Design and Evaluation of a Novel Convolutional Neural Network for Short-Term Vehicle Multi-Traffic Prediction

by

Danilo Carvalho Grael

A thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science

in

Computer Science

University of Ontario Institute of Technology

Oshawa, Ontario, Canada

August 2019

Thesis Examination Information

Submitted by: **Danilo Carvalho Grael**

Thesis title: Design and Evaluation of a Novel Convolutional Neural Network for Short-Term Vehicle Multi-Traffic Prediction

An oral defense of this thesis took place on August 15, 2019 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Loutfouz Zaman
Research Supervisor	Dr. Richard Pazzi
Co-Research Supervisor	Dr. Khalil El-Khatib
Co-Research Supervisor	Dr. Abdulaziz Almehmadi
Thesis Examiner	Dr. Karthik Sankaranarayan
Examining Committee Member	Dr. Sharam Heydari

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Short-term vehicle traffic forecasting is about predicting how traffic indicators are going to be in the near future. The main traffic parameters are: traffic volume, traffic speed, and congestion state. In this thesis, we propose a convolutional neural network model that performs traffic forecasting for all three parameters, using historical integrated traffic data over a large area. The proposed model also predicts all three parameters for all 5-minute intervals from the initial time up to one hour into the future. Our proposed method was compared with the state of the art Stacked Long Short-Term Memory (S-LSTM) model, and showed 20% proportionally smaller percentage error and about 2% better recall. Our model also showed comparable results to Google Maps when employed for route travel time estimation, outperforming it in most scenarios. We concluded that our model is better than the current S-LSTM models and also its applications are comparable to established industry equivalents.

Keywords: Intelligent transportation systems; traffic forecasting; deep learning; congestion detection; estimated travel time.

Acknowledgements

I would like to thank my supervisor Dr. Richard Pazzi and my co-supervisors Dr. Khalil El-Khatib and Dr. Abdulaziz Almehmadi for their expert knowledge and great advice during my graduate studies. I also would like to thank them for their continuous encouragement and endless patience during this process.

Additionally, I would like to thank my colleagues from the Advanced Networking Technology and Security (ANTS) lab for their help and support. I would like to mention especially Mr. David Arppe for his support, Mr. Felipe Lobo for his advice, and Mr. Thiago Pereira for his help running my final experiments.

Finally, I would like to thank my family and personal friends who provided me with love and encouragement, especially my parents Mr. Wagner Grael and Mrs. Claudia Grael, my brother and sister Ms. Fernanda Grael and Mr. Diego Grael, and my grandmothers Mrs. Lucia Carvalho and Mrs. Haydee Grael.

Author's Declaration

Statement of Contributions

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

Adam Adaptive Moments.

ARIMA Auto-Regressive Moving Average.

BTMB Beijing Traffic Management Bureau.

Caltrans California Department of Transportation.

CNN Convolutional Neural Network.

CNNs Convolutional Neural Networks.

DBN Deep-Belief Network.

DBNs Deep-Belief Networks.

ETT Estimated Travel Time.

FFNN Feed-Forward Neural Network.

GPS Global Positioning System.

ITS Intelligent Transportation Systems.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

MAPE Mean Absolute Percentage Error.

MSE Mean Standard Error.

PeMS Caltrans Performance Measurement System.

RBMs Restricted Boltzmann Machines.

ReLU	Rectified Linear Unit.
RMSE	Root Mean Square Error.
RNN	Recurrent Neural Network.
RNNs	Recurrent Neural Networks.
S-LSTM	Stacked Long Short-Term Memory.
SMAPE	Symmetric Mean Absolute Percentage Error.
SVMP-CNN	Short-term Vehicle Multi-traffic Prediction CNN.

Chapter 1

Introduction

1.1 Motivation

With the recent increase in population, there is also an increase in traffic. This leads to mobility issues in urban areas due to congestion, which poses challenges to urban planing, public transport, and the mobility of emergency vehicles [1]. Intelligent Transportation Systems (ITS) is a set of applications that aim to improve the way traffic is managed. According to Junping Zhang et. al, the traffic variables such as traffic volume, traffic speed, and congestion state are the most used for those systems, especially being able to forecast them in the short-term.

In the literature, there have been many methods to predict traffic on the road. Historically, the Auto-Regressive Moving Average (ARIMA) method is used for short-term traffic forecasting [2], which is a parametric model. However, it has been proposed that non-parametric models can benefit from large amounts of data and potentially have superior performance [3]. With recent advances in hardware performance, new neural network approaches have emerged to fill this gap, since they are non-parametric.

Recently, many neural networks models have been presented to solve the traffic forecasting problem using different architectures. There is the Deep-Belief Network (DBN) type of model [4, 5], which is unsupervised learning. Then, the Recurrent Neural Network (RNN) type [6, 7] that uses time recurrence to improve its results, the Long Short-Term Memory (LSTM) type [8–15], which is a sub-type of RNN, that adds a short-term memory unit to improve its results, and the Convolutional Neural Network (CNN) type [16–21], which uses matrix convolutions to find context and structure in the data.

Each of these models predict at least one of the traffic variables, which are traffic volume, traffic speed, or congestion state. One of the issues with these models is that they only predict one of the variables, or at most two of them simultaneously, requiring extra models for the other variables. Also, none of them perform the predictions for the entirety of one-hour and over a large area of traffic. Lastly, most of them do not use large amounts of data that are available to train their models. In order to amend those issues, our proposed model is able to perform all of those tasks at the same time, and the predictions are separated into 5-minute intervals for each of the target points for one-hour.

1.2 Problem Statement

Due to the mobility problems in cities, providing short-term traffic variable prediction for ITS becomes a critical problem. This thesis addresses this issue by filling the gaps left by the other models through integrating predictions and taking advantage of the large amounts of traffic data that are available.

Our proposed model, named Short-term Vehicle Multi-traffic Prediction CNN (SVMP-CNN), is based on Convolutional Neural Networks (CNNs), which are a specific type of a Feed-Forward Neural Network (FFNN) that can be used to perform multi-traffic prediction (flow, speed, and congestion simultaneously). The predictions are for multiple points in traffic at the same time and for the entirety of one hour split into 5-minute intervals.

1.3 Contributions

The main contributions from this thesis are the following:

- A model that integrates the prediction of traffic flow, traffic speed, and congestion detection
- The proposed model is trained with recent data (from 2019) with almost 6 months of historical traffic information.
- The proposed model has higher accuracy and lower error than a model based on LSTM networks.
- The proposed model integrates the prediction for a large area, and performs the prediction simultaneously for each traffic station in that area, and for the entire short-term window of one hour
- The proposed model can be used for the estimation of travel time on specific routes and for one-hour detailed congestion detection

1.4 Thesis Organization

The following chapter of this thesis are structured as follows: Chapter 2 discusses the literature that is relevant to our proposed solution, Chapter 3 discusses the background theories that support our proposed model, Chapter 4 discusses the methodology of our model, how it is built and how it is validated, Chapter 5 discusses the experiments that support our model, and Chapter 6 discusses the conclusions, final remarks, and future works.

Chapter 2

Related Work

This chapter covers traffic prediction related to the SVMP-CNN. These are short-term traffic flow or congestion prediction models. The main factor that groups them together is that the prediction is for the short-term (one hour or less into the future) and they are based on deep learning algorithms. The following definitions (def. 2.1, 2.2, 2.3) are used in the literature to refer to the following traffic variables: traffic flow, traffic speed, traffic occupancy [4, 6–17, 17–22].

Definition 2.1 *Traffic flow (or traffic volume) is the number of vehicles that go trough a checkpoint on a road during a set period of time.*

Definition 2.2 *Traffic congestion is the state in which the average vehicle travel speed on a specific road is lower than a set threshold. This threshold is represented as a percentage of the maximum allowed speed of that road.*

Definition 2.3 *Traffic occupancy is the percentage of time that a specific checkpoint on the road is being occupied by a vehicle in a set window of time.*

This chapter is structured as follows: Section 2.2 covers the traffic data sources and structure, Section 2.3 covers the dimensions that are added to the data to improve predictions, Section 2.4 covers the external data that is added to improve predictions, Section 2.5 covers the metrics that are used to evaluate the methods, and Section 2.6 covers the base algorithms that are used to perform the predictions.

2.1 Traffic forecasting data

Traffic flow forecasting consists of predicting what the traffic flow (def. 2.1) on a specific road is going to be in the future, given the historical data from that road and, optionally, some other sources of information that could help improve the prediction, such as weather data or social media information. In general, the historical data is represented as a time series.

Additionally, traffic congestion forecasting consists of predicting if the traffic speed state of a specific road in the future is classified as congested (def. 2.2). It also uses historical data from that road and other external data to increase its prediction accuracy, similarly to traffic flow forecasting.

Typically, the data that is used for traffic flow prediction is a time series, i.e. a sequence of measurements taken as time passes, and the structure of the traffic flow on point x is displayed in Equation 2.1. This time series has a window size of n measurements, aggregated in intervals of t minutes.

$$F_x^t = [f_0, f_1, f_2, ..., f_n] \qquad \text{(Eq. 2.1)}$$

Other than traffic flow data, there is also the average traffic speed, and the traffic occupancy (def. 2.3). Those are used to perform traffic congestion forecasting and also side-by-side with traffic flow to improve traffic flow prediction and to perform traffic speed prediction.

An example of a sample day of data is displayed in Figure 2.1. The data is from

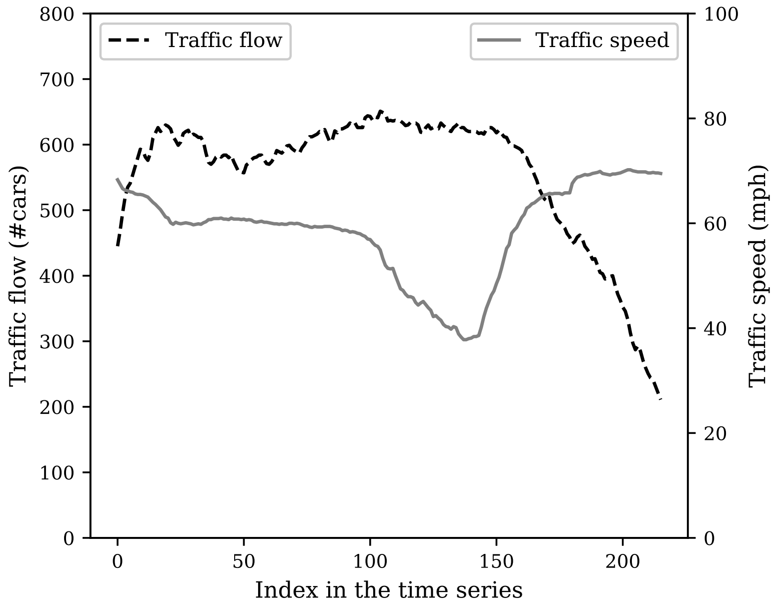


Figure 2.1: Traffic flow and speed data from January 5th, 2019

the Caltrans Performance Measurement System (PeMS) [23] dataset, and consists of 216 entries from a single day of measurements extracted from a single sensing station (district 7, from 6am to midnight). It consists of the traffic flow and the average speed measurements from the same period.

2.2 Data Set Sources

The most widely used dataset is the PeMS [23] dataset from the California Department of Transportation (Caltrans). PeMS is an online service from Caltrans that allows users to access real-time and historical traffic data, such as traffic flow, speed and occupancy, from the entire California road network.

The data is collected by road sensors, such as inductive loops, all over the road network and then stored in the PeMS. The sensors send the sampled data to the central system every 30 seconds, which then aggregates it into 5-minute intervals.

In PeMS, the road network is divided into districts. The historical data each district can be accessed directly from the system already aggregated. The readings are indexed by the ID of each station and the timestamp.

Another dataset that is used more often [5, 22, 24] is the Beijing Traffic Management Bureau (BTMB), which provides data from the Beijing road network area that is very similar to the PeMS dataset. The main difference is that the BTMB does its aggregation in 2-minute intervals.

Some other research work used other datasets that are sourced locally. For instance, the model from [17] sources its experiment data from Global Positioning System (GPS) information from taxis. They aggregate the data into road sections using specific GPS locations and generate data similar to PeMS.

2.3 Space and Time Context

As described in Section 2.1 and Equation 2.1, the data is structured in a time series and the models use it for their predictions (def. 2.4). However, there are other ways of using this data in order to provide the model with more context.

In general, the traffic on roads could be influenced by the current traffic status in nearby roads. If the time series data from those roads is provided to the model with the time series for the target road, the model could learn how they influence its traffic (def. 2.5).

In addition to that, the traffic in roads could display recurrent patterns from time to time. For models to capture that, they can add the time series for a point and also for that same point separated by a time gap, that way the model could learn the recurrent pattern (def. 2.6).

Definition 2.4 *Models that use the historical time series for the same point as it is trying to predict are using the time dimension.*

Definition 2.5 *Models that use the historical time series from nearby points to the one that it is trying to predict are using the space dimension.*

Definition 2.6 *Models that use the historical time series from the target point at day d and also day $d - q$ are using the period dimension with a q-day time gap.*

2.4 Additional Sources

On top of the time series data, some models use other external data to improve its results. One of the the elements that can influence traffic is weather, and the models from [4, 22, 24] add the weather conditions associated with the time series to improve the forecasting.

The model from [4] also used social media data, such as Twitter, to extract traffic events near its target station. The nature of the event is identified based on natural language analysis and those results are added to the series data for an improved prediction.

2.5 Evaluation Metrics

There are two main types of metrics that are used in the literature to evaluate the models: the error metrics and the classification metrics. The error metrics measure how far the predictions are from the actual result, and they are applied to traffic flow forecasting (Equations 2.2, 2.3, 2.4, 2.5, 2.6, 2.7). These metrics evaluate the distance between the forecast values and the expected values, which means that a model with lower metrics has better predictions [4, 6–11, 13–22].

The classification metrics are applied to traffic flow and congestion models that predict traffic as a general condition, instead of precise values, such as low, medium or heavy traffic (Equations 2.8, 2.9, 2.10, 2.11). These metrics evaluate how often the models assign the correct label to the input data, which means that a model with higher metrics has better predictions [12, 17].

Both error metrics and classification metrics are described in the following equations. In these equations, $y_{p,x}$ is the x$^{\text{th}}$ forecast value, $y_{a,x}$ is the x$^{\text{th}}$ expected (actual) result, $\bar{y_a}$ is the average of the expected results, N is the number of predictions, TP is the number of correct positive predictions, TN is the number of correct negative predictions, FP is the number of wrong positive predictions, and FN is the number of wrong negative predictions.

The definition of positive prediction depends on how the problem is stated, in our case, a positive prediction indicates that congestion was detected, while a negative prediction indicates that there is no congestion. The Precision score indicates the percentage of correct positive predictions among all positive predictions, this metric is useful for simultaneously evaluating that the model makes correct positive predictions and not many false positive predictions. The Recall score evaluates how many positive predictions are correct among the correct positive predictions, it differs from the Precision in the sense that it is not affected by false-positives and is used to measure positive predictions in isolation. Then, the Specificity score evaluates the rate of correct negative predictions, it is exactly like the Recall, but for negative predictions instead.

$$\text{Mean Standard Error (MSE)} = \frac{1}{N}\sum_{i=1}^{N}(y_{p,i} - y_{a,i})^2 \qquad \text{(Eq. 2.2)}$$

$$\text{Root Mean Square Error (RMSE)} = \sqrt{\text{MSE}} \tag{Eq. 2.3}$$

$$\text{Mean Absolute Error (MAE)} = \frac{1}{N}\sum_{i=1}^{N} |y_{p,i} - y_{a,i}| \tag{Eq. 2.4}$$

$$\text{Mean Absolute Percentage Error (MAPE)} = \frac{100}{N}\sum_{i=1}^{N} \frac{(y_{p,i} - y_{a,i})}{y_{a,i}} \tag{Eq. 2.5}$$

$$\text{SMAPE} = \frac{200}{N}\sum_{i=1}^{N} \frac{|y_{p,i} - y_{a,i}|}{|y_{p,i} + y_{a,i}|} \tag{Eq. 2.6}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_{p,i} - y_{a,i})^2}{\sum_{i=1}^{N}(\bar{y_a} - y_{a,i})^2} \tag{Eq. 2.7}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{Eq. 2.8}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{Eq. 2.9}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{Eq. 2.10}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{Eq. 2.11}$$

2.6 Model Categories

There are three main categories of deep learning algorithms that are used for traffic flow and congestion prediction. All of the models evaluated in this chapter either belongs to one of these categories or fall in more than one category, and are called hybrid models. The categories are: Deep-Belief Networks (DBNs), Recurrent Neural Networks (RNNs), and CNNs. Each of these categories will be introduced here and further discussed in Chapter 3.

2.6.1 Deep-Belief Networks (DBN)

A DBN is a category of deep neural networks that uses a stacked architecture of Restricted Boltzmann Machines (RBMs). It is a type of neural network that uses unsupervised learning to model statistical trends.

There are a couple of different approaches for applying the DBNs to traffic prediction. The first model [4] uses the traffic flow data coupled with social networks and weather data for its traffic flow prediction. It first uses DBN to predict traffic flow using historical data and weather data. In parallel, another network does the prediction with historical data and social media data. The result from each network is clustered into four traffic flow levels (low, medium, high, and very high). Then the result from each clustering is merged into a final prediction.

The other model [5] uses a standard DBN architecture for traffic speed prediction. It uses a DBN to extract the features and then uses a fully-connected layer to perform the speed prediction. Both DBN models evaluated are very limited in regards to space-context. They are only able to predict the traffic for one specific target point and also do not take into account nearby roads.

2.6.2 Recurrent Neural Networks (RNN) and Long Short-Term Memory Networks (LSTM)

A RNN is a category of deep neural networks that forms a recurrent connection. The output of the network is redirected as the input of the network again with the new data forming a temporal chain. This type of network architecture allows the RNN to better fit to temporal series and also to form memory.

The model described on [6] shows that RNNs can be applied successfully for short-term traffic flow prediction. The model described takes a sequence of 6 elements and uses that to make a prediction through its recurrent architecture. The main issue is that it is only done for one specific spot, and it does not take into account the overall traffic from that area, which limits its predicting power.

Another study attempts to evaluate the performance of RNNs applied in a iterative fashion [7]. The SVMP-CNN is applied to to the data and the the results are used to extend the prediction by iteratively taking the result and using it again in the model with the previous input. However, the results do not show significant performance improvements over other approaches.

LSTM networks are a specific subcategory of RNNs in which the networks have a memory element. This memory element is developed over training to be able to learn and remember the necessary data so that the network can make better predictions and take the past measurements into account for it.

The models described in [8,9] use single layer LSTM to make its predictions. This type of network feeds its input into a LSTM layer and then creates the output. This architecture is used to generate the traffic prediction.

Another model described in [10] also uses LSTM in a similar way, however it focuses more on finding out missing data rather than making future predictions. That is because they have used a main dataset that has recurrent missing data issues. Both of those models have the goal of predicting the traffic flow value.

A third model that uses single layer LSTM is described in [11], however, this model does not feed the data directly into the LSTM layer. The data is pre-processed into congestion states and that data is fed to the LSTM layer. The goal of that model is to predict the future traffic state.

Another way of using LSTM is in a stacked architecture. By feeding the output of a LSTM layer into another one before feeding the results to a final layer, it is possible to create deeper LSTM models that can work better for more complex data. The models described in [12–14] are very similar in the sense that they all use a stacked LSTM architecture at the core of their models to predict the traffic flow. However, the study done by [13] shows that on their dataset, stacked LSTM architecture did not have better results than single layer LSTM.

A unique LSTM model is described in [15], where the authors propose a stacked LSTM network that is stacked both in breadth and in depth. That allows the LSTM to sense more context before the final regression layer for traffic flow.

2.6.3 CNN

A CNN is a category of deep neural network that uses one or more convolutional layers. One of the most famous applications is the ImageNet classifier [25]. A convolutional layer applies its operation on the input and generates a feature map. The output feature map can be chained for more convolutions, or used for the model output.

The are two main types of CNN models for traffic prediction, there are the models which predict speed and the ones that predict flow. The traffic speed based models [16–18] use a window of the traffic speed time series to predict how it is going to behave in the near future.

The model in [17] uses traffic speed as its only input, while the model in [16] also uses traffic flow and occupancy to improve its accuracy. Both of them attempt to predict the actual traffic speed in future. However, the model in [17] also attempts to predict the congestion state, which indicates if the target road segment is going to be congested. The model in [18] also predicts the congestion state, but forfeits the actual traffic speed prediction. Another distinction for that model is that in its evaluation its dataset is limited to weekdays during day time (from 6:00am to 12:00pm).

Each of these models have their limitations. The model in [16] uses space-context relationships to improve its predictions, but only use segments in the same road, which fail to take into account nearby road segments that may influence its traffic but are not in that specific road. The model in [17] has limited data, they use GPS from taxis and split the results into a grid. Inside each grid segment there may be more than one road, which makes it hard to extend those results to each individual road. Also, the authors arbitrarily classify free flow traffic as over 40 km/h for any road in the entire city, which can still mean a road is congested depending on the type of road. The model in [18] takes into account space-context relationships, but limits its predictions to a single road segment, while their dataset has 614 distinct road segments.

The traffic flow models [19–21] use a window of traffic flow time series to predict it in the near future. The models in [19, 20] only use traffic flow as its input, while the model in [21] also adds the traffic flow average and mode. All of the models function similarly by creating a matrix with the traffic flow time series data for each of the target road segments and using that as the input for the convolution. The main difference is for the model in [20], which uses multiple networks to capture the space-context relationships, the other ones use the convolution itself to capture those relationships. Another distinction is that the evaluation done for model [19] limits its dataset to weekdays only.

The limitations of these traffic flow models is that some of them have limited predictions. The model in [20] only predicts for a single road segment without taking into account the space-context relationship. The model in [21] uses a very costly neural network process to group its road segments and creates a model that only works for a specific group. That model also uses trajectory data for its group selection, which is the sequence of road segments a vehicle uses to get to its destination and also is more limited than only the traffic flow data, which may result sub-optimal groups.

Another limitation of all the CNN models is the amount of data used. All of them use a maximum of 3 months of data for their experiments, with some of them using only a month of data for training their model. It is likely that using a bigger dataset could improve the results of the CNN models.

The specific details regarding each type of model is displayed in Table 2.1 and the last row in the table is of the SVMP-CNN. The metrics legends are in Table 2.2, the features legends are in Table 2.3, and the modifiers legends are in Table 2.4.

2.7 Proposed model (SVMP-CNN)

As pointed out in Table 2.1, our model is the only one that can perform all of the following, at the same time:

- Multi-traffic prediction: traffic flow, traffic speed, and congestion detection
- One-hour multi-step prediction, every 5 minutes
- Approximately 6 months of recent data (2019)
- Integrated prediction for a large area of multiple stations

Authors	Forecast	Prediction (min.)	Features	Metrics	Dataset	Stations #	Period	Modifiers	Split	Type
Liu, Qingchao *et al.* [16]	s	15,30	f,s,o	1,2,3	PeMS	12	1 month		26/5	CNN
Ma, Xiaolei *et al.* [17]	s, c	10,20	s	4, 6	GPS	236	37 days		30/7	CNN
Liao, Shijie *et al.* [19]	f	15,30,60	f	1,3,2	PeMS	100	2 months	w	1/1	CNN
Chen, Meng *et al.* [18]	c	5,10,15,30,60	t	1,2,5	CV	614	42 days	w,d	20/5	CNN
Chen, Yuanfang *et al.* [20]	f	15	f	2	England	2501	3 months		2/1	CNN
Yu, Donghai *et al.* [21]	f	12	f, fa, fm	2, 5	Custom	1323	3 months		72/18	CNN
Soua, Ridha *et al.* [4]	f	15	f, sn, we	1, 2	PeMS	1	4 months		3/1	DBN
Jia, Yuhan *et al.* [22]	s	2, 10, 30	s	1, 2, 3	Beijing	1	91 days		84/7	DBN
Qu, Jiabin *et al.* [6]	f	15	f	3, 6	PeMS	1	130 days	w	100/30	RNN
Fandango, Armando *et al.* [7]	f	15, 30, 45	f	1, 3, 6, 8	PeMS	2	6 months		5/1	RNN
Fu, Rui *et al.* [8]	f	5	f	3,6	PeMS	50	4 weeks		3/1	LSTM
Kong, Fanhui *et al.* [9]	f	5	f	2, 5	PeMS	10	X		X	LSTM
Tian, Yan *et al.* [10]	f	5, 15, 60	f	1, 2, 5	Beijing	34	274 days		4/1	LSTM
Chen, Yuan-yuan *et al.* [12]	c	30	c	7, 9, 10	Beijing	20	37 days		85/15	LSTM
Kong, Fanhui *et al.* [11]	f	5	t	2, 5	PeMS	13	3 days	d	~1/1	LSTM
Shao, Hongxin *et al.* [13]	f	5	t	2, 3	PeMS	1	31 days	w	X	LSTM
Kang, Danqing *et al.* [14]	f	5	t,s,o	1, 2, 3	PeMS	1	2 months		2/1	LSTM
Wang, Jingyuan *et al.* [15]	f	15	t	2, 3	PeMS	1	6 months		5/1	LSTM
SVMP-CNN	f, s, c	5, 10, ..., 60	fs	4, 6, 8	PeMS	82	6 months	w,d	2/1	CNN

Table 2.1: Models Summary

Index	Metric
1	MAE
2	RMSE
3	MAPE
4	Accuracy
5	MRE
6	MSE
7	F_1
8	SMAPE
9	Precision
10	Recall

Table 2.2: Metrics

Index	Feature
f	Traffic flow
s	Traffic speed
o	Road occupancy
t	Travel time
c	Congestion level
fa	Traffic flow mean
fm	Traffic flow mode
sn	Social networks
we	Weather

Table 2.3: Features

Index	Modifier
w	Week days
d	Daytime 6am-12pm

Table 2.4: Data Modifiers

The models that were evaluated either use large amounts of data or large number of stations, while the SVMP-CNN does both. Also, using recent data enables the SVMP-CNN to capture recent traffic patterns and for it to be used with current and live traffic prediction.

Therefore, our model has better predictions than the the other models described in this chapter and has more flexible usage due to its multi-traffic multi-step prediction. In the following chapters, the SVMP-CNN is compared to the S-LSTM model proposed by Kang, Danqing *et al.* [14] and also evaluated using travel time estimation to support these claims.

Chapter 3

Feed-forward Neural Networks (FFNN)

3.1 Introduction to Feed-forward Neural Networks

Feed-forward neural networks are a type of Artificial Neural Networks that can approximate a function by chaining matrix operations based on weight parameters. These parameters are used to map the input and output to the best approximation of such function. It is named feed-forward because the data propagates from the input to the output in a single direction. This behaviour is defined by a sequence of computations that, ultimately, result in the approximation of the target function [26].

The model is built as a network of approximated functions that are composite. The network uses the output of each function and propagates forward as the input for the following function to allow more flexibility to approximating a model. The functions are structured in layers, as the illustration in Figure 3.1 and the composite function can be represented as the following composite function $f(x) = f^{(x)}(f^{(h)}(f^{(y)}(x)))$.

The hierarchy in Figure 3.1 is a graphical representation of the function structure

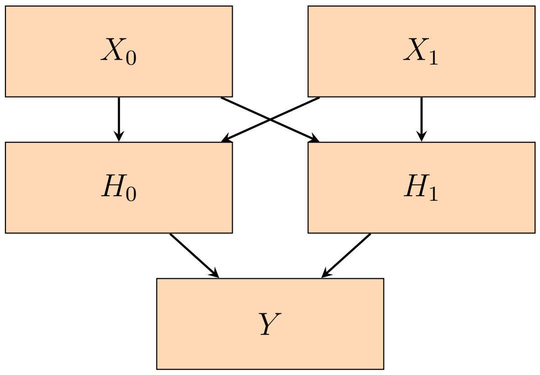


Figure 3.1: Feed-forward neural network

described in the composite function. The function $f^{(x)}$ is the input layer, the function $f^{(h)}$ is the hidden layer, the function $f^{(y)}$ is the output layer. The composition indicates the propagation direction. The depth of the network is the number of functions that are composed together, and each layer that is in between the input and output layers is a hidden layer. In the previous example, there is only one hidden layer, but there can be more than one, or none.

The reason this type of network is called neural is because the way the model works takes inspiration from the way brain cells work in neuroscience. Each node sends information based on its internal values and has multiple paths that connect inputs to outputs, as in brains.

Each function is connected to the next through hidden units. In matrix representation, the computation of the hidden layer, which is the hidden unit h, is in Equation 3.1, where W is the weight matrix of each unit and b is the bias vector. This type of hidden unit is called linear activation.

$$h = W^T x + b \tag{Eq. 3.1}$$

3.1.1 ReLU Activation

In the proposed model, the activation function that is used is the Rectified Linear Unit (ReLU) activation. It works differently than the linear activation through some simple changes.

The ReLU activation [27–29] works as the hidden unit for the neural network hidden layer. The output of this function is zero for half of its inputs (inputs between -1 and 0), which is helpful for training the network with with derivative-based optimizers, as its derivative is always going to be zero or larger, making it more flexible for optimization. The formula is displayed in Equation 3.2, where the ReLU is applied for the hidden unit activation.

$$g(x) = max(0, x), h = g(W^T x + b) \qquad \text{(Eq. 3.2)}$$

3.1.2 Sigmoid Activation

The Sigmoid activation uses the sigmoid function (Equation 3.3) to achieve binary probability distribution values, that range from 0 to 1. This activation function is typically used for output activation for binary classification problems.

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \widehat{y} = \sigma(w^T + b) \qquad \text{(Eq. 3.3)}$$

3.2 Convolutional Neural Networks

CNNs are a type of feed-forward neural networks, it works with grid-like input data and applies the mathematical operation of convolution [30]. If a feed-forward neural network has at least one convolution layer, then it is classified as a CNN.

3.2.1 Convolution and Pooling

The convolution is a smoothing operation that is based on a weight matrix that is created from a function approximation, this matrix is called the kernel. For a kernel K of size $m \times n$ and a matrix M with dimensions equal to or larger than K, the convolution operation is a simple matrix multiplication, as described in Equation 3.4.

$$C_{i,j}^{K} = C_{i,j} * K = \sum_{m} \sum_{n} M_{m+i,n+j} K_{m,n} \qquad \text{(Eq. 3.4)}$$

For a convolutional layer, each unit represents one element in the convolution input matrix, so the units are organized in a grid. The convolution is applied directly to the input for each possible grid that matches the dimensions of the kernel. This is done by applying the convolution to the grid at the first row and first column, then the kernel follows a raster scan pattern and the convolution is applied to the entire input.

The output of the convolution then goes through the hidden unit activation, which can be linear, or ReLU, or any other type of activation function. Since the raster scanning process results in a increase in units, the pooling layer is employed. The pooling layer evaluate each output grid of the convolution and selects one of the values of that grid to be the output of its hidden unit. Therefore, it reduces the dimensions of the hidden unit grid of its previous layer.

One of the methods for pooling that is used with convolutional networks is max pooling [31]. The max pooling operation does a similar process of raster scanning with a grid window, however, for every input region, it copies the output of only the unit with the maximum value, therefore performing the pooling operation.

3.3 Training

Optimization algorithms are generally used for training feed-forward neural networks [26]. The training process works with the parameters θ and a cost function $J(\theta)$. The cost function uses the θ parameters to set the weights and biases of every layer, and then calculate the average cost over the training set by applying the composite functions as matrix operations, and the using the result to calculate the cost.

The main parameter that guides the optimization algorithms is the learning rate, which it uses when changing the θ parameters. The higher the learning rate, the more dramatically the θ parameters change after every iteration.

3.3.1 Adam

One of the optimization algorithms that is used with feed-forward neural networks is Adaptive Moments (Adam) [32]. The Adam optimizer is a gradient descent method with adaptive moments, it optimizes the cost function using gradient calculation, that is then influenced by two moments that change as the training process continues. At every iteration, the parameters θ are updated based on the gradient of every iteration and the two moments.

The proposed model employs a CNNs that is optimized using Adam and two cost functions that are described on equations 4.2 and 4.3. These cost functions and the training process will be further explained on section 4.3.

Chapter 4

Short-term Vehicle Multi-traffic Prediction (SVMP-CNN)

This chapter covers the short-term traffic prediction model proposed in this thesis. This model has two main goals: to accurately predict the traffic parameters (flow and speed) and also to predict congestion. Applications of this model will be covered in Chapter 5.

The SVMP-CNN is a CNN-based model that uses historical traffic data as well as the relationship between each target point's traffic to perform traffic forecasting, which is traffic flow, speed, and congestion. The SVMP-CNN is trained using a large volume of historical traffic data and can make short-term predictions by processing a window of traffic data.

the SVMP-CNN can be divided into two main components: the data component and the CNN component. The data component is divided into data gathering and data pre-processing. The CNN component is divided into model definition, model training, and model validation.

The remainder of this chapter is structured as follows: Section 4.1.1 covers the

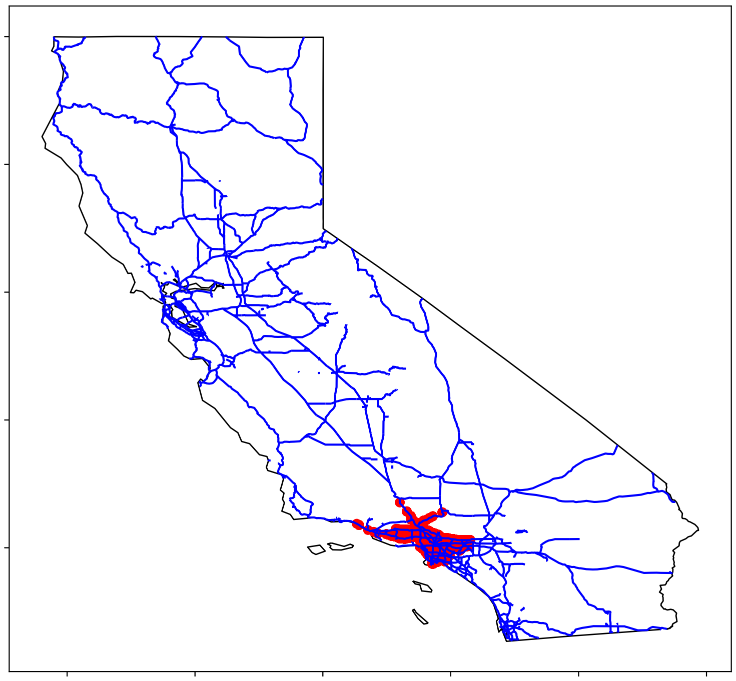


Figure 4.1: Dataset Stations - California View

data gathering process from the dataset, Section 4.1.2 covers the data pre-processing for the model, Section 4.2 covers the model architecture and definitions, Section 4.3 covers the model training process, and Section 4.4 covers the model validation process.

4.1 Data Component

4.1.1 Data gathering

The data used for our model was obtained from the PeMS dataset listed in Section 2.2. The PeMS dataset gathers its data from a large network of inductive road sensors distributed over several districts of the California road network.

The PeMS data is available in an online system upon request. The dataset used for this model is dated from January 1st to June 17th of 2019 and is located in district 7 (Figure 4.2). That area is located around the city of Los Angeles in California and its sensors are located exclusively on highways.

The data is collected every 30 seconds, but it is made available consolidated into 5-minute intervals. Each day of data consists of 288 entries starting at 12:00am and ending 11:55pm of the same day. The data is collected for each individual traffic sensing station. All the stations for district 7 are displayed in Figures 4.1, 4.2, and 4.3. The first figure shows the stations from district 7 in a state-wide view, the second figure shows a close-up of the same stations, and the third figure shows a crop of the dataset on the target area of the Los Angeles centre.

District 7 has 4859 sensing stations and only 2737 of them have complete data for the analysis period. Out of those, there are 1886 stations that are located on the actual roads.

To select the target stations for the model, all stations further than 22km from the center of Los Angeles were excluded. This radius was chosen to limit the model to the central Los Angeles area and include the commute routes from the neighbour cities. Also, stations that are closer than 4.4km of each other were excluded, unless their traffic flows in opposite direction. This process was done to make sure traffic stations have more spacing between them to avoid redundant data. This resulted in 82 target stations to be used for the mode, that are highlighted in Figure 4.4. Each

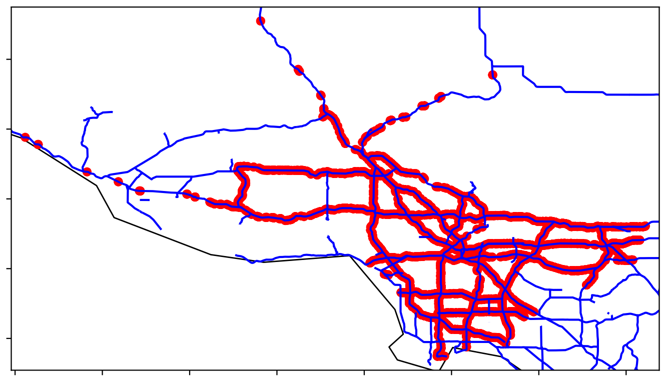


Figure 4.2: Dataset Stations - District 7 View

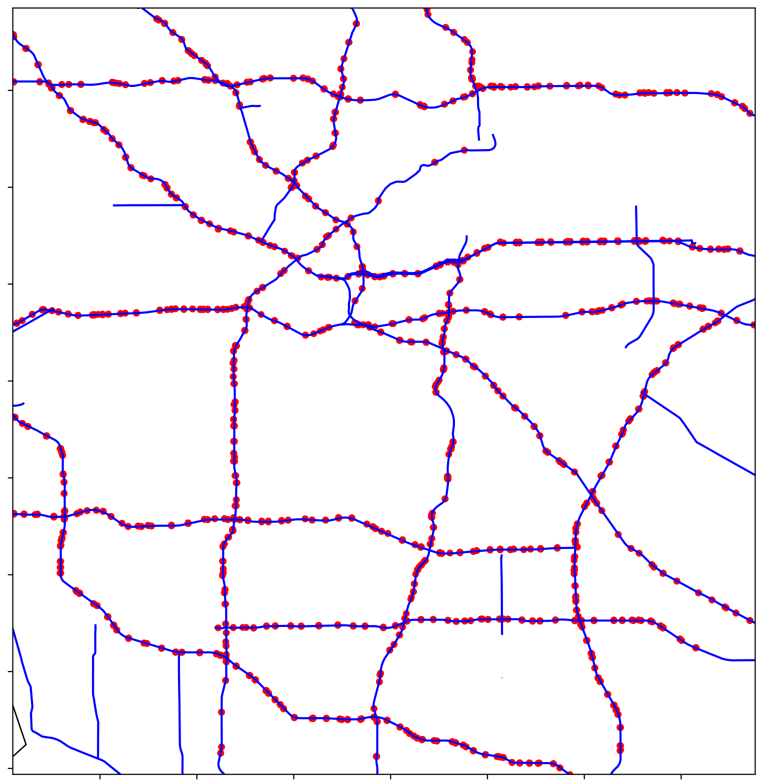


Figure 4.3: Dataset Stations - Target Area View

target station has its index and the traffic flow direction (North, East, West, and South).

The historic data is available in text files, in with each file having one day of data. Each file is composed of multiple lines and each line has the data for one station at one of the 5-minute aggregation intervals, totaling 288 lines for each station. This data includes traffic direction, average traffic flow, traffic speed, among other data.

Each text file is parsed and only the lines with data from one of the target stations are kept. These lines are then used to create the source matrix $A^d_{m\times\ n}$, where $m = n_{timestamps}$ and $n = n_{stations}$ and d is the index of the day that the file represents. Each element $a^d_{i,j}$ has a pair of values $(f^d_{i,j}, s^d_{i,j})$, in which $f^d_{i,j}$ is the traffic flow and $s^d_{i,j}$ is the traffic speed for timestamp i, target station j, and day d.

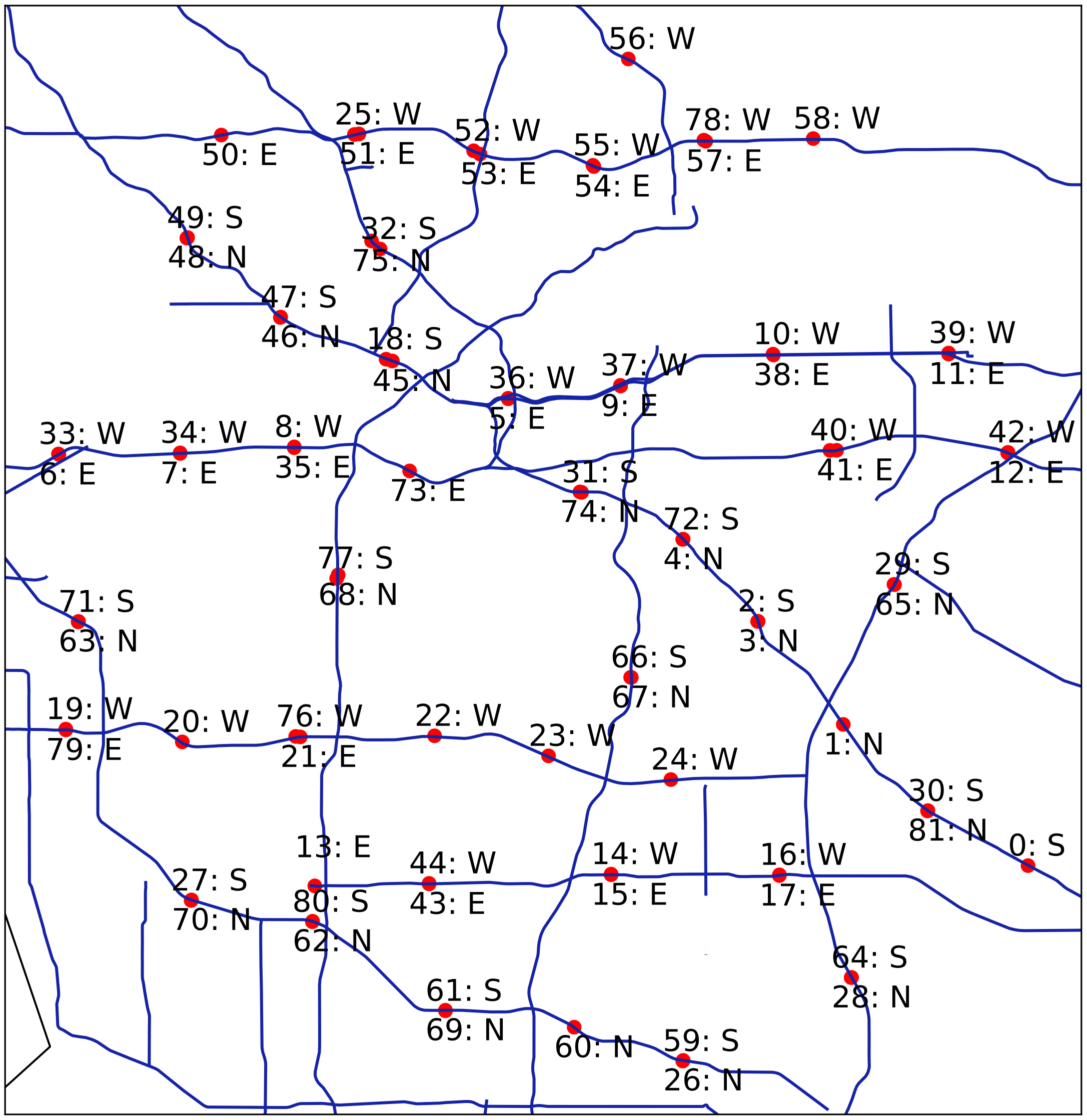


Figure 4.4: Target Stations

After all matrices A^d are created for every single day, they are filtered. All matrices that are created from weekends are discarded. And each matrix is filtered so the data related to the times of the day from 12:00am to 6:00am are discarded, leaving each day with 216 timestamps. After that is done, the matrices are sent for pre-processing. Discarding the data from 12:00am to 06:00am and also discarding weekend data is a common practice for traffic prediction [6, 13, 18, 19], since the traffic patterns are much different at those times.

4.1.2 Data pre-processing

The data at this step is basically raw values of traffic flow, which can range from around 0 to more than a thousand, and traffic speed which can range between 0 and a hundred. Since the two types of input are in different orders of magnitude, the data has to be re-scaled. After that, since the SVMP-CNN is CNN-based, the data has to be re-organized in higher dimension matrices so it can be used as an input for the model. So, there are five steps that are taken for the pre-processing of the input data: scaling, window generation, shuffling, congestion detection, and train/test splitting.

The first step is the scaling, and for that the min-max scaling process is used. Let U be the set of the matrices A^d for every day d in the dataset, for all matrices in the set U, f_{max} and s_{max} are respectively the maximum values for traffic flow and traffic speed. The minimum values for the dataset are set as 0, since that is the minimum speed possible and also the minimum traffic flow possible. Then, every value of traffic flow in the set U is divided by f_{max} and every value of traffic speed is divided by s_{max}. That will result in values ranging from 0 to 1.0.

The next step is the window generation. For every matrix A^d in U, a sliding window is created, which will generate multiple samples. For this process there are two main parameters: w and v (which will be described in Section 4.2), where w is the input window size and v is the output window size. Starting at timestamp t, the sample S_t^d is created as described in the Equation 4.1 with $t = \{1, 2, ..., (n_{timestamps} - w - v)\}$. The S_t^d matrix has $(w + v)$ columns and $n_{stations}$ lines. The timestamp t is increased by one to slide the window to the next set of values up until the end is reached.

$$S_t^d = \{a_{t,j}^d, a_{t+1,j}^d, ..., a_{w,j}^d, a_{w+1,j}^d, ..., a_{w+v,j}^d\} \qquad \text{(Eq. 4.1)}$$

Then, the matrices S_t^d are split into $S_t^d = \mid X_t^d \mid Y_t^d \mid$ such that X_t^d are the first w columns of S_t^d and Y_t^d are the last v columns of S_t^d. All matrices X_t^d are put into the set $X^{(f)}$ and the matrices Y_t^d are put into the set $Y^{(f)}$.

The next step is the data shuffling. Let $n_{samples}$ be the number of samples inside the set $X^{(f)}$ and $I_{samples}$ be the indexes of each sample in the set $X^{(f)}$. A random permutation $R_{samples}$ is a random shuffling of the indices in $I_{samples}$. Using the permutation $R_{samples}$, the elements in the sets $X^{(f)}$ and $Y^{(f)}$ are re-ordered following the indexes in the permutation.

By using the same permutation for both sets we assure that the elements from $X^{(f)}$ are in the same index as their correspondents from $Y^{(f)}$ after the process is completed. The resulting sets are named $X^{(s)}$ and $Y^{(s)}$ and are respectively the first input and the first output of the model.

After the data is shuffled, the second input and output pair can be created. This step is the congestion detection and it has only one parameter: the congestion threshold c_t (which will be detailed in Section 4.2).

The congestion detection process takes each traffic flow and speed pair and if the traffic speed is less than or equal to the average speed (calculated using all instances of traffic speed for that station) of that road, then the congestion value is set as 1.0, otherwise the value is set as 0.0. The congestion detection process is applied to the last u (which will be detailed in Section 4.2) columns of the samples in the set $X^{(s)}$ and all columns of the set $Y^{(s)}$ and the result is the second input $C^{(s)}$ and the second output $L^{(s)}$.

Finally, the sets are divided into train and test. The first 70% samples make the training sets $X^{(s)}_{train}$, $Y^{(s)}_{train}$, $C^{(s)}_{train}$, and $L^{(s)}_{train}$ and the remaining 30% make the test sets $X^{(s)}_{test}$, $Y^{(s)}_{test}$, $C^{(s)}_{test}$, and $L^{(s)}_{test}$.

4.2 Model definition

The model created is a CNN structured with 2D Convolution layers and Dense layers. The main parameters for the model are the past window size (w), the future window prediction size (v), and the past window size for congestion (u), which has to be smaller than or equal to w. These parameters are the number of time steps that are used for the mode and, since the data is divided into 5-minute intervals, each parameter can be multiplied by 5 to find the number of minutes that it actually represents.

Initially, the parameters were defined as $u = 4$, $v = 12$, and $w = 9$, which means that our model uses the data from the past 45 minutes (9×5) to predict how the traffic is going to behave for the following hour (12×5) and our second input uses the congestion state for the past 20 minutes (4×5). The reasons for $w = 9$ choice will be further explained in the validating section (Section 4.4).

The value v was defined as 12 in order to cover the period of one hour, which is usually the furthest into the future that is still considered short-term. Also, the congestion parameter was set at $c_t = 0.5$, which means that we classify a road as congested if its current speed is below 50% of its average speed [18].

The model is divided in two parts: the convolution part and the classification part. The convolution part processes the first input of the model ($X^{(s)}$) and does the traffic flow and traffic speed prediction. The second part of the model uses the output from the first part and the second input of the model ($C^{(s)}$) to predict the traffic congestion state.

The model processes one input at a time, and in order to explain the functionality of each layer, one instance of each input will be used. For the first input we have one sample X_i from the set $X^{(s)}$ and another sample C_i from the set $C^{(s)}$. As explained in Section 4.2, the matrix X_i is $9 \times 82 \times 2$ and the matrix C_i is 4×82.

The functionality of each layer on the first part of the model is as follows:

- Main Input Layer: This layer feeds the input X_i to the network. It starts with dimensions $9 \times 82 \times 2$ (9 rows, 82 columns, and 2 features).

- Reshape 1 Layer: This layer takes the input from the Main Input layer and reshapes it to $9 \times 164 \times 1$

- 2d Convolution 1 Layer: This layer takes the reshaped input and applies a 2d convolution that creates 32 filters with a 3×3 convolution size. This results in a matrix that is $7 \times 162 \times 32$.

- 2d Convolution 2 Layer: This layer takes the output of the first 2d convolution layer and applies another convolution that creates 96 filters with a 3×3 convolution size. This results in a matrix that is $5 \times 160 \times 96$.

- 2d Max Pooling Layer: This layer takes the output of the second 2d convolution layer and applies a 2×2 max pooling operation. The output of this layer is a matrix with size $2 \times 80 \times 96$.

- Dropout 1 Layer: This layer takes the output of the 2d max pooling layer and randomly sets values to zero with 25% chance during training to avoid overfitting. The input size $2 \times 80 \times 96$ is unchanged after this.

- Flatten Layer: The flatten layer takes the output of the previous Dropout layer and simply flattens the input into a single vector with size 15360. This is the convolution feature list.

- Dense 1 Layer: This layer takes the output of the flatten layer and feeds it to 640 dense neurons. The output is size 640.

- Dropout 2 Layer: This layer takes the output of the Dense 1 layer and randomly sets values to zero with 50% chance during training. The output size remains 640.

- Main Output: This layer takes the output of the Dropout 2 layer and feeds it to a dense output of size 1968 ($12 \times 82 \times 2$).

For the second part of the model the layers will be detailed in the following list:

- Second Input Layer: This layer feeds the input C_i into the network. It starts flattened with size 328 (4×82).

- Reshape 2 Layer: This layer takes the Main Input and flattens it into a vector with size 1476.

- Concatenate Layer: This layer takes the output of the Reshape 2, the Main Output, and the Second input and concatenates them into a single large vector of size 3772.

- Dense 2 Layer: This layer takes the output from the Concatenate layer and feeds it to 256 dense neurons. The output size is 256.

- Second Output Layer: This layer takes the output of the Dense 2 layer and feeds it to a dense output of size 984 (12×82).

The activation function is the ReLU (Section 3.1.1) for all layers in the first and second parts of the model, with the exception of the Second Output layer, which uses Sigmoid activation (Section 3.1.2). The layer architecture and connections of the model is displayed in Figure 4.5.

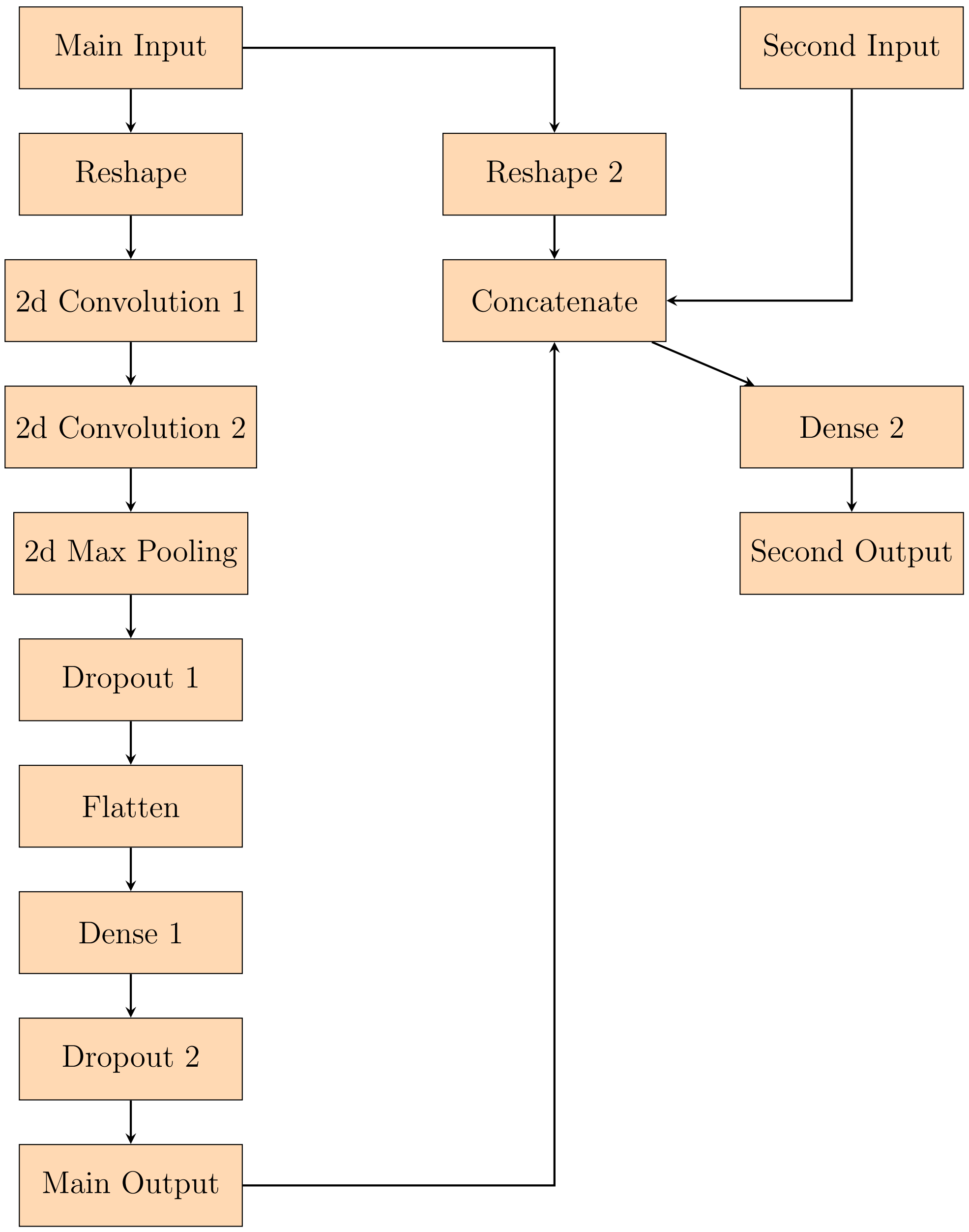


Figure 4.5: Model Layer Architecture

4.3 Model Training

With the model architecture setup, the model can now be trained with the dataset. For this process, the Adam optimizer is used (Section 3.3.1). This optimizer is an iterative algorithm that takes two main parameters for training: the learning rate

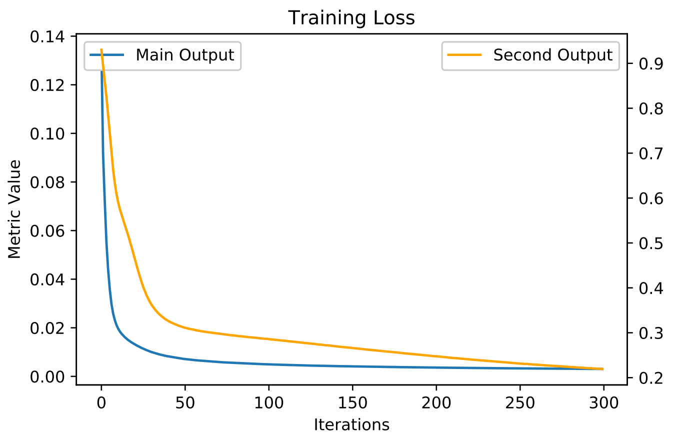


Figure 4.6: Training process

and the decay. The learning rate defines how the optimizer updates the model for every iteration and the decay is simply how much the learning rate reduces after each iteration.

In order for the optimizer to evaluate the current performance of the model, before updating the models weights, it uses the results from loss metrics. The loss metrics are applied for each output of the model after every iteration done by the algorithm. The metric chosen for the Main Output and the Second Output respectively are: Mean Squared Logarithmic Error (Equation 4.2) and Weighted Binary Crossentropy (Equation 4.3). This process is illustrated in Figure 4.6, which shows how the loss changes as the model is trained.

For the following equations, $y_{a,x}$ is the xth actual value, $y_{p,x}$ is the xth predicted value, and h_w is the weight used for the binary crossentropy. The value chosen for h_w is $h_w = 6.35$. The reason that a weight is employed is because for every sample in the dataset that is classified as congested there are 6.35 other samples that are not. The weight is applied during training to offset this class imbalance.

$$MSLE = \frac{1}{N}\sum_{i=1}^{N}(log(y_{p,i}+1) - log(y_{a,i}+1))^2 \qquad \text{(Eq. 4.2)}$$

$$WBCE = -\frac{1}{N}\sum_{i=1}^{N}(y_{a,i} \times log(y_{p,i}) \times h_w + (1-y_{a,i}) \times log(1-y_{p,i})) \qquad \text{(Eq. 4.3)}$$

4.4 Model Validation

After the model is trained with the dataset training sets, the model can be evaluated with the test sets, which were not used for training. For the Main Output of the model the following metrics are used: SMAPE (Equation 2.6) and MSE (Equation 2.2). These are error metrics, and they are applied to the traffic flow and traffic speed predictions.

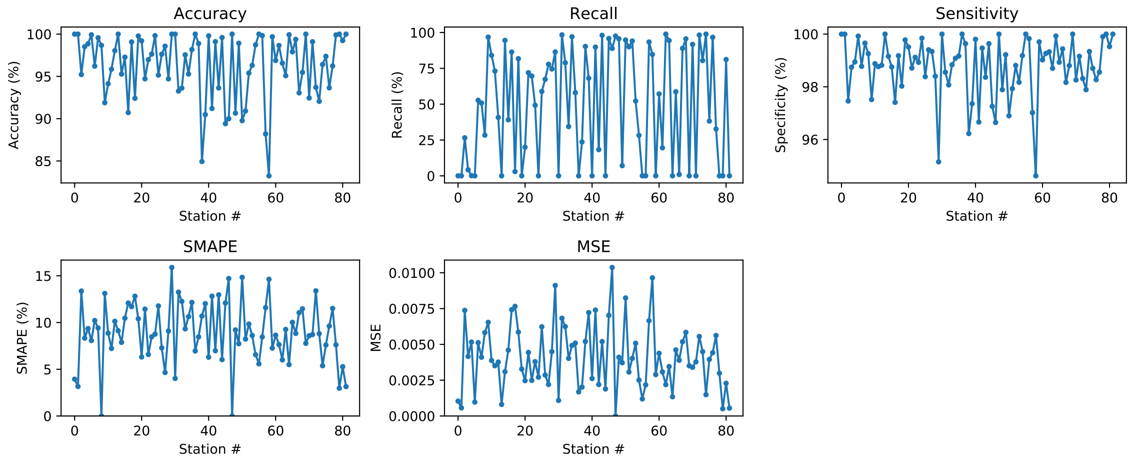


Figure 4.7: Performance by Station

Then, for the Second Output of the model the following metrics are applied: Accuracy, Recall (Equation 2.9), and Specificity (Equation 2.10). Those are classification

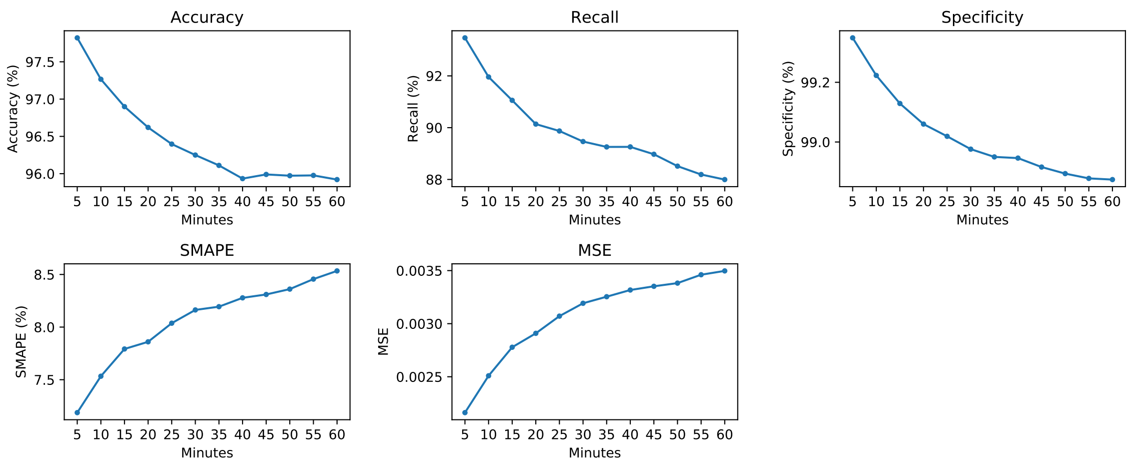


Figure 4.8: Performance by prediction time

metrics, the accuracy measures the rate of correct predictions by the model, the recall metric measures the true positive rate, and the specificity measures the true negative rate. In the dataset, the positive samples are the congestion samples.

The first step in evaluating the model is choosing the best value for w, which is the window size for the Main Input. The model was trained for 300 iterations for each different value of w and the metrics were applied. The values evaluated were $w = 6, 9, 12$. The minimum value for the w variable is 6 (30 minutes) because of the 2d convolutions. Then we evaluated the next two increments of 3 (15 minutes) up until 12 (one hour). For this test the v value was 12 (one hour) and u value was 4 (20 minutes).

Parameter w	MSE ($\times 10^{-3}$)	SMAPE	Accuracy	Recall	Specificity
6	4.410	9.67%	95.4%	87.9%	98.8%
9	4.267	9.46%	95.8%	87.8%	98.8%
12	4.381	9.80%	95.5%	88.5%	98.9%

Table 4.1: w evaluation

The results from this evaluation are displayed in Table 4.1. The metrics show that by choosing $w = 9$ the model has lower MSE and SMAPE and also higher accuracy than compared to the other options. Therefore $w = 9$ was used for all further experiments and validation.

After comparing the different values for w, the metrics were applied separately for each individual target station using the test data. The results are displayed in Figure 4.7. The metrics show that there is not a large variation on each individual station, with a few exceptions. The recall metric is 0% for some stations, and that is because there are a few stations on the dataset that are never congested over the period of validation, which drive that metric to zero for those specific cases.

Prediction w (min.)	MSE ($\times 10^{-3}$)	SMAPE	Accuracy	Recall	Specificity
5	2.162	7.19%	97.8%	93.5%	93.5%
10	2.509	7.53%	97.3%	92.0%	92.0%
15	2.778	7.79%	96.9%	91.1%	91.1%
20	2.909	7.86%	96.6%	90.1%	90.1%
25	3.071	8.04%	96.4%	89.9%	89.9%
30	3.192	8.16%	96.2%	89.5%	89.5%
35	3.254	8.19%	96.1%	89.3%	89.3%
40	3.317	8.28%	95.9%	89.3%	89.3%
45	3.352	8.31%	96.0%	89.0%	89.0%
50	3.382	8.36%	96.0%	88.5%	88.5%
55	3.461	8.46%	96.0%	88.2%	88.2%
60	3.497	8.53%	95.9%	88.0%	88.0%

Table 4.2: Prediction over time

Also, the model was evaluated for every future prediction separately (5, 10, 15, ..., 55, 60 minutes), the results are in Table 4.2 and in Figure 4.8. The results show a decrease in metrics quality as the predictions start to be further into the future. For the next chapter, those metrics and validation process are used to compare the model with another one from the literature.

Chapter 5

Performance Evaluation

This chapter covers the experiments that were performed in order to evaluate the performance of the SVMP-CNN. The model is compared to the Stacked LSTM model, which is the most common type of model from the literature (Chapter 2) in Section 5.1. Afterwards, the model is used in the analysis of congestion focal points through heat maps in Section 5.2. Then, the model is used to calculate the Estimated Travel Time (ETT) for specific routes in Section 5.3. For this evaluation, the model ETT is compared with the actual route travel time and with the ETT calculated by Google Maps. Finally, the conclusions are presented in Section 5.4.

5.1 SVMP-CNN vs. Stacked LSTM

The first evaluation compares the SVMP-CNN with a S-LSTM model. There are many such models described in Chapter 2, and the model chosen for this comparison is a 2-layer S-LSTM model proposed by Kang, Danqing *et al.* [14]. This model was chosen because it is the most similar to the SVMP-CNN, since it uses nearby traffic data combined with historical data to perform its predictions.

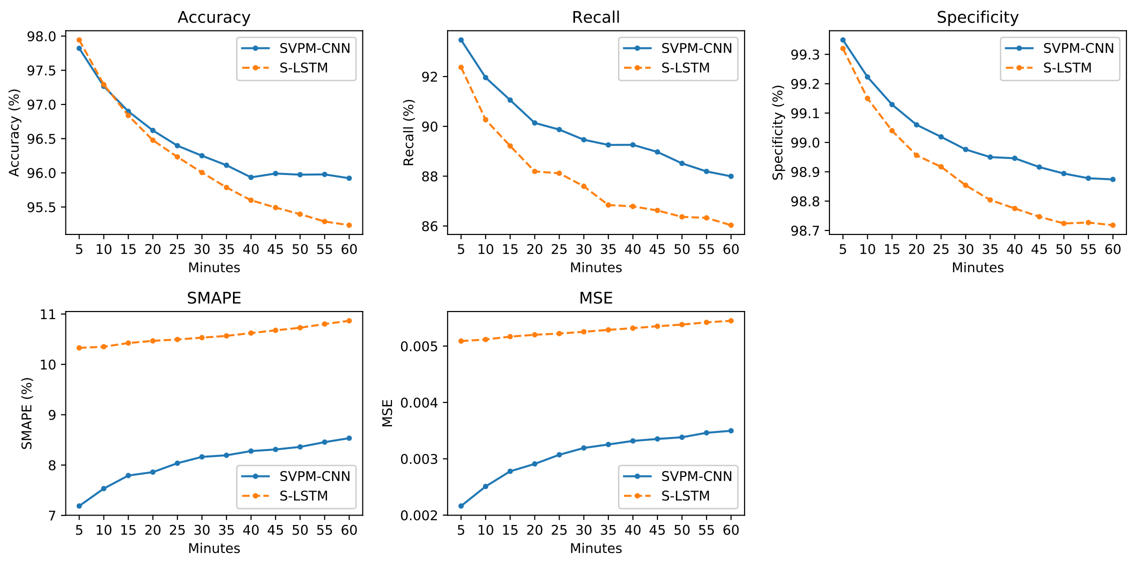


Figure 5.1: S-LSTM metrics over time

For this experiment, both models were trained using the exact same dataset from PeMS district 7 used in Section 4.1.1 from January 1st to June 17th, 2019 (excluding weekends and data from 12:00am to 6:00am), split into training and testing in the same way. The models were trained using 300 iterations with the same learning rate using Adam optimizer.

The results are shown in Table 5.1 and in Figure 5.1. The table displays the model metrics for every 5-minute interval for the future prediction, up to 1 hour. The metrics are in the same scale as the metrics in Table 4.2 in the previous chapter, so they can be compared directly.

Figure 5.1 shows the results side by side (our model vs. the S-LSTM model). The metrics show that, with the exception of the congestion accuracy for 5-minute predictions, the SVMP-CNN outperforms S-LSTM with lower error metrics and higher accuracy, recall and sensitivity metrics from 5 minutes up until one hour. Even though the accuracy for the SVMP-CNN is worse for the 5-minute prediction, the recall metric and the specificity metric shows that the SVMP-CNN makes less mistakes for

Prediction w (min.)	5	10	15	20	25	30	35	40	45	50	55	60
MSE SVPM-CNN	2.162	2.509	2.778	2.909	3.071	3.192	3.254	3.317	3.352	3.382	3.461	3.497
MSE S-LSTM	5.089	5.117	5.168	5.201	5.222	5.254	5.288	5.318	5.352	5.382	5.421	5.449
SMAPE (%) SVPM-CNN	7.19	7.53	7.79	7.86	8.04	8.16	8.19	8.28	8.31	8.36	8.46	8.53
SMAPE (%) S-LSTM	10.33	10.35	10.42	10.47	10.49	10.53	10.57	10.62	10.68	10.73	10.80	10.87
Accuracy (%) SVPM-CNN	97.8	97.3	96.9	96.6	96.4	96.2	96.1	95.9	96.0	96.0	96.0	95.9
Accuracy (%) S-LSTM	97.9	97.3	96.8	96.5	96.2	96.0	95.8	95.6	95.5	95.4	95.3	95.2
Recall (%) S-SVPM-CNN	93.5	92.0	91.1	90.1	89.9	89.5	89.3	89.3	89.0	88.5	88.2	88.0
Recall (%) S-LSTM	92.4	90.3	89.2	88.2	88.1	87.6	86.8	86.8	86.6	86.4	86.3	86.0
Specificity (%) SVPM-CNN	99.3	99.2	99.1	99.1	99.0	99.0	99.0	98.9	98.9	98.9	98.9	98.9
Specificity (%) S-LSTM	99.3	99.2	99.0	99.0	98.9	98.9	98.8	98.8	98.7	98.7	98.7	98.7

Table 5.1: S-LSTM comparison over time

detecting actual congestion and also detecting non-congestion.

Therefore, we can conclude that the SVMP-CNN performs better than the S-LSTM model for the dataset that was used for training. Even though S-LSTMs are more used for time series forecasting, our model performs better because our CNN architecture is good at capturing contextual features. That is because the traffic on each road influences the traffic on other roads, and the SVMP-CNN is able to learn those context relationships better than a S-LSTM model.

5.2 Congestion Focal Points

As described in Section 4.2 of the previous chapter, the second output of the SVMP-CNN is a class prediction of whether or not each target station is going to be congested in the short-term. However, before the final output of the class, the model actually performs a likelihood calculation, and if that exceeds a 50% threshold, then that target station is classified as congested.

It is possible to identify each target station and give it a score from 0% to 100% of how confident the model is that each place is going to be congested in the future with the model output before applying the threshold. That is displayed in Figure 5.2, which shows a heatmap of congestion likelihood calculated from the model. Each traffic station is displayed, and each of them have a size and shade based on the congestion confidence scale on the right-hand size depending on how close or far they are from the 100% mark.

In the first column there is the actual congestion at the time, and in the three following columns we have the likelihood plot for 15, 30, and 45 minutes into the future. Each target station size is proportional to the likelihood with the largest size possible being for 100%. This information can be used to make route suggestions in the short-term, such as choosing a route that can avoid areas that have a high score for congestion in the near future.

For instance, if a car that is attempting to reach a point that could take more than 15 minutes to drive, it may choose its route based on the congestion likelihood of each route. In order to do that, the routing system has to find all potential routes to reach the destination, and then drop the ones with with higher congestion confidence.

5.3 Travel Time

For the travel time experiment, the SVMP-CNN is used for estimating the travel time for several routes. The result is compared with our baseline and the results from Google Maps.

5.3.1 Baseline Travel Time

The baseline benchmark used for comparison is calculated using historical data from the PeMS dataset. The data does not have to be the same exact data used for evaluating the model. The travel time is computed by segment, using the average speed for each traffic station in its route. The data from each day is available on the following day, so the baseline can only be calculated on the next day after collecting the data.

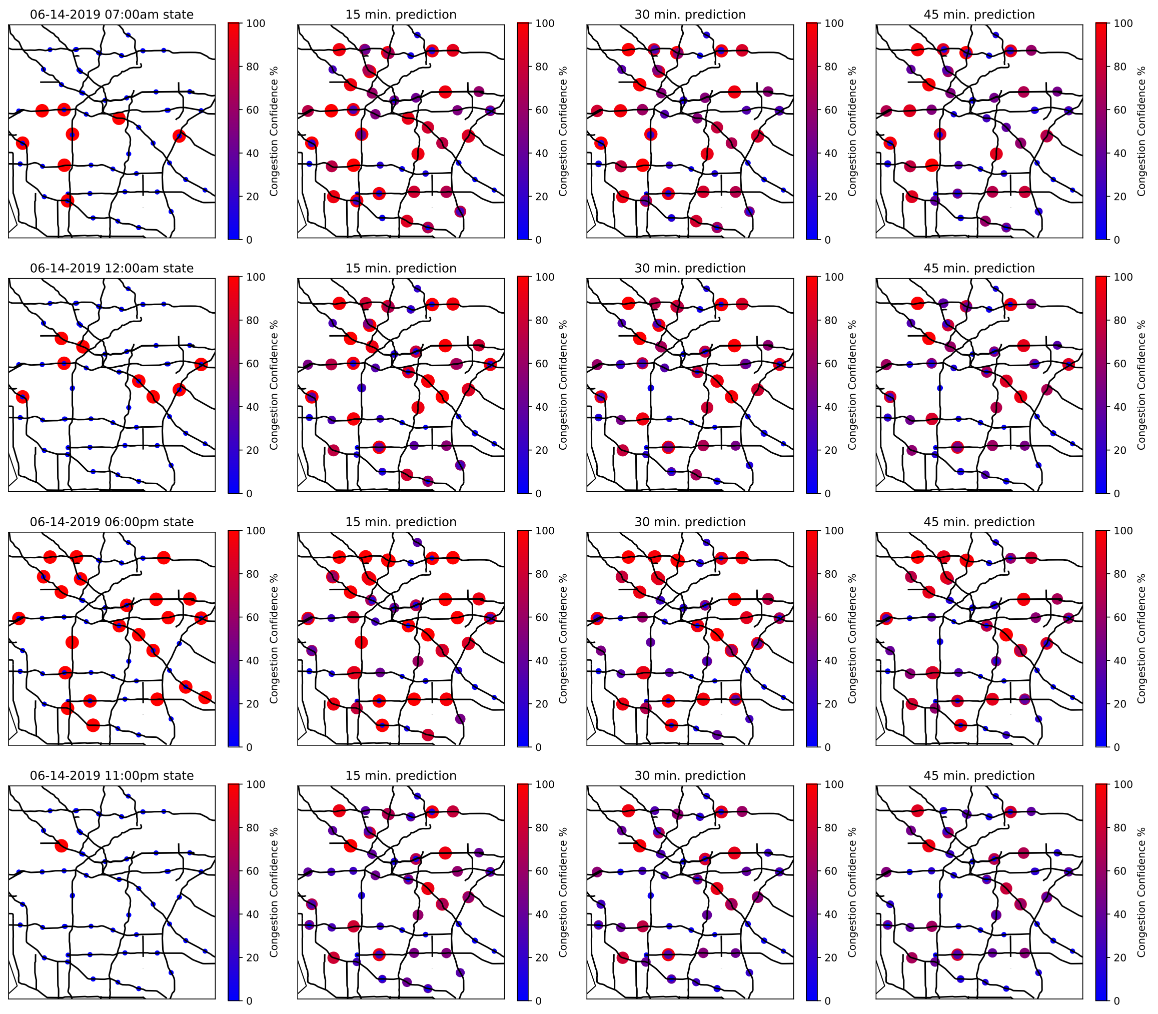


Figure 5.2: Congestion heatmaps

The first step is computing the travel time between two traffic stations TS_i and TS_j. Let $D_{i,j}$ be the road distance departing from TS_i and arriving at TS_j and V_i be the average traffic speed at TS_i. The time $T_{i,j}$ required to go from TS_i to TS_j is displayed in Equation 5.1.

$$T_{i,j} = \frac{D_{i,j} \times 3600}{V_i} \qquad \text{(Eq. 5.1)}$$

For our target dataset, the speed is measured in miles per hour, and $M_{i,j}$ is measured in seconds. The measurement V_i can be obtained by looking at the dataset directly with the proper index. As described in Section 4.1.1, each day of data has 288 entries for each target station.

Let the time $h : m$ be the target time of day, where h represent the hour from 0 to 23 and m is the minute from 0 to 59. The index d_{idx} for retrieving the traffic speed at this time is described in Equation 5.2. The floor function computes the largest integer that is less than or equal to its argument. The result ranges between 0 and 287.

$$d_{idx} = floor(h \times 12 + floor(\frac{m}{5})) \qquad \text{(Eq. 5.2)}$$

The next step is applying the same method for an entire route. Let the time $h_0 : m_0$ be the target time of day and the route R be the a list of target stations to be followed in order, such as $R = [T_0, T_1, .., T_f]$ where f is the number of traffic stations in the route. Starting at the first segment from T_0 to T_1, the travel time $M_{0,1}$ is calculated using the target time $h_0 : m_0$.

Then, the result is added to the target time to be used for the next segment, as follows $(h_1, m_1) = (h_0, m_0) + M_{0,1}$. This process is repeated for the next segment until the last one is processed. The final time of arrival is (h_{f-1}, m_{f-1}), which is the resulting time from the final segment. The result (h_{f-1}, m_{f-1}) is subtracted from the starting time (h_0, m_0) to compute the travel time in minutes T_R, as described in the algorithm 5.1. The algorithm uses two auxiliary functions, the *addSeconds* function adds a set number of seconds to a pair of hour and minute and *subtractTime* subtracts two time stamps in hour-minute format.

The inputs for the following algorithm are: the start time (h_0, m_0), the target route R with size f, the table of traffic speeds for the day S, and the segment road distances D. The output is the travel time in minutes T_R.

Algorithm 5.1 Travel time algorithm

```
1: Input: $(h_0, m_0)$, $R$, $f$, $S$, $D$
2: Output: $T_R$
3: $(h, m) = (h_0, m_0)$
4: i = 0
5: while $i < f - 1$ do
6:     d = floor(h * 12 + floor(m))
7:     V = S[d]
8:     M = ($D_{R_i,R_{i+1}}$ * 3600) / V
9:     $(h, m)$ = addSeconds(h, m, M)
10:    i = i + 1
11: $(h, m)$ = subtractTime($(h, m)$, $(h_0, m_0)$)
12: return h * 60 + m
```

5.3.2 Selected Routes

One of the key elements to calculating the travel time is the actual route, i.e. the sequence of target stations to be followed. For that purpose, six different routes were selected, each route has at least 4, up to 5, target stations. The routes are displayed over the target area in Figure 5.3. The traffic stations that are part of each route are displayed with a larger marker and each of them have its starting point labeled as 1, and its final point labeled as 4 for routes A, B, C, E, and F, or 5 for route D.

These routes were chosen because they connect nearby cities to the centre of Los Angeles, therefore are common commute routes which may display patterns of heavy traffic during rush hours. The target times for the evaluation were chosen spread out over the day to observe different recurrent traffic patterns, the values are: 7:00am, 12:00pm, 6:00pm, and 10:00pm.

5.3.3 SVPM-CNN ETT vs. Baseline

With the baseline computed, it is possible to compare it with the SVMP-CNN over the target routes. Instead of using the historical data for traffic speed, it has to be

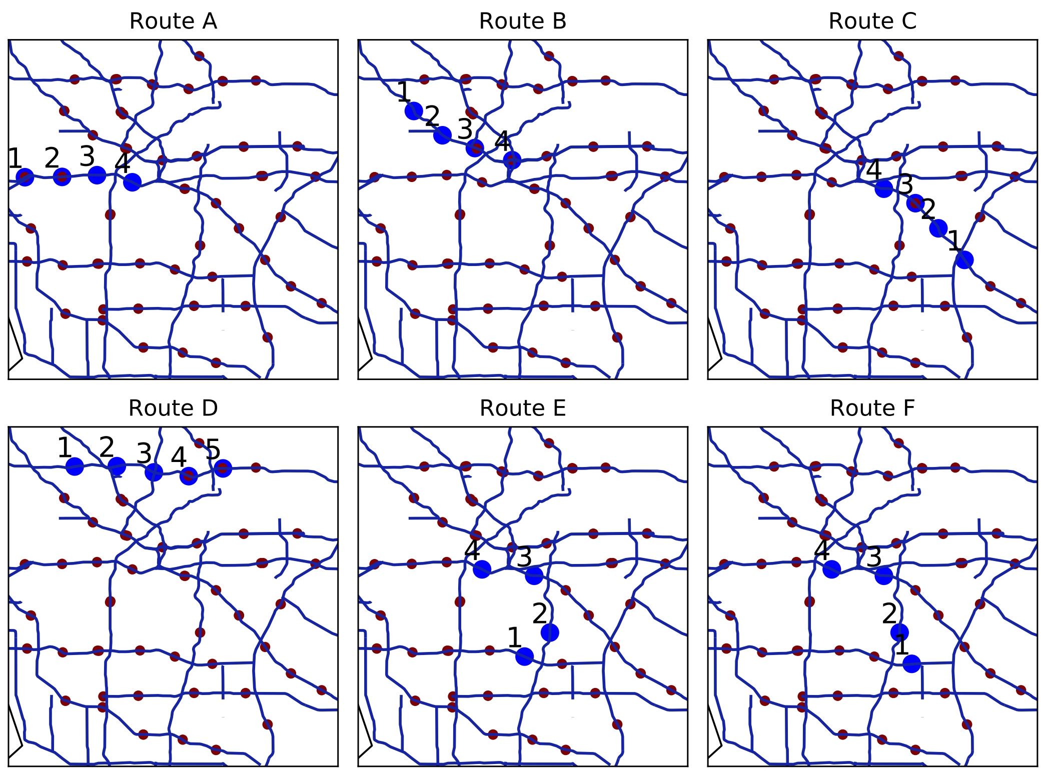


Figure 5.3: Selected routes

forecast from the current traffic speeds up until the target time. For that, the data needs to be processed using the same procedure that was applied for preparing the dataset for training the model.

Using the same steps for the baseline travel time described in Algorithm 5.1, it is possible to replace the speed table for the current day with the historical data up until the target start time appended with the one hour of traffic predicted by the SVMP-CNN. That way, the output of the algorithm will be the ETT based on those predictions.

The first results are shown separated by route over time (Figure 5.4) and also separated by time over the routes (Figures 5.5, 5.6, and 5.7). The Table 5.2 has the indexes for the timestamps in Figure 5.4 for readability.

Index	Time	Index	Time
1	07-1-2019 07:00	2	07-1-2019 12:00
3	07-1-2019 18:00	4	07-1-2019 22:00
5	07-2-2019 07:00	6	07-2-2019 12:00
7	07-2-2019 18:00	8	07-2-2019 22:00
9	07-3-2019 07:00	10	07-3-2019 12:00
11	07-3-2019 18:00	12	07-3-2019 22:00
13	07-4-2019 07:00	14	07-4-2019 12:00
15	07-4-2019 18:00	16	07-4-2019 22:00
17	07-5-2019 07:00	18	07-5-2019 12:00
19	07-5-2019 18:00	20	07-5-2019 22:00

Table 5.2: Time index table

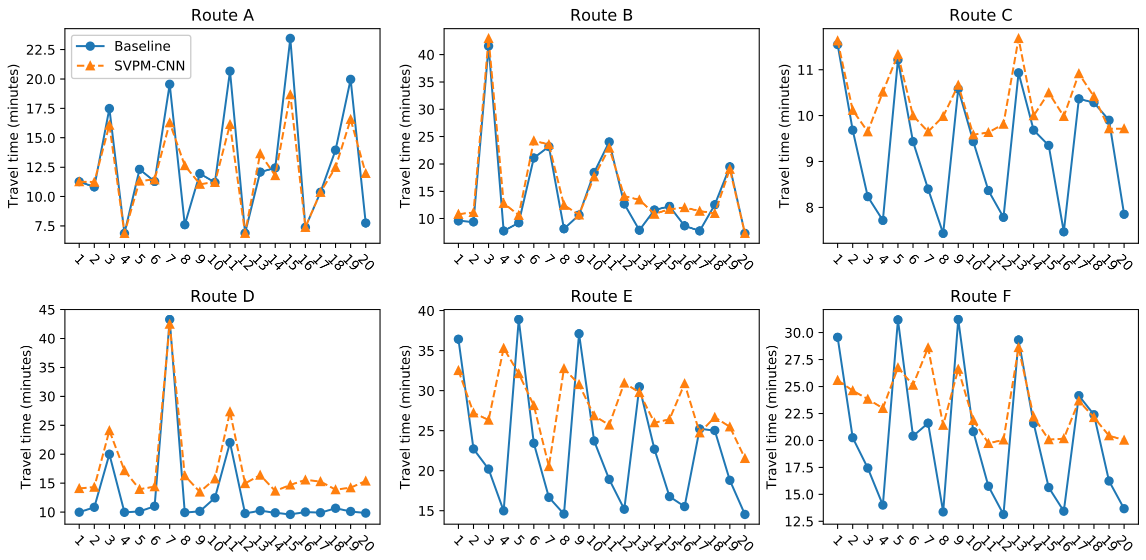


Figure 5.4: SVPM-CNN vs. Baseline over time

Figure 5.4 shows that the model behavior is different for each route. The uninterrupted travel time of each station is displayed by the values from 10:00pm, by which time all congestion has finished and the traffic flows freely.

The results show that the model comes closer to the travel time when routes are congested, and makes larger mistakes when roads have lighter traffic. Another tendency from the SVMP-CNN is that it deviates further from the baseline at the 10:00pm time, as shown on the individual time plots. That could be an indicator that

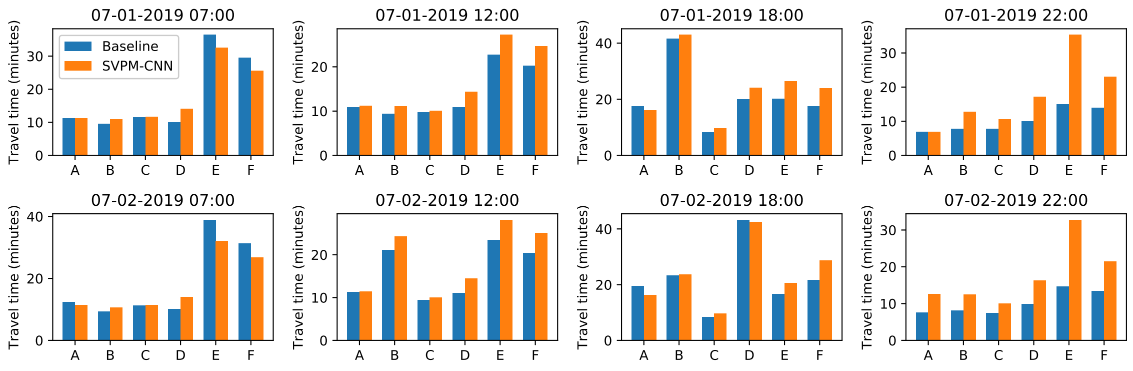


Figure 5.5: SVPM-CNN vs. Baseline over routes: Day 1 and 2

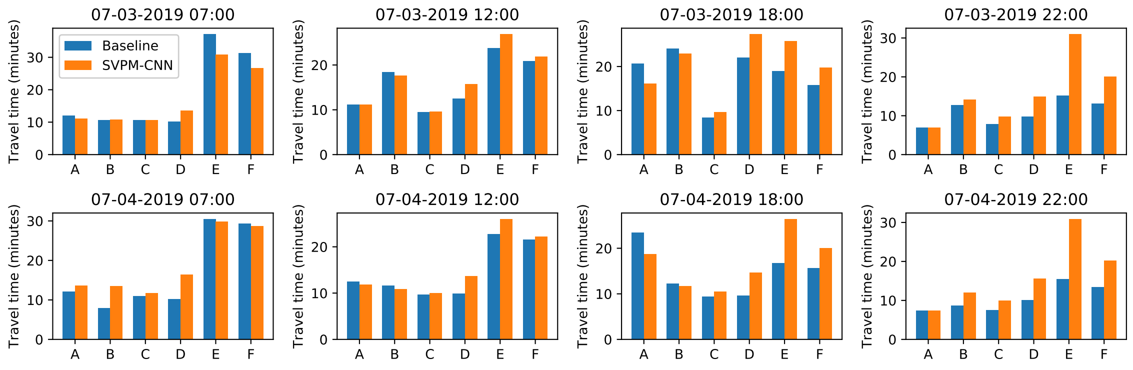


Figure 5.6: SVPM-CNN vs. Baseline over routes: Day 3 and 4

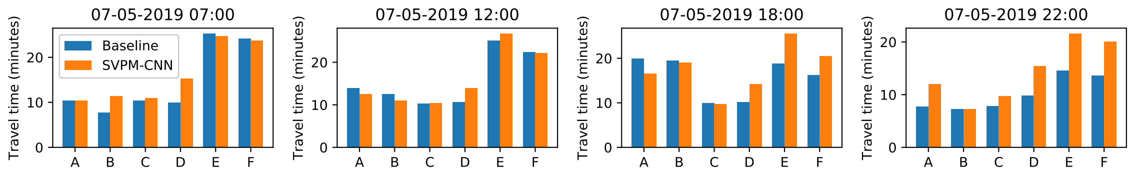


Figure 5.7: SVPM-CNN vs. Baseline over routes: Day 5

the model is worse when traffic is very light, and better when traffic is heavy, due to fact that the model was trained using less data from periods of very light traffic; our dataset is limited to data from 6am to midnight.

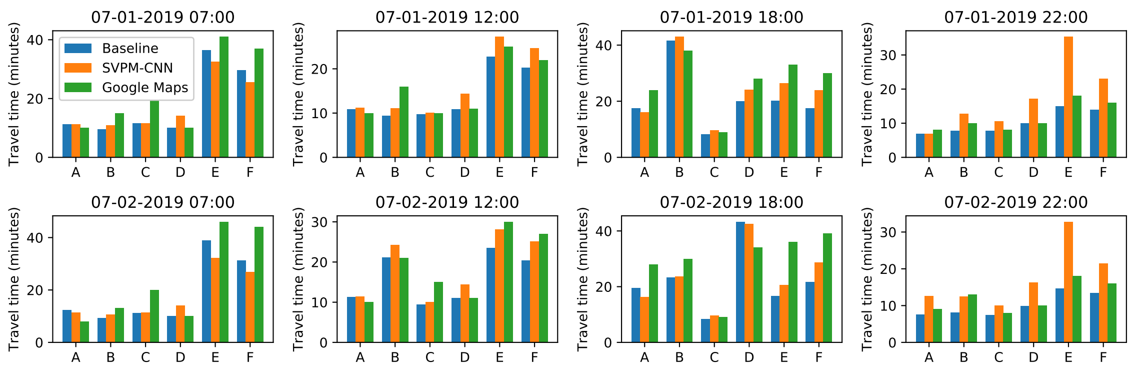


Figure 5.8: SVPM-CNN vs. Baseline vs. Google Maps over routes: Day 1 and 2

5.3.4 Comparing with Google Maps

Finally, in order to understand how close the previous results are to other applications, the ETT is compared with the one provided by Google Maps. During each one of the times displayed in Table 5.2, the Google Maps navigation service was accessed live and the provided ETT was recorded. Then, this result was compared with the ETT provided by the SVMP-CNN the same way as it was compared with the baseline.

The results of the computation of travel time by route over time is in Figure 5.9, and the travel time by time over each route are shown in Figures 5.8, 5.10, 5.11. The results show that, as opposed to the SVMP-CNN, Google Maps has a tendency of having larger error when traffic is heavy, and lower errors when traffic is light. As opposed to the SVMP-CNN, the Google Maps ETT does not display bigger errors around 10:00pm, however, it does show larger errors around 6:00pm, which the most congested time of day.

The mean absolute error from the SVMP-CNN and from Google Maps are then computed to further compare both results. There are two mean errors, the mean error by route and the mean error by time of day and they are displayed in Figures 5.12a and 5.12b. The SMAPE compared against Google Maps for each route is displayed in Table 5.3 and for each time of day in Table 5.4.

This result shows that for that the SVMP-CNN has a lower average error than Google Maps when estimating ETT, for four out of the six routes analyzed. Furthermore, the SVMP-CNN has a lower average error than Google Maps for predicting ETT for three (7:00am, 12:00pm, 6:00pm) out of four target times. Generally speaking, for the routes considered and for the target times, the SVMP-CNN has a lower error when estimating ETT than Google Maps.

Thus, the SVMP-CNN performs better than Google Maps at estimating the travel time on the specific times and routes that were analyzed, more so at high congestion moments. However, our approach preforms worse at predicting traffic on low traffic times of the day. One reason for that is that during training our model is weighted more heavily to focus on congestion, which may reward a result that is better at predicting heavy traffic, leading to a final result that is better during congestion.

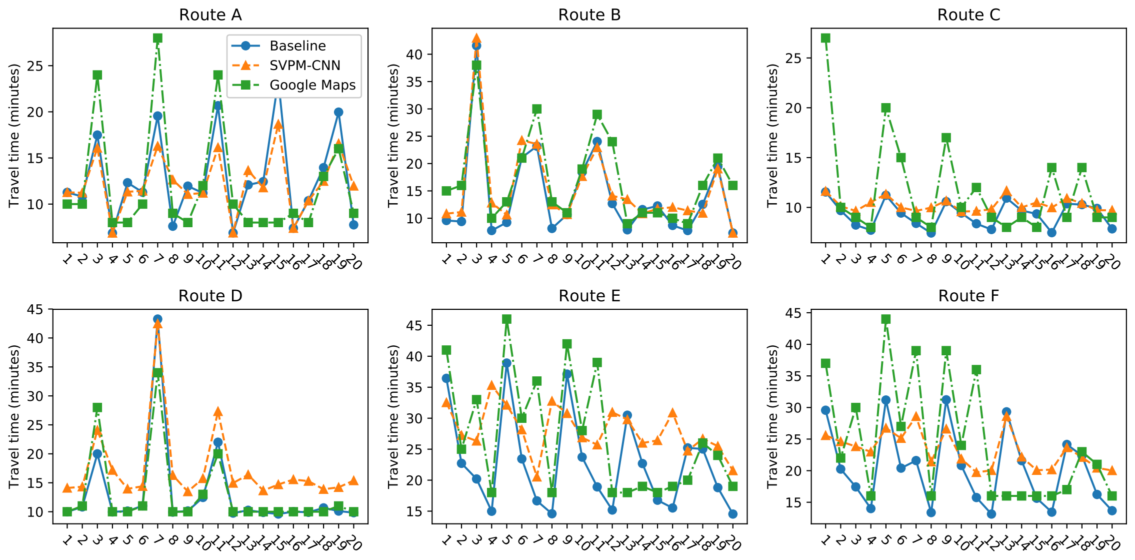


Figure 5.9: SVPM-CNN vs. Baseline vs. Google Maps over time

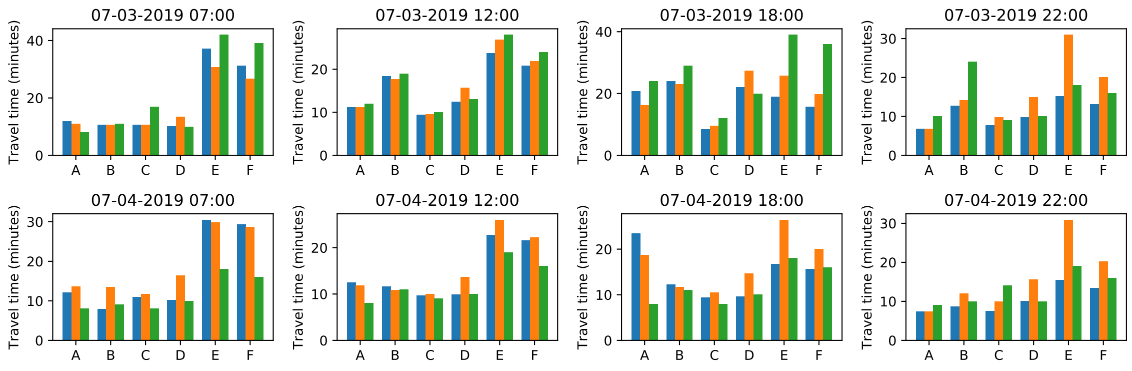


Figure 5.10: SVPM-CNN vs. Baseline vs. Google Maps over routes: Day 3 and 4

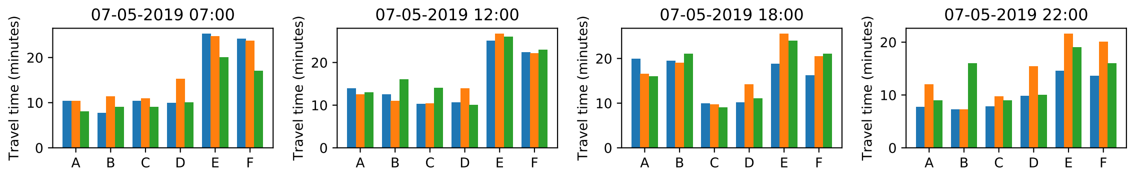
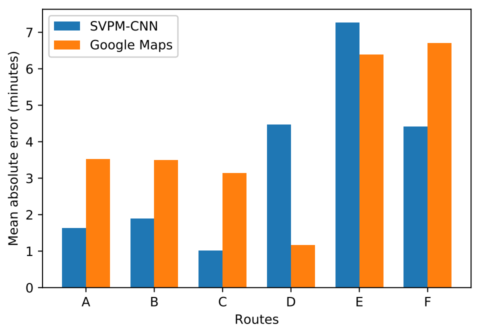


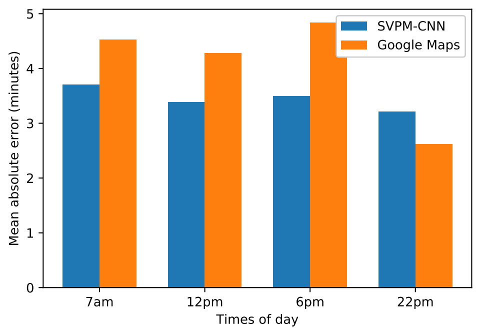
Figure 5.11: SVPM-CNN vs. Baseline vs. Google Maps over routes: Day 5

5.4 Conclusion

The experiments that were performed and then described in this chapter show that the SVMP-CNN can precisely perform traffic forecasting and accurately perform congestion detection. The results show that our model can predict traffic with lower error and higher accuracy than the S-LSTM model, which is the state for the art model



(a) Mean error vs. Google Maps over routes



(b) Mean error vs. Google Maps over time

Route	SMAPE	Google Maps SMAPE
A	11.74	27.18
B	16.12	24.38
C	11.05	24.86
D	33.39	5.21
E	30.29	25.65
F	22.13	27.78

Table 5.3: Route SMAPE vs. Google Maps

Time of day	SMAPE	Google Maps SMAPE
7:00am	20.63	23.25
12:00pm	19.40	20.64
6:00pm	21.31	27.29
22:00pm	21.82	18.86

Table 5.4: Time of day SMAPE vs. Google Maps

from the literature. Also, our traffic prediction results can be used to make travel time estimations comparable to those performed by Google Map. Those results support the claim that our model is better for traffic flow, speed, and congestion forecasting.

Chapter 6

Conclusions

6.1 Conclusion

In order to help improve the mobility issue in cities and urban areas, performing short-term traffic forecasting is essential to ITS. In this thesis, we solve this problem by doing short-term forecasting of the traffic main variables (traffic flow, traffic speed, and congestion state) using large amounts of traffic data that is available.

We have proposed a CNN-based model for short-term multi-traffic forecasting. The SVMP-CNN performs integrated traffic prediction (traffic flow, traffic speed, and congestion state) simultaneously over a large area, for a future window of one-hour at 5-minute intervals. In order to achieve that, we have gathered almost 6 months of data from the PeMS [23] dataset in the Los Angeles, CA area (district 7) and used that data to train and validate our model.

The SVMP-CNN was compared against the S-LSTM model proposed by Kang, Danqing *et al.* [14], then used to generate short-term congestion focal points and to estimate route travel time. The SVMP-CNN showed higher accuracy and recall than the S-LSTM model for detecting future congestion in the one-hour future interval.

Also, the SVMP-CNN has on average 2.7% smaller SMAPE error (7.19%-8.53% from our model vs. 10.33%-10.87% from the S-LSTM model). The main reason that SMAPE gives the best picture of the error is because it stabilizes small values, which is useful for our model because since the data is re-scaled to numbers between 0.0 and 1.0, there are several results that are close to 0.0.

Finally, the SVMP-CNN was used to estimate the travel time for six different routes at four different times of day, with an average error of around 20.8% for different times of day. Our results were compared with Google Maps for route travel time estimation, and our model has a smaller average error for four out of the six routes and also for three out of the four target times of day. There is one result that stands out, our model SMAPE percentage error at 6pm is around 20% better than the one from Google Maps, which shows that our model is able to perform especially better when traffic is heavy.

In conclusion, our model outperformed the state of the art model in all traffic flow and speed errors and also in congestion detection accuracy. Also, our model show comparable results to Google Maps when used for route travel time estimation, outperforming it in most scenarios, specially during rush hour when traffic is more congested.

The SVMP-CNN can be used for many ITS applications that require traffic forecasting, and it can also be used in real-time. The reason for that is that the computational cost is high only for training the model, using the model to perform forecasting has a negligible cost; the model only requires the updated window of 45 minutes of traffic data. Also, since the SVMP-CNN is trained using only data form 6am to midnight, the model can be updated every day by re-training it using new data from the previous day during the downtime. The process of training deep learning models with large amounts of data takes a significant amount of time, as many GPU computations are required for that, the SVMP-CNN performs multi-traffic prediction, so only one model is required to forecast the main traffic variables, which has a significantly lower cost than training three deep learning models.

There were a few challenges that were faced during this research. Due to the nature of the dataset, it needs to be parsed separately for each day of the year. Each day of the dataset has around 1 million text entries, and that needs to be filtered for the dataset extraction process. Also, the traffic speed and traffic flow are different orders of magnitude, and in order to make the model work properly, the correct scaling had to be applied, otherwise the model would converge to zeroes every single time.

Another difficulty of our model is to find the correct number of units for each layer in the model architecture. Since our model uses a window size of 9, and each element is a matrix of size 82×2, the data input is large, which poses a memory issue when training and also high processing times. However, it is possible to reduce the number of units in such a way that is does not deteriorate the performance and it takes less time to train and evaluate.

6.2 Limitations

The SVMP-CNN is trained using data from the target traffic stations from the dataset. The model learns the relationships between the traffic in each of the target traffic stations during training, so that it can perform better predictions.

Due to this nature, the model could not be used in any other area in trained form. In other words, if the model is trained for a specific group of traffic stations, then it can only be used to predict traffic in those stations. The model can be used in other areas, but it has to be re-trained with that from the other areas or during low traffic times.

6.3 Future Works

The SVMP-CNN has proven to perform well as compared to the literature, however it shows higher error during times of very light traffic. It would be valuable to the model in the future to update it to handle low traffic better. For instance, it is possible to evaluate each road and identify the ones that are more likely to be congested and feed this to the model. That way, the model is able to tune its predictions and avoid overshooting in low traffic areas.

On top of that, the SVMP-CNN is set to perform 1 hour of predictions at 5-minute intervals. That way, it is not possible to estimate travel time for routes that could potentially take more than one-hour. It could be interesting to extend the model to perform medium-term predictions to cover longer routes and evaluate the results in that scenario.

Currently, the congestion detection portion of our model is limited by a single threshold of 50% [18]. For future research, it would be valuable to introduce a multi-class congestion detection. That way, it is possible to have a more complete view of the traffic by using ranges, such as normal traffic, light congestion and heavy congestion. Another option would be to perform a thorough traffic evaluation to identify the optimal congestion threshold for each individual traffic station.

Another limiting factor for our model is that it is only trained with highway data, therefore it cannot be used for urban traffic and congestion prediction. For future work, it is possible to collect urban traffic data to validate the model in urban scenarios and compare it with highway scenarios, to evaluate the model accuracy and error for very different use scenarios.

Bibliography

[1] J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, and C. Chen, “Data-driven intelligent transportation systems: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 1624–1639, Dec 2011.

[2] L. N. N. Do, N. Taherifar, and H. L. Vu, “Survey of neural network-based models for short-term traffic state prediction,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 1, 2019.

[3] B. L. Smith, B. M. Williams, and R. K. Oswald, “Comparison of parametric and nonparametric models for traffic flow forecasting,” *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 4, pp. 303 – 321, 2002.

[4] R. Soua, A. Koesdwiady, and F. Karray, “Big-data-generated traffic flow prediction using deep learning and dempster-shafer theory,” vol. 2016-October, (Vancouver, BC, Canada), pp. 3195 – 3202, 2016.

[5] Y. Jia, J. Wu, and Y. Du, “Traffic speed prediction using deep learning method,” vol. 0, (Rio de Janeiro, Brazil), pp. 1217 – 1222, 2016.

[6] J. Qu, X. Gu, and L. Zhang, “Improved ugrnn for short-term traffic flow prediction with multi-feature sequence inputs,” vol. 2018-January, (Chiang Mai, Thailand), pp. 13 – 17, 2018.

[7] A. Fandango and R. P. Wiegand, "Towards investigation of iterative strategy for data mining of short-term traffic flow with recurrent neural networks," (Lakeland, FL, United states), pp. 65 – 69, 2018.

[8] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," (Wuhan, China), pp. 324 – 328, 2016.

[9] F. Kong, J. Li, B. Jiang, T. Zhang, and H. Song, "Big data-driven machine learning-enabled traffic flow prediction," *Transactions on Emerging Telecommunications Technologies*, 2018.

[10] 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.

[11] F. Kong, J. Li, and Z. Lv, "Construction of intelligent traffic information recommendation system based on long short-term memory," *Journal of Computational Science*, vol. 26, pp. 78 – 86, 2018.

[12] Y.-Y. Chen, Y. Lv, Z. Li, and F.-Y. Wang, "Long short-term memory model for traffic congestion prediction with online open data," vol. 0, (Rio de Janeiro, Brazil), pp. 132 – 137, 2016.

[13] H. Shao and B.-H. Soong, "Traffic flow prediction with long short-term memory networks (lstms)," vol. 0, (Singapore, Singapore), pp. 2986 – 2989, 2016.

[14] D. Kang, Y. Lv, and Y.-Y. Chen, "Short-term traffic flow prediction with lstm recurrent neural network," vol. 2018-March, (Yokohama, Kanagawa, Japan), pp. 1 – 6, 2018.

[15] J. Wang, F. Hu, and L. Li, "Deep bi-directional long short-term memory model for short-term traffic flow prediction," vol. 10638 LNCS, (Guangzhou, China), pp. 306 – 316, 2017.

[16] Q. Liu, B. Wang, and Y. Zhu, "Short-term traffic speed forecasting based on attention convolutional neural network for arterials," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 11, pp. 999 – 1016, 2018.

[17] 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 (Switzerland)*, vol. 17, no. 4, 2017.

[18] M. Chen, X. Yu, and Y. Liu, "Pcnn: Deep convolutional networks for short-term traffic congestion prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 11, pp. 3550 – 3559, 2018.

[19] S. Liao, J. Chen, J. Hou, Q. Xiong, and J. Wen, "Deep convolutional neural networks with random subspace learning for short-term traffic flow prediction with incomplete data," vol. 2018-July, (Rio de Janeiro, Brazil), 2018.

[20] Y. Chen, F. Chen, Y. Ren, T. Wu, and Y. Yao, "Poster: Deeptfp: Mobile time series data analytics based traffic flow prediction," vol. Part F131210, (Snowbird, UT, United states), pp. 537 – 539, 2017.

[21] D. Yu, Y. Liu, and X. Yu, "A data grouping cnn algorithm for short-term traffic flow forecasting," vol. 9931 LNCS, (Suzhou, China), pp. 92 – 103, 2016.

[22] Y. Jia, J. Wu, and M. Xu, "Traffic flow prediction with rainfall impact using a deep learning method," *Journal of Advanced Transportation*, vol. 2017, 2017.

[23] "Caltrans Performance Measurement System (PeMS)." http://pems.dot.ca.gov/. Accessed: 2019-03-01.

[24] Y. Jia, J. Wu, M. Ben-Akiva, R. Seshadri, and Y. Du, "Rainfall-integrated traffic speed prediction using deep learning method," *IET Intelligent Transport Systems*, vol. 11, no. 9, pp. 531 – 536, 2017.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[27] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, Sep. 2009.

[28] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[30] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard, "Handwritten digit recognition: appli-

cations of neural network chips and automatic learning," *IEEE Communications Magazine*, vol. 27, pp. 41–46, Nov 1989.

[31] Zhou and Chellappa, "Computation of optical flow using a neural network," in *IEEE 1988 International Conference on Neural Networks*, pp. 71–78 vol.2, July 1988.

[32] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.