade-off between smaller window size and a greater window size, if the window size is small we cannot use enough traffic information surrounding the missing data point to estimate the missing value. If the window size is more we use irrelevant traffic information to impute the missing value. Hence, in all our experiments, we use the window size eight which means we use the one hour traffic data before the missing data point and one-hour traffic data after the missing data point to estimate the missing value. We also used two naive approaches, simple average of available data surrounding the missing data in both spatial and temporal range. For these experiments, we used the same window size as collaborative filtering techniques.

Before imputing the missing data in train data we analyze various methods for imputing the data. For this purpose we use October month's data as test data. we generated the missing data from 5% to 50% in test data(October 2018) as this month has more percentage of available data. We have tested the following methods.

1. CF-S: Collaborative filtering Spatial

2. CF-T: Collaborative filtering Temporal

3. HA: Historical average of all available data in the same location, same hour, same time quarter and same weekday as the missing data.

4. Avg-S: Simple Average of available data surrounding the missing data in the spatial range

5. Avg-T: Simple Average of available data surrounding the missing data in the temporal range

6. Multi-view 1: Combination of collaborative filtering spatial and temporal. Here, if both predictions are available, then the final predicted value is the average of these predictions. If either of them is available, then the available prediction is the final. If none of them are available then the historical average is the final predicted value.

7. Multi-view 2: Combination of simple average spatial and temporal. Here, also if both predictions are available, then the final predicted value is the average of these predictions. If either of them is available, then the available prediction is the final. If none of them are available then the historical average is the final predicted value.

8. Multi-view 3: The final predicted value is the collaborative filtering temporal value and if that value doesn't exist then the historical average value is the predicted value.

9. Multi-view 4: The final predicted value is the simple average temporal value and if that value doesn't exist then the historical average value is the predicted value.

   Whenever there are more number of available neighboring points, CF techniques show good predictions as shown in Table 5.7 from the multi-view 3 results with 5% of missing data. By increasing the percentage of missing values, the CF techniques cannot perform well as these approaches completely depend on the neighboring data and calculated weights cannot represent the

actual similarity measure. When the percentage of missing data is more, the simple average of nearby data to the missing data in the temporal range (Avg-T) works slightly better as we can see from the Table 5.7. Also from multi-view results 1 and 2, combining the spatial predictions with the temporal predictions are better than the temporal and spatial single views. In multi-view 3 and 4 methods we use the combination of temporal and historical average and results from these methods are better than the multi view 1 and 2 which also included spatial information. From multi view 3 and 4 results we can see that the results are better without adding spatial information. Overall when the missing percentage is more, the simple average of available neighboring data to the missing data in the temporal range shows better predictions and if the neighboring data is not available then the historical average shows better prediction. We can see this from the multi view-4 results.

Table 5.7: Comparison of different imputing methods for missing values - MAPE

Method	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
CF-S	1.9117	2.9543	2.7698	2.7466	2.6863	2.6788	2.6686	2.6428	3.0113	3.4114
CF-T	1.1719	1.9917	2.0114	2.0437	2.1598	2.4465	2.8172	3.2785	3.9855	4.9770
HA	1.6559	1.6559	1.6559	1.6559	1.6559	1.6559	1.6559	1.6559	1.6559	1.6559
Avg-S	2.8043	2.9906	2.9965	3.0043	3.0600	3.0824	3.0913	3.0877	3.4512	3.7864
Avg-T	1.6626	1.9148	1.9235	1.9985	2.0793	2.3607	2.7611	3.1740	3.9077	4.9089
Multi-view 1	1.0369	1.7724	1.7047	1.6742	1.6248	1.6074	1.6179	1.6227	1.6346	1.6935
Multi-view 2	1.6240	1.7305	1.7375	1.7361	1.7378	1.7517	1.7742	1.7672	1.7968	1.8376
Multi-view 3	**0.8090**	1.4833	1.5031	1.4626	1.5060	1.5156	1.5234	1.6006	1.5885	1.5834
Multi-view 4	1.2997	**1.4065**	**1.4152**	**1.4174**	**1.4254**	**1.4297**	**1.4672**	**1.4961**	**1.5107**	**1.5153**
nPred	1375	1375	1375	1375	1375	1375	1375	1375	1375	1375

The results in table 5.8 show the MAPE error rate for network-wide prediction with different missing rates.

Table 5.8: CNN model with different missing rates in train data - MAPE

Location	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
Location 12	3.5879	3.8406	4.4761	5.6434	5.2266	5.4273	8.1067	9.8360	9.8948	9.8450
Location 13	0.8971	0.8931	1.1848	0.4562	1.3204	1.0166	0.5129	1.7296	2.4232	4.3852
Location 14	1.6252	1.8214	1.6292	1.2596	1.5489	1.3515	1.6773	2.5997	2.8924	3.6124
Location 15	0.3418	0.9093	0.6643	0.4338	0.9419	0.4067	1.2535	3.9688	4.0945	5.9836
Location 16	0.5062	0.9663	0.7622	0.5753	1.2085	0.8475	0.7145	1.3843	2.2993	2.4288
Location 17	2.4548	2.5039	2.3877	2.0621	2.3235	2.3142	1.8215	1.8849	2.3257	4.2660
Location 18	6.9696	6.8243	6.9415	7.0468	6.9789	7.1047	8.0245	9.4836	10.7852	11.8835
Location 19	0.3223	0.9021	0.5651	0.4967	0.6060	0.4647	1.6888	2.6438	3.2861	5.7184
Location 20	0.4342	0.9907	0.9671	0.4737	0.9312	0.9510	1.0987	2.2128	3.1841	3.8570
Location 21	0.4068	0.4977	0.4198	0.6342	0.5608	0.5110	1.1683	1.5287	2.8943	3.6565

In this experiment, we have generated the missing data from 5% to 50% in the train data of the CNN model for single step prediction and compared the performance of the model after imputing the missing data with multi view-4 approach.

From Table 5.9, we can see that the performance of the CNN model has improved significantly.

Table 5.9: CNN model before and after filling the missing values in train data - MAPE

Locations	Before filling	After filling
Location 12	9.8450	**3.5541**
Location 13	4.3852	**0.29617**
Location 14	3.6124	**1.3567**
Location 15	5.9836	**0.3761**
Location 16	2.4288	**0.3563**
Location 17	4.2660	**2.4624**
Location 18	11.8835	**5.2746**
Location 19	5.7184	**0.3123**
Location 20	3.8570	**0.6175**
Location 21	3.6565	**0.5366**

5.3 Analysis on unusual traffic speed

The unusual sequences are generated from 12 to 21 locations from November 2018 to September 2018. The traffic data contains many atypical observations. In the presence of these unusual traffic speeds, both human administrators and automated algorithms attempt to discover incidents in the road network. We aggregate the number of individual unusual traffic speeds that occur in close spatial and temporal proximity into the atypical sequences. These sequences describe the features like duration, severity of anomalies, etc. Analyzing such features helps in improving the performance of many algorithms. In this study, we investigate the impact of these sequences in imputing the missing data. The traffic variable speed captures the intuition of traffic irregularities as moderate or fast-moving traffic; hence it has been used in many incident detection algorithms [66]. We assume the histori-

cal average speed represents usual traffic speed and the speed which is deviating from this represents the unusual speed. It is considered as unusual speed if the difference between historical speed and the actual speed is greater than the threshold value. In this study, we considered the threshold values 0, 10, 20, and 30. If such unusual speeds appear continuously in the temporal range, we call them as atypical temporal sequence, and if they occur spatially, we call them as atypical spatial sequences.

Table 5.10: Atypical sequence information where historical speed- actual speed >0 - Temporal

Duration (Min)	Sequences	Duration (Min)	Sequences
15	8111	120	13
30	1811	135	12
45	616	150	12
60	227	165	7
75	104	180	3
90	53	210	1
105	31	225	2

We have one sequence for each of the durations (255, 270, 285, 300, 315, 465, 525, 600, 690) in minutes.

Table 5.11: Atypical sequence information where historical speed- actual speed >10 - Temporal

Duration (Min)	Sequences	Duration (Min)	Sequences
15	1112	90	8
30	166	105	3
45	33	120	3
60	29	135	4
75	10	240	2

Table 5.12: Atypical sequence information where historical speed- actual speed >20 - Temporal

Duration (Min)	Sequences	Duration (Min)	Sequences
15	362	75	7
30	58	90	1
45	20	120	1
60	8	135	3

Table 5.13: Atypical sequence information where historical speed- actual speed >30 - Temporal

Duration (Min)	Sequences	Duration (Min)	Sequences
15	164	75	4
30	36	105	1
45	10	120	1
60	7	135	1

Total number of atypical sequences where the minimum length starting from two for each threshold in the temporal range are:

- Total sequences with threshold 0: 2908
- Total sequences with threshold 10: 258
- Total sequences with threshold 20: 98
- Total sequences with threshold 30: 60

Similarly, we have atypical sequences in the spatial range. Total number of atypical sequences where the minimum length starting from two for each threshold in the spatial range are:

- Total sequences with threshold 0: 1985

- Total sequences with threshold 10: 126
- Total sequences with threshold 20: 63
- Total sequences with threshold 30: 31

We can infer from the tables above, that the number of sequences decrease as the threshold increases. Most of the anomaly patterns are covered within the ten time steps.

From the above details, we can clearly see that there are 50% less unusual traffic sequences in the spatial range compared to temporal range. Moreover in most of the scenarios whenever there is unusual traffic in spatial range, we can also see them in temporal range. However, it is not vice versa which means whenever there is unusual traffic in temporal range, it may or may not be present in spatial range. One reason for this might be the distance between locations. In our dataset, we have 5km distance between every two locations and it might be the case that the traffic becomes abnormal to normal within this distance. Due to these kind of scenarios, in most of our experiments we found temporal predictions are better than the spatial predictions. In the future work, we plan to use this kind of information for imputing the missing values and also for prediction.

5.4 Analysis on recursive traffic speed prediction using dynamic k-NN

In this experiment, for the first future time step, we used the model prediction and then updated the obtained predicted values for locations 12, 17 and 18 using dynamic k-NN. These updated values are appended along with the previous nine time lags to get the next time step prediction and likewise the recursive prediction is made till the eight future time steps. Which means here we are predicting the traffic for the next two hours in the future. The results for this network-wide recursive traffic speed prediction are shown in Table 5.14.

Table 5.14: CNN recursive multi-step prediction with dynamic- kNN - MAPE

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.4144	3.5717	3.7988	3.9484	4.1348	4.1513	4.3763	4.5458
Location 13	0.4058	0.4212	0.3826	0.3408	0.3191	0.3021	0.3237	0.3624
Location 14	1.3100	1.3340	1.3426	1.3154	1.3108	1.3161	1.2977	1.3229
Location 15	0.3278	0.3354	0.3183	0.2997	0.3648	0.4453	0.5258	0.6450
Location 16	0.0813	0.0843	0.0936	0.0905	0.0891	0.0875	0.1108	0.1324
Location 17	2.0388	2.0683	2.1414	2.1316	2.1214	2.1579	2.1133	2.1892
Location 18	5.6010	5.7720	6.0044	5.8717	5.9404	5.8318	5.9539	6.2302
Location 19	0.0526	0.0542	0.0681	0.0604	0.0604	0.0449	0.0511	0.0496
Location 20	0.5249	0.5698	0.6039	0.6287	0.6070	0.6565	0.7355	0.7912
Location 21	0.1118	0.0931	0.0837	0.0900	0.0875	0.0906	0.0952	0.0891

In Table 5.14, we can see that the error rate improved significantly in locations 17 and 18 for almost all time steps. However, for location 12 the error improved till the fourth time step. These results can be compared with the results in Table 5.2.

5.5 Analysis on real-time traffic speed prediction with missing values

The real-time values may contain missing values. Wherever we notice a missing value, we replace it with the model prediction value and generate an updated list of values for that time step. This updated time step is appended to the previous time steps and used as the new input for the prediction model to predict the next time step. Likewise the recursive prediction is made till the eight future time steps. Which means here we are predicting the traffic for the next two hours in the future. The results for this experiment are shown in Tables 5.15 to 5.19 and they belong to the model shown in Figure 4.18. We can see that till 20% missing rate the results are same as actual results and as the percentage of missing

rate increases, the results are almost same as the recursive results. For actual results we can compare with the results reported in Table 5.3 and for recursive results we can compare with the results in Table 5.2.

Table 5.15: Real-time prediction with 10% missing rate

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.6728	3.6937	3.7799	3.7123	3.7239	3.7676	3.8305	3.8747
Location 13	0.4850	0.4872	0.4752	0.4708	0.4750	0.4724	0.4759	0.4834
Location 14	1.3083	1.3014	1.2999	1.2894	1.2812	1.2701	1.2824	1.2946
Location 15	0.4515	0.4575	0.4548	0.4602	0.4645	0.4703	0.4732	0.4758
Location 16	0.2109	0.2125	0.2133	0.2165	0.2243	0.2201	0.2232	0.2247
Location 17	2.7497	2.8446	2.8780	2.8960	2.8409	2.8650	2.8874	2.8970
Location 18	5.8078	5.8582	5.8724	5.6956	5.5882	5.4536	5.5193	5.5789
Location 19	0.1718	0.1735	0.1815	0.1795	0.1804	0.1796	0.1818	0.1811
Location 20	0.5204	0.5284	0.5317	0.5323	0.5294	0.5304	0.5396	0.5410
Location 21	0.2342	0.2264	0.2276	0.2232	0.2113	0.2146	0.2165	0.2246

Table 5.16: Real-time prediction with 20% missing rate

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.6728	3.7437	3.8128	3.7895	3.8035	3.8366	3.8946	3.9329
Location 13	0.4850	0.4831	0.4604	0.4540	0.4528	0.4591	0.4634	0.4759
Location 14	1.3083	1.2953	1.3007	1.2837	1.2782	1.2732	1.2769	1.2856
Location 15	0.4515	0.4541	0.4515	0.4529	0.4658	0.4716	0.4731	0.4910
Location 16	0.2109	0.2125	0.2174	0.2147	0.2291	0.2224	0.2264	0.2227
Location 17	2.7497	2.8680	2.9082	2.9225	2.8588	2.8703	2.9012	2.9258
Location 18	5.8078	5.8751	5.9009	5.7320	5.6389	5.5009	5.5841	5.6231
Location 19	0.1718	0.1752	0.1829	0.1785	0.1774	0.1756	0.1789	0.1780
Location 20	0.5204	0.5304	0.5348	0.5347	0.5297	0.5371	0.5496	0.5488
Location 21	0.2342	0.2235	0.2252	0.2227	0.2018	0.2146	0.2112	0.2199

Table 5.17: Real-time prediction with 40% missing rate

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.6728	3.7942	3.8596	3.8974	3.9489	4.0109	4.0949	4.1267
Location 13	0.4850	0.4839	0.4460	0.4228	0.4174	0.4203	0.4220	0.4406
Location 14	1.3083	1.2981	1.2983	1.2777	1.2814	1.2735	1.2782	1.2747
Location 15	0.4515	0.4494	0.4482	0.4457	0.4627	0.4714	0.4770	0.4988
Location 16	0.2109	0.2147	0.2243	0.2214	0.2375	0.2362	0.2432	0.2378
Location 17	2.7497	2.9338	2.9674	2.9766	2.9138	2.9134	2.9233	2.9646
Location 18	5.8078	5.9159	5.9254	5.8334	5.6891	5.6474	5.7395	5.8466
Location 19	0.1718	0.1763	0.1853	0.1799	0.1745	0.1727	0.1772	0.1756
Location 20	0.5204	0.5363	0.5409	0.5424	0.5264	0.5387	0.5549	0.5548
Location 21	0.2342	0.2210	0.2259	0.2178	0.1984	0.2156	0.2087	0.2145

Table 5.18: Real-time prediction with 80% missing rate

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.6728	3.8896	4.0890	4.2116	4.4175	4.5340	4.6550	4.7796
Location 13	0.4850	0.4734	0.4044	0.3560	0.3457	0.3397	0.3377	0.3619
Location 14	1.3083	1.2953	1.2887	1.2665	1.2620	1.2589	1.2706	1.2742
Location 15	0.4515	0.4384	0.4358	0.4368	0.4696	0.4790	0.4882	0.5242
Location 16	0.2109	0.2189	0.2311	0.2405	0.2622	0.2595	0.2763	0.2803
Location 17	2.7497	3.0066	3.0974	3.1168	3.0579	3.0259	3.0204	3.0419
Location 18	5.8078	5.9515	6.0718	5.9787	5.9791	5.9417	6.0374	6.1858
Location 19	0.1718	0.1804	0.1904	0.1858	0.1781	0.1757	0.1783	0.1918
Location 20	0.5204	0.5488	0.5564	0.5613	0.5344	0.5660	0.5994	0.6027
Location 21	0.2342	0.2147	0.2162	0.2096	0.1982	0.2162	0.2224	0.2241

Table 5.19: Real-time prediction with 100% missing rate

locations	15 min	30 min	45 min	60 min	75 min	90 min	105 min	120 min
Location 12	3.6728	3.9268	4.2048	4.3894	4.6136	4.7912	5.0181	5.1710
Location 13	0.4850	0.4711	0.3818	0.3248	0.3102	0.2938	0.2777	0.3053
Location 14	1.3083	1.2921	1.2821	1.2648	1.2458	1.2520	1.2663	1.2657
Location 15	0.4515	0.4337	0.4297	0.4324	0.4676	0.4748	0.4871	0.5290
Location 16	0.2109	0.2212	0.2364	0.2542	0.2792	0.2802	0.3025	0.3151
Location 17	2.7497	3.0408	3.1419	3.1653	3.1330	3.1364	3.1107	3.0894
Location 18	5.8078	5.9835	6.1827	6.0659	6.1322	6.0816	6.2857	6.4136
Location 19	0.1718	0.1847	0.1950	0.1882	0.1814	0.1769	0.1811	0.1980
Location 20	0.5204	0.5543	0.5671	0.5700	0.5412	0.5845	0.6287	0.6344
Location 21	0.2342	0.2126	0.2133	0.2078	0.2051	0.2218	0.2341	0.2443

The results shown in Table 5.19 are similar to results in Table 5.2.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

- In this research work, we have adopted the CNN model for network-wide traffic speed prediction.
- In our dataset there is no significant difference between the traffic patterns for different weekdays and also there are no clear established traffic patterns related to the time of the day. We can observe the difference between the normal traffic patterns and unusual traffic patterns.
- We have used the multi-view learning approach based on simple average and historical average techniques for imputing the missing values.
- Experimental results show that imputing the missing data considering only the temporal information is better than considering both the temporal and spatial information. The performance of the CNN model significantly improved after imputing the missing values with multi-view approach.
- We have used three different models (ANN, LSTM, CNN) for prediction and we have investigated these three model performances with different missing rates in the training data. Overall, the ANN models were found to be more sensitive to the percentage of missing data than LSTM and CNN models.
- We have implemented the CNN model combined with KNN for recursive prediction in which k-NN was used only in the locations where we have more

unusual traffic conditions.

- In our k-NN method, to choose the k value we have implemented dynamic k method where k value is calculated dynamically for each sample from the dataset.

- Adopted CNN model for real time traffic speed prediction with missing data

6.2 Future Work

Future work could be using different clustering techniques for obtaining the various groups of atypical patterns in the traffic speed and use this information for both imputing the missing values and also for traffic speed prediction. As the traffic speed is always normal which is around the speed limit for that particular road segment, we believe that by adding the pattern view to the multi-view approach can reduce the error rate. This pattern view can handle the atypical patterns in the traffic speed and could enhance the prediction results. And also the neural network models cannot work efficiently without enough training data and some traffic congestions are non recurrent, we intend to use the pattern based information obtained from clusters and modify the results of CNN model. We believe especially this information would be useful in the case of recursive multi step prediction. Because in the first step if we modify the results with accurate predictions, the error will not propagate to future time steps. Analyzing the impact of weather, accidents, etc. on the traffic speed could be our future work.

BIBLIOGRAPHY

[1] J. Park, D. Li, Y. L. Murphey, J. Kristinsson, R. McGee, M. Kuang, and T. Phillips, "Real time vehicle speed prediction using a neural network traffic model," in *The 2011 International Joint Conference on Neural Networks*. IEEE, 2011, pp. 2991–2996.

[2] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[3] "Ontario highway 401 — Wikipedia, the free encyclopedia." [Online]. Available: https://en.wikipedia.org/wiki/Ontario_Highway_401

[4] W. P. Anderson, H. F. Maoh, and K. Gingerich, "Cross-border freight movements in the great lakes and st. lawrence region, with insights from passive gps data," *The Canadian Geographer/Le Géographe canadien*, vol. 63, no. 1, pp. 69–83, 2019.

[5] L. Li, J. Zhang, Y. Wang, and B. Ran, "Missing value imputation for traffic-related time series data based on a multi-view learning method," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[6] X. Yi, Y. Zheng, J. Zhang, and T. Li, "St-mvl: filling missing values in geo-sensory time series data," 2016.

[7] A. K. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, no. 3, pp. 31–44, 1996.

[8] M. Moniruzzaman, H. Maoh, and W. Anderson, "Short-term prediction of border crossing time and traffic volume for commercial trucks: A case study for the ambassador bridge," *Transportation Research Part C: Emerging Technologies*, vol. 63, pp. 182–194, 2016.

[9] "Activation function — Wikipedia, the free encyclopedia." [Online]. Available: https://en.wikipedia.org/wiki/Activation_function

[10] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[11] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[14] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[15] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[16] V. Vapnik, S. E. Golowich, and A. J. Smola, "Support vector method for function approximation, regression estimation and signal processing," in *Advances in neural information processing systems*, 1997, pp. 281–287.

[17] L. Li, Y. Li, and Z. Li, "Efficient missing data imputing for traffic flow by considering temporal and spatial dependence," *Transportation research part C: emerging technologies*, vol. 34, pp. 108–120, 2013.

[18] B. L. Smith, W. T. Scherer, and J. H. Conklin, "Exploring imputation techniques for missing data in transportation management systems," *Transportation Research Record*, vol. 1836, no. 1, pp. 132–142, 2003.

[19] S. Tak, S. Woo, and H. Yeo, "Data-driven imputation method for traffic data in sectional units of road links," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1762–1771, 2016.

[20] D. Ni and J. D. Leonard, "Markov chain monte carlo multiple imputation using bayesian networks for incomplete intelligent transportation systems data," *Transportation research record*, vol. 1935, no. 1, pp. 57–67, 2005.

[21] L. Qu, L. Li, Y. Zhang, and J. Hu, "Ppca-based missing data imputation for traffic flow volume: A systematical approach," *IEEE Transactions on intelligent transportation systems*, vol. 10, no. 3, pp. 512–522, 2009.

[22] C. Chen, J. Kwon, J. Rice, A. Skabardonis, and P. Varaiya, "Detecting errors and imputing missing data for single-loop surveillance systems," *Transportation Research Record*, vol. 1855, no. 1, pp. 160–167, 2003.

[23] M. Zhong, P. Lingras, and S. Sharma, "Estimation of missing traffic counts using factor, genetic, neural, and regression techniques," *Transportation Research Part C: Emerging Technologies*, vol. 12, no. 2, pp. 139–166, 2004.

[24] H. Chen, S. Grant-Muller, L. Mussone, and F. Montgomery, "A study of hybrid neural network approaches and the effects of missing data on traffic forecasting," *Neural Computing & Applications*, vol. 10, no. 3, pp. 277–286, 2001.

[25] M. T. Asif, N. Mitrovic, J. Dauwels, and P. Jaillet, "Matrix and tensor based methods for missing data estimation in large traffic networks," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 7, pp. 1816–1825, 2016.

[26] X. Chen, Z. He, and J. Wang, "Spatial-temporal traffic speed patterns discovery and incomplete data recovery via svd-combined tensor decomposition," *Transportation research part C: emerging technologies*, vol. 86, pp. 59–77, 2018.

[27] X. Chen, Z. He, and L. Sun, "A bayesian tensor decomposition approach for spatiotemporal traffic data imputation," *Transportation research part C: emerging technologies*, vol. 98, pp. 73–84, 2019.

[28] B. Bae, H. Kim, H. Lim, Y. Liu, L. D. Han, and P. B. Freeze, "Missing data imputation for traffic flow speed using spatio-temporal cokriging," *Transportation Research Part C: Emerging Technologies*, vol. 88, pp. 124–139, 2018.

[29] W. C. Ku, G. R. Jagadeesh, A. Prakash, and T. Srikanthan, "A clustering-based approach for data-driven imputation of missing traffic data," in *2016 IEEE Forum on Integrated and Sustainable Transportation Systems (FISTS)*. IEEE, 2016, pp. 1–6.

[30] Y. Duan, Y. Lv, Y.-L. Liu, and F.-Y. Wang, "An efficient realization of deep learning for traffic data imputation," *Transportation research part C: emerging technologies*, vol. 72, pp. 168–181, 2016.

[31] I. Laña, I. I. Olabarrieta, M. Vélez, and J. Del Ser, "On the imputation of missing data for road traffic forecasting: New insights and novel techniques," *Transportation research part C: emerging technologies*, vol. 90, pp. 18–33, 2018.

[32] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "Lstm-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, pp. 297–305, 2018.

[33] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert systems with applications*, vol. 36, no. 3, pp. 6164–6173, 2009.

[34] F. Guo, J. W. Polak, and R. Krishnan, "Comparison of modelling approaches for short term traffic prediction under normal and abnormal conditions," in *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2010, pp. 1209–1214.

[35] T. Wu, K. Xie, D. Xinpin, and G. Song, "A online boosting approach for traffic flow forecasting under abnormal conditions," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*. IEEE, 2012, pp. 2555–2559.

[36] F. Guo, R. Krishnan, and J. W. Polak, "Short-term traffic prediction under normal and abnormal traffic conditions on urban roads," in *Transportation Research Board 91st Annual Meeting*, no. 12-1627, 2012.

[37] T.-I. Theodorou, A. Salamanis, D. D. Kehagias, D. Tzovaras, and C. Tjortjis, "Short-term traffic prediction under both typical and atypical traffic conditions using a pattern transition model." in *VEHITS*, 2017, pp. 79–89.

[38] A. Salamanis, G. Margaritis, D. D. Kehagias, G. Matzoulas, and D. Tzovaras, "Identifying patterns under both normal and abnormal traffic conditions for short-term traffic prediction," *Transportation research procedia*, vol. 22, pp. 665–674, 2017.

[39] B. Sun, W. Cheng, P. Goswami, and G. Bai, "Flow-aware wpt k-nearest neighbours regression for short-term traffic prediction," in *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2017, pp. 48–53.

[40] B. Sun, W. Cheng, L. Ma, and P. Goswami, "Anomaly-aware traffic prediction based on automated conditional information fusion," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 2283–2289.

[41] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic

flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 2016, pp. 324–328.

[42] M. S. Ahmed and A. R. Cook, *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*, 1979, no. 722.

[43] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, 2011.

[44] S. Kim, J. Kim, and K. R. Ryu, "Comparison of different k-nn models for speed prediction in an urban traffic network," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, no. 2, pp. 419–422, 2017.

[45] L. Vanajakshi and L. R. Rilett, "A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed," in *IEEE Intelligent Vehicles Symposium, 2004*. IEEE, 2004, pp. 194–199.

[46] P. Theja and L. Vanajakshi, "Short term prediction of traffic parameters using support vector machines technique," in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. IEEE, 2010, pp. 70–75.

[47] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387–399, 2011.

[48] J. Hua and A. Faghri, "Apphcations of artificial neural networks to intelligent vehicle-highway systems," *Transportation Research Record*, vol. 1453, p. 83, 1994.

[49] Y.-y. Chen, Y. Lv, Z. Li, and F.-Y. Wang, "Long short-term memory model for traffic

congestion prediction with online open data," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 132–137.

[50] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

[51] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*. IEEE, 2015, pp. 153–158.

[52] D. Kang, Y. Lv, and Y.-y. Chen, "Short-term traffic flow prediction with lstm recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[53] H. Shao and B.-H. Soong, "Traffic flow prediction with long short-term memory networks (lstms)," in *2016 IEEE Region 10 Conference (TENCON)*. IEEE, 2016, pp. 2986–2989.

[54] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning*, no. 34, 2017, pp. 1–5.

[55] Z. Cui, R. Ke, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," *arXiv preprint arXiv:1801.02143*, 2018.

[56] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv preprint arXiv:1703.04691*, 2017.

[57] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding," *Transportation Research Part C: Emerging*

Technologies, vol. 90, pp. 166–180, 2018.

[58] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "Lc-rnn: A deep learning model for traffic speed prediction." in *IJCAI*, 2018, pp. 3470–3476.

[59] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *arXiv preprint arXiv:1802.07007*, 2018.

[60] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: implications for previous and future research," *Knowledge and information systems*, vol. 8, no. 2, pp. 154–177, 2005.

[61] "Geotab." [Online]. Available: https://www.geotab.com/about/

[62] G. Chang and T. Ge, "Comparison of missing data imputation methods for traffic flow," in *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*. IEEE, 2011, pp. 639–642.

[63] C. F. Keras, "Github; 2015," 2017.

[64] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems, 2015," *Software available from tensorflow. org*, vol. 1, no. 2, 2015.

[65] Y. Kim, P. Wang, Y. Zhu, and L. Mihaylova, "A capsule network for traffic speed prediction in complex road networks," in *2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*. IEEE, 2018, pp. 1–6.

[66] H. Gonzalez, J. Han, Y. Ouyang, and S. Seith, "Multidimensional data mining of traffic anomalies on large-scale road networks," *Transportation research record*, vol. 2215, no. 1, pp. 75–84, 2011.

VITA AUCTORIS

NAME: Sindhuja Gutha

PLACE OF BIRTH: Hyderabad, India

YEAR OF BIRTH: 1990

EDUCATION: Narayana junior college

Hyderabad, India, 2007

JNTU University, ACE Engineering college, B.Tech in Computer Engineering

Hyderabad, India, 2011

University of Windsor, M.Sc

Windsor, ON, 2019