

TRAFFIC SPEED PREDICTION WITH NEURAL NETWORKS

by

UMUT CAN ÇAKMAK

Submitted to the Graduate School of
Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University

July, 2017

TRAFFIC SPEED PREDICTION WITH NEURAL NETWORKS

APPROVED BY:

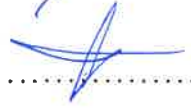
Prof. Dr. Bülent Çatay
(Thesis Supervisor)



Asst. Prof. Dr. Mehmet Serkan Apaydın



Assoc. Prof. Dr. Kemal Kılıç



DATE OF APPROVAL: 27/07/2017

© Umut Can akmak 2017

All Rights Reserved

ABSTRACT

TRAFFIC SPEED PREDICTION WITH NEURAL NETWORKS

UMUT CAN ÇAKMAK

M.Sc. Thesis, July 2017

Thesis Supervisor: Prof. Dr. Bülent Çatay

Keywords: Neural networks, forecasting, time series analysis, exponential smoothing, moving average

With the increasing interest in creating Smart Cities, traffic speed and flow prediction have attracted more attention in contemporary transportation research. Neural networks have been utilized in many recent studies to tackle this problem; yet, these methods have focused on the short-term traffic prediction while longer forecast horizons are needed for more reliable mobility and route planning. This work aims at filling this gap by trying to address the mid-term forecasting as well as the short-term. The study employs feed-forward neural networks that combine different time series forecasting techniques such as naïve, moving average and exponential smoothing where the predicted speed values are fed into the network as inputs. We train our neural networks and select the hyper-parameters of the network structures to minimize the error; thus, yielding the best possible setup for further forecast input. In our experimental study, we analyzed two nearly 20-km multi-segment routes from the city of Istanbul in Turkey. The speed data on these routes are collected by GPS for every minute for a 5-month horizon. Our computational tests showed that forecasts are more successful when performed on a route with shorter segments as well as a combination of conventional predictive methods are input to a neural network. We also discovered that depending on the characteristic of the analyzed road, it is possible to utilize the information from neighboring segments.

ÖZET

YAPAY SİNİR AĞLARI İLE KARAYOLU HIZ TAHMİNİ

UMUT CAN ÇAKMAK

Yüksek Lisans Tezi, Temmuz 2017

Tez Danışmanı: Prof. Dr. Bülent Çatay

Anahtar Kelimeler: Yapay sinir ağları, öngörüleme, zaman dizisi analizi, üstel düzleştirme, hareketli ortalama,

Akıllı Şehirler olarak adlandırılan şehircilik anlayışına yönelimin artmasıyla birlikte, trafik hız ve yoğunluk tahmini araştırma konuları da daha çok ilgi çekmeye başlamıştır. Bu konularda yapılan yakın tarihteki çalışmaların birçoğunda yapay sinir ağlarının kullanıldığı görülmektedir; fakat, bu çalışmalar genelde yakın gelecek tahminleri üzerine odaklanmıştır. Ancak yakın gelecek tahminleri günümüzdeki ihtiyaçları karşılamamaktadır ve daha güvenilir güzergâh planlaması için orta ve uzak dönem hız tahminleri yapan çalışmalara ihtiyaç duyulmaktadır. Bu çalışma da literatürdeki bu yetersizliği doldurmaya ve yakın dönem tahmin gibi orta dönem tahmin üzerinde de çalışmaktadır. Kullanılan ileri beslemeli yapay sinir ağı farklı zaman serisi öngörüleme yöntemlerini birleştirerek tahminler üretmektedir. Sinir ağıımızın eğitimi esnasında hata değerlerini en küçükleyecek parametreler kullanılmıştır. Çalışmamızda İstanbul şehrinden yaklaşık 20 kilometre uzunluğunda iki adet çok ayrıtlı güzergâh incelenmiştir ve bu güzergâhlar üzerindeki hız verileri Küresel Konumlama Sistemi (KKS) ile 5 aylık bir süre boyunca her dakika için toplanmıştır. Deneysel çalışmalarımız sonucunda daha kısa ayrıtlara bölünmüş bir güzergâhta ortalama mutlak sapma açısından daha iyi öngörüleme sonuçları alındığını gözlemledik. Aynı şekilde, belirli zaman serisi tahmin yöntemlerinin birleştiği yapay sinir ağlarının da bu tahmin yöntemlerinin kendilerinden daha iyi sonuç verdiğini gördük. Bunun dışında, incelenen yolun özelliklerine bağlı olarak, komşu ayrıtlı bilgilerinin de tahminlemede yararlı olduğunu gözlemledik.

Acknowledgments

I would like to first thank my parents for their extreme generosity and dedication towards their children and their education. I also want to thank my brother and trustworthy editor Bulut.

I then want to thank Hazal. Together we managed to complete our theses by continuous support and I am grateful that she was always helpful. She makes life a lot more meaningful, keeps the worries away, and is always there for me.

I am also grateful that I had the chance to work with my supervisor Prof. Dr. Bülent Çatay, and I want to thank him for his advice and guidance throughout my master's degree. I also want to thank Asst. Prof. Dr. Mehmet Serkan Apaydın for his continuous supervision throughout my research, as well. Finally, I would like to thank Assoc. Prof. Dr. Kemal Kılıç for his valuable feedback. Of course, I would also like to thank Başarsoft Inc.; because, without their contribution, this work would not have been possible.

I want to thank my professors in the Industrial Engineering department for all of their contributions throughout my education. And I want to thank my friends from Grad Office.

I want to thank Sabancı University Information Center and the many authors they have brought me in contact with. Most importantly, I want to especially mention the works of Isaac Asimov as they have been extremely inspiring and gave me hope for the future.

Contents

1	Introduction	1
2	Literature Review	3
3	Methodology	6
3.1	Data Collection and Cleaning	6
3.2	Prediction Methods	8
3.2.1	Naïve Method Prediction	8
3.2.2	(Weighted) Moving Average Method Prediction	8
3.2.3	Simple Exponential Smoothing Method Prediction	9
3.2.4	Double Exponential Smoothing (Holt’s) Method Prediction	10
3.2.5	Triple Exponential Smoothing (Winters) Method Prediction	11
3.3	Machine Learning	12
3.3.1	Feedforward Neural Networks (FFNN) / Multilayer Perceptrons (MLP)	12
3.3.2	Training Algorithms	14
3.3.2.1	Stochastic / Minibatch Gradient Descent	14
3.3.2.2	Adaptive Moment Estimation	15
4	Experimental Setup	16
4.1	Route Selection	16
4.2	Temporally and Spatially Connected Approach (1-step Ahead Forecast) (TS–1 Network)	18
4.3	Single Segment Approach (1-step Ahead Forecast) (SS–1 Network)	19
4.4	Single Segment Approach (Multi-step Ahead Forecast) (SS–M Network)	19
4.5	Recursive Single Segment Approach (Multi-step Ahead Forecast) (RSS– M Network)	20
5	Computational Results	22
5.1	1-step Ahead Predictions	23
5.2	Multi-step Ahead Predictions	24

6 Concluding Remarks and Future Research	28
Bibliography	30
Appendix A 1-Step Ahead Results	34
Appendix B Multi-Step Ahead Results	39

List of Tables

5.1	Lengths of the Analyzed Segments for 1-step Ahead Prediction	23
5.2	1-step Ahead Prediction Average of MAD (km/h) values of Different Setups and Methods for Both Routes over 500 epochs and with a minibatch size of 1000	23
5.3	Lengths of the Analyzed Segments for Multi-step Ahead Prediction . . .	24
5.4	Multi-step Ahead Prediction Average of MAD (km/h) values of Different Setups and Methods for Both Routes over 30 epochs and with a minibatch size of 1000	25
A.1	1-step Ahead Prediction for Segment #5 of Route 1 (0.2170 km in length)	35
A.2	1-step Ahead Prediction for Segment #15 of Route 1 (0.0441 km in length)	35
A.3	1-step Ahead Prediction for Segment #18 of Route 1 (0.0507 km in length)	36
A.4	1-step Ahead Prediction for Segment #50 of Route 1 (0.0454 km in length)	36
A.5	1-step Ahead Prediction for Segment #5 of Route 2 (0.4219 km in length)	37
A.6	1-step Ahead Prediction for Segment #15 of Route 2 (0.4419 km in length)	37
A.7	1-step Ahead Prediction for Segment #18 of Route 2 (0.4443 km in length)	38
A.8	1-step Ahead Prediction for Segment #50 of Route 2 (0.0072 km in length)	38
B.1	Multi-step Ahead Prediction Results for Segment #18 of Route 1 (0.0507 km in length)	40
B.2	Multi-step Ahead Prediction Results for Segment #19 of Route 1 (0.0461 km in length)	40
B.3	Multi-step Ahead Prediction Results for Segment #20 of Route 1 (0.0466 km in length)	40
B.4	Multi-step Ahead Prediction Results for Segment #46 of Route 1 (0.0822 km in length)	41
B.5	Multi-step Ahead Prediction Results for Segment #47 of Route 1 (0.0813 km in length)	41
B.6	Multi-step Ahead Prediction Results for Segment #48 of Route 1 (0.1019 km in length)	41

B.7 Multi-step Ahead Prediction Results for Segment #94 of Route 1 (0.2084 km in length)	42
B.8 Multi-step Ahead Prediction Results for Segment #95 of Route 1 (0.1617 km in length)	42
B.9 Multi-step Ahead Prediction Results for Segment #96 of Route 1 (0.1069 km in length)	42
B.10 Multi-step Ahead Prediction Results for Segment #121 of Route 1 (0.0738 km in length)	43
B.11 Multi-step Ahead Prediction Results for Segment #122 of Route 1 (0.0887 km in length)	43
B.12 Multi-step Ahead Prediction Results for Segment #123 of Route 1 (0.0682 km in length)	43
B.13 Multi-step Ahead Prediction Results for Segment #1 of Route 2 (0.5538 km in length)	44
B.14 Multi-step Ahead Prediction Results for Segment #2 of Route 2 (0.3711 km in length)	44
B.15 Multi-step Ahead Prediction Results for Segment #3 of Route 2 (0.2252 km in length)	44
B.16 Multi-step Ahead Prediction Results for Segment #6 of Route 2 (0.3849 km in length)	45
B.17 Multi-step Ahead Prediction Results for Segment #7 of Route 2 (0.3174 km in length)	45
B.18 Multi-step Ahead Prediction Results for Segment #8 of Route 2 (0.3567 km in length)	45
B.19 Multi-step Ahead Prediction Results for Segment #17 of Route 2 (0.4784 km in length)	46
B.20 Multi-step Ahead Prediction Results for Segment #18 of Route 2 (0.4443 km in length)	46
B.21 Multi-step Ahead Prediction Results for Segment #19 of Route 2 (0.5675 km in length)	46
B.22 Multi-step Ahead Prediction Results for Segment #26 of Route 2 (0.7029 km in length)	47
B.23 Multi-step Ahead Prediction Results for Segment #27 of Route 2 (0.9742 km in length)	47
B.24 Multi-step Ahead Prediction Results for Segment #28 of Route 2 (1.121 km in length)	47

List of Figures

3.1	Close-up Version of Custom Data Cleaning vs Median Filtering	7
3.2	Naïve Method Prediction vs the Observed Speeds	9
3.3	Weighted Moving Average Method Prediction vs the Observed Speeds . .	10
3.4	Simple Exponential Smoothing Method Prediction vs the Observed Speeds	11
3.5	Holt’s Method Prediction vs the Observed Speeds	12
3.6	Winters Method Prediction vs the Observed Speeds	13
4.1	Map and Features of Route 1	17
4.2	Map and Features of Route 2	17
4.3	Temporally and Spatially Connected Approach (1-step Ahead Forecast) (TS–1 Network) (Representation of only a single (m^{th}) method out of a total of M methods)	18
4.4	Single Segment Approach (1-step Ahead Forecast) (SS–1 Network) . . .	19
4.5	Single Segment Approach (Multi-step Ahead Forecast) (SS–M Network)	20
4.6	Recursive Single Segment Approach (Multi-step Ahead Forecast) (RSS– M Network)	21
5.1	NMS Method on SS–M over one day	26
5.2	NMW Method on SS–M over one day	26
5.3	NMSW Method on SS–M over one day	27

List of Algorithms

1	Data cleaning process (Applied separately for each segment)	7
2	Gradient Descent	14
3	Adaptive Moment Estimation	15

Chapter 1

Introduction

Traffic congestion has negative economic, environmental, and health-related effects on our world and our lives. According to Cookson and Pishue (2017), the traffic congestion costs £6.2 billion annually in the city of London alone with a total annual cost of £30.8 billion countrywide. Research like that of Guerreiro et al. (2016) also suggests that it has adverse effects on our health, being the second-largest source of particulate matter emission and a source of CO₂ emission that is 40% above EU regulations. Additionally, according to a 2016 report titled A European Strategy for Low-Emission Mobility, since more than 70% of the greenhouse gas emissions stem from the road transport, the extra time spent in traffic leads to a high emission of greenhouse gases (European Commission, 2016). To overcome these problems, the European Commission Directorate-General for Mobility and Transport (2011) is aiming for the widespread and effective use of the Smart Transportation Systems. Similarly, the authorities at Republic of Turkey Ministry of Transport, Maritime Affairs and Communications (2014) stated the need to direct our attention to the advancement of the information technologies that would solve such problems.

In this light, the congestion has substantial adverse consequences. A key part of reducing the congestion would be the accurate prediction of real-time traffic speed and optimize the route planning for individual people and companies alike.

A crucial element of the Smart Transportation Systems are smartphones and navigation systems. Ericsson AB (2015) states that there were 2.6 billion smartphone subscribers in 2014 and 3.2 billion in 2015, and expects the number will reach 6.3 billion in 2021. Similarly, Mobile Fact Sheet of Pew Research Center (2017) indicates there is a positive trend in both cell phone usage and smartphone usage with a greater trend in smartphone usage (up to 77% percent of U.S. adults from 35% in only 5.5 years.) Aside from the smartphones, the traditional navigation and Global Positioning Systems equipment are also somewhat favored among drivers. Jenness et al. (2008) states that the majority of the drivers (88% of the surveyed) are content with their built-in navigation systems and

would repurchase said systems.

The worldwide rise of the use of smartphones is clear, and while traditional navigation systems are starting to lose their users, they are simultaneously being replaced by smartphone navigation applications. uShip.com (2012) states that among the surveyed U.S. truckers, 75% of them use smartphones, 52% of them use their smartphone's GPS navigation capabilities, compared to 22% in 2010 and 24% in 2011. Meanwhile, U.S. Navigation Usage and Satisfaction Study of J.D. Power (2013) finds that while only 37% of vehicle owners downloaded a smartphone application for navigation in 2011, 47% responders say the same only a year later. Smith et al. (2015) underline that 67% of adult smartphone owners use their turn-by-turn navigation systems at least occasionally while 31% use them frequently.

Furthermore, there exist some research on whether the navigation systems are beneficial for areas of road traffic other than the shortened journey times. Vonk et al. (2007) report that drivers feel and act safer when guided by a navigation system.

In Turkey, according to Poushter (2016), adult smartphone owners made up 59% of the population in 2015, ranked 12th in the world. It amounts to a 42-point increase in only 2 years, from 17% ownership ratio. Meanwhile, 72% of the adult population uses the internet at least occasionally, ranked 11th in the world along with Palestinian territories, Italy, and Russia. It amounts to a 31-point increase in only 2 years, from 41%. While there is no country-specific forecast on the smartphone ownership numbers in the coming years, Turkey is clearly at the forefront when it comes to adapting to smartphones.

Given its negative effects, it would be highly beneficial to improve our predictions, which is facilitated by this rapid growth of smartphone technology. In this work, it is expected to provide a close prediction of real-life speeds over a forecasting horizon instead of a fixed point in the future. We also expect our predictions to have a real-life effect on the end user. The remainder of this thesis is organized as follows: Chapter 2 is a review of the related literature. Chapter 3 is a thorough presentation of the methodology including the data collection and correction processes, the introduction to the prediction methods and the machine learning concepts. Chapter 4 presents the experimental setup. Chapter 5 reports and discusses the results. Finally, Chapter 6 concludes the thesis with suggestions for future research.

Chapter 2

Literature Review

There are certain methods frequented by other researchers on data collection and speed prediction. The survey of Vlahogianni et al. (2014) shows that early studies mainly collected their data from the highway sensors, and GPS data collection was not commonplace until after 2011. This is probably because GPS data only recently started to become abundant; however, the highway sensors were already present because the public institutions opted to use such equipment on traffic monitoring. The difference in methods is not only present in data collection but also on the essence of the data. Vlahogianni et al. (2014) again show that the data collected is mainly one of the following three types: speed data, journey time, and volume.

The speed prediction methods, on the other hand, are much more diverse than the data collection methods. The research mainly groups into two: discrete or continuous analyses. Discrete analyses can be summed up to binary or multiclass classification methods while continuous analyses employ function approximation, time series analysis and the like.

In Wang et al. (2015), GPS data was used to assess the predictability of the transitions between six different congestion states separated by certain speed limits. The authors report around 75% predictability for speeds less than 50 km/h and 80%-90% predictability for speed larger than 50 km/h over three different ring roads. The authors also comment that as the number of the segments on a route is increased, better results are likely to be acquired. Similarly, in Khan et al. (2012), the researchers assumed multiple congestion levels based on the level of occupancy of a road. Due to lack of accessible data, the authors analyzed on a simulated model with space-time autoregressive integrated moving average (STARIMA). The main source of their data was motorway sensors and their results turned out quite impressive (around 90% accuracy) for short-term predictions up to 5 minutes on the highway; however, in the urban setting, they report poor results. Additionally, they also emphasize their simulation models' inadequacy to reflect the seasonal characteristics of rush hours and the weekends. Hu et al. (2014) employ binary classification to determine if an intersection is congested and needs to be avoided or not. Based on their limit speed

of congestion, a correct congestion prediction of up to 89% was achieved. However, it is reasonable to assume that, in a larger road network, independently assigning a congestion threshold to every area and possibly every time of a day will not be feasible. It should also be noted that not all of these works inspect the effects of the neighboring segments or the past speed data.

Lippi et al. (2010) are among the first of the (binary) classification works that state the spatial and temporal neighboring data is important in performing more accurate predictions. In their study, they use road sensors to gather data and they occasionally outperform support vector regression (SVR). Similarly, Li et al. (2016) also employ binary classification to their GPS data and obtain 2.5% to 4% root mean squared error depending on the departure time identifying congestions. Additionally, Yang (2013) emphasizes the consideration of spatial and temporal relations in prediction results in a very large problem. That is why the author tries to rank the features and perform analyses only with the most relevant ones among them on a binary classification problem. Though the results are around 50-60% for mean prediction precision, this work is highly important and perhaps can be improved via GPS data collection as opposed to Yang's data collection choice of road sensors. Georgescu et al. (2012) apply regression to working days speed data. Since their dataset is limited and repetitive, their relative mean error values are starting from 5.8 for 1-step ahead prediction. Dauwels et al. (2014), stating the importance of neighboring data, also outperform SVR in long-term prediction. Xie et al. (2010) use road sensors to read the volume of the road and predict the volume of a road for multiple horizons into the future. The work focuses on Gaussian processes and outperforms autoregressive integrated moving average (ARIMA) and compete with ν -Support Vector Machines (ν -SVM).

Although some of the research has mainly focused on route planning and congestion avoidance, their insights are invaluable to the speed prediction literature. Liang and Wakahara (2014) suggest that information coming from the neighboring segments will help improve the forecasts. The use of the machine learning methods in traffic speed prediction, lately, has been on the rise with the rapidly improving technology and computing power capacity. Ma et al. (2015) employ recurrent neural networks (RNN) and similar deep learning methods to predict the propagation of a congestion in a network on long-term prediction. Their results indicate that a larger data, naturally, will take more time to analyze and will yield worse results. That is why they have better results when data aggregation is over 60 minutes (88.2%) compared to 5 minutes (68.9%). In Ye et al. (2012), the authors work on irregularly observed data and state that incorporating the information from both the past speed data and the data of the neighboring segments improved their algorithm's performance significantly. They achieve an average mean absolute deviation (MAD) value of 6.60 km/h for 1-second ahead and 12.47 km/h for 5-second ahead prediction over 20 segments.

This work will employ a feedforward neural network (FFNN) to perform a continuous prediction. The idea to use a neural network is mainly inspired by the works and largely acceptable solutions of Ye et al. (2012) employing an FFNN, the success of Park et al. (2011) also employing an FFNN on short-term congestion forecast as a binary classification problem, and the combination of the rain data with the number of vehicles information also with an FFNN by Dunne and Ghosh (2013).

The survey of Djahel et al. (2015) suggests that ARIMA, neural networks, statistical models, and GPS data based techniques are not adequate to make accurate predictions by themselves. This work is aimed to cover the positive aspects of the most promising literature and combine them in an improved manner.

Routing problems have always been problems of the sort that needs to be solved quickly for the sake of the urban population. However, as per the stochastic nature of the traffic, these predictions will always have higher error margins as the prediction horizon gets larger. That is why predictions must be made quite frequently and continuously. In this work, discrete methods are not preferred because it would be unrealistic to classify every road segment with its unique and seasonally changing congestion threshold speed. Similarly, since it is not reasonable to equip every road segment in an urban area with highway sensors, data collected by GPS is preferred to perform broader analyses. Additionally, the data and the predictions will be the speed data type because a congestion classification analysis will be difficult as each road and each segment would have different congestion properties. Similarly, volume data would force the analysis to make separate distinctions for different segments and journey time data would direct the analysis to perform from a point to another point. And unlike Ye et al. (2012), these data will not be collected at irregular intervals; thus, making the predictions more accurate for the long-term forecasting.

Chapter 3

Methodology

3.1 Data Collection and Cleaning

The historical speed data is obtained from Başarsoft Information Technologies Inc. It includes floating car speeds collected on Turkey road network with 1-minute time intervals over a 5-month horizon from October 2016 to February 2017. The locations, lengths, and other features of the segments of the road network were predetermined by Başarsoft.

After extracting the segment data from the original country data, first, the missing data were linearly interpolated. Moreover, the maximum speed is determined to be 120 km/h since it is the legal speed limit. Any speed data exceeding this limit was fixed to 120 km/h.

However, there emerged a need to clean the data of unrealistic observations. For example, a segment could experience a jump of 80 km/h in speed from one minute to the following which seems highly unlikely in a normal traffic flow. The most likely cause of these irregularities seems to be the source of data. Since the raw data can be collected from numerous different types of vehicles and without a lane restriction, it is possible that a segment's average speed can be affected greatly by a high-speed vehicle traveling on a hard shoulder like an ambulance for one minute then come back down to the regular speed value the next.

The custom method of data cleaning process is as follows: First, we calculated the minute-by-minute speed differences. Then, the minute-to-minute speed difference limit was set to be 5 km/h. If there existed a difference greater than this limit, speed differences falling outside 3 standard deviations of the data were replaced with linear interpolations in the speed data. After these replacements, if there still existed a difference greater than 5 km/h in consecutive time intervals in a given segment, the maximum standard deviation was reduced by 85% ($3 \times 0.85 = 2.55$). So, the points falling outside 2.55 standard deviations of the data were replaced with the same steps. This process continued until there was no speed difference greater than 5 km/h. The process is also outlined in

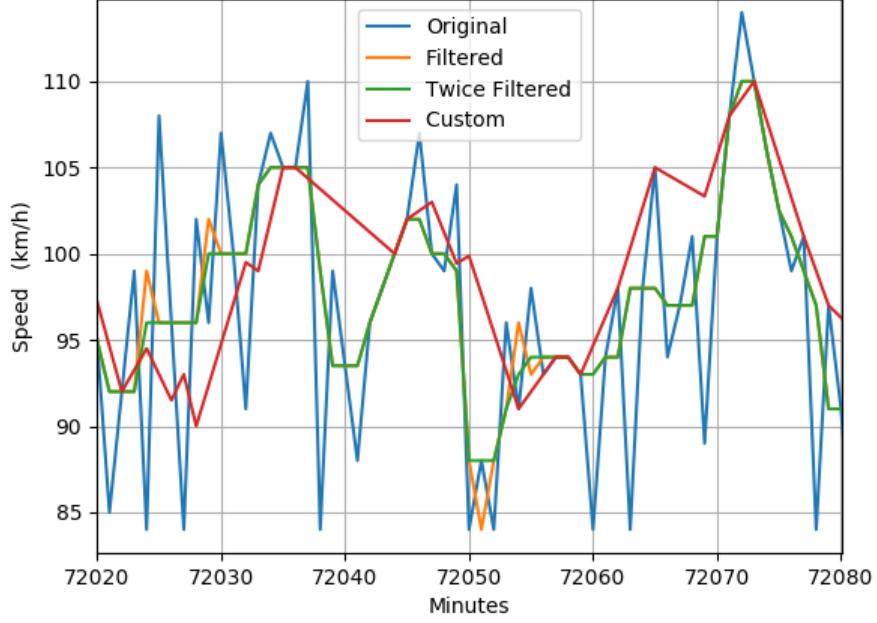


Figure 3.1: Close-up Version of Custom Data Cleaning vs Median Filtering

Algorithm 1 below. The resulting dataset is named *Cleaned* data in the computational experiments.

Algorithm 1: Data cleaning process (Applied separately for each segment)

- 1 Calculate minute-by-minute *speed difference data* with respect to *speed data*
 - 2 Calculate standard deviation of speed changes in a given segment
 - 3 Speed difference limit $:= 5 \text{ km/h}$
 - 4 Max standard deviation $K := 3$
 - 5 **while** *there exists speed difference* $>$ *Speed difference limit* **do**
 - 6 Find *speed difference data points* falling outside K standard deviations
 - 7 Remove and linearly interpolate the corresponding data point from *speed data*
 - 8 **if** *no point left outside* K *standard deviations* **then**
 - 9 $K \leftarrow K \times 0.85$
 - 10 **end**
 - 11 **end**
-

One of the many data smoothing or data cleaning techniques is median filtering. It is mainly used for image and signal noise reduction. Median filtering performs a sliding window analysis on the series on hand. Median of the *sorted* contents of the window replaces the value at the center of the window. This way, if there is an erratic value, it would be removed and the overall data will be much smoother (Delmas, 2010). We employed it on this dataset to see the differences with our custom method. The differences between the two methods over an hour for segment #9 of Route 2 can be seen in Figure 3.1.

Clearly, our cleaning method results in a more reasonable plot. Median filtered and twice median filtered (*Filtered* and *Twice Filtered* in the plot, respectively) are also re-

moving the spikes in the data; however, as it can be seen from the zoomed plot, filtered speeds still make irrational minute-to-minute jumps.

Some observations can be made from these findings and the general characteristics of the routes and the dataset. First, clearly, these two routes are of significantly different types. While the first route contains a lot of traffic lights, the second route is a highway route that has a bottleneck that is among the most effective ones in the city. The average segment length is more than five times larger for the second route compared to the first route while the number of segments is more than five times larger in the first route when compared to the second route. The first route had 35.7% of its original data replaced with an average of 18.77 km/h decrease and an average of 5.94 km/h increase. The second route had 60.6% of its original data replaced with an average of 10.74 km/h decrease and an average of 8.33 km/h increase. It is, then, also possible to say that the data read originally from the second route seems more smooth as the average change in speed after cleaning is much less than the first one. This is expected and the reason for this might be that the first route had frequent intersections with traffic lights and different lanes are used by vehicles of varying speeds simultaneously while the second route is a highway which has similar cruising speeds throughout. It is also clear that average speed observed on a longer segment would be better at balancing out the adverse effects of outliers.

3.2 Prediction Methods

In the following methods, s_t represents the observation at time t while f_{t+k} represents the prediction of the speed at time $t+k$.

3.2.1 Naïve Method Prediction

Naïve method prediction used in this study is the simplest prediction method where the forecast is equal to the data last observed. So, a lagged trace of the observed data is what is forecasted (Stevenson, 2012). This method may perform well for short-term predictions because 1-step/1-minute ahead prediction alone can deviate at most 5 km/h from the actual value as per the limit set in data cleaning process. A part of the naïve prediction on the data can be seen in Figure 3.2 and the forecast equation is below.

$$f_{t+1} = s_t \tag{3.1}$$

3.2.2 (Weighted) Moving Average Method Prediction

In regular moving average method prediction, the forecast is the average of the observed data over a period (see (3.2)). The moving average method basically smooths the observed

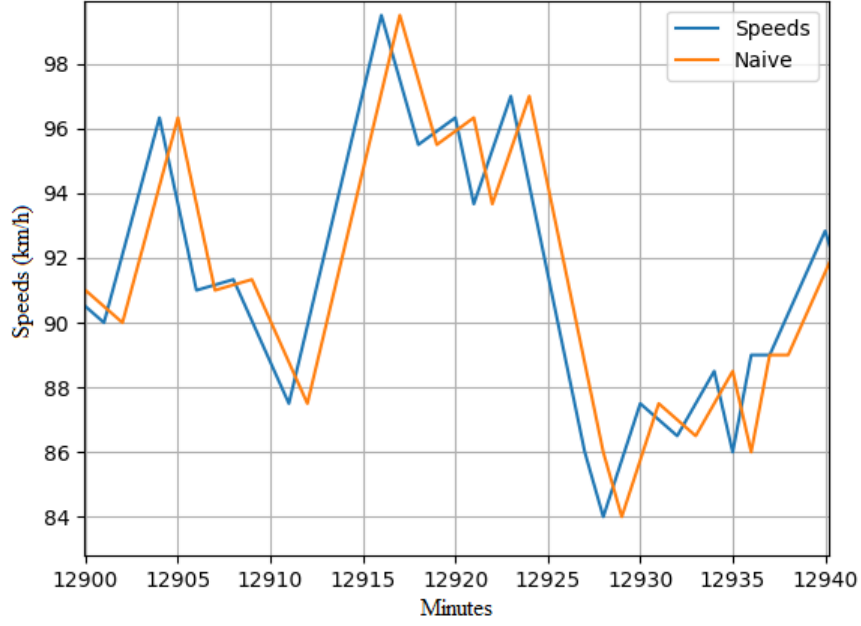


Figure 3.2: Naïve Method Prediction vs the Observed Speeds

data (Stevenson, 2012).

$$f_{t+k} = \frac{s_t + s_{t-1} + s_{t-2} + \cdots + s_{t-(n-1)}}{n} \quad (3.2)$$

It is better to keep the parameter n smaller as a larger n would result in the forecast to be too smooth (Stevenson, 2012). In weighted moving average method prediction, on the other hand, a weighted average of the last n observations is taken as the forecast as can be seen in (3.3).

$$f_{t+k} = w_t s_t + w_{t-1} s_{t-1} + \cdots + w_{t-(n-1)} s_{t-(n-1)} \quad (3.3)$$

where w_i is the weight associated with the observation at time i with $\sum_{i=t-(n-1)}^t w_i = 1$ and $0 \leq w_i \leq 1, \forall i$. The benefit of weighted moving average is that it can be tuned to give the most proper past data more importance (Stevenson, 2012). For example, in a speed prediction model, the latest data is likely the most reflective of the current situation. A part of the weighted moving average prediction on the data can be seen in Figure 3.3.

3.2.3 Simple Exponential Smoothing Method Prediction

This method is similar to the weighted moving average where a weight is given to the last observation and another weight is given to the last forecast. This recursive relationship makes the forecast take into account the whole past observations. While a larger alpha

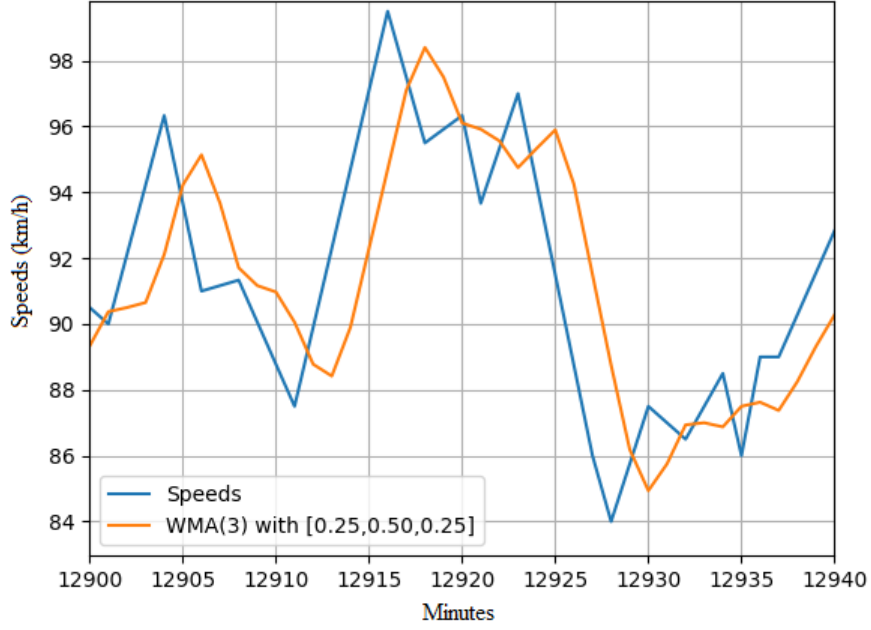


Figure 3.3: Weighted Moving Average Method Prediction vs the Observed Speeds

value increases the responsiveness of the forecast, a smaller one can make the forecasts smoother (Stevenson, 2012).

$$f_{t+k} = \alpha s_t + (1 - \alpha)f_t, \quad 0 \leq \alpha \leq 1 \quad (3.4)$$

where α is the *smoothing constant* used to smooth the data (Croarkin et al., 2006). A part of the simple exponential smoothing prediction on the data can be seen in Figure 3.4.

3.2.4 Double Exponential Smoothing (Holt's) Method Prediction

This method is used when there is a trend in the data. While a regular linear fit would not adapt to the changes in the trend through the observations, this method changes its slope with the changing trend (Stevenson, 2012). This method is also tested on this dataset because there are clear positive and negative *microtrends* over the transitions from rush hours to remaining parts of a day.

$$L_t = \alpha s_t + (1 - \alpha)(L_{t-1} + T_{t-1}), \quad 0 \leq \alpha \leq 1 \quad (3.5)$$

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1}, \quad 0 \leq \beta \leq 1 \quad (3.6)$$

$$f_{t+k} = L_t + kT_t \quad (3.7)$$

where L_i is known as *Level* or *Smoothed Observation* at time i and α is the *smoothing*

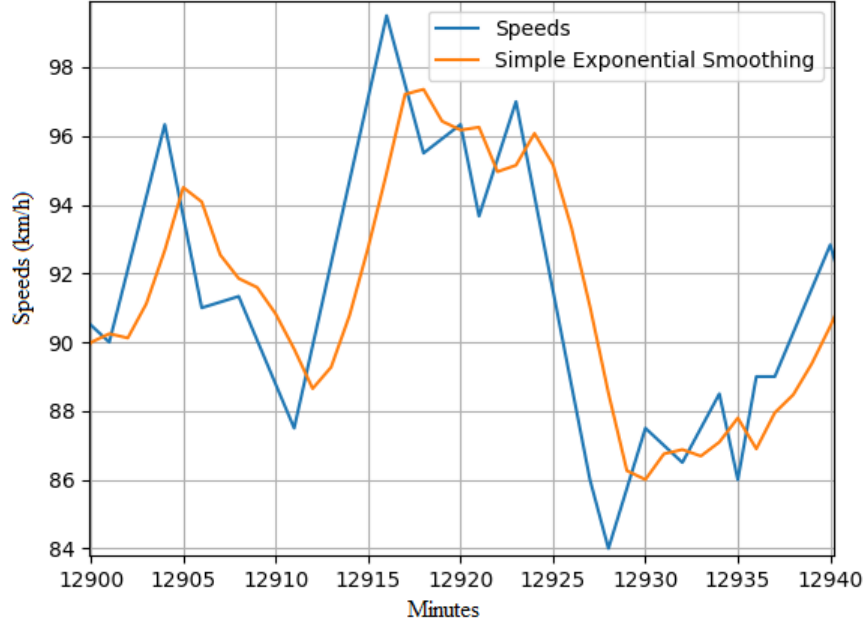


Figure 3.4: Simple Exponential Smoothing Method Prediction vs the Observed Speeds

constant to find it. Similarly, T_i is known as *Trend* or *Trend Factor* at time i and β is the *trend smoothing constant* which helps to smooth the trend in the data (Croarkin et al., 2006). A part of the double exponential smoothing prediction on the data can be seen in Figure 3.5.

3.2.5 Triple Exponential Smoothing (Winters) Method Prediction

This method is developed to handle trend and seasonality simultaneously. Thus, it is employed when even the double exponential smoothing method cannot be used (Croarkin et al., 2006). It is decided to be used on this dataset because there are microscopic seasons over the course of this five months such as the rush hours of weekdays in addition to the trends mentioned above.

$$L_t = \alpha \frac{S_t}{S_{t-M}} + (1 - \alpha)(L_{t-1} + T_{t-1}), \quad 0 \leq \alpha \leq 1 \quad (3.8)$$

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1}, \quad 0 \leq \beta \leq 1 \quad (3.9)$$

$$S_t = \gamma \frac{S_t}{L_t} + (1 - \gamma)S_{t-M}, \quad 0 \leq \gamma \leq 1 \quad (3.10)$$

$$f_{t+k} = (L_t + kT_t)S_{t+k-M} \quad (3.11)$$

Similarly, for Winters, L is the *Smoothed Observation* and α is the *smoothing constant*. T

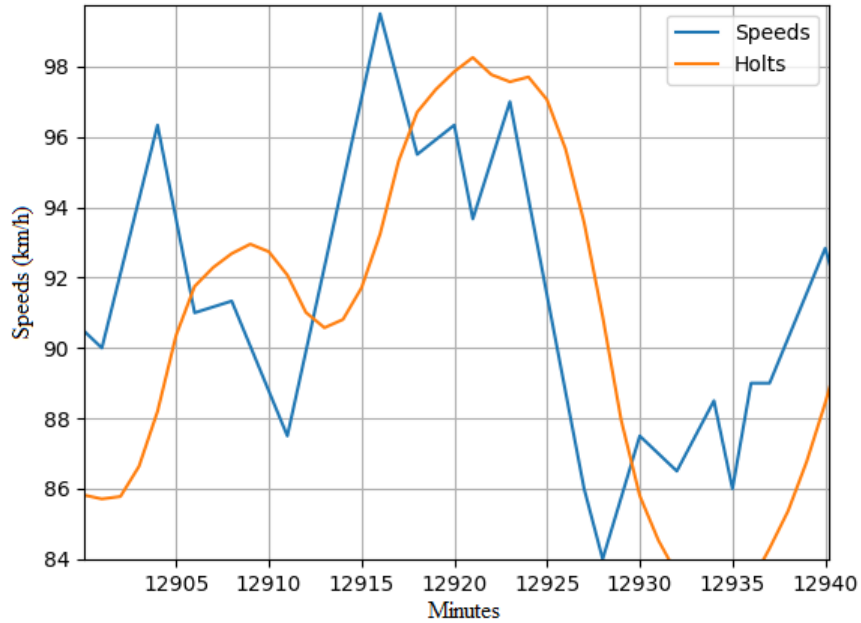


Figure 3.5: Holt's Method Prediction vs the Observed Speeds

is the *Trend Factor* and β is the *trend smoothing constant*. Here, S_i is called the *Seasonal Index* at time i and γ is the *seasonality smoothing constant* (Croarkin et al., 2006). And finally, M is the number of seasons. In our case, the seasons consist of 1-minute time intervals and we have 1440 seasons in a day throughout the entire horizon. A part of the triple exponential smoothing prediction on the data can be seen in Figure 3.6.

3.3 Machine Learning

For this problem, artificial neural networks are employed. While a concept of the 1950s (McCarthy et al., 2006), artificial intelligence improved significantly in the recent years with advancement in computing power.

3.3.1 Feedforward Neural Networks (FFNN) / Multilayer Perceptrons (MLP)

Feedforward neural networks or perceptrons are created to mimic the workings of the human brain. A simple perceptron model has an output unit y_i and input units x_i along with an extra bias unit, a set of weights that connect the inputs and the bias unit to the output (Alpaydın, 2014).

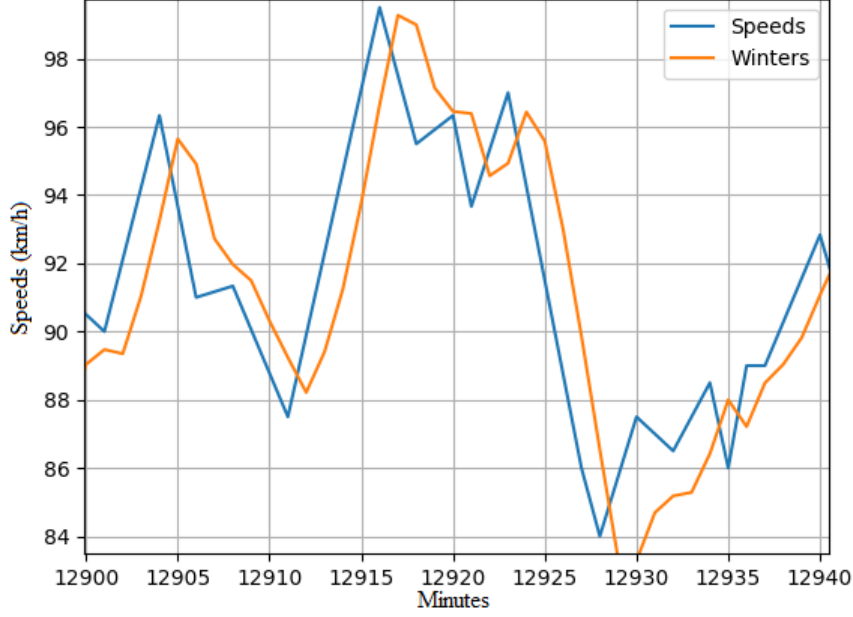


Figure 3.6: Winters Method Prediction vs the Observed Speeds

$$y = w_0x_0 + \sum_{j=1}^d w_jx_j \quad (3.12)$$

where d is the number of input neurons excluding the bias unit. A bias unit is an input unit of $x_0 = 1$. It acts, as can be seen from (3.12), as the constant in a linear equation.

$$y = \mathbf{w}^T \mathbf{x}, \text{ where } \mathbf{w} = [w_0, w_1, \dots, w_d]^T \text{ and } \mathbf{x} = [1, x_1, \dots, x_d]^T \quad (3.13)$$

A multilayer perceptron is capable of more, compared to a single layer perceptron. It has the advantage of handling nonlinear functions (Alpaydm, 2014). The multilayer perceptrons have at least one hidden layer in addition to input and output layers.

To train these models, a dataset with input and target data is required. In this work, the input data is the forecasts acquired by the forecasting methods mentioned in Section 3.2. The training starts with an initial set of weights and progresses forward over the system to yield an output value. This forward propagation uses (3.13). Every layer has an activation function, and the input from the previous layer is multiplied by the connection weights to be fed into the next layer nodes as a *forward input*. This input then is fed to the activation function yielding the output of the node. This process continues from the input layer to the output layer similarly for all neurons in all layers. The output layer then, again through an activation function, yields an output value. This output value is then compared to the

target value provided.

To minimize the resulting error, the system then does a backward propagation, updating the weights. The backpropagation process can be seen as a forward propagation from the output layer to the input layer with the gradients of the activation functions. The gradient of the error function is fed into the output layer where it is multiplied with *forward input* of the neuron passed through the gradient of the activation function. The resulting value is *error signal*. Similarly, for the hidden layers, the error signals of a later layer multiplied by connection weights are called *backward input*. This *backward input* is again multiplied with *forward input* passed through the gradient of the activation function and the resulting *error signal* is used for other layers. After obtaining *error signals*, gradients of the weights are calculated as the multiplication of *error signal* and *forward output* of a neuron. Finally, the weights are updated with a predetermined *learning rate* multiplied by the gradient of the weights. This procedure continues until the error is minimized. Interested readers are encouraged to read in more detail in Goodfellow et al. (2016).

Once the algorithm stops and obtains the minimum error value, another dataset is tested or validated on the set of tuned weights to measure the network's performance, as well.

3.3.2 Training Algorithms

3.3.2.1 Stochastic / Minibatch Gradient Descent

Stochastic gradient descent is among the most preferred training algorithms in machine learning. The method works on single points of the data, updating the weights at each epoch. And the update is done according to (3.14).

$$w^{\tau+1} = w^{\tau} - \eta \nabla E_n(w^{\tau}) \quad (3.14)$$

Since the data point for which the weights will be updated is randomly selected, stochastic gradient descent is more likely to escape the local solutions compared to the methods with full gradient calculations (Bishop, 2006).

However, the minibatch update handles multiple data points and thus more efficient compared to stochastic gradient descent in the same amount of time.

Algorithm 2: Gradient Descent

- 1 Sample a minibatch of m examples from the training set \mathbf{x} with corresponding targets \mathbf{y}
 - 2 Compute gradient estimate $\frac{\nabla E \mathbf{w}^{\tau}}{m}$ over these m samples
 - 3 Apply weight update: $\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\nabla E \mathbf{w}^{\tau}}{m}$
-

Algorithm 2 from Goodfellow et al. (2016) is a more general representation of the

stochastic and minibatch gradient descent methods. If $m = 1$, then this update is known as *stochastic gradient descent*; if $1 < m < d$, then the update is known as the *minibatch gradient descent*; and if $m = d$, the update is known as the *batch gradient descent*.

3.3.2.2 Adaptive Moment Estimation

Adam is the algorithm used in this work. Over the years, a lot of extensions were added to the stochastic gradient descent and Adam is one of them. It only uses the first order gradient and some constants (Ruder, 2016; Kingma and Ba, 2014). The steps are outlined in Algorithm 3.

Algorithm 3: Adaptive Moment Estimation

```

1 while not converged do
2    $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
3    $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
4    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \frac{\eta}{\epsilon + \sqrt{\frac{v_t}{1 - \beta_2^t}}} * \frac{m_t}{1 - \beta_1^t}$ 
5    $t \leftarrow t + 1$ 
6 end

```

Adam’s one advantage is that it makes use of the momentum. Momentum helps in cases where other methods might make large oscillations and miss the optimum due to poor *learning rate* selections. Momentum takes the average of the updates over the iterations thus creating a corrective update direction that would not miss the optimum (Goodfellow et al., 2016). The downside is the extra memory requirement from storing information on previous updates. Another advantage of Adam is that it “computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.” (Kingma and Ba, 2014)

Chapter 4

Experimental Setup

We propose four main approaches to the problem. To this end, four different MLPs were constructed. Two of these perform 1-step ahead forecasts and the other two perform multi-step ahead forecasts. Error function of all of these perceptrons is mean absolute deviation and it can be formulated as shown in (4.1) for a given segment.

$$MAD = \frac{1}{n} \sum_{i=1}^n |e_i| \quad (4.1)$$

where n is the number of forecasts and $|e_i|$ is the absolute error.

4.1 Route Selection

The analyses were performed on two nearly 22-km routes in the city of Istanbul. The routes and their features can be seen in Figures 4.1 and 4.2.

The first route covers 324 segments over a distance of 21.4908 kilometers, with a mean segment length of 0.0663 kilometers and a median segment length of 0.0455 kilometers. It is an urban route with many intersections and that is why there are a lot of segments on it.

The second route covers 63 segments over a distance of 22.7513 kilometers, with a mean segment length of 0.3611 kilometers and a median segment length of 0.2482 kilometers. It is mostly a highway route with a Bosphorus bridge crossing acting as a bottleneck.

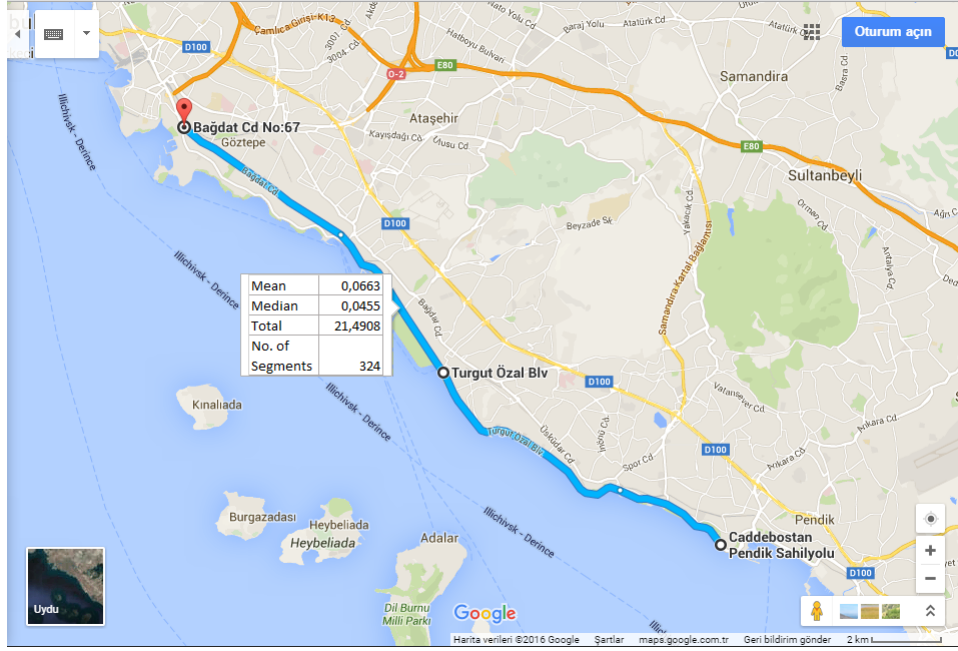


Figure 4.1: Map and Features of Route 1

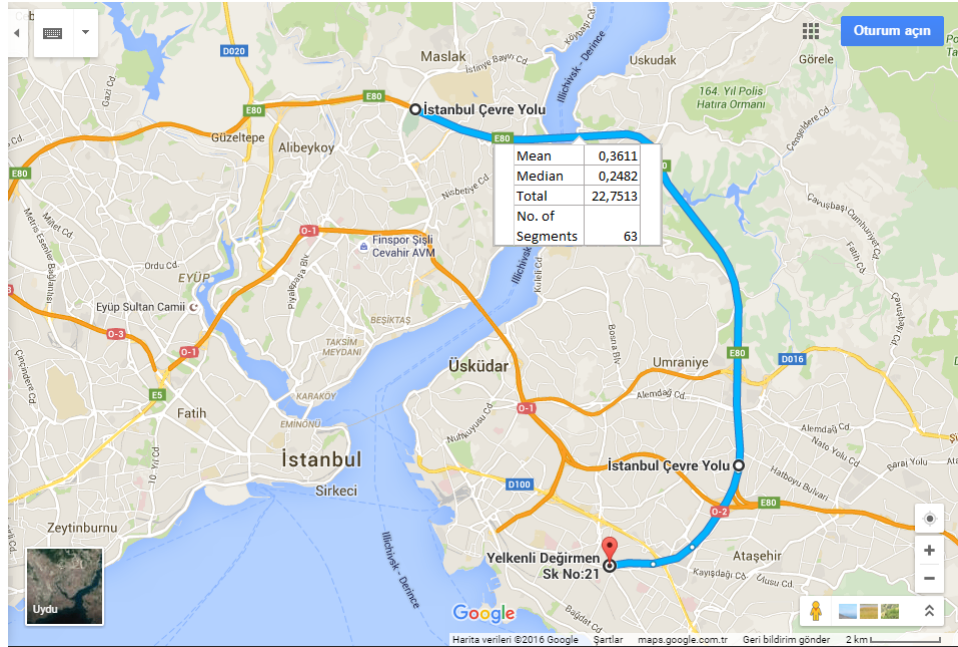


Figure 4.2: Map and Features of Route 2

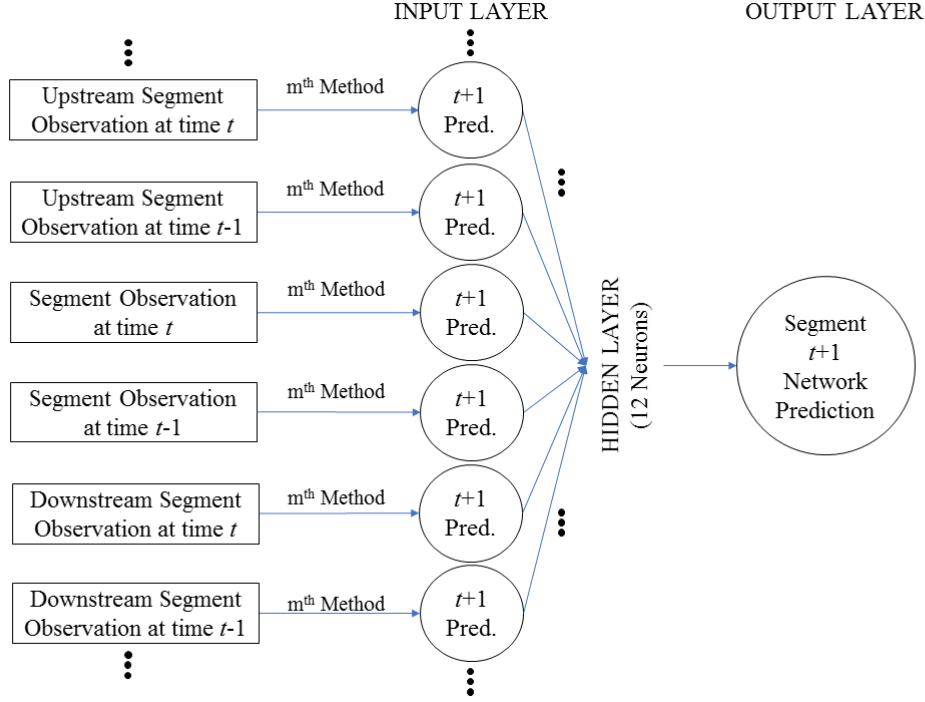


Figure 4.3: Temporally and Spatially Connected Approach (1-step Ahead Forecast) (TS-1 Network) (Representation of only a single (m^{th}) method out of a total of M methods)

4.2 Temporally and Spatially Connected Approach (1-step Ahead Forecast) (TS-1 Network)

For a representation of the TS-1 network, see Figure 4.3. Like in Ye et al. (2012), inputs of this network include forecasts from the neighboring segments and forecasts from 1-step behind. For every prediction method, there are six input neurons and they forecast the speeds at $(t + 1)^{st}$ time. While two of them forecast the speeds for the segment in question, another pair of neurons forecast the speeds for the neighboring upstream segment (U.S.), and the last pair performs forecasts for the neighboring downstream segment (D.S.). One of each pair of input neurons perform the forecast based on the speed at t^{th} time, and the other does the same based on the speed at the $(t - 1)^{st}$ time. Also, there are 12 neurons in the hidden layer and a single output neuron in the output layer. In short, this network has $12 \times ((6 \times M) + 1) + (12 + 1) = 72 \times M + 25$ weight parameters where M is the number of predictive methods used.

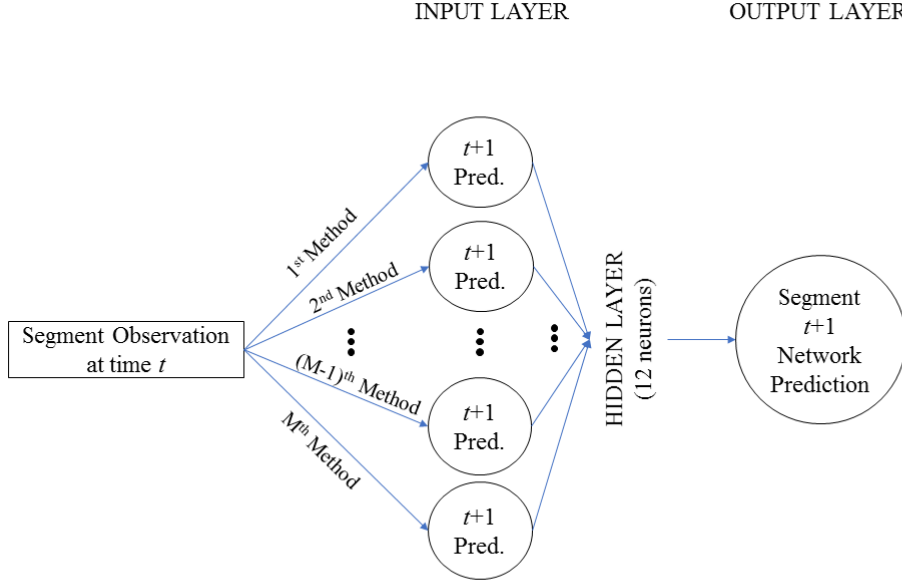


Figure 4.4: Single Segment Approach (1-step Ahead Forecast) (SS-1 Network)

4.3 Single Segment Approach (1-step Ahead Forecast) (SS-1 Network)

For a representation of the SS-1 network, see Figure 4.4. This approach is similar to the previous one; however, there is only a single input neuron for every prediction method this time and that is the prediction of the speed value at $(t + 1)^{st}$ time with respect to the speed observation at t^{th} time. This network also has 12 neurons in the only hidden layer and a single output neuron in the output layer. Thus, this network has $12 \times (M + 1) + (12 + 1) = 12 \times M + 25$ weight parameters where M is the number of predictive methods used.

4.4 Single Segment Approach (Multi-step Ahead Forecast) (SS-M Network)

For a representation of the SS-M network, see Figure 4.5. This method is similar to the previous methods with one crucial difference: This network performs multi-step ahead forecasts. For a forecast over the next N timesteps, every prediction method has N input neurons. First input neuron contains the prediction for the speed at $(t + 1)^{st}$ time, the second neuron is for the prediction for the speed at $(t + 2)^{nd}$ time, and so on. The hidden layer in this network contains 50 neurons and it has N output neurons with each neuron

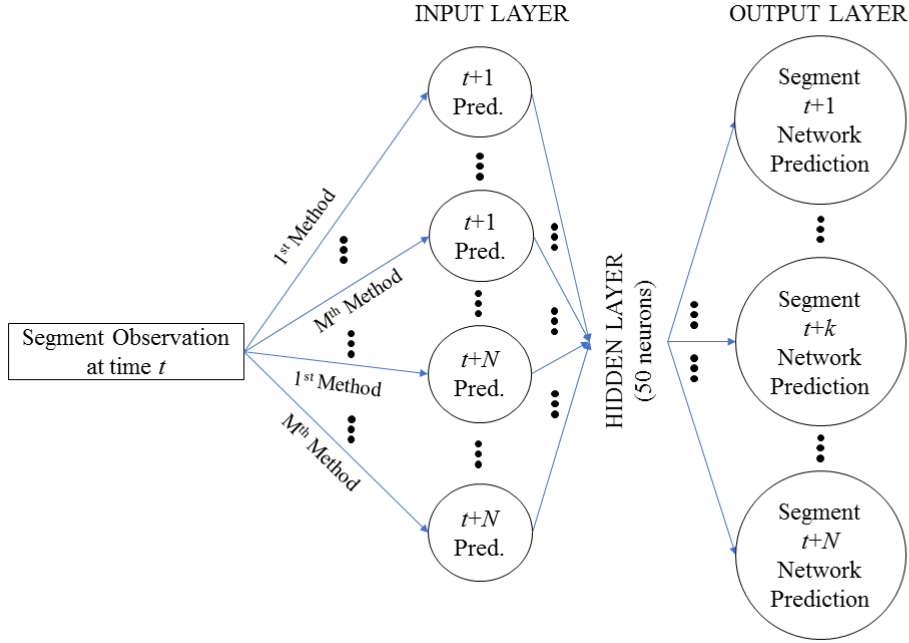


Figure 4.5: Single Segment Approach (Multi-step Ahead Forecast) (SS-M Network)

outputting a prediction for its corresponding step-ahead time. The reason SS-M contains 50 hidden neurons while other networks have 12 hidden neurons is because the number of input and output neurons is significantly higher compared to the others. This number, of course, could have been determined with a more thorough analysis. This network has $50 \times ((M \times N) + 1) + (50 + 1) = 50 \times M \times N + 101$ weight parameters where M is the number of predictive methods used and N is the forecasting horizon.

4.5 Recursive Single Segment Approach (Multi-step Ahead Forecast) (RSS-M Network)

For a representation of the RSS-M network, see Figure 4.6. This method is an addition to the second approach. The network has the same number of input, hidden, and output neurons. However, in this network, every 1-step ahead forecast output is treated as the actual observed value and a new 1-step ahead forecast is made with this input until the N^{th} time in the future. Finally, this network has $12 \times (M + 1) + (12 + 1) = 12 \times M + 25$ weight parameters where M is the number of predictive methods used.

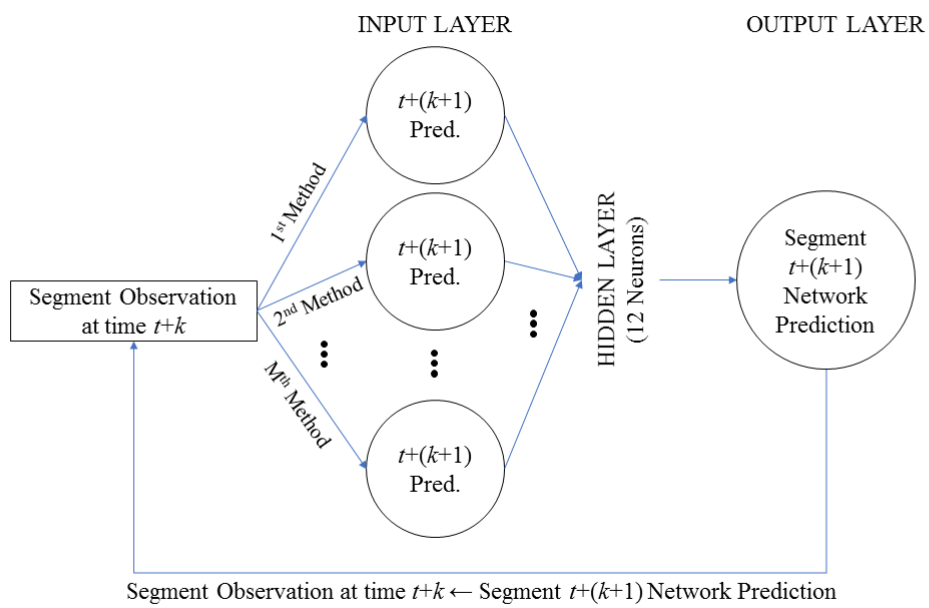


Figure 4.6: Recursive Single Segment Approach (Multi-step Ahead Forecast) (RSS-M Network)

Chapter 5

Computational Results

The computational experiments were carried on a workstation with a 64-bit Windows 7 Professional operating system, a memory of 128 GB, and two 10-core Intel Xeon CPU E5-2640 v4 @ 2.40 GHz processors. We have implemented the FFNN using Keras library built on Theano with Python 2.7 on Jupyter Notebook via Continuum Anaconda. For data analysis and computations, openpyxl, matplotlib, pandas, NumPy, SciPy, scikit-learn libraries are used.

In the following analyses, 90% (4.5 months) of the dataset was used for training and the remaining 10% (15 days) was used for test. Activation functions for both input–hidden layer transition and hidden–output layer transition were purely linear functions. Furthermore, as noted before, the networks tried to minimize the mean absolute deviation. Also, unless otherwise stated, the parameters of the predictive methods are as follows: Weighted moving average method uses a 3-minute horizon with 0.25, 0.50, and 0.25 as corresponding weights in the analyses. Simple exponential smoothing method takes 0.50 as alpha; Holt’s method takes 0.20 and 0.30 as its alpha and beta values, respectively; and finally, Winters method takes 0.45, 0.30, and 0.20 as its alpha, beta and gamma values, respectively. When we tried to optimize the parameters, we obtained exceptionally low and high values. Since too high *smoothing constants* would yield a too smooth and too low *smoothing constants* would yield a too erratic prediction (Croarkin et al., 2006), we decided to decrease the values empirically.

5.1 1-step Ahead Predictions

Segment	Route 1	Route 2
	Length (km)	Length (km)
5	0.2170	0.4219
15	0.0441	0.4419
18	0.0507	0.4443
50	0.0454	0.0072
Average	0.0893	0.3288

Table 5.1: Lengths of the Analyzed Segments for 1-step Ahead Prediction

Methods	Setups	Route 1		Route 2	
		MAD (km/h)		MAD (km/h)	
		Train	Test	Train	Test
N			0.0232		1.5643
M			0.0448		2.3202
S			0.0438		2.0416
H			0.0513		3.2077
W			0.0483		1.9861
NMS	TS-1	0.0725	0.0725	1.1317	1.9133
NMS	SS-1	0.0500	0.0583	1.1708	1.2742
NMW	TS-1	0.0775	0.0800	1.1800	1.3350
NMW	SS-1	0.0550	0.0575	1.1675	1.2100
NMSHW	TS-1	0.0700	0.0858	1.2082	1.5618
NMSHW	SS-1	0.0567	0.0650	1.1975	1.4308

Table 5.2: 1-step Ahead Prediction Average of MAD (km/h) values of Different Setups and Methods for Both Routes over 500 epochs and with a minibatch size of 1000

Both routes are tested with two different 1-step ahead setups introduced in Chapter 4 over the same four segments (#5, #15, #18, #50) and with a batch size of 1000 for 100, 300, and 500 epochs. The lengths of the segments analyzed with 1-step ahead forecast can be found in Table 5.1. We also employed regularization and trained them with Adam training algorithm (as it is an improvement to SGD (Kingma and Ba, 2014)). Both routes employed the NMS, NMW, and NMSHW predictive method combinations for 1-step analyses.

While the average results can be found in Table 5.2 and the detailed results are explained below, related tables of 1-step ahead forecast methods can be found in Tables A.1 to A.8 in Appendix A.

For both routes and for all methods, SS-1 is superior to TS-1 on average (see Table 5.2). This suggests that taking neighboring segments and previous time steps into consideration, unlike the findings of Ye et al. (2012), turns out to be redundant. However, it is important to note that their findings were based on data recorded at irregular

intervals while we have a complete data. It is also important to note that the scope of our custom data cleaning method covered only the temporal irregularities while including a spatial cleaning as well could have helped TS-1 perform better. Furthermore, all methods perform better on the shorter segments for both routes, which supports the findings of Wang et al. (2015). Apparently, a shorter segment’s readings would not fluctuate much in just 1-step later, while the same cannot be said for a longer segment; hence, the differences between error values. The results are also benchmarked with individual predictive methods. We observe that the Naïve method is a powerful predictive method by itself. Especially for Route 1, no other option comes close to its error values. For Route 2, even though it is not the best predictive method, it falls short by only a MAD difference of 0.3543 km/h. Considering the individual predictive methods take much less time than constructing and training a neural network, it might be more reasonable to perform the 1-step ahead predictions with the Naïve method.

5.2 Multi-step Ahead Predictions

Route 1		Route 2	
Segment	Length (km)	Segment	Length (km)
18	0.0507	1	0.5538
19	0.0461	2	0.3711
20	0.0466	3	0.2252
46	0.0822	6	0.3849
47	0.0813	7	0.3174
48	0.1019	8	0.3567
94	0.2084	17	0.4784
95	0.1617	18	0.4443
96	0.1069	19	0.5675
121	0.0738	26	0.7029
122	0.0887	27	0.9742
123	0.0682	28	1.1210
Average	0.093		0.5415

Table 5.3: Lengths of the Analyzed Segments for Multi-step Ahead Prediction

Setup	Method	Route 1	Route 2
	W	0.409	7.403
	S	0.403	7.436
	M	0.404	7.555
	N	0.394	7.533
RSS–M	NMS	1.651	13.554
RSS–M	NMW	1.402	12.985
RSS–M	NMSW	1.437	13.027
SS–M	NMS	0.299	6.175
SS–M	NMW	0.299	6.165
SS–M	NMSW	0.304	6.155

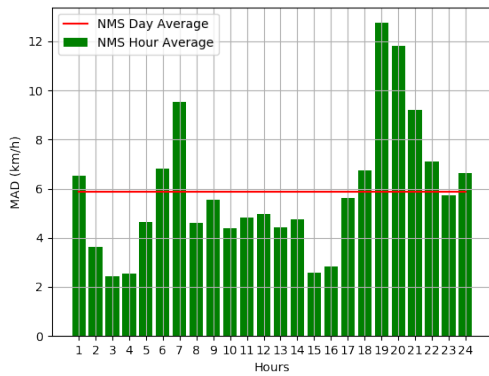
Table 5.4: Multi-step Ahead Prediction Average of MAD (km/h) values of Different Setups and Methods for Both Routes over 30 epochs and with a minibatch size of 1000

Both routes are also tested with two different multi-step (30 minutes) ahead setups introduced in Chapter 4 over twelve segments each. This time, different set of segments were selected to see the performance of the multi-step ahead setups in varying conditions. The lengths of the analyzed segments can be seen in Table 5.3. We employed NMS, NMW, and NMSW predictive methods. Multi-step ahead predictions work on methods not employing Holt’s method because RSS–M setup cause the predictions with a trend component to explode. Additionally, that is why the Winters method components consider the data without a trend component. Since the RSS–M setup is not compatible with the method, to have a fair comparison between the multi-step ahead prediction methods, Holt’s method (or, trend in general) was not included in the combinations. These tests were exclusively performed with the Adam optimizer. Also, number of epochs and number of batch size values are fixed to 30 and 1000, respectively. We benchmarked our results with the multi-step ahead forecasts of Naïve, Weighted Moving Average, Simple Exponential Smoothing, and Winters predictive methods separately.

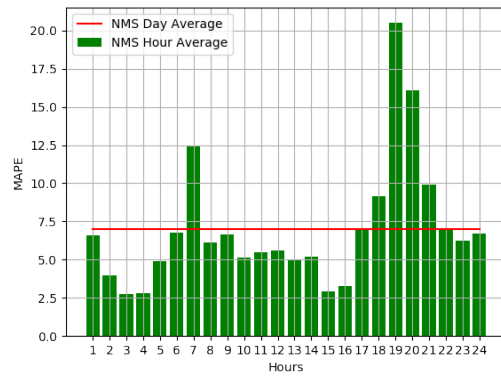
While the average results can be found in Table 5.4 and the detailed results are explained below, tables of the detailed results of multi-step ahead forecast methods can be found in Tables B.1 to B.24 in Appendix B.

Results of multi-step ahead forecasts show that RSS–M is not successful. In all segments, it performs worse than individual predictive methods. This is to show that treating a 1-step ahead prediction as the new observation would actually amplify the deviation as time progresses. On the other hand, SS–M, almost always, outperforms the individual methods. However, for Route 1, even the individual methods return error values that are less than 0.5 km/h. For Route 2, SS–M, usually, improves the predictions by 1–2 km/h, which is quite helpful.

It is also important to note the difference of the error values between Route 1 and Route 2. In all cases, Route 2 underperformed. A plausible reason for these poor error

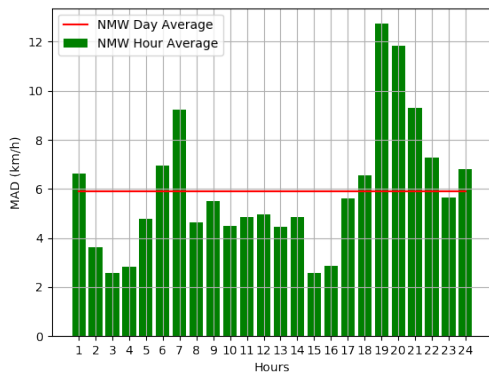


(a) Average MAD (km/h) values over 24 hours

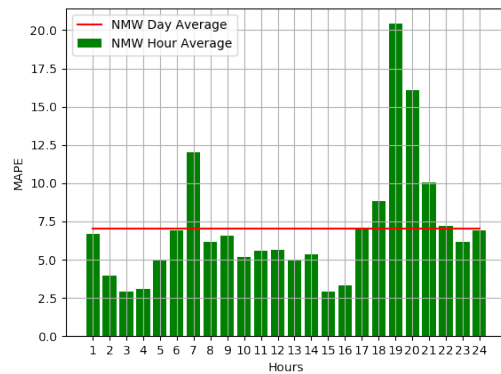


(b) Average MAPE values over 24 hours

Figure 5.1: NMS Method on SS-M over one day



(a) Average MAD (km/h) values over 24 hours

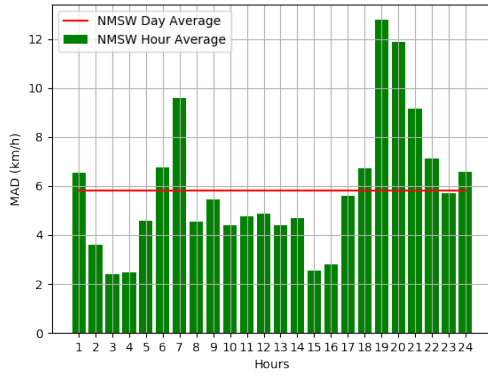


(b) Average MAPE values over 24 hours

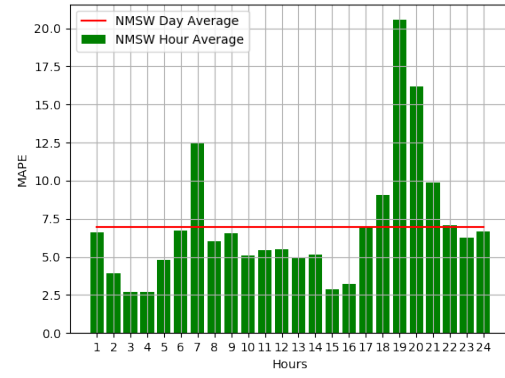
Figure 5.2: NMW Method on SS-M over one day

values seems to be the large length of the segments. While the high frequency of traffic lights on Route 1 might cause a more stable (and slow) cruising speed on every lane, a highway would have more variance in terms of speeds from one minute to the next gathering data from different lanes with different average speed values.

To further investigate the performance of multi-step ahead predictions, Figures 5.1, 5.2 and 5.3 are provided. In these figures, segment #18 of Route 2 is analyzed in more detail with respect to the observed speeds of February 27, 2017, Monday. This segment is located on the Asian side exit of the FSM bridge. The three methods have close error values throughout the day and on average. The rush hour predictions have the highest mean absolute deviation from the actual observations which is expected given the highly unpredictable characteristic of rush hour traffic. When mean absolute percent error (MAPE) values, as given in (5.1), are compared with MAD values, we can see the poor results of morning and afternoon rush hours stressed even further.



(a) Average MAD (km/h) values over 24 hours



(b) Average MAPE values over 24 hours

Figure 5.3: NMSW Method on SS-M over one day

$$MAPE = \frac{1}{n} \sum_{i=1}^n \frac{|e_i|}{s_i} \quad (5.1)$$

where n is the number of forecasts, $|e_i|$ is the absolute error, and s_i is the observation.

Chapter 6

Concluding Remarks and Future Research

In this work, we tackled the problems of short and medium-term speed prediction on two different routes in Istanbul. We proposed four neural network setups that take forecasts from individual predictive methods to perform predictions on a segment. Two of these setups perform 1-minute ahead predictions while the remaining two predict the next 30 minutes. These neural networks were trained with Adam optimizer and employed mean absolute deviation error as their loss functions. To observe the performances of the introduced networks, we benchmarked our results with individual predictive methods. Our findings indicate that machine learning methods are useful for this task. Our 1-step ahead analyses returned mean absolute deviation values of less than 0.55 km/h for Route 1 and less than 1.75 for Route 2 with the appropriate parameter combinations, considerable improvements over the results of Ye et al. (2012). Our findings also show that the Naïve method is a powerful individual predictive method when employed on 1-step ahead prediction. However, our single segment multi-step ahead forecasts are quite successful when compared to the conventional predictive methods. The difference in error values for two separate routes show that a route with shorter segments would return lower error values. The results show that neural networks are capable of providing accurate predictions with appropriate parameter selection.

To improve the performance of our approach, several extensions can be considered. First, the parameters of individual prediction methods can be fine tuned independently. In this work, these parameters were fixed at empirically chosen values. Secondly, it is also possible to change the definition of *season* to capture the seasonal characteristics of sections of a day instead of using minutely seasons which may be very precise but inefficient. Thirdly, increasing the number of epochs from 30 would return better results in multi-step ahead forecasts as in 1-step ahead prediction. It is also important to note that the custom data cleaning process contains empirically chosen parameters. The 5 km/h speed difference limit, 3 standard deviations and the decision to decrease the maximum standard deviation limit by 0.85 at every iteration were all determined to have a balanced

smoothing process. A data collection source like occupied taxis of Ye et al. (2012) would be much more accurate to the actual traffic flow. On the other hand, it might not be possible to cover all of the streets every minute on the road network. Finally, as mentioned in Chapter 5, performing a spatial cleaning in addition to a temporal one might return improved results.

There are several possible future directions this research can take. First, recurrent neural networks (RNN) can be implemented. Recurrent neural networks are currently being used on text generation problems, and it is likely that they are suitable for time series sequence generation. A problem-specific advantage of RNNs is that they would not need the predictive methods and significantly decrease the problem size. Among RNNs, Long short-term memory (LSTM) networks and gated recurrent units (GRU) can be employed. Secondly, employing Winters prediction over a horizon longer than 30 minutes might reflect seasonal characteristics better and provide improvements. Thirdly, random forest regression is also a simple method we can use to combine the individual prediction methods. Finally, training and testing on respective days instead of full data might yield better results as well as trying to accurately predict the speed on a network of segments instead of a single one.

Bibliography

Ethem Alpaydın. *Introduction to Machine Learning*. MIT press, 2014.

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

François Chollet. Keras. <https://github.com/fchollet/keras>, 2015.

Graham Cookson and Bob Pishue. INRIX Global Traffic Scorecard. Technical report, INRIX Research, 2017.

Carroll Croarkin, Paul Tobias, JJ Filliben, et al. *NIST/SEMATECH e-Handbook of Statistical Methods*. 2006.

Justin Dauwels, Aamer Aslam, Muhammad Tayyab Asif, Xinyue Zhao, Nikola Mitrovic, Andrzej Cichocki, and Patrick Jaillet. Predicting traffic speed in urban transportation subnetworks for multiple horizons. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pages 547–552. IEEE, 2014.

Patrice Delmas. Computer Graphics and Image Processing (COMPSCI 373) Lecture Notes: Image Filtering Median Filtering. 2010.

Soufiene Djahel, Ronan Doolan, Gabriel-Miro Muntean, and John Murphy. A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches. *IEEE Communications Surveys & Tutorials*, 17(1):125–151, 2015.

Stephen Dunne and Bidisha Ghosh. Weather adaptive traffic prediction using new-wavelet models. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):370–379, 2013.

Ericsson AB. Ericsson Mobility Report: On the Pulse of the Networked Society. *Ericsson, Sweden, Tech. Rep. EAB-14*, 61078, 2015.

European Commission Directorate-General for Mobility and Transport. *White Paper on Transport: Roadmap to a Single European Transport Area: Towards a Competitive and Resource-efficient Transport System*. Publications Office of the European Union, 2011.

- European Commission. A European Strategy for Low-Emission Mobility. http://europa.eu/rapid/press-release_MEMO-16-2497_en.htm, 2016.
- Luana Georgescu, David Zeitler, and Charles Robert Standridge. Intelligent transportation system real time traffic speed prediction with minimal data. *Journal of Industrial Engineering and Management*, 5(2):431, 2012.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.
- Cristina Guerreiro, Alberto González Ortiz, Frank de Leeuw, Mar Viana, and Jan Horálek. *Air Quality in Europe – 2016 Report*. Publications Office of the European Union, 2016.
- Wenbin Hu, Huan Wang, and Liping Yan. An actual urban traffic simulation model for predicting and avoiding traffic congestion. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 2681–2686. IEEE, 2014.
- John D Hunter. Matplotlib: A 2D Graphics Environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.
- J.D. Power. Vehicle owners ask for smartphone integration and better voice controls, as satisfaction with factory-installed navigation systems declines. www.jdpower.com/press-releases/2012-us-navigation-usage-and-satisfaction-study, 2013.
- James W Jenness, Neil D Lerner, Steven D Mazor, J Scott Osberg, and Brian C Tefft. Use of advanced in-vehicle technology by young and older early adopters. Survey results on navigation systems. Technical report, 2008.
- Eric Jones, Travis Oliphant, and Pearu Peterson. SciPy: Open Source Scientific Tools for Python. 2014.
- RAI Khan, B Landfeldt, and A Dhamdher. Predicting travel times in dense and highly varying road traffic networks using starima models. Technical report, Technical report, School of Information Technologies, The University of Sydney and National ICT Australia, 2012.
- Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter Notebooks – A publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.

- Fuliang Li, Junfeng Gong, Yunyi Liang, and Jiali Zhou. Real-time congestion prediction for urban arterials using adaptive data-driven methods. *Multimedia Tools and Applications*, 75(24):17573–17592, 2016.
- Zilu Liang and Yasushi Wakahara. Real-time urban traffic amount prediction models for dynamic route guidance systems. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):85, 2014.
- Marco Lippi, Matteo Bertini, and Paolo Frasconi. Collective traffic forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 259–273. Springer, 2010.
- Xiaolei Ma, Haiyang Yu, Yunpeng Wang, and Yinhai Wang. Large-scale transportation network congestion evolution prediction using deep learning theory. *PloS one*, 10(3): e0119044, 2015.
- John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the Dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12, 2006.
- Wes McKinney et al. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. SciPy Austin, TX, 2010.
- Jungme Park, Dai Li, Yi L Murphey, Johannes Kristinsson, Ryan McGee, Ming Kuang, and Tony Phillips. Real time vehicle speed prediction using a neural network traffic model. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2991–2996. IEEE, 2011.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.
- Pew Research Center. Mobile Fact Sheet. Technical report, 2017.
- Jacob Poushter. Smartphone ownership and internet usage continues to climb in emerging economies. *Pew Research Center*, 22, 2016.
- Python Software Foundation. Python Language Reference, version 2.7. <http://www.python.org/>, 2014.
- Republic of Turkey Ministry of Transport, Maritime Affairs and Communications. Ulusal Akıllı Ulaşım Sistemleri Strateji Belgesi (2014–2023) ve eki Eylem Planı (2014-2016). Technical report, 2014.

- Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- Aaron Smith et al. US smartphone use in 2015. *Pew Research Center*, 1, 2015.
- William J Stevenson. *Operations Management: Theory and Practice*. McGraw-Hill/Irwin, 2012.
- The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.
- uShip.com. uShip survey: 7 in 10 truckers using mobile more often for business; smartphones, GPS units on collision course? <https://goo.gl/nf3t8M>, 2012.
- Eleni I Vlahogianni, Matthew G Karlaftis, and John C Golias. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 43:3–19, 2014.
- T Vonk, T Van Rooijen, J Hogema, and P Feenstra. Do navigation systems improve traffic safety. *Report TNO*, 2007.
- Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.
- Jingyuan Wang, Yu Mao, Jing Li, Zhang Xiong, and Wen-Xu Wang. Predictability of road traffic and congestion in urban areas. *PloS One*, 10(4):e0121825, 2015.
- Yuanchang Xie, Kaiguang Zhao, Ying Sun, and Dawei Chen. Gaussian processes for short-term traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, (2165):69–78, 2010.
- Su Yang. On feature selection for traffic congestion prediction. *Transportation Research Part C: Emerging Technologies*, 26:160–169, 2013.
- Qing Ye, Wai Yuen Szeto, and Sze Chun Wong. Short-term traffic speed forecasting based on data recorded at irregular intervals. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1727–1737, 2012.

Appendix A

1-Step Ahead Results

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
TS-1	5	NMS	500	1000	0.08	0.09	1296.08
SS-1	5	NMW	500	1000	0.08	0.09	1443.72
SS-1	5	NMSHW	500	1000	0.07	0.11	273.10
SS-1	5	NMSHW	300	1000	0.07	0.11	156.32
SS-1	5	NMS	500	1000	0.10	0.11	1103.15
TS-1	5	NMS	100	1000	0.11	0.12	323.50
SS-1	5	NMS	300	1000	0.11	0.12	910.93
SS-1	5	NMS	100	1000	0.12	0.12	412.52
TS-1	5	NMSHW	300	1000	0.09	0.13	457.37
TS-1	5	NMW	500	1000	0.12	0.13	480.42
TS-1	5	NMSHW	500	1000	0.11	0.15	2078.76
SS-1	5	NMSHW	100	1000	0.14	0.15	90.34
TS-1	5	NMSHW	100	1000	0.12	0.15	307.43
TS-1	5	NMS	300	1000	0.15	0.15	663.22

Table A.1: 1-step Ahead Prediction for Segment #5 of Route 1 (0.2170 km in length)

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
SS-1	15	NMS	100	1000	0.01	0.01	552.40
SS-1	15	NMSHW	300	1000	0.02	0.02	266.43
SS-1	15	NMSHW	100	1000	0.02	0.02	54.32
SS-1	15	NMS	300	1000	0.02	0.02	842.82
SS-1	15	NMSHW	500	1000	0.03	0.03	508.15
TS-1	15	NMS	500	1000	0.03	0.03	1152.13
SS-1	15	NMW	500	1000	0.03	0.03	1925.36
TS-1	15	NMS	300	1000	0.04	0.04	687.38
TS-1	15	NMS	100	1000	0.04	0.04	183.01
TS-1	15	NMSHW	100	1000	0.04	0.04	665.44
TS-1	15	NMW	500	1000	0.05	0.05	385.54
TS-1	15	NMSHW	300	1000	0.05	0.05	2286.84
TS-1	15	NMSHW	500	1000	0.06	0.06	1416.96
SS-1	15	NMS	500	1000	0.06	0.06	789.38

Table A.2: 1-step Ahead Prediction for Segment #15 of Route 1 (0.0441 km in length)

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
SS-1	18	NMS	300	1000	0.07	0.08	154.07
SS-1	18	NMSHW	500	1000	0.07	0.08	477.62
SS-1	18	NMS	100	1000	0.08	0.08	527.67
SS-1	18	NMW	500	1000	0.08	0.08	2037.79
SS-1	18	NMS	500	1000	0.09	0.09	1345.93
SS-1	18	NMSHW	300	1000	0.06	0.09	228.50
TS-1	18	NMS	100	1000	0.10	0.10	258.11
TS-1	18	NMSHW	500	1000	0.09	0.10	2831.35
TS-1	18	NMSHW	300	1000	0.09	0.11	629.86
TS-1	18	NMW	500	1000	0.11	0.11	632.78
TS-1	18	NMS	300	1000	0.11	0.11	686.15
SS-1	18	NMSHW	100	1000	0.08	0.11	109.91
TS-1	18	NMS	500	1000	0.12	0.12	735.50
TS-1	18	NMSHW	100	1000	0.08	0.13	279.33

Table A.3: 1-step Ahead Prediction for Segment #18 of Route 1 (0.0507 km in length)

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
SS-1	50	NMS	500	1000	0.01	0.01	702.37
SS-1	50	NMS	100	1000	0.01	0.01	200.69
TS-1	50	NMS	300	1000	0.02	0.01	517.67
SS-1	50	NMS	300	1000	0.01	0.01	596.49
SS-1	50	NMW	500	1000	0.01	0.01	1698.55
SS-1	50	NMSHW	300	1000	0.01	0.02	439.94
SS-1	50	NMSHW	500	1000	0.02	0.02	953.06
TS-1	50	NMS	100	1000	0.02	0.02	332.07
SS-1	50	NMSHW	100	1000	0.02	0.02	133.06
TS-1	50	NMW	500	1000	0.03	0.03	358.52
TS-1	50	NMSHW	300	1000	0.03	0.03	1519.08
TS-1	50	NMSHW	500	1000	0.03	0.03	1024.43
TS-1	50	NMS	500	1000	0.05	0.04	924.16
TS-1	50	NMSHW	100	1000	0.05	0.05	173.71

Table A.4: 1-step Ahead Prediction for Segment #50 of Route 1 (0.0454 km in length)

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
TS-1	5	NMW	500	1000	1.71	1.71	520.28
TS-1	5	NMS	500	1000	1.71	1.71	893.60
TS-1	5	NMS	100	1000	1.75	1.75	176.72
TS-1	5	NMS	300	1000	1.77	1.76	639.02
SS-1	5	NMS	500	1000	1.82	1.82	863.10
SS-1	5	NMS	300	1000	1.83	1.82	349.76
SS-1	5	NMS	100	1000	1.85	1.85	637.24
SS-1	5	NMW	500	1000	1.85	1.86	2080.85
SS-1	5	NMSHW	500	1000	1.82	1.87	1036.11
TS-1	5	NMSHW	100	1000	1.77	2.02	380.00
TS-1	5	NMSHW	500	1000	1.81	2.30	2112.99
SS-1	5	NMSHW	300	1000	1.86	2.40	481.34
SS-1	5	NMSHW	100	1000	1.91	2.66	224.57
TS-1	5	NMSHW	300	1000	1.83	2.81	1592.85

Table A.5: 1-step Ahead Prediction for Segment #5 of Route 2 (0.4219 km in length)

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
TS-1	15	NMW	500	1000	1.12	1.17	333.19
TS-1	15	NMS	100	1000	1.15	1.19	82.71
SS-1	15	NMS	300	1000	1.21	1.27	332.15
SS-1	15	NMS	100	1000	1.21	1.27	283.84
TS-1	15	NMS	300	1000	1.23	1.28	690.91
SS-1	15	NMS	500	1000	1.24	1.29	881.18
SS-1	15	NMW	500	1000	1.23	1.28	2042.19
SS-1	15	NMSHW	500	1000	1.22	1.29	231.92
SS-1	15	NMSHW	300	1000	1.25	1.47	491.54
TS-1	15	NMS	500	1000	1.71	1.77	823.03
TS-1	15	NMSHW	100	1000	1.35	1.92	225.59
TS-1	15	NMSHW	300	1000	1.54	2.64	1140.84
SS-1	15	NMSHW	100	1000	1.46	2.77	786.40
TS-1	15	NMSHW	500	1000	1.48	3.27	6298.04

Table A.6: 1-step Ahead Prediction for Segment #15 of Route 2 (0.4419 km in length)

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
TS-1	18	NMW	500	1000	1.38	1.43	570.49
TS-1	18	NMS	500	1000	1.40	1.45	807.46
TS-1	18	NMS	100	1000	1.42	1.46	171.69
TS-1	18	NMS	300	1000	1.42	1.47	283.80
SS-1	18	NMS	500	1000	1.48	1.53	155.12
SS-1	18	NMW	500	1000	1.49	1.55	1529.77
SS-1	18	NMS	100	1000	1.50	1.56	66.88
SS-1	18	NMSHW	500	1000	1.48	1.56	890.16
SS-1	18	NMS	300	1000	1.51	1.56	119.07
SS-1	18	NMSHW	300	1000	1.53	1.66	377.42
SS-1	18	NMSHW	100	1000	1.57	1.94	129.43
TS-1	18	NMSHW	500	1000	1.59	4.47	764.18
TS-1	18	NMSHW	300	1000	1.39	6.64	2220.88

Table A.7: 1-step Ahead Prediction for Segment #18 of Route 2 (0.4443 km in length)

Setup	Seg.	Method	Epochs	Batch	MAD (km/h)		Train Time (s)
					Train	Test	
SS-1	50	NMS	500	1000	0.09	0.10	254.19
SS-1	50	NMW	500	1000	0.12	0.13	1944.16
SS-1	50	NMS	300	1000	0.12	0.13	2236.66
SS-1	50	NMS	100	1000	0.14	0.14	32.70
TS-1	50	NMSHW	500	1000	0.09	0.14	768.18
TS-1	50	NMW	500	1000	0.15	0.15	486.78
SS-1	50	NMSHW	500	1000	0.08	0.16	734.96
SS-1	50	NMSHW	300	1000	0.10	0.18	349.85
SS-1	50	NMSHW	100	1000	0.12	0.18	171.43
TS-1	50	NMS	300	1000	0.18	0.19	434.04
TS-1	50	NMS	100	1000	0.18	0.19	165.22
TS-1	50	NMS	500	1000	0.19	0.19	3233.63
TS-1	50	NMSHW	300	1000	0.13	0.19	540.73
TS-1	50	NMSHW	100	1000	0.14	0.21	1103.66

Table A.8: 1-step Ahead Prediction for Segment #50 of Route 2 (0.0072 km in length)

Appendix B

Multi-Step Ahead Results

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.095	
	S		0.091	
	M		0.091	
	N		0.089	
RSS-M	NMS	0.031	0.390	24
RSS-M	NMW	0.059	0.542	22
RSS-M	NMSW	0.081	0.830	25
SS-M	NMS	0.146	0.074	1799
SS-M	NMW	0.141	0.068	1480
SS-M	NMSW	0.133	0.061	1821

Table B.1: Multi-step Ahead Prediction Results for Segment #18 of Route 1 (0.0507 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.679	
	S		0.671	
	M		0.672	
	N		0.658	
RSS-M	NMS	0.384	3.761	19
RSS-M	NMW	0.402	3.815	21
RSS-M	NMSW	0.197	1.929	21
SS-M	NMS	0.578	0.595	1826
SS-M	NMW	0.487	0.507	1551
SS-M	NMSW	0.542	0.559	1501

Table B.2: Multi-step Ahead Prediction Results for Segment #19 of Route 1 (0.0461 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.072	
	S		0.069	
	M		0.069	
	N		0.068	
RSS-M	NMS	0.066	0.747	24
RSS-M	NMW	0.084	0.851	28
RSS-M	NMSW	0.027	0.283	23
SS-M	NMS	0.111	0.051	2154
SS-M	NMW	0.120	0.060	1737
SS-M	NMSW	0.124	0.063	1885

Table B.3: Multi-step Ahead Prediction Results for Segment #20 of Route 1 (0.0466 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.840	
	S		0.831	
	M		0.832	
	N		0.812	
RSS-M	NMS	0.209	2.219	17
RSS-M	NMW	0.355	2.962	15
RSS-M	NMSW	0.434	4.407	16
SS-M	NMS	0.552	0.593	40
SS-M	NMW	0.579	0.621	1485
SS-M	NMSW	0.538	0.579	34

Table B.4: Multi-step Ahead Prediction Results for Segment #46 of Route 1 (0.0822 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.935	
	S		0.925	
	M		0.927	
	N		0.905	
RSS-M	NMS	0.409	4.168	25
RSS-M	NMW	0.378	4.012	20
RSS-M	NMSW	0.244	2.483	21
SS-M	NMS	0.662	0.697	64
SS-M	NMW	0.613	0.649	1750
SS-M	NMSW	0.689	0.724	52

Table B.5: Multi-step Ahead Prediction Results for Segment #47 of Route 1 (0.0813 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.106	
	S		0.102	
	M		0.102	
	N		0.100	
RSS-M	NMS	0.148	1.219	12
RSS-M	NMW	0.152	1.287	9
RSS-M	NMSW	0.143	1.161	14
SS-M	NMS	0.143	0.110	85
SS-M	NMW	0.183	0.15	1747
SS-M	NMSW	0.230	0.197	73

Table B.6: Multi-step Ahead Prediction Results for Segment #48 of Route 1 (0.1019 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.085	
	S		0.082	
	M		0.082	
	N		0.080	
RSS-M	NMS	0.101	0.924	11
RSS-M	NMW	0.087	0.900	13
RSS-M	NMSW	0.033	0.208	12
SS-M	NMS	0.127	0.068	136
SS-M	NMW	0.131	0.073	1735
SS-M	NMSW	0.131	0.073	95

Table B.7: Multi-step Ahead Prediction Results for Segment #94 of Route 1 (0.2084 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.106	
	S		0.104	
	M		0.104	
	N		0.101	
RSS-M	NMS	0.060	0.669	10
RSS-M	NMW	0.061	0.662	12
RSS-M	NMSW	0.061	0.653	12
SS-M	NMS	0.146	0.085	173
SS-M	NMW	0.142	0.082	1672
SS-M	NMSW	0.141	0.081	148

Table B.8: Multi-step Ahead Prediction Results for Segment #95 of Route 1 (0.1617 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.225	
	S		0.220	
	M		0.221	
	N		0.215	
RSS-M	NMS	0.054	0.492	13
RSS-M	NMW	0.071	0.806	15
RSS-M	NMSW	0.094	0.901	13
SS-M	NMS	0.230	0.174	163
SS-M	NMW	0.242	0.184	1857
SS-M	NMSW	0.244	0.184	172

Table B.9: Multi-step Ahead Prediction Results for Segment #96 of Route 1 (0.1069 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.415	
	S		0.408	
	M		0.408	
	N		0.399	
RSS-M	NMS	0.165	1.588	13
RSS-M	NMW	0.167	1.599	10
RSS-M	NMSW	0.164	1.563	12
SS-M	NMS	0.388	0.337	197
SS-M	NMW	0.400	0.346	1825
SS-M	NMSW	0.357	0.303	182

Table B.10: Multi-step Ahead Prediction Results for Segment #121 of Route 1 (0.0738 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		1.228	
	S		1.221	
	M		1.221	
	N		1.187	
RSS-M	NMS	0.162	2.698	12
RSS-M	NMW	0.124	2.415	14
RSS-M	NMSW	0.095	2.234	12
SS-M	NMS	0.358	0.712	217
SS-M	NMW	0.373	0.726	1867
SS-M	NMSW	0.379	0.734	213

Table B.11: Multi-step Ahead Prediction Results for Segment #122 of Route 1 (0.0887 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		0.120	
	S		0.116	
	M		0.116	
	N		0.113	
RSS-M	NMS	0.090	0.938	12
RSS-M	NMW	0.096	1.021	11
RSS-M	NMSW	0.063	0.588	12
SS-M	NMS	0.186	0.089	253
SS-M	NMW	0.214	0.118	1774
SS-M	NMSW	0.190	0.093	232

Table B.12: Multi-step Ahead Prediction Results for Segment #123 of Route 1 (0.0682 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		7.028	
	S		7.057	
	M		7.171	
	N		7.149	
RSS-M	NMS	2.179	13.964	24
RSS-M	NMW	2.183	14.454	25
RSS-M	NMSW	2.258	16.223	25
SS-M	NMS	5.741	5.808	54
SS-M	NMW	5.770	5.834	1398
SS-M	NMSW	5.751	5.817	41

Table B.13: Multi-step Ahead Prediction Results for Segment #1 of Route 2 (0.5538 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		7.251	
	S		7.285	
	M		7.405	
	N		7.381	
RSS-M	NMS	2.288	17.257	20
RSS-M	NMW	2.216	13.058	18
RSS-M	NMSW	2.169	12.593	14
SS-M	NMS	5.915	5.964	76
SS-M	NMW	5.905	5.948	1530
SS-M	NMSW	5.900	5.943	64

Table B.14: Multi-step Ahead Prediction Results for Segment #2 of Route 2 (0.3711 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		6.978	
	S		7.007	
	M		7.132	
	N		7.125	
RSS-M	NMS	2.128	10.078	15
RSS-M	NMW	2.254	14.957	16
RSS-M	NMSW	2.317	16.568	22
SS-M	NMS	5.745	5.874	97
SS-M	NMW	5.736	5.858	1570
SS-M	NMSW	5.711	5.838	88

Table B.15: Multi-step Ahead Prediction Results for Segment #3 of Route 2 (0.2252 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		8.203	
	S		8.233	
	M		8.382	
	N		8.384	
RSS-M	NMS	2.600	14.753	17
RSS-M	NMW	2.590	14.027	12
RSS-M	NMSW	2.590	13.883	14
SS-M	NMS	6.985	6.834	227
SS-M	NMW	7.004	6.850	1530
SS-M	NMSW	6.951	6.798	207

Table B.16: Multi-step Ahead Prediction Results for Segment #6 of Route 2 (0.3849 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		7.539	
	S		7.582	
	M		7.710	
	N		7.697	
RSS-M	NMS	2.425	16.689	22
RSS-M	NMW	2.296	14.985	20
RSS-M	NMSW	2.358	14.466	21
SS-M	NMS	6.770	6.281	268
SS-M	NMW	6.728	6.226	1732
SS-M	NMSW	6.703	6.200	252

Table B.17: Multi-step Ahead Prediction Results for Segment #7 of Route 2 (0.3174 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		6.916	
	S		6.961	
	M		7.089	
	N		7.090	
RSS-M	NMS	2.329	11.862	12
RSS-M	NMW	2.319	11.953	15
RSS-M	NMSW	2.325	11.331	26
SS-M	NMS	6.645	5.862	299
SS-M	NMW	6.636	5.848	1600
SS-M	NMSW	6.617	5.825	243

Table B.18: Multi-step Ahead Prediction Results for Segment #8 of Route 2 (0.3567 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		6.978	
	S		7.012	
	M		7.120	
	N		7.094	
RSS-M	NMS	2.190	14.974	13
RSS-M	NMW	2.512	15.057	9
RSS-M	NMSW	2.043	8.783	11
SS-M	NMS	6.117	5.841	561
SS-M	NMW	6.111	5.831	1633
SS-M	NMSW	6.082	5.800	531

Table B.19: Multi-step Ahead Prediction Results for Segment #17 of Route 2 (0.4784 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		7.112	
	S		7.145	
	M		7.251	
	N		7.218	
RSS-M	NMS	2.046	9.835	13
RSS-M	NMW	2.208	13.201	14
RSS-M	NMSW	2.160	12.585	12
SS-M	NMS	6.066	5.927	663
SS-M	NMW	6.058	5.918	1593
SS-M	NMSW	6.041	5.903	598

Table B.20: Multi-step Ahead Prediction Results for Segment #18 of Route 2 (0.4443 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		8.230	
	S		8.272	
	M		8.399	
	N		8.367	
RSS-M	NMS	2.400	15.541	12
RSS-M	NMW	2.402	15.467	9
RSS-M	NMSW	2.318	12.147	11
SS-M	NMS	6.882	6.772	705
SS-M	NMW	6.890	6.776	1691
SS-M	NMSW	6.850	6.733	666

Table B.21: Multi-step Ahead Prediction Results for Segment #19 of Route 2 (0.5675 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		8.504	
	S		8.525	
	M		8.632	
	N		8.561	
RSS-M	NMS	1.959	11.921	13
RSS-M	NMW	1.920	12.183	15
RSS-M	NMSW	1.996	12.915	12
SS-M	NMS	7.546	7.097	827
SS-M	NMW	7.529	7.027	1727
SS-M	NMSW	7.527	7.065	752

Table B.22: Multi-step Ahead Prediction Results for Segment #26 of Route 2 (0.7029 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		6.777	
	S		6.804	
	M		6.913	
	N		6.890	
RSS-M	NMS	2.043	10.517	12
RSS-M	NMW	2.108	12.029	11
RSS-M	NMSW	2.121	12.777	13
SS-M	NMS	6.236	5.689	918
SS-M	NMW	6.261	5.724	1841
SS-M	NMSW	6.298	5.766	845

Table B.23: Multi-step Ahead Prediction Results for Segment #27 of Route 2 (0.9742 km in length)

Setup	Method	MAD (km/h)		Time (s)
		Train	Test	
	W		7.317	
	S		7.343	
	M		7.461	
	N		7.436	
RSS-M	NMS	2.278	15.260	11
RSS-M	NMW	2.250	13.782	13
RSS-M	NMSW	2.187	12.054	10
SS-M	NMS	6.674	6.152	997
SS-M	NMW	6.659	6.137	1691
SS-M	NMSW	6.677	6.168	977

Table B.24: Multi-step Ahead Prediction Results for Segment #28 of Route 2 (1.121 km in length)