Forecasting Vehicle Traffic with Big Data

by

Hao Peng

(Under the Direction of John A. Miller)

Abstract

Traffic forecasting is an important issue in several aspects. It may help governments and city planners make better decisions in regards to an intelligent transportation system. Traffic app developers and everyday travelers/commuters would also be interested in such matters. This work contains an overview of the recent developments in the area of traffic forecasting. In recent years, the availability of large amounts of traffic data has paved the way for data scientists to train models with big data to obtain better accuracy. An extensive study on forecasting traffic flow is given, covering various statistical and machine learning models, while shedding light on the most recent and state-of-art modeling techniques in this field. Furthermore, we studied the traffic forecasting problem using a situation-aware approach. Differing from a purely data-driven modeling approach, in which the models are tasked with learning everything from the data, we have chosen to be proactively aware of traffic affecting situations that could help guide the model building process. Examples may include the appropriate selection or removal of certain features and the choice of training data when we are aware of certain events that may cause traffic patterns to deviate from the norm, such as a weather condition or a holiday. As a result, we can obtain forecasts that are generally more accurate and models that are more interpretable. Remaining aware of

certain situations can therefore effectively complement the popular data-driven modeling approach. We also present the Quadratic Extreme Learning Machine model in this work. The model generally exhibits improved performance over the standard Extreme Learning Machine model while remaining relatively efficient. It may be a viable alternative to the generally more computationally costly Neural Networks.

INDEX WORDS: Vehicle Traffic Forecasting, Big Data, Data Science, Machine Learning, Deep Learning, Time Series Analysis

Forecasting Vehicle Traffic with Big Data

by

Hao Peng

B.S., University of Georgia, 2013

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2019

Forecasting Vehicle Traffic with Big Data

by

Hao Peng

Approved:

Major Professor: John A. Miller

Committee: Ingrid Maria Hybinette
Khaled M. Rasheed

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
August 2019

To My Father, My Safety and Strength

To My Mother, My Comfort and Delight

To My Brother, My Help in Times of Trouble

To My Closest Friend, Ever Faithful and Always By My Side

Acknowledgments

I would like to first thank my family for their support during this Ph.D. process. It has been a long and meaningful journey, and I couldn't have done it without you.

I extend my gratitude to my major professor, Dr. John A. Miller. He is truly an example of a scholar with diligence and integrity.

I would like to thank my committee members, Dr. Hybinette and Dr. Rasheed, for their valuable feedback. I also greatly appreciate Dr. Ma's help in answering various questions about statistics that I've had.

Last but not least, thanks to my lab mates whom I have the pleasure of working with: Michael, Mustafa, Santosh, Casey, and Nick.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traffic forecasting is an issue that is of interest to many people who would like to avoid traffic congestions when commuting to work or traveling to other places. Traffic apps are developed to better guide travelers/commuters in the fastest ways to their destinations. Furthermore, governments and city planners are interested in traffic forecasting to gain insights that will help them make better decisions in regards to road expansions, smart traffic lights, and other aspects of an intelligent transportation system.

Realizing the need to collect large amounts of high-quality traffic data, many states have deployed large numbers of traffic sensors on their major highways and urban areas. The Caltrans Performance Measurement System[1] (PeMS) is one such system that provides real-time traffic data from more than 39,000 sensors in the state of California. Historical traffic data in 5-minute resolutions are also available, making big data analytics possible in the area of traffic forecasting.

The contributions of this work are as follows:

- An extensive literature review is provided, focusing on both the various models used in traffic forecasting and a chronological overview of the developments in this field.

[1] `http://pems.dot.ca.gov/`

- Using big data, the performance of various statistical and machine learning models traffic flow forecasting is evaluated.
- The impacts of incorporating spatially dependent data into multivariate forecasting models are studied and compared against the univariate cases.
- The performance of multi-step forecasts and the impacts of varying data resolutions are discussed.
- The trade-offs/pros and cons of the models in terms of accuracy, stability, computational cost, and ease of use are explored.
- A situation-aware approach to forecasting traffic is discussed. Being aware of situations such as holidays, special weather conditions, and the locations of sensors can be effectively used to guide the model building process and complement the popular data-driven modeling approach.
- The Quadratic Extreme Learning Machine model is presented. It generally improves upon the standard Extreme Learning Machine while remaining relatively efficient. Its performance can be competitive with Neural Networks.

The rest of this work is organized as follows: Chapter 2 provides an extensive literature review on both the various models that have been used in traffic forecasting and a chronological overview of the developments in this area. Chapter 3 is a submitted manuscript to *International Journal of Data Science and Analytics*. It focuses on evaluations of the effectiveness of the various models in a variety of settings. Chapter 4 is a manuscript to be submitted to *IEEE Transactions on Big Data*. The focus is to use a situation-aware approach to help with the model building process. The Quadratic Extreme Learning Machine model is also presented. Finally, Chapter 5 concludes the work with a summary of major findings.

Chapter 2

Literature Review

The recent revolutions in big data and deep learning have swept across many fields of study, and vehicle traffic forecasting is no exception. The availability of large quantities of sensor data and the ample hardware computing resources have enabled data scientists to pursue research in this direction. This section reviews literature in the field of vehicle traffic forecasting. It can be observed that many studies have shifted focus to applying deep learning models with big data in very recent years.

2.1 Overview of Traffic Forecasting Models

A plethora of techniques have been successfully applied to traffic forecasting. An overview of some of the most commonly used models and discussions of their effectiveness in regards to traffic forecasting is located in the subsequent sections.

2.1.1 Regression Models

One of the most important and fundamental techniques for analytics is Regression [Galton, 1886]. Sometimes it may be helpful to regress on polynomial powers of the input features.

Such a model is known as a polynomial regression model. The products of pairs of features, known as interaction/cross-terms, may also be incorporated into the polynomial regression models. To model data that exhibit periodic patterns, trigonometric regression may also be used, in which sine and cosine functions are used to transform the input features.

Both polynomial and trigonometric regression can certainly be used to capture the daily traffic patterns such as flow and speed. There have been a few forecasting studies that rely on local linear or polynomial regressions, which only consider data in the very recent past and aim to produce forecasts in the immediate future [Sun et al., 2003, Zhong et al., 2005, Yue et al., 2010]. In those scenarios, forecasts are produced through extrapolation. Others have attempted to apply additional forecasting techniques after the standard traffic patterns have been extracted using Trigonometric Regression [Zou et al., 2015, Tang et al., 2017].

Another type of Regression is known as Poisson Regression, which is a type of Generalized Linear Model that assumes the response follows a Poisson distribution It is typically used to model the number of times a particular event occurs within a given time interval, and the most common traffic application is to predict traffic collisions [Ma et al., 2008, El-Basyouny and Sayed, 2009, Abdel-Aty and Radwan, 2000, Hadayeghi et al., 2010, Li et al., 2013]. Rarely is Poisson Regression used to forecast traffic flow, though there has been one recent work that attempted it in that direction [Okawa et al., 2017]. To apply Poisson Regression to forecast flow, the flow data must be divided into separate time slots and each time slot would require a separate Poisson Regression model be trained to forecast flow from the same time slot (e.g., let Y represents the number of vehicles that pass through this road/sensor from 10:00 AM to 10:15 AM). Speed and travel times are both continuous variables and therefore not suitable for applying Poisson Regression.

2.1.2 Univariate Time Series Models

In univariate time series, the AutoRegressive Integrated Moving Average (ARIMA) [Box and Jenkins, 1970] family of models is by far the most common and prevalent classical statistical tool for forecasting. To train an ARIMA model, an order of Integration (I) must be chosen first to difference a non-stationary time series as necessary. Seasonal AR and MA components may also be incorporate to build a Seasonal ARIMA (SARIMA) model. Other eXogenous (X) covariates may also be added to construct ARIMAX or SARIMAX models.

Many researchers in the field of traffic forecasting have chosen to apply univariate time series methods due to its simplicity. Some of the earliest studies that applied ARIMA to traffic forecasting can be traced back to [Ahmed and Cook, 1979] and [Levin and Tsao, 1980]. The effectiveness of SARIMA has been studied in work like [Williams and Hoel, 2003]. The ARIMA family of models has the advantage of simple model structure, quick training time and in many cases, decent forecasts. Nevertheless, they can suffer from issues such as the tendency to miss extreme peaks and valleys [Davis et al., 1990, Hamed et al., 1995], which are common scenarios in traffic patterns. They are also unable to take advantage of other useful factors such as spatial dependencies by including the traffic information in the neighboring road. ARIMAX and SARIMAX models can be used to alleviate some of these issues by allowing exogenous variables. Some related studies include [Cools et al., 2009, Tsirigotis et al., 2012, Wu et al., 2014], in which the exogenous variables can include the day of the week, a representation of a clustered traffic pattern, weather data like rainfall, and other related traffic data such as flow and percentage of trucks. Multivariate generalizations of the ARIMA family of models are also commonly used to address these issues and are discussed in more detail in Section 2.1.3.

Another classical statistical univariate time series forecasting model is Exponential Smoothing (ES) [Brown, 1956, Holt Charles, 1957]. The Exponential Smoothing techniques, like ARIMA, are simple to use and can produce somewhat decent forecasts. A study in [Chrobok

et al., 2004] found Exponential Smoothing to work well, at least in the immediate short terms but then was outperformed by historical averages in longer terms. In [Castro-Neto et al., 2009], [Hong et al., 2011b] and [Guo et al., 2014], Exponential Smoothing was used for comparisons with other better performing techniques. Work in [Tan et al., 2009] used Exponential Smoothing as one of the building blocks of an aggregated model. A study in [Chan et al., 2012] used Exponential Smoothing as a preprocessing technique to smooth the data before training a Neural Network. It can be seen in these studies that Exponential Smoothing is rarely the center of attention and the best performing model.

Both ARIMA and Exponential Smoothing, as univariate time series models, have existed for decades. Plenty of research has been devoted to these models. Inevitably, due to the recent progress in deep learning, there seems to be a significant leap from the traditional statistical/theory-based analytics approach to a more data-driven, computational intelligence and data-mining based approach [Vlahogianni et al., 2014]. For the lack of better terms, ARIMA and Exponential Smoothing are somewhat outdated and less and less research has been devoted to these models in the more recent years, except as a comparison technique to shine the spotlight on other models. Nevertheless, they have been proven to be simple and decent techniques that can serve as the new floor in performance for research in fields like traffic forecasting.

2.1.3 Multivariate Time Series Models

In Multivariate Time Series analysis, relevant time series can be incorporated into the model to help to forecast the primary time series or all time series simultaneously. Relevant information can include traffic data from neighboring upstream and downstream sensors, other traffic parameters (e.g., using speed data to help forecast flow), or data that can have significant impacts on traffic, such as weather, special events and holidays. Multivariate Time Series models include the multivariate generalization of the univariate ARIMA family of

models, known as the VARMA family of models, as well as the specific Space-Time variant, STARIMA family of models. Notable studies that employ multivariate time series models to forecast traffic include:

- [Kamarianakis and Prastacos, 2005]: Forecasting flow using 7.5-minute resolution data from 25 sensors for two months in Athens, Greece. The authors concluded that the STARIMA model achieved a somewhat similar performance with an ARIMA model.

- [Chandra and Al-Deek, 2008]: Forecasting traffic speed using 5-minute resolution speed data from 5 sensors on I-4 (Orlando). Forecasts were produced for 5 minutes ahead into the future. The VAR model which exploited spatial dependencies of upstream/-downstream traffic information from neighboring sensors were shown to outperform the basic ARIMA model.

- [Lin et al., 2009]: Forecasting flow using data in 15-minute resolutions from 78 sensors. STARIMA model was used to encompass spatial dependencies, but no comparison was done with other models.

- [Ding et al., 2011]: Forecasting flow using 5-minute resolution data for 10 weeks in Beijing. Forecasts were produced for 5, 10, 15 and 30 minutes ahead. The STARIMA model outperformed an ARIMA model.

- [Min and Wynter, 2011]: Forecasting traffic speed from 5 minutes to 1 hour ahead into the future. Data were available in 5-minute resolutions. A VARMA variant was used to predict traffic speed considering spatial dependencies of multiple sections/links connected to a particular road. Forecasts were compared based on different road types but unfortunately not compared with any other model.

- [Tsirigotis et al., 2012]: Forecasting traffic speed using data from a major freeway in Athens, Greece. Data were collected in 10-minute resolutions and available for mul-

tiple lanes of the freeway. Rainfall data were also available in 10-minute resolutions and used as an exogenous variable. Variations of VAR models were used to simultaneously predict traffic speed for all lanes. Flow and traffic mix (e.g., percentages of trucks) were also separately used as an exogenous variable in addition to rainfall. The authors concluded that the exogenous variables were generally helpful for predicting traffic speed. VARX and Bayesian VARX models slightly outperformed ARIMA and ARIMAX while VARMAX yielded significant improvements.

- [Vlahogianni and Karlaftis, 2013]: Forecasting speed based on both minutely and 5-minute resolution data of speed and flow by lane as well as rainfall on a major freeway between the airport and center of town in Athens, Greece. One-step ahead forecasts were made. The RNN model outperformed VARMAX and ARIMAX models, especially on datasets with the higher resolutions.

- [Salamanis et al., 2015]: Forecasting traffic speed based on speed probe data collected using Tom-Tom GPS for 2 weeks in Berlin. Forecasts were made for 1 hour ahead. The data were aggregated every 5 minutes. The authors proposed to segment the time series based on different traffic patterns and train separate models for each pattern. As a result, the authors' STARIMA model, taking advantage of the segmentation, outperformed its standard counterparts.

- [Duan et al., 2016]: Forecasting flow using 2-minute resolution data collected over 6 sensors on I-80 for 10 days. Forecasts were produced for up to 1 hour ahead (30 steps). The authors proposed a STARIMA variant that changes the number of lags to be used dynamically based on traffic speed data. The results were shown to be better than the standard STARIMA model.

- [Wang et al., 2016b]: Predicting travel time in London. Twenty-two links were used, varying from about 200 meters to 15.5 kilometers. The travel time per link was av-

eraged every 5 minutes. Data were collected for 166 days. Only daytime data between 6:00 AM and 9:00 PM were considered. Forecasts were made for 5 - 30 minutes ahead into the future. The authors proposed a variant of Time Delay Neural Network (TDNN) that also considered spatial correlations and named it Space-Time Delayed Neural Network (STDNN), which slightly outperformed STARIMA and significantly outperformed ARIMA.

2.1.4 State Space Models

Another popular approach is using State Space Model, and Kalman Filter (KF) [Kalman et al., 1960] is a popular algorithm for estimating the states. Kalman Filter was designed to work with linear systems. Other variants of Kalman Filter include Extended Kalman Filter [Smith et al., 1962, McElhoe, 1966] and Unscented Kalman Filter [Julier and Uhlmann, 1997] that can work with nonlinear systems. A primary advantage of Kalman Filter is that only the previous state estimate and current measurement are needed to produce an estimate of the current state without requiring any further history of observations. This characteristic makes it ideal for real-time forecasting. A major difficulty in applying Kalman Filter to forecasting is that all the input matrices must be supplied by the user. The user has the responsibility of specifying the underlying linear system, which can be based on well-established theories, models or laws, such as physical laws of motion, or other models that can be represented in State Space forms. Based on the user's expertise in his/her field of study, the transition matrix can be carefully designed to represent an appropriate linear system.

A work in [Gardner et al., 1980] provided a way of representing an ARMA process using the Kalman Filter. The transition matrix was designed to encompass the autoregressive coefficients. The process noise was represented by taking the outer product of the Moving Average coefficients vector with itself. Simplifying assumptions are made regarding the measurement, for which the observations are assumed to be perfectly measured without

error, so H_t becomes a variant of identity matrix that simply retrieves the first element of the state vector as an actual observation and R_t is assumed to be a 1×1 matrix of 0. Another recent study in [Guo et al., 2014] used the Kalman Filter to represent an ARIMA + GARCH process to take advantage of Kalman Filter's ability to do efficient real-time processing.

The design of a Kalman Filter can vary greatly with different researchers. A work in [Okutani and Stephanedes, 1984] may be the earliest work that applies Kalman Filter to predict traffic flow. The transition matrix was constructed by column-wise appending l matrices (l is the number of lags), in each of which the diagonal and neighboring elements were constructed using vectors of traffic characteristics such as occupancy and density for that lag and values of 0's were filled in elsewhere. Another work in [Whittaker et al., 1997] used traffic density and flow of hundreds of stations to represent the state vector and transition matrix, though not explicitly given (due to its large size), was designed to encompass physical laws of motion by using equations relating density, flow, area of lanes, velocity, etc. In [Chen and Chien, 2001, Chien and Kuchipudi, 2003], the state vector represents travel time and the transition matrix is learned from historical data that establishes relationships between travel time at time t and $t + 1$. A work in [Barceló et al., 2010] adopted a similar style. Another work in [Stathopoulos and Karlaftis, 2003] used MLE to estimate the matrices. A study in [Fei et al., 2011] simply uses the identity matrix for both the transition and observation matrices; the study was on travel time and the data resolution was minute by minute.

It is evident that the responsibility of designing an appropriate Kalman Filter largely rests on the individual researcher. In a sense this is the extreme opposite of the Neural Network model, usually for which the researchers simply provide the data and let Neural Network figure out everything in its black box of knowledge. The Kalman Filter, on the other hand, cannot be applied without explicit equations, systems or theories that are supposed to effectively model a real-life scenario such as traffic forecasting.

2.1.5 Hybrid Models

Hybrid or combined models have no specific shape or form. For example, the Dynamic Regression model, as described in Section 9.1 of [Hyndman and Athanasopoulos, 2014], combines regression and ARIMA models. It has the same forecasting ability as to an ARIMAX model, but the primary reason for preferring a Dynamic Regression model over an ARIMAX model is the easy interpretation of the regression parameter, which has its usual interpretation in a regression model. Though it does seem that in the literature, the ARIMAX related terminology can be more popular.

There are some, but not much, work that utilized ensemble learning techniques for traffic forecasting. Other ways to build hybrid models can vary greatly. A work in [Van Der Voort et al., 1996] first applied a Kohonen self-organizing map as an initial classifier, and then used a separate ARIMA or NN model for each class. Another work in [Zeng et al., 2008] joined ARIMA and Neural Networks for traffic flow prediction. A study in [Hong et al., 2011b] combined SVR with genetic algorithms and simulated annealing. A work in [Chan et al., 2012] used Exponential Smoothing to smooth the data as a pre-processing step before applying Neural Networks. This general technique can certainly be applied to other types of models. A study in [Zhang et al., 2014] applied spectral analysis, ARIMA and GJR-GARCH to model the intra-day/periodic trends, deterministic components, and volatility in traffic flow patterns, respectively. Another work in [Zou et al., 2015] used Trigonometric Regression to extract the basic traffic pattern and then applied time series models. In [Tang et al., 2017], a similar approach was adopted by combining Trigonometric Regression with Neural Networks.

2.1.6 Support Vector Regression

The original Support Vector Machine [Cortes and Vapnik, 1995] algorithm was designed as a binary classifier. Support Vector Regression (SVR) [Drucker et al., 1997] can be considered as an extension to SVM. The SVR algorithm is a simple and effective tool that has been very popular up until the early 2010s, and then it seemed to be mostly eclipsed by the great wave of deep learning. Studies in [Castro-Neto et al., 2009, Hong et al., 2011a, Hong et al., 2011b] have shown SVR to be an effective forecaster. In recent years, work involving SVR has seen a reduction, except as a comparison, such as in [Lippi et al., 2013, Lv et al., 2015, Ma et al., 2015, Tang et al., 2017, Zhao et al., 2017].

2.1.7 Neural Networks

Since the great breakthroughs in deep learning [Krizhevsky et al., 2012], deep Neural Networks have taken the center of the stage in many fields and traffic forecasting is no exception. Though traditional Neural Networks have been used in this field for many years, the performance of the then shallow and typically small Neural Networks was usually not the most impressive. The class of Recurrent Neural Networks (RNN), which can take advantage of the temporal dependencies in the data, are inherently suitable for forecasting. In general, RNN allows edges to loop backward, as opposed to standard feedforward Neural Networks, in which edges are only allowed to connect to neurons in the subsequent layer. The parameters of an RNN or LSTM may be learned through algorithms such as the basic Backpropagation Through Time [Robinson and Fallside, 1987, Werbos, 1988, Mozer, 1995]. Newer algorithms such as Adam [Kingma and Ba, 2014] have also been developed in recent years.

In theory, an RNN should able to hand long term dependency, but in practice, it seems to be having trouble doing so [Hochreiter, 1991, Bengio et al., 1994]. To resolve this issue, LSTM was designed to retain long term memory. LSTM is arguably the most prominent

type of RNN and has been successfully and widely applied in many fields such as speech recognition [Fernández et al., 2007], machine translation [Sutskever et al., 2014], automatic image captioning [Vinyals et al., 2015], among many others. It is also known to be used in Google Android [Zen and Sak, 2015] and in particular, Google Voice Search [Sak et al., 2015].

There are also other variants of the standard LSTM. One such recently developed variant is known as the Gated Recurrent Unit (GRU) [Cho et al., 2014]. It has a simpler structure than the standard LSTM and has been gaining increasing popularity.

In recent years, LSTM has also gained increasing popularity in traffic forecasting. Very recent work in [Ma et al., 2015, Zhao et al., 2017, Jia et al., 2017] all concluded that LSTM is the top-performing model when compared with a plethora of statistical and machine learning models, including some deep learning models.

Another variant of RNN is known as State Space Neural Network (SSNN), which was developed based on the insights that RNN can be used to represent spatiotemporal patterns given in [Elman, 1990]. SSNN, just like an Elman Network, which is also known as a Simple Recurrent Network due to its simple structure, has a context layer that stores the previous state, which is like short term memory, and the context layer loops back to the hidden layer. The input layer is usually used to represent spatial relationships in the data (e.g., different segments of the road), and connections between the input layer and the hidden layer may be selected based on spatial dependencies.

A study in [Van Lint et al., 2002] tested several variants of SSNN and concluded the SSNN can be used to effectively predict travel time, though no other models were used in the comparison. A work in [Van Lint et al., 2005] experimented with predicting travel time with SSNN under the impact of missing data and concluded that SSNN can be resistant to missing data. No other model was included in the comparison. A study in [Liu et al., 2006] proposed to use an Extended Kalman Filter to train an SSNN, which yielded better performance than

a standard Kalman Filter and an SSNN trained with the LevenbergMarquardt algorithm. A work in [van Hinsbergen et al., 2009] compared SSNN with standard NN and found NN performed slightly better for 5 minutes ahead predictions but SSNN performed better for 15 minutes ahead predictions. All of the work that used SSNN in the field of traffic forecasting seems to come from the same group of people. It may be better or more appropriate to consider SSNN as a type of Elman Network that considers spatial dependencies.

Another type of Neural Network that handles temporal data is known as Time Delay Neural Network (TDNN) [Waibel et al., 1990]. It is typically implemented as a feedforward Neural Network but differs in some significant ways. The neurons in a TDNN received outputs from neurons in the previous layer just like a feedforward NN, but they could also receive the time-delayed (past) outputs from the same neurons in the previous layer. During backpropagation, time-shifted copies of the network are made and error gradients are computed across the copies and then averaged before weight updates. Thus position dependency can be removed from the training process. TDNN can be considered to be extremely similar to a one-dimensional Convolutional Neural Network that performs convolution on the time axis. It differs from RNN, which uses hidden layers to keep track of the past as opposed to time-varied input.

A study in [Lingras and Mountford, 2001] applied genetic algorithms for selecting connections between the input and hidden layers of a TDNN and demonstrated that the proposed model worked better than a standard TDNN. No other models were included for the comparison. A work in [Ishak et al., 2003] concluded that TDNN performed slightly better than simple RNN variants. Another study in [Zhong et al., 2005] showed TDNN can perform better than weighted local regression if the training data are not separated into days of the week, but would perform worse otherwise. Recent work in [Ma et al., 2015] used TDNN as a comparison for LSTM, which outperformed TDNN. Besides that, not much recent work seems to have been conducted using TDNN in the field of traffic forecasting.

Both SSNN and TDNN seem to be fading away in recent years. LSTM seems to outshine TDNN in terms of effectively exploiting temporal dependencies. To account for spatial dependencies, recent work in [Wu and Tan, 2016] proposed a hybrid model combining Convolution NN and LSTM. The performance of the hybrid model outperformed LSTM, SAE, NN, and gradient boosting regression tree. Another recent study in [Yu et al., 2017] that combined Convolutional NN with LSTM obtained similar top-performing results.

2.1.8 Concluding Remarks

The field of traffic forecasting has been well researched for many decades. A work in [Vlahogianni et al., 2004] provided a great survey of short term traffic forecasting up until 2004 and has been a great source for learning. The work has been updated in [Vlahogianni et al., 2014], which has examined the shift from traditional classical statistical models to the more data-driven machine learning models. Another work in [Mori et al., 2015] provided a comprehensive review of travel time forecasting. No comprehensive review seems to exist that cover the recent revolution in deep learning and its impact on traffic forecasting. Certainly, this is such a recent and dynamic field and many researchers have only delved into it in the very recent years.

2.2 Chronological Survey of Related Work

This section primarily consists of a time-ordered table on the notable studies on traffic forecasting since the early 2000s. Some of the studies were briefly discussed or mentioned in the previous sections. The five columns of the Table 2.1 carry the following meanings:

- **Models**: a list of models used in a study.
- **Type**: the type of traffic data to forecast, such as flow or speed.

- **Data**: a description of the data; the first item is always the resolution of the data.
- **Horizon**: the forecasting horizon, which is how far ahead into the future the forecasts were made.
- **Summary**: an overall summary of findings in a study.

Table 2.1: Chronological Survey of Traffic Forecasting Work

Models	**Type**	**Data**	**Horizon**	**Summary**
RNN, TDNN, Partially Recurrent Networks	Speed	5-min, 28 weekdays (2001), 3 sensors on I-4	up to 20 minutes	See [Ishak et al., 2003]. TDNN yielded the best results overall, but the performance of Elman NN (a simple RNN) and Partially Recurrent Networks were not too far behind.
Simple Exponential Smoothing, Linear Regression/Extrapolation, Combined Model using Exponential Smoothing and Historical Averages	Flow	minutely, 350 sensors, over 2 years in Germany	up to 1 hour	See [Chrobok et al., 2004]. Considered both directions of roads and special events including a soccer game and a solar eclipse. Separates days into Mon-Thu, Fri and days before holidays, Sat except holidays, and Sun and holidays. Found ES to perform well on immediate short terms (around half an hour), and then loses to historical averages. Linear extrapolation generally did not perform as well. A combined model that used ES during short terms and historical averages in longer terms was proposed.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
Locally weighted regression, TDNN	Flow	hourly, 6 sensors, 4-5 years in Canada	1 hour	See [Zhong et al., 2005]. This work made an effort to cover roads with different traffic patterns such as regional commuter, rural long-distance, summer recreational, etc. GAs were used to filter the most recent 168 data points (1 week) to include only 24 input data points that have maximum correlations with the value to be predicted. TDNN performed better than locally weighted regression if they were both trained with all the available data. Regression performed better than TDNN when the training data were separated into different days of the week.
SARIMA, ARIMAX, SARIMAX	Flow	daily, 3 years, 4 sensors in Belgium	1 day	See [Cools et al., 2009]. This paper, in particular, studied the effect of holidays on daily traffic counts. A binary-encoded exogenous variable was used to indicate if a day is a holiday. Six binary-encoded variables were also used to represent seven days of a week, with Sunday represented by all zeros in the variables. The authors found that the holiday effects were significant on commuter roads but not as much on leisure roads.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
SVR based model, outperforming Gaussian maximum likelihood-based model, Double Exponential Smoothing and NN	Flow	5-min, 16 days, 7 sensors from PeMS	5 min	See [Castro-Neto et al., 2009]. This work primarily focused on forecasting traffic flow under abnormal conditions and special events such as holidays and traffic collisions. The NN structure, in terms of the number of neurons in each layer, was 10-4-1. The authors' proposed model performed the best under both typical and atypical conditions.
VAR, outperforming ARIMA and SARIMA	Flow, Speed	5-min flow and speed data, 5 sensors, March 2003 on I-4 (near Disney)	5 min	See [Chandra and Al-Deek, 2009]. The VAR model took advantage of the spatial dependencies between sensors that are on the same freeway. Flow and Speed were forecasted using only flow and speed data, respectively, from multiple sensors.
ARIMA, MA, and Double Exponential Smoothing are aggregated using NN, outperforming basic ARIMA, nonparametric regression and NN	Flow	hourly, 1 year, 1 sensor in Guangzhou (southern China)	up to 3 hours	See [Tan et al., 2009]. Three time series were extracted from data, hourly, daily and weekly; ARIMA, MA and double ES were used to forecast each of the three aforementioned time series, respectively. A NN was then trained by using the forecasts from the three base models as inputs to produce the final output forecast. Extremely similar to Stacking, except that each base model was trained with a different time series.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
SSNN, NN	Travel Time	5-min, 39 morning rush hours (2007), 19 sensors in Netherland	up to 15 minutes	See [van Hinsbergen et al., 2009]. Each segment of the road was represented by a neuron in the hidden layer of SSNN. The inputs included both flow and speed for a particular road segment. The performance of SSNN and NN were similar for 5 minutes ahead forecasts (NN slightly better), and SSNN performed better for 15 minutes ahead forecasts than NN.
STARIMA, outperforming Multivariate Adaptive Regression Splines and model based on chaos theory	Flow	15-min, 1 day (10/21/06), 10 sensors on 2nd Ring of Beijing	15 minutes	See [Min et al., 2009]. The spatial matrix of STARIMA was based on Turn Ratio Prediction Model, which attempted to model the incoming number of vehicles at a location based on the vehicles from neighboring streets that could make turns into the location of interest.
GSTARIMA, outperforming STARIMA slightly	Flow	Same as above, except the chosen day is 10/21/06	Same as above	See [Min et al., 2010]. An extended version of the above paper. The GSTARIMA differs from STARIMA by allowing different locations to have different autoregressive and moving average parameters. The authors suggested that more data could potentially further improve the results.
Local cubic and linear regression	Speed	5-min, 2 hours, 1 sensor in LA	5 min	See [Yue et al., 2010]. Using local cubic and linear regression to estimate traffic speed. Cubic regression generally performed better than the linear one.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
SVR based model, outperforming SARIMA	Flow	hourly, 1-2 months, 3 sensors in Taipei, Taiwan	1 hour	See [Hong et al., 2011a]. The parameters of the SVR model were optimized using the ant colony optimization algorithm.
SVR based model, outperforming SARIMA, Double and Triple Exponential Smoothing and NN	Flow	Same as above	Same as above	See [Hong et al., 2011b]. An extended version of the above paper. The proposed SVR model used a hybrid genetic algorithm-simulated annealing technique to learn the parameters of SVR. The structure of the NN was rather small, with only 3 neurons in the hidden layer.
NN, in combination with Simple Exponential Smoothing and Levenberg-Marquardt algorithm	Flow	minutely, morning rush hours only for a couple of weeks, 30 sensors from western Australia	5 min	See [Chan et al., 2012]. Simple ES was used as a pre-processing/smoothing step before training the NN on the smoothed data using the Levenberg-Marquardt (LM) algorithm to minimize the cost function. Evaluations were mostly done with other NN that used variants of the LM algorithm, wavelet NN and a Bayesian NN. The authors' proposed NN generally performed better.
ARIMA, SARIMA, Kalman Filter, SVR based models and NN	Flow	15-min, 9 months, 16 sensors from PeMS	15 minutes	See [Lippi et al., 2013]. The NN structure was of size 5-10-1 and generally did not perform well. The SARIMA model whose parameters were estimated using a Kalman Filter was the top-performing model. However, SVR did run much faster at the expense of losing some accuracies.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
Geographically Weighted Poisson Regression, Generalized Linear Model	Crash	Annual, 4 years of crash counts data from 58 counties in CA	1 year	See [Li et al., 2013]. Explanatory Variables included traffic information such as Daily Vehicle Miles Traveled and percentage of trucks/trailers, types of road (e.g., urban, freeway) and social-demographic information (e.g., age, poverty level). The Geographically Weighted Poisson Regression model was compared with a GLM model that had the same setup except that it did not include the geographical/county information. The Poisson Regression model performed better by having extra county information.
Wavelet Transform + NN, outperforming NN	Flow	hourly, 1 month (Jan 2009), 2 sensors in Dublin, Ireland	1 hour	See [Dunne and Ghosh, 2013]. Rainfall data were incorporated into the models. The study showed that incorporating rainfall data certainly helped improve prediction.
Adaptive KF representation of SARIMA + GARCH, outperforming Triple ES, SARIMA and regular KF	Flow	15-min, 3-12 months, 36 sensors in UK and US	15 minutes	See [Guo et al., 2014]. The joint SARIMA + GARCH model was able to produce both traffic flow prediction (mean) and the prediction interval (variance). The Adaptive Kalman Filter representation of such model was advantageous in 1) efficiency in real-time processing 2) ability to update process variances as time passes.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
Hybrid model of Trigonometric Regression and time series models including an ST model, VAR and ARIMA	Speed	5-min, 7 months, 5 sensors on I-394E	up to 1 hour	See [Zou et al., 2015]. The Space-Time (ST) model was based on linear combinations of the lagged values of all stations (at most 2 lags, not all lags from all stations were used). The variables were chosen in a stepwise forward feature selection manner using BIC. Both the ST and VAR models took advantage of the spatial dependencies by including information from neighboring traffic sensors. The hybrid model that extracted the periodic components first and then fed the residuals into time series models yielded improved results for forecast-horizon that was greater than half an hour. ST models generally performed the best and VAR was in second place.
Deep NN built with SAE, outperforming NN, SVR, and RBF NN	Flow	5-min, all sensors (about 15000) for the first 3 months of 2013 from PeMS	up to 1 hour	See [Lv et al., 2015]. The deep NN took advantage of both temporal and spatial dependencies by incorporating the lagged values of all freeway time series as inputs. No manual selection was done to determine the relevant/connected ones. Grid search was performed to determine the number of hidden layers to be 2-4, with the size of each layer 200-500, depending on the actual forecasting horizon.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
LSTM, outperforming Elman NN, TDNN, NARX NN, SVR, ARIMA and KF	Speed	2-min, 1 month (June 2013), 2 sensors on 2nd Ring of Beijing	2 minutes	See [Ma et al., 2015]. Both Speed and Flow data were used to forecast speed. The performance of LSTM was overwhelmingly well comparing with the other techniques.
kNN, outperforming SARIMA and Kalman Filter based models	Flow	15-min, 3-12 months, 30 sensors in the UK and US	up to 90 minutes	See [Habtemichael and Cetin, 2016]. Forecasts were produced using kNN to find the closet historical traffic pattern profiles and aggregate them. The kNN algorithm was a very simple algorithm and the authors suggested that it may be very suitable for real-time short term traffic forecasts.
Deep Belief Network, ARIMA and NN	Flow	15-min, 4 months in late 2013, 47 sensors from PeMS	15 minutes	See [Koesdwiady et al., 2016]. Incorporated weather information such as rain, temperature, humidity, etc., into a deep belief network. Performance comparisons were done with ARIMA and a neural network model of three layers. The authors' proposed deep belief network outperformed ARIMA significantly and did better than the three-layer neural network, though the margins were not as great.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
Bilinear Poisson Regression	Flow	probe data converted to 20-min resolution, 1 month in 2015, Tokyo	up to 48 hours	See [Okawa et al., 2017]. Data were divided into 14 major routes/segments and 72 20-minute intervals for every 24 hours. The authors' proposed Bilinear Poisson Regression model was able to take advantage of the spatial dependencies and produced forecasts for all segments. However, this model was only compared with two other forms of Bilinear Poisson Regression models, one trained on all the data (baseline 1) and the other trained on each segment separately (baseline 2). The authors' proposed model performed much better than baseline 1 and slightly better overall than baseline 2.
Fuzzy NN with Trigonometric Regression, outperforming NN, SVR, ARIMA, VAR	Speed	2-min, 1 month (Dec 2014), 3 sensors from 4th Ring in Beijing	up to 20 minutes	See [Tang et al., 2017]. Trigonometric regression was fit on the data to capture periodic patterns in the daily traffic speed. The periodic components were then given to the authors' proposed fuzzy NN, which yielded improved accuracies for multi-step forecasts.
LSTM, outperforming ARIMA, SVM, RBF NN, SAE NN and RNN	Flow	5-min, the first half-year of 2015, 500 sensors in 5th Ring in Beijing	up to 1 hour	See [Zhao et al., 2017]. LSTM structures consisted of 2-6 layers depending on the forecast-horizon using trial and error. The authors' proposed LSTM structure took advantage of both spatial and temporal dependencies by representing each sensor at each time point as a memory unit.

Continued on next page

Table 2.1 – *Continued from previous page*

Models	**Type**	**Data**	**Horizon**	**Summary**
LSTM, outperforming ARIMA, NN, DBN	Flow	2-min, Jun-Aug in 2013 and 2014, 4 sensors between 2nd and 3rd Ring of Beijing	10 and 30 minutes	See [Jia et al., 2017]. Rainfall data were incorporated into the models. The authors concluded the incorporation of rainfall data generally improved forecasting performance for most models that were tested.
RNN with GRU	Flow	hourly, two months in late 2016, 1 sensor in Santa Clara county (CA PeMS)	up to 12 hours	See [Zhang and Kabuka, 2018]. The weather data included precipitation, speed, and temperature. The authors demonstrated that the incorporation of weather data can improve forecasting accuracies; however, no other models were used for comparison purposes. Only the authors' proposed model, with and without weather data, was included in the performance evaluations.

Chapter 3

Traffic Flow Forecasting on Highways Evaluation and Comparison of Models[1]

[1]Hao Peng and John A. Miller. Submitted to *International Journal of Data Science and Analytics*, 06/20/19

Abstract

Traffic flow forecasting is valuable to both governments for designing intelligent transportation systems and everyday commuters or travelers who are interested in the best routes to their destinations. This work focuses on forecasting traffic flow on major highways in the San Diego, California area using data from a large number of sensors. A large variety of models is considered, including seasonal Autoregressive Integrated Moving-Average model, seasonal Vector Autoregressive Integrated Moving-Average model, regression models, Support Vector Regression, Extreme Learning Machine, Feedforward Neural Networks, and two types of Long Short-Term Memory Neural Networks. Forecasting is performed in both a univariate manner by relying on the historical temporal data of a particular sensor as well as in a multivariate manner by considering a neighborhood of three closely located sensors. Two data resolutions are also used in the experiment, a 5-minute resolution, and a 15-minute resolution, both are commonly found in the existing literature. Multivariate forecasters generally improve upon their univariate counterparts. The various neural networks, in particular, Encoder-Decoder Long Short-Term Memory Neural Networks, can achieve the highest level of accuracy but are also the most computationally expensive. Extreme Learning Machine can be trained in a very short amount of time while achieving great accuracies. The trade-off between accuracy and computational costs, effects of down-sampling to a lower data resolution, as well as issues of parameter tuning and overfitting are discussed.

3.1 Introduction

Traffic flow forecasting is an important aspect of designing intelligent transportation systems for cities and highways. It is also of great interest to everyday travelers who may desire to know in advance the congestion levels of roads and the amount of time it would take to reach their destinations. Much research has been devoted to studying this topic in recent years,

as can be evident in very recent work such as [Okawa et al., 2017, Tang et al., 2017, Zhao et al., 2017, Jia et al., 2017]. The fruits of such studies can be of good use to city planners in governments, traffic app developers as well as everyday commuters and travelers. A congested road segment may be identified by low traffic flow and speed, which may lead a commuter to choose a different route.

The dawn of the big data era has also created great opportunities for data scientists to push for advancements in research on traffic forecasting. Seeing the need to collect large amounts of high-quality, high-resolution traffic data, numerous states in the United States have invested in deploying a great number of traffic sensors on their busiest roads and highways. The Caltrans Performance Measurement System (PeMS)[2] from the state of California is an example of such systems. High-resolution traffic data such as flow and speed are collected in real-time from more than 39000 sensors deployed in major urban areas and highways across the state. This work is devoted to studying traffic flow forecasting using PeMS data collected in the San Diego, California area during the entire year of 2018. Figure 3.1 illustrates the location of all the sensors in this study.

The vast majority of existing literature on the topic of traffic forecasting has devoted to forecasting in the immediate short terms, such as a couple of minutes ahead. It is certainly justified, as the immediate short terms can usually best capture the dynamic nature of traffic situations and are usually of the greatest interests. For example, a commuter would be very interested in the optimal routes to avoid the most traffic congestions during morning rush hours; or the amount of time, hopefully in minutes, for the commuter to arrive at his or her workplace. Longer-term traffic forecasts can certainly be done by relying more on the historical data of a particular location but may suffer from relatively poorer accuracies due to larger time gaps (e.g., forecasting many hours in advance may need to heavily rely on

[2] `http://pems.dot.ca.gov/`

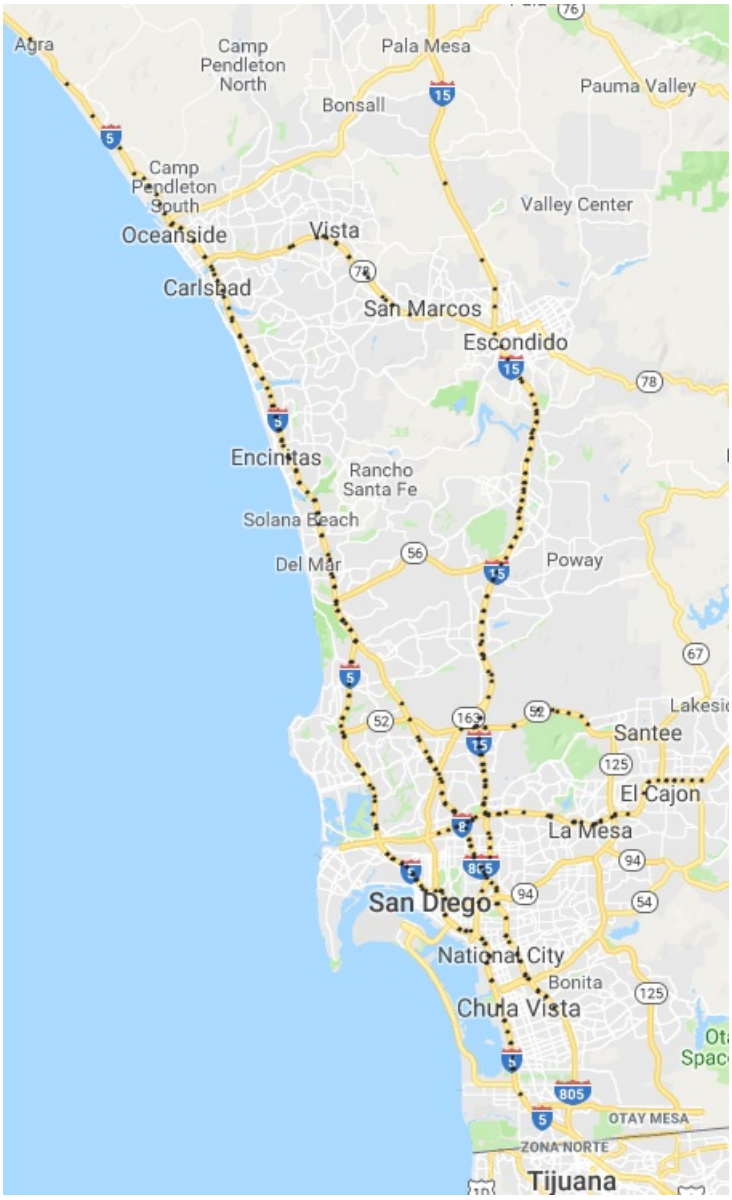

Figure 3.1: Location of Sensors. Generated by mapmakerapp.com on Google Maps.

historical averages, but if exceptional circumstances occur, such as an accident or rainy weather, then clearly such forecasts may not be as reliable).

Many researchers have exclusively studied one-step-ahead forecasts [Lippi et al., 2013, Guo et al., 2014, Ma et al., 2015, Koesdwiady et al., 2016]; in other words, if the data are of the 15-minute resolution, then forecasts are produced for only 15 minutes ahead. Others, especially recently, have also studied multi-step traffic forecasting, from minutes to a couple of hours ahead, such as in [Lv et al., 2015, Zhao et al., 2017, Jia et al., 2017]. This study focuses on producing forecasts for up to 12 steps in the future, which is equivalent to up to 1 hour and 3 hours ahead, in 5-minute and 15-minute data resolutions, respectively.

Univariate forecasting, meaning producing forecasts by relying on historical data from one particular sensor alone, is also predominant in the literature, such as in [Tan et al., 2009, Hong et al., 2011b, Lippi et al., 2013, Ma et al., 2015, Jia et al., 2017]. Multivariate forecasting often involves using data from multiple spatially dependent sensors to produce improved forecasts over their univariate counterparts, such as in [Chandra and Al-Deek, 2009, Lv et al., 2015, Zhao et al., 2017]. Very recently, some researchers have also chosen to simply give data from very large numbers of sensors to a deep neural network and task it to determine and establish any dependencies among the data [Lv et al., 2015, Zhao et al., 2017]. Some studies have also included external variables such as weather data into their forecasting models, such as in [Koesdwiady et al., 2016, Jia et al., 2017].

Commercial companies that provide some type of traffic/travel predictions often rely on data such as travel distance, speed limits, and historical traffic conditions. Some companies are also able to estimate the current traffic condition by collecting real-time data from users' mobile devices with their permissions. To provide reliable traffic forecasting, it is crucial to have access to accurate and current traffic data and analyze them using appropriate models.

A very recent study in [Neilson et al., 2019] provided a thorough review of the various research questions and challenges in using big data in the domain of transportation and

smart city. The collection of large amounts of traffic data requires careful consideration of issues such as storage space, quality control, and data security. It is also important to determine what types of data to collect as well as how frequently the data should be collected. There are also various ways scientists must decide as to what and how to analyze the data, including analysis of historical data, predictive analytics, real-time forecasting, video/image analytics, collision preventions, etc. Such research questions and challenges must be properly addressed to build better transportation systems. Our study focuses on predictive analytics, traffic flow forecasting in particular, and for future work, we plan to extend our work to real-time forecasting.

This work is an extended version of a conference proceeding [Peng and Miller, 2019]. Additional models are considered, two new experiments are conducted using 15-minute resolution data, and more detailed discussions of the models and the results are included. The contributions of this work are as follows: 1) to evaluate the effectiveness of various statistical and machine learning models on univariate traffic flow forecasting using large amounts of temporal data; 2) to study the impacts of incorporating spatially dependent data into multivariate forecasting models; 3) to examine the performance of multi-step forecasts and the impacts of varying data resolutions; 4) to explore the various trade-offs/pros and cons of the models in terms of accuracy, stability, computational cost, and ease of use.

Most forecasting models used in this work are provided by the ScalaTion project [Miller, 2018]. It is an open-source, MIT licensed, Scala-based project designed for analytics and simulation using big data. For more details, please visit ScalaTion homepage at `http://www.cs.uga.edu/~jam/scalation.html`. Various neural network models in this study are provided by Keras [Chollet et al., 2015] using the Tensorflow [Abadi et al., 2015] backend. The ScalaTion project is also actively developing various neural network models and currently provides implementations of feedforward neural networks.

The rest of this paper is organized as follows: Section 3.2 discusses the basic background of various statistical and machine learning models included in this study. Section 3.3 is about Related Work in traffic forecasting. Section 3.4 explains the detailed experimental, performance evaluations, and detailed discussions of the results. Finally, Section 3.5 concludes the paper and offers potential directions for future work.

3.2 Background

For any model to produce accurate forecasts, it must rely on appropriate sources of inputs. Define $\mathbf{y}_{tl_1}$ as the vector $[y_{t-1}, \ldots, y_{t-l_1}]$, $\boldsymbol{\epsilon}_{tl_2}$ as the vector $[\epsilon_{t-1}, \ldots, \epsilon_{t-l_2}]$, and X_{tl_3} as a matrix of row vectors $[\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-l_3}]$; where y_t is the response at discrete time t, ϵ_t is the residual at time t, $\mathbf{x}_t$ is a vector of exogenous predictors at time t, $\mathbf{y}_{tl_1}$ is the vector of historical values of the response up to lag l_1, $\boldsymbol{\epsilon}_{tl_2}$ is the vector of past residuals (interpreted as shocks) up to lag l_2, and X_{tl_3} is the matrix containing exogenous predictors at previous time points up to lag l_3. Then a very general form of a univariate forecasting model may take on the form of

$$y_t = f_1(\mathbf{y}_{tl_1}; \boldsymbol{\alpha}) + f_2(\boldsymbol{\epsilon}_{tl_2}; \boldsymbol{\theta}) + f_3(X_{tl_3}; \boldsymbol{\beta}) + \epsilon_t \tag{3.1}$$

where $f1$, $f2$, and $f3$ are functions that optimize parameters $\boldsymbol{\alpha}$, $\boldsymbol{\theta}$, and $\boldsymbol{\beta}$, from input data $\mathbf{y}_{tl_1}$, $\boldsymbol{\epsilon}_{tl_2}$, and X_{tl_3}, respectively, so that some type of norm of the residuals such as the Sum of Squared Error (SSE) or Mean Squared Error (MSE) are minimized; $\hat{y}_t$ is the forecasted value at time t, i.e., $y_t = \hat{y}_t + \epsilon_t$. Sometimes a differencing operator ∇^d of order d may be appropriately applied to the original time series

$$z_t = \nabla^d y_t \ , \tag{3.2}$$

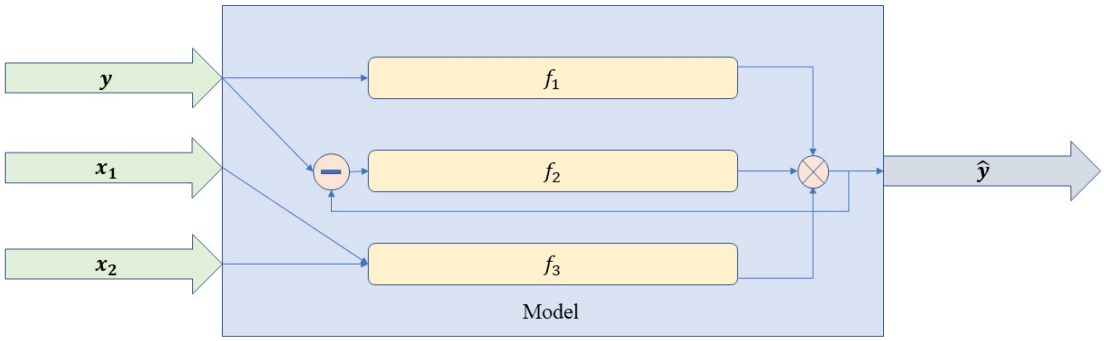

Figure 3.2: A general model. It considers three streams of data: a time series of interest and two exogenous predictors.

where z_t is the value of the differenced time series at time t and then the model in Equation 3.1 may be fitted on z_t instead. The univariate model may also be generalized into the multivariate case by including additional inputs such as the lagged values of other time series.

The same general model in Equation 3.1 can be illustrated in Figure 3.2. Three streams of inputs are sent to the model. The feedback of $y - \hat{y}$ is needed to computed the residuals/shocks used in f_2. The outputs of the three functions are aggregated to produce a stream of forecasts. In Equation 3.1, the aggregation operator ("x" within a circle) represents summation, but the means of aggregation may vary in other contexts.

Most models would need $\mathbf{y}_{tl_1}$ as input to make reliable forecasts. Often time series models could make use of $\boldsymbol{\epsilon}_{tl_2}$, but it is not very common for machine learning models. The exogenous predictors X_{tl_3} can include information such as weather conditions, time of the day, real-time collision data, etc., and may improve models' performance.

3.2.1 Model Inputs and Outputs

The time series models in this study, named seasonal ARIMA and VARIMA models, just require the time series of interests as inputs. Other models typically require the data from the time series to be organized into training instances of input and output features. Such a design allows more flexibility in adding and removing features. Models like neural networks may also require the appropriate scaling of the data.

Another major difference between the time series model and other models is the time series models must rely on previously produced forecasts as inputs to make subsequent forecasts. For example, to produce two-step ahead forecasts, the forecasted value at step one is needed as input to produce a forecast at step two. Other models typically would only rely on the actual values of their input features to produce forecasts for any forecast-horizon.

The orders of the time series models can be chosen based on the plots of the autocorrelation function (ACF) and partial autocorrelation function (PACF). Sample ACF and PACF plots are in Figure 3.3. From the ACF plots, the last significant lag is 3, which suggests an moving average order of 3. The PACF plots are slightly more difficult to interpret the autoregressive order. The lags quickly drop off at around lag 5, but stays at roughly the same level of significance throughout. An autoregressive order of 5 can be a reasonable choice. Alternatively, we also used an automated approach proposed by [Hyndman et al., 2007] to find the appropriate orders by using a grid-search like algorithm based on some type of scoring function, such as Bayesian Information Criterion (BIC) [Schwarz et al., 1978]. Though it would be feasible to try to choose the best orders for the time series models, it would be difficult to make perfectly fair comparisons with other models [Karlaftis and Vlahogianni, 2011], mostly due to the differences in inputs and forecasting equations.

To train other models, typically an $n \times k_i$ input training matrix V, where n is the number of instances and k_i is the number of input features, and an $n \times k_o$ output/response matrix W, where k_o is the number of outputs/responses, are required. In this study, the following

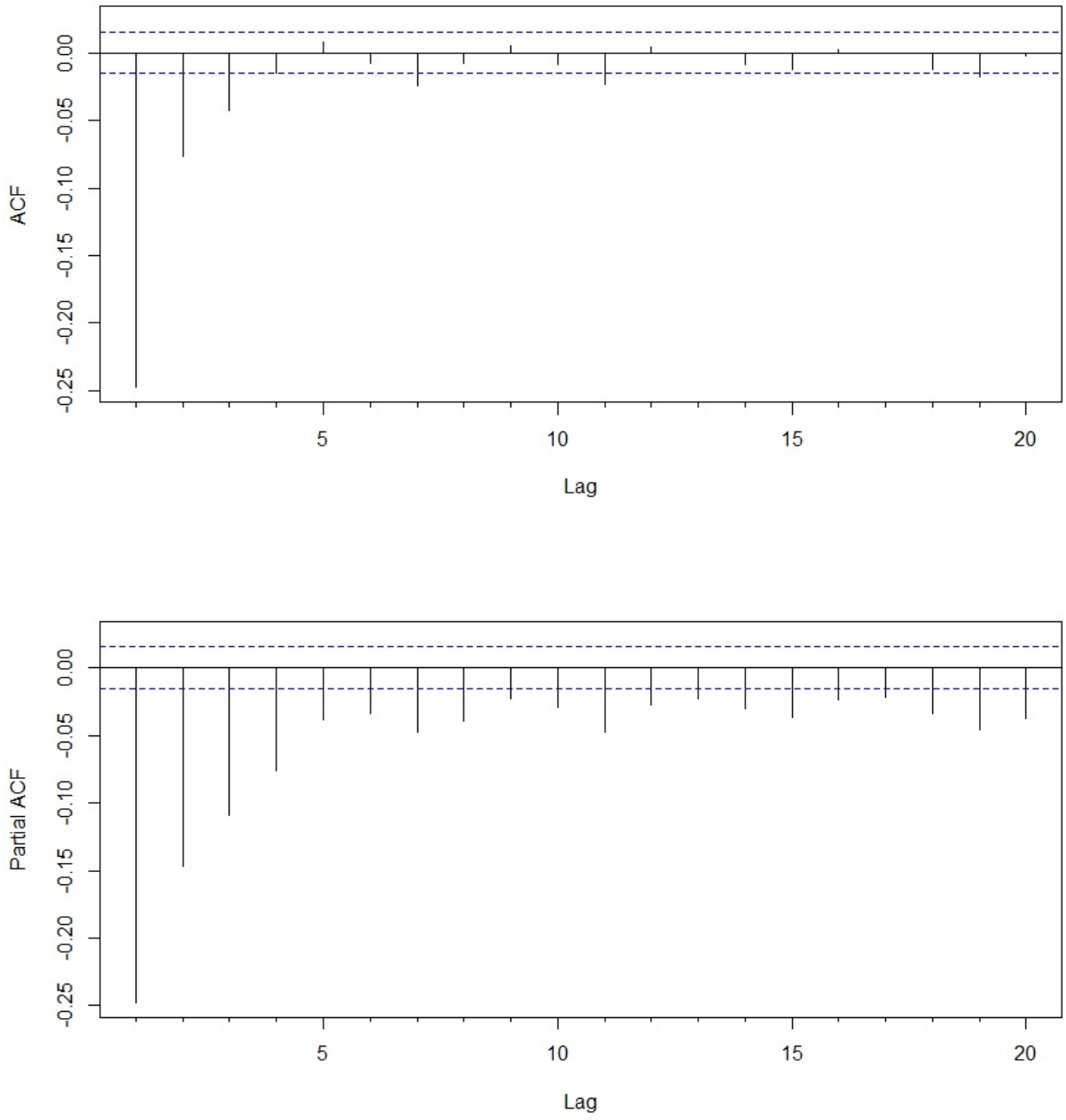

Figure 3.3: ACF and PACF. Generated using 15-minute resolution data from sensor ID 1108415 located on I-15 N. Order 1 seasonal differencing and order 1 simply differencing have been applied. Plots are generated by the "forecast" package [Hyndman et al., 2019] in R.

input features are considered for input matrix V: 12 most recent traffic flow observations, 12 observations from the previous seasonal period (e.g., if using 5-minute resolution data to forecast this coming Monday's traffic from 8:00 AM to 9:00 AM, then last Monday's traffic flow data from the same time window are used), 12 historical averages computed from previous 4 weeks, and the time of the day. The output training matrix W simply includes traffic flow data for the next 12 steps. In mathematical notations, instance/row t of the input training matrix V may be defined as

$$\mathbf{v}_t = [\bar{y}_t, \ldots, \bar{y}_{t+h-1}, y_{t-s}, \ldots, y_{t+h-1-s}, y_{t-h}, \ldots, y_{t-1}, hr_t] \,, \tag{3.3}$$

and its associated instance/row t of the output training matrix may be defined as

$$\mathbf{w}_t = [y_t, \ldots, y_{t+h-1}] \,, \tag{3.4}$$

where h is the forecasting horizon, 12 in this study; s is the seasonal period; hr_t is the time of the day (e.g., 7.5 would represent 7:30 AM) associated with y_t; and $\bar{y}_t$ is the historical average computed weekly for 4 weeks before time t (e.g., if y_t represents traffic flow this coming Monday at 8:00 AM, then $\bar{y}_t$ is the historical average of traffic flow for the last four Mondays at 8:00 AM). The training input matrix V may also be expanded for the multivariate experiments by including similar data from neighboring sensors.

Note that in this design, each column of W represents a particular forecasting step. Depending on the model, the training process may need to consider one output feature, represented by a column in W, at a time or all output features at once.

The input features of V can be highly correlated with each other, therefore introducing multicollinearity into the training matrix. However, as stated in Section 5.9 of [Hyndman and Athanasopoulos, 2018], this study is not concerned with the interpretations of parameters on the highly correlated features and the future traffic flow values are within the recent historical

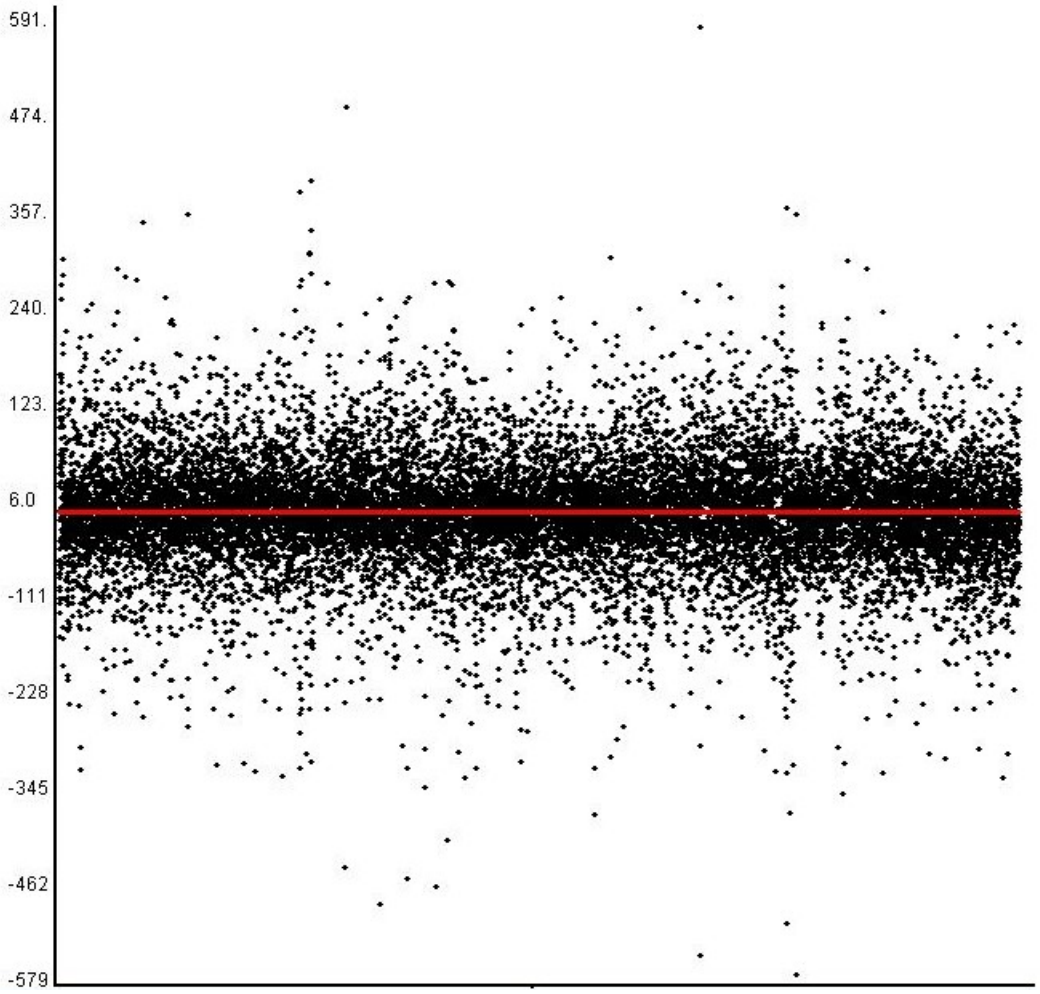

Figure 3.4: Residuals vs. time. Generated using 15-minute resolution data from sensor ID 1108615 located on I-5 S. One-step ahead forecast values produced by Extreme Learning Machine are used to compute the residuals.

range, therefore the problem of multicollinearity is not an issue. A sample residual plot is also available in Figure 3.4, which shows no significant trends or patterns. The histogram of the same residuals is in Figure 3.5.

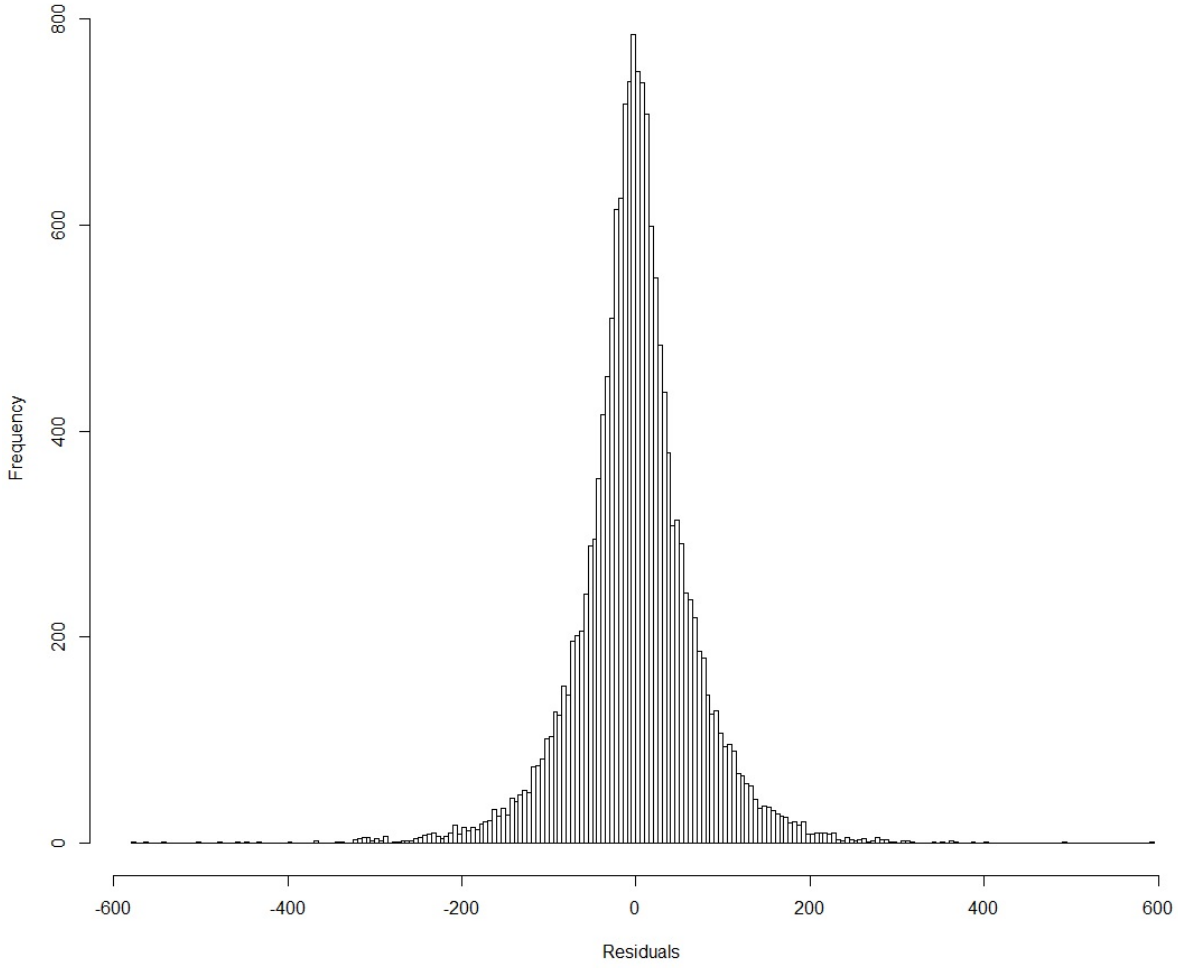

Figure 3.5: Histogram of residuals. The mean and standard deviation are -0.0008047318 and 67.13218, respectively. Plot generated by the "hist" function in R.

3.2.2 Time Series Models

Time series models generally involve formalization of equations with the assumption that there are dependencies amongst the **y** stream (referring back to Figure 3.2) that can be explained. Commonly use time series forecasting models include seasonal Autoregressive Integrated Moving-Average (ARIMA) model [Box and Jenkins, 1970] and its multivariate generalization seasonal Vector Autoregressive Integrated Moving-Average (VARIMA) model [Sims, 1980].

Seasonal ARIMA

The seasonal Autoregressive Moving-Average model may be defined as

$$z_t = \nabla^d \nabla_s^D y_t \tag{3.5}$$

$$z_t = c + \sum_{i=1}^{p} \phi_i z_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \sum_{i=1}^{P} \Phi_i z_{t-is} + \sum_{i=1}^{Q} \Theta_i \epsilon_{t-is} + \epsilon_t \ , \tag{3.6}$$

where the differencing operator ∇ and seasonal differencing operator ∇_s, of orders d and D, respectively, may need to be applied to the time series stabilize the mean before fitting the parameters. the parameters include an intercept c, p autoregressive parameters ϕ's, q moving-average parameters θ's, and their seasonal counterparts, P Φ's and Q Θ's. Notation-wise, it is common to express a seasonal ARIMA model as SARIMA $(p, d, q) \times (P, D, Q)_s$.

Seasonal VARIMA

The seasonal Vector Autoregressive Integrated Moving-Average model is the multivariate generalization of the seasonal ARIMA model. Instead of only relying on the lagged values of a single time series to make forecasts, the seasonal VARIMA model incorporates lagged values from m time series to help make forecasts for each time series. The seasonal VARIMA

model may be expressed as

$$\mathbf{z}_t = \mathbf{c} + \sum_{i=1}^{p} A_i \mathbf{z}_{t-i} + \sum_{i=1}^{q} M_i \boldsymbol{\epsilon}_{t-i} + \sum_{i=1}^{P} U_i \mathbf{z}_{t-is} + \sum_{i=1}^{Q} O_i \boldsymbol{\epsilon}_{t-is} + \boldsymbol{\epsilon}_t \ , \tag{3.7}$$

where $\mathbf{z}_t$ is a vector of dimension m representing the vector of differenced responses at time t, generated similarly as in Equation 3.5 for each of the m concurrent time series; $\boldsymbol{\epsilon}_t$ is a vector of residuals at time t; $\mathbf{c}$ is a vector of intercepts; the parameter matrices A's, M's, U's, and O's[3] are all of dimensions $m \times m$ and are the multivariate generalizations of ϕ's, θ's, Φ's, and Θ's in Equation 3.6, respectively;

3.2.3 Regression Models

Regression models such as linear regression and polynomial regression are usually very efficient and effective. The linear regression model may be expressed as

$$y_t = \mathbf{b} \cdot [1, \mathbf{v}_t] + \epsilon_t \ , \tag{3.8}$$

where $\mathbf{b}$ is a vector of parameters and a value of 1 is prepended to $\mathbf{v}_t$ (in Equation 3.3) to account for the intercept. The left-hand side of Equation 3.8 is taken from the first column of W, representing 1-step ahead training output values. Separate regression models would need to be fitted for subsequent columns of W (see Equation 3.4).

Polynomial regression introduces non-linearity into the model by including additional input features such as the powers of the original input features. A Response Surface model would also include products of all possible pairs of original input features, known as interaction/cross-terms, in addition to the power terms. Recently, there has been inter-

[3]The letters of the matrices are chosen from the first two letters of AUtoregressive and MOving-average.

est in polynomial regression as an alternative to feedforward neural networks [Cheng et al., 2018].

3.2.4 Support Vector Models

Support Vector Regression (SVR) [Drucker et al., 1997, Smola and Schölkopf, 2004], in its simplest linear form, may be expressed as

$$y_t = c + \mathbf{b} \cdot \mathbf{v}_t + \epsilon_t \ , \tag{3.9}$$

where c is the intercept and $\mathbf{b}$ is a vector of parameters. Optimization is performed to minimize

$$\frac{1}{2}||\mathbf{b}||^2 \ , \tag{3.10}$$

subject to the constraint of

$$|y_t - (c + \mathbf{b} \cdot \mathbf{v}_t)| \leq \rho \ , \tag{3.11}$$

that is, the predicted value must be within a threshold ρ of the observed value for all training instances. Often, a non-linear kernel function may be used to transform the training instances into higher dimensional space to fit a curve rather than a line. The parameter vector $\mathbf{b}$ may also be expressed as a linear combination of selected training instances, known as support vectors [Smola and Schölkopf, 2004].

3.2.5 Neural Networks Models

Neural Networks have garnered much attention in recent years, primarily due to the advancement in deep learning research.

Feedforward Neural Networks

A standard 3-layer feedforward Neural Network (NN) with more than one output neuron may be expressed as,

$$\mathbf{y}_t = f_o^a(B_o^T f_h^a(B_h^T \mathbf{v}_t + \mathbf{c}_h) + \mathbf{c}_o) + \boldsymbol{\epsilon}_t \ , \tag{3.12}$$

let k_h represent the number of hidden nodes, then parameters include the $k_i \times k_h$ parameter/weight matrix B_h, the k_h dimensional bias/intercept vector $\mathbf{c}_h$, the $k_h \times k_o$ parameter/weight matrix B_o, and the k_o dimensional bias vector $\mathbf{c}_o$. These parameters are typically learned using gradient-based back-propagation algorithms [Rumelhart et al., 1988]. The two activation functions, f_h^a and f_o^a, output signals from the input layer to the hidden layer, and from the hidden layer to the output layer, respectively. $\mathbf{y}_t$ and $\boldsymbol{\epsilon}_t$ are the k_o-dimensional vectors of outputs and residuals, respectively; Additional hidden layers may be added to a neural network and its forecasted/predicted values may be produced in a similar layer-by-layer manner. The neural network model is also flexible to consider one output feature at a time or all output features at once. Since the information can only be passed in a forward manner, and every pair of adjacent layers are completely connected by edges, such neural networks are also more precisely called feedforward fully connected neural networks.

If a neural network contains many layers, then it is generally referred to as a deep neural network, though there is no universal consensus as to how many layers are considered deep enough [Schmidhuber, 2015]. However, everyone does seem to agree that a neural network with a single hidden layer is considered "shallow". Other structurally varying neural networks also exist; some have been revived from the past and others were newly invented in the wave of deep learning.

Extreme Learning Machine

Extreme Learning Machine (ELM) [Huang et al., 2006] may be viewed as a special type of feedforward neural network. A simple, 3-layer ELM may be efficiently learned in a two-step process.

1. Input-to-Hidden: apply an activation function to the inputs with fixed random weights. This would help introduce non-linearity into the model and the state of the hidden layer may be viewed as encrypted features that have been non-linearly transformed from the inputs.

2. Hidden-to-Output: use linear regression to learn the outputs from the hidden features. The linear regression problem may be effectively solved using pseudo-inverse techniques.

The ELM model has several advantages over a traditional feedforward neural network, such as minimal parameter tuning and low computational cost.

Long Short-Term Memory Neural Networks

The Long Short-Term Memory (LSTM) Neural Network [Hochreiter and Schmidhuber, 1997] is a type of recurrent neural network designed to work with temporal data. The core of an LSTM NN is an LSTM unit, which may also be viewed as a special layer. The input to an LSTM unit/layer must contain an additional temporal dimension to the standard instances $\times$ features training input matrix V used in other machine learning models. In other words, a training input instance to an LSTM unit/layer contains the temporal evolution of the values of the features. The additional temporal dimension can lead to deep LSTM structures that take longer to train, but often produce great results. The designs of the 3-dimensional training input structures are based on the contents of V and are explained in more detail

in Section 3.4.3. Let $\mathbf{v}_t$ be redefined as a vector of input features of a particular training instance at time step t, then an LSTM unit may be defined as

$$\mathbf{f}_t = \sigma(B_f[\mathbf{h}_{t-1}, \mathbf{v}_t] + \mathbf{c}_f) \ , \tag{3.13}$$

$$\mathbf{i}_t = \sigma(B_i[\mathbf{h}_{t-1}, \mathbf{v}_t] + \mathbf{c}_i) \ , \tag{3.14}$$

$$\mathbf{o}_t = \sigma(B_o[\mathbf{h}_{t-1}, \mathbf{v}_t] + \mathbf{c}_o) \ , \tag{3.15}$$

$$\mathbf{m}_t = \mathbf{f}_t * \mathbf{m}_{t-1} + \mathbf{i}_t * \tanh(B_m[\mathbf{h}_{t-1}, \mathbf{v}_t] + \mathbf{c}_m) \ , \tag{3.16}$$

$$\mathbf{h}_t = \mathbf{o}_t * \tanh(\mathbf{m}_t) \ . \tag{3.17}$$

An illustration of the inner workings of an LSTM unit is in Figure 3.6. An LSTM unit/layer contains a cell state that serves as a memory unit ($\mathbf{m}_t$) responsible for maintaining valuable information throughout time. At each time step t, the inputs to the current temporal layer include both the outputs from the previous layer ($\mathbf{h}_{t-1}$) and current input features ($\mathbf{s}_t$). Three gates exist within an LSTM unit/layer that affects the information stored in the cell state: 1) the forget gate ($\mathbf{f}_t$) determines what old information is no longer relevant in the cell state; 2) the input gate ($\mathbf{i}_t$) determines new information that needs to be added to the cell state; 3) the output gate ($\mathbf{o}_t$) determines what output signals to produce based on the contents of the cell state. The four parameter/weight matrices B's and the four intercept/bias vectors $\mathbf{c}$'s may be learned through the backpropagation through time algorithm [Werbos et al., 1990].

Once the final output from an LSTM unit/layer has been obtained at the last time step, then the output may be fed into another neural network layer, such as a fully connected layer described in the previous section, and the final forecasted output may be obtained in a similar layer by layer manner described in Equation 3.12.

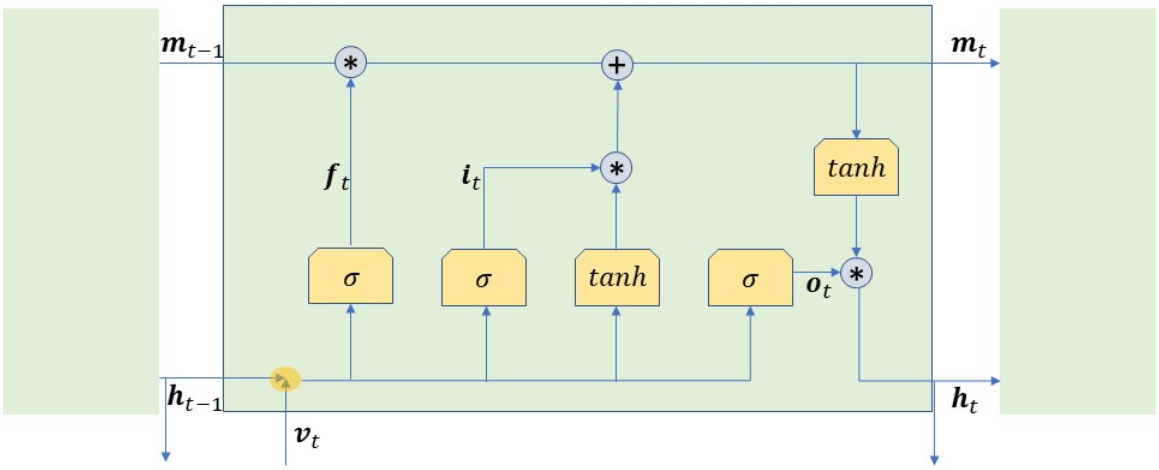

Figure 3.6: An LSTM unit.

A special type of LSTM NN is known as Encoder-Decoder (ED) LSTM Neural Networks [Cho et al., 2014], specifically designed for Sequence-to-Sequence (Seq2Seq) problems. There are two LSTM units in an ED LSTM NN. The first LSTM unit is the encoder which maps the input sequence to a fix-length vector representation. The second LSTM unit is the decoder that maps the fixed-length vector to the output sequence.

Other Deep Learning Models

Other neural network architectures, though not directly used in this study, have been applied in the field of traffic forecasting in recent years. Some of the most important ones are briefly discussed in this section.

Stacked Autoencoders An autoencoder [Ballard, 1987] is a neural network that is designed to reproduce its inputs. The dimensions of the hidden layers should be smaller than those of the input and output layers so that the information from the input neurons can be encoded with fewer hidden neurons, thus achieving an effect of dimensionality reduction. By

stacking layers of autoencoders, successful applications have found in filtering out noise and recover distorted inputs [Vincent et al., 2010]. The dimensionality reduction capabilities of autoencoders can be effective when given large quantities of input traffic data.

Convolutional Neural Network Best known for achievements in image recognition, convolutional neural networks [Krizhevsky et al., 2012] model spatial dependencies by using shared weights in a specific region. The number of parameters is therefore significantly reduced from a fully connected neural network. As a result, convolutional neural networks are much less likely to overfit. Some researchers have used convolutional neural networks to model the spatial dependencies among the traffic sensors.

Gated Recurrent Unit Considered as a variant of LSTM, the Gated Recurrent Unit [Cho et al., 2014] simplifies the architecture of LSTM and is, therefore, able to be trained faster. The forget gates and input gates are merged into a single gate. The cell state and the hidden state are also combined into one.

3.3 Related Work

In an attempt to survey recent studies in the field of traffic forecasting, the following are chosen from the last 10 years due to their importance and relevance in the field. Many studies began in a univariate setting, typically the dataset consists of a few sensors. As much more data have been collected in recent years, researchers have chosen to conduct their studies in larger scales and multivariate experiments have become more common.

Selected studies on univariate traffic flow forecasting are as follows. Cools et al. in 2009 [Cools et al., 2009] studied the effect of holidays on daily traffic flow from 3 years of daily data collected by 4 sensors in Belgium. Forecasts were made for 1-step ahead, the equivalent of one day ahead. The ARIMA family of models were used with additional binary-encoded

exogenous variables representing holidays. No models were included outside the ARIMA family for comparisons. Castro-Neto et al. in 2009 [Castro-Neto et al., 2009] found SVR to outperform exponential smoothing and a small neural network. The size of each of the 3 layers of the neural network was 10, 4 and 1. It would be possible for the neural network to achieve better accuracies if its structure is expanded. The traffic data span 16 days in 5-minute resolution, collected from 7 sensors in PeMS, and forecasts were made for 1-step ahead, the equivalent of 5 minutes ahead. Tan et al. in 2009 [Tan et al., 2009] used a neural network as a meta-learner trained by the outputs of ARIMA, Moving-Average, and exponential smoothing models. The performance was better than ARIMA and a small NN consisting of 3 layers of sizes 3, 16, and 1. Hourly data were collected for 1 year from a single sensor in the Guangzhou province of southern China. Forecasts were produced for up to 3 steps, the equivalent of 3 hours. Hong et al. in 2011 [Hong et al., 2011b] proposed an SVR model that outperformed SARIMA, exponential smoothing, and a small neural network that only has 3 neurons in its hidden layer. The proposed SVR model used a hybrid of genetic algorithm and simulated annealing technique to learn the parameters. Hourly data were collected for two months from 3 sensors in Taipei, Taiwan. Forecasts were made for 1-step ahead, the equivalent of one hour. Lippi et al. in 2013 [Lippi et al., 2013] compared SARIMA, SVR, and NN using 15-minute resolution data collected over 9 months by 16 sensors from PeMS. The SARIMA model performed the best, but the authors' proposed SVR model ran much faster without losing much accuracy. A small NN was also included but did not perform as well. The size of each layer in the NN was 5, 10, and 1. Nine months of data from 16 sensors in PeMS were collected. The data were aggregated into 15-minute resolutions and forecasts were produced for 1-step ahead, the equivalent of 15 minutes. In a study done by Jia et al. in 2017 [Jia et al., 2017], LSTM NN outperformed ARIMA and other types of neural networks. Rainfall data were also included to improve forecasting accuracies. The LSTM NN structure is made up of a single hidden layer, which is the LSTM layer.

The other neural networks contained one to two hidden layers, with 300 to 400 neurons per hidden layer depending on the forecasting step. The data were of 2-minute resolutions and collected from June to August in 2013 and 2014. Four sensors in Beijing were used in the study and forecasts were produced for 10 and 30 minutes ahead.

Multivariate time series forecasting generally relies on using spatially dependent sensor data to improve performance. In a study conducted by Chandra and Al-Deek in 2009 [Chandra and Al-Deek, 2009], a VAR model took advantage of the spatial dependencies between sensors that are on the same freeway and outperformed univariate ARIMA and SARIMA models. The data were of 5-minute resolutions and collected from 5 sensors on I-4 in Florida (near Disney) during March 2003. Forecasts were made for 1-step ahead, the equivalent of 5 minutes. Min et al. in 2010 [Min et al., 2010] used Generalized Space-Time ARIMA, which can be considered as a special case of VARIMA, to model spatial dependencies among sensors on connected roads. Its performance was only compared with Space-Time ARIMA. Data were collected in 15-minute resolutions for one day from 10 sensors in Beijing. Forecasts were made for 1-step ahead, the equivalent of 15 minutes. Zhao et al. in 2017 [Zhao et al., 2017] proposed an LSTM structure, consisting of 2-6 hidden layers depending on the forecast-horizon, outperformed ARIMA, SVR, and other types of neural networks. The data were in 5-minute resolution, collected during the first half-year of 2015, from 500 sensors in the 5th Ring (city bypass) in Beijing. Forecasts were made for up to 1 hour ahead.

It can be seen that the general trend within the last decade has been a transition from mostly univariate studies to multivariate ones, single-step forecast to multi-step forecasts, small amount of data to much larger amount of data, and simple models with few parameters to structurally complex models. Most recently, many researchers have focused their attention on applying deep learning models to traffic flow forecasting. In particular, several such papers studied traffic forecasting using datasets from PeMS. Although not directly comparable with the findings of the study presented here since the datasets used in the various studies are

from different locations in the state of California, one may still be able to obtain some general ideas on the most recent developments in the field of traffic forecasting, particularly the various level of accuracy measures when using datasets from PeMS.

A commonly cited study done by Lv et al. in 2015 [Lv et al., 2015] studied traffic flow forecasting up to 1 hour. The exact geographical locations in which the datasets come from were not explicitly stated by the authors, so they may have used all available datasets in the first 3 months of 2013. The data were also appeared to be aggregated into 15-minute resolution, as a couple of graphs illustrating observed vs. predicted traffic flow values are shown in 15-minute intervals. The authors' proposed model was a deep neural network built using stacked autoencoders and its performance is compared with other machine learning models, namely two other types of neural networks and SVR. The number of hidden layers varies from 2 to 4, depending on the forecasting horizon, and the number of neurons in the hidden layers varies from 200 to 500. The authors proposed model obtained MAPE[4] in the 6% to 7% range for the forecasts in the first 4 steps, though it is rather intriguing that their 45-minute and 60-minute ahead forecasts were more accurate than their 15-minute and 30-minute ahead forecasts. The MAPE values were mostly comparable with the ones in our study, in which slightly lower MAPE values are obtained for the first 2 steps and slightly higher MAPE values for the next 2 steps. Our MAPE values also gradually degrade as the forecasting horizon increases, which is more consistent with common reasoning.

Shao and Soong in 2016 [Shao and Soong, 2016] provided a very small-scale study that focused on a single PeMS sensor located in Irvine, CA. The authors' primary model was an Encoder-Decoder LSTM NN, which achieved an optimal MAPE of 5.4% in the one-step-ahead forecast. The network structure consists of a single hidden LSTM layer with 32 neurons. Other models that the LSTM NN outperformed include machine learning models such as SVR and a stacked autoencoder NN containing 3 hidden layers of 40 neurons each.

[4]In the paper it was called Mean Relative Error

It is more difficult to attempt to draw comparisons between our study and our findings since only the MAPE value from a single sensor was reported. In our results, the MAPE values from different locations can vary quite a bit.

Wu et al. in 2018 [Wu et al., 2018] used data from 33 sensors in Highway I-405 to train their proposed deep neural network built with Convolutional Neural Networks and Gated Recurrent Units to capture spatial and temporal dependencies among the data, respectively. All data are given to a single deep neural network to produce concurrent forecasts for all 33 sensors. The training data were from April 1, 2014, to June 20, 2015, and the testing data were the remaining 10 days from June 2015. The authors' approach was to let a large deep neural network determine any spatial-temporal dependencies among all of the sensors, while our study takes a more "local" approach that focuses on forecasting traffic at a single sensor while considering data from upstream and downstream sensors. Their forecasts were made for up to 9 steps, the equivalent of a 45-minute forecasting horizon, with MAPE values ranging from around 7% to 9%. The best MAPE values for the 9-step ahead forecasts using 5-minute resolution data are around 6% to 8%, which are slightly lower than their MAPE values.

Yang et al. in 2019 [Yang et al., 2019] chose 50 PeMS sensors from the San Diego/Imperial area, which is by far the most similar to the ones in our study, though not as many. The data from March and April of 2017 were used as training data and those from May were used for testing. The authors' proposed model is an LSTM variant that achieved a MAPE of 6.54%, which outperformed several other machine learning models including the stacked autoencoder based deep neural network previously mentioned in [Lv et al., 2015]. Our best 1-step ahead forecast MAPE in 15-minute data resolution is around 5.5%.

While most of the recent deep learning papers focused on accuracy, not many provided the computational costs of their deep neural networks. Another common theme among the recent studies has been to build larger and deeper neural networks, but the actual

performance improvements, say over smaller networks focusing on a local area, do not seem to be sufficiently addressed. The models, including various neural networks, in our study, are not necessarily as large in terms of model structures as those in some of the studies mentioned above, but the forecasting performance seems comparable. Our study also covers a greater number of sensor locations or time range than many of the studies as mentioned earlier, for which the researchers, in an attempt to build huge network structures, may not be able to do so to ensure the training process completes in a reasonable amount of time.

3.4 Evaluations

This section provides details on the datasets, experimental setup, performance evaluations, and thorough discussions of the results.

3.4.1 Dataset Description and Preprocessing

The traffic data are obtained from the Caltrans Performance Measurement System. This study focuses on the San Diego, California area, or district 11 as classified by the California Department of Transportation. All data are from major highways, or Mainline (ML) according to PeMS classification. The data are made available to download in the 5-minute resolution. A separate dataset is also created by down-sampling the original 5-minute resolution data to 15-minute resolution. The traffic flow values are aggregated and traffic speed values averaged for every three 5-minute intervals. This process has an inherent smoothing effect on the data and could help lower the level of noise or sudden fluctuations in the data. A total of 373 sensors are chosen in this study by filtering out only highways with sensors that span at least 10 miles for the multivariate experiment. The size of all the data is approximately 1.5 gigabytes. Table 3.1 contains a summary of the number of sensors selected

Table 3.1: Number of Sensors from Each Highway.

Highway	15 N	15 S	5 N	5 S	52 E	52 W
Sensors	38	54	78	79	4	8
Highway	8 E	8 W	78 E	78 W	805 N	805 S
Sensors	21	20	5	11	28	27

from each highway. All sensors chosen for this study must contain data for the entire year of 2018.

The quality control system of PeMS is very robust. If missing data arise due to sensor failure, PeMS automatically imputes the data and provides the imputed data to the users. The users are also given information on the percentage of observed (non-imputed) data across all lanes at any sensor location. On extremely rare occasions, the data provided by PeMS may contain missing data for certain timestamps. In such cases, we would impute the missing data through linear interpolation. The overall missing value rate is about 10%. All imputed data are included to train models but excluded for performance evaluations.

3.4.2 Evaluation Metrics

Three normalized evaluation metrics are considered, Mean Absolute Percentage Error (MAPE), Normalized Root Mean Squared Error (NRMSE), and coefficient of determination R^2

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{y_t - \hat{y}_t}{y_t}\right|, \tag{3.18}$$

$$NRMSE = \frac{T}{\sum_{t=1}^{T} y_t}\sqrt{\frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{T}}, \tag{3.19}$$

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{T} y_t^2 - \frac{1}{T}(\sum_{t=1}^{T} y_t)^2}, \tag{3.20}$$

where T is the total number of instances in the evaluation set. Any instances in which y_t is imputed or takes on the value of 0 are removed from evaluation. A total of four experiments are conducted, univariate and multivariate experiments using two data resolutions. All experiments are conducted on compute nodes each with a 32-core Intel Xeon Skylake processor and 187GB of RAM from the Sapelo2 cluster of Georgia Advanced Computing Resource Center[5].

3.4.3 Problem Analysis and Modeling

Data from the first 8 months of 2018 are used to train models and the last 4 months are used to evaluate the performances. Only data from workdays are considered, as weekend data are usually of significant different patterns. Such practice is common in the literature, as can be seen in [Lippi et al., 2013, Lv et al., 2015]. Furthermore, the evaluation is focused on daytime traffic from 7:00 AM to 7:00 PM since traffic is most congested and dynamic during the daytime.

Some preliminary work has also been done to produce rolling forecasts by repeatedly re-training models using the most recent data. Minor improvements result from re-training models in weekly and daily frequencies. If models are re-trained hourly, which is more like real-time forecasting, some decent accuracy improvements can be obtained. We plan to devote more effort to real-time forecasting in future work.

Forecasts are produced for 12 steps ahead, or up to 1 hour for the 5-minute resolution dataset and up to 3 hours ahead for the 15-minute resolution dataset. A baseline, weekly historical averages computed from the previous 4 weeks, is included to compare against other forecasting models. Figure 3.7 provides a sample on the forecasting performance of the baseline in the 15-minute data resolution. For comparison purposes, Figure 3.8 illustrates

[5] `https://gacrc.uga.edu/`

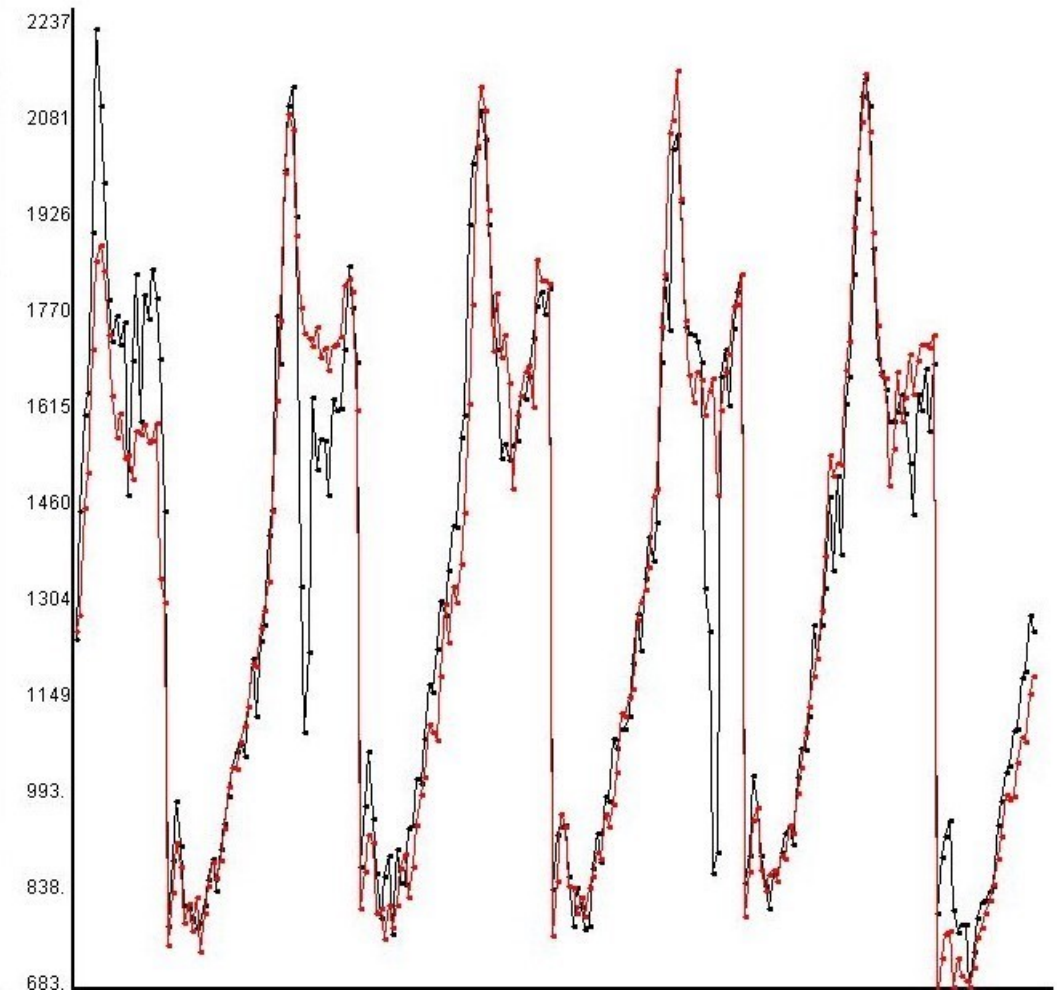

Figure 3.7: Actual traffic flow (black) vs. baseline forecasts (red). Sep 10-14 (M-F), 2018, 7:00 AM-7:00 PM only, sensor ID 1111542 on I-805 S near intersection with El Cajon Blvd in San Diego.

the 1-step ahead forecasting performance on the same location and time by using the linear regression model, which is more accurate than the baseline.

Univariate Experiments

In the univariate experiment, a model is trained only with historical data from one particular sensor. The SARIMA $(1, 0, 1) \times (0, 1, 1)$ model is simple model found in literature [Shekhar

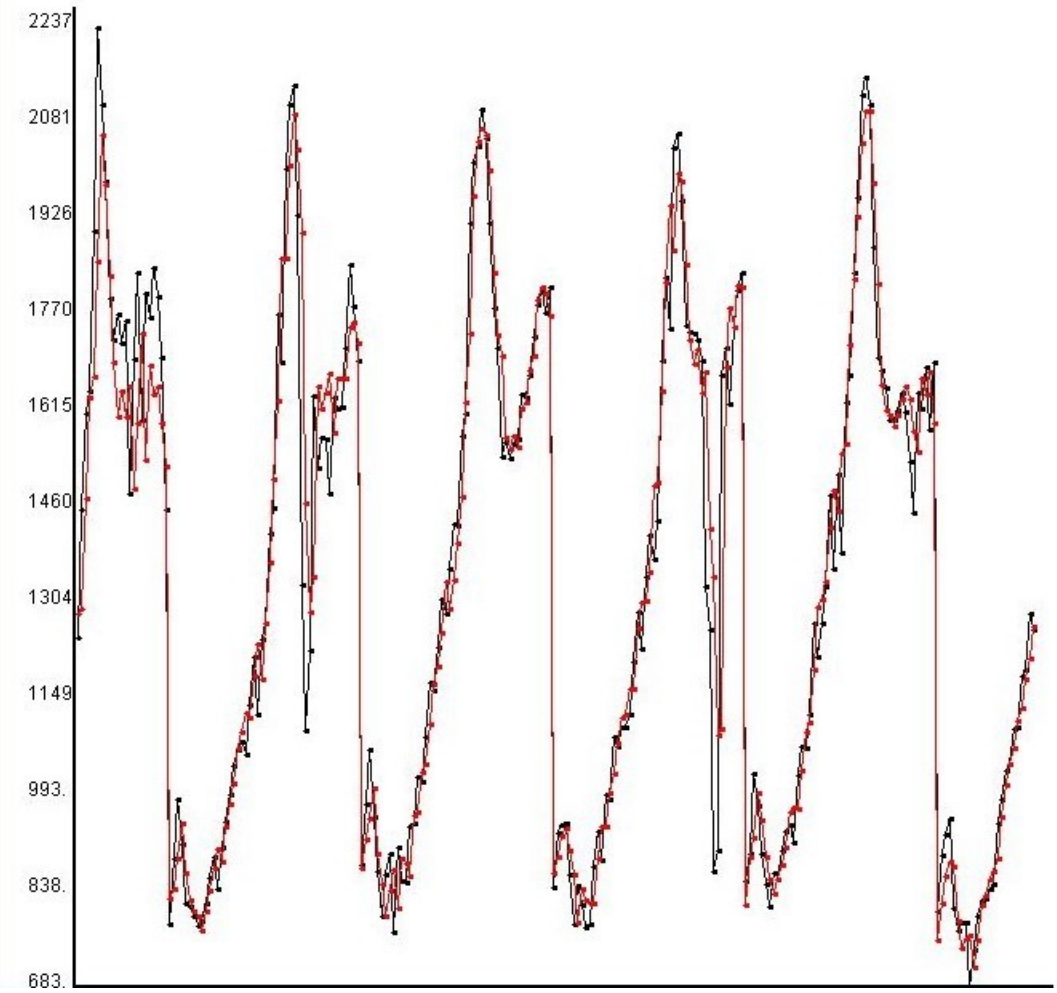

Figure 3.8: Actual traffic flow (black) vs. linear regression forecasts (red). Sep 10-14 (M-F), 2018, 7:00 AM-7:00 PM only, sensor ID 1111542 on I-805 S near intersection with El Cajon Blvd in San Diego.

and Williams, 2008, Lippi et al., 2013, Peng et al., 2018]. The seasonal period is one week. The parameters are optimized using Maximum Likelihood Estimation (MLE). This model is used for both the 5-minute and 15-minute resolution data. Due to the model's simplicity, it may be considered as an alternative baseline to benchmark other models.

Using randomly selected sensor data, we have also attempted to construct other SARIMA models using a grid search like algorithm proposed by [Hyndman et al., 2007] based on the BIC [Schwarz et al., 1978] scoring function. For the 5-minute resolution data, the SARIMA $(12, 1, 4) \times (1, 1, 1)_{1440}$ is found to have the best BIC value. The seasonal period of 1440 still refers to one week (excluding weekends) in 5-minute resolution. For the 15-minute resolution data, the SARIMA $(5, 0, 3) \times (1, 1, 1)_{480}$, using weekly seasonality, is found to be the best. Both of these two models are referred to as SARIMA2 in later figures and tables, and the distinction can be made obvious based on the data resolution.

For the majority of other models, the training input and output matrices, V and W, are scaled within the range of 0 and 1 using Min-Max normalization. Various regression models are considered, including multiple linear regression (Reg), quadratic regression (QuadReg), response surface (RespSurf), which is quadratic regression with additional interaction/cross-terms, and cubic regression (CubicReg). These regression models tend to be very fast and their parameters are solved using the QR factorization technique [Francis, 1961, Kublanovskaya, 1962].

The ν-SVR model, in which the parameter ν controls the number of support vectors, is used in this study. The ScalaTion implementation is based on the LIBSVM package [Chang and Lin, 2011]. The alternative Epsilon-SVR was attempted but resulted in a higher number of chosen support vectors but lower accuracy. The Radial Basis Function (RBF) kernel is used. Other kernels such as linear and polynomials kernels were also tested, but the RBF kernel gives the best performance. Grid search is used to determine the parameter ν and the cost. For the 5-minute resolution data, ν is set to 0.05 and cost is set to 1.0; for the

15-minute resolution data, ν is set to 0.1 and cost is set to 5.0. The values of the parameter ν are rather small because traffic flow data tend to have very strong seasonal/repeated patterns, therefore the proportion of boundary data points serving as support vectors is generally low. All other parameters are left to LIBSVM recommended defaults.

For the Extreme Learning Machine, the *tanh* activation function connecting the input layer and the hidden layer is found to work well. The training data are therefore scaled between -1 and 1. The number of neurons in the hidden layer is tuned to be 8 times the number of input neurons plus one, to draw similarities with a regression model that contains non-linearly transformed features that are 8 times the original number of features plus one to account for the intercept.

The Feedforward Neural Network, denoted as NN in later figures, is made up of 4 layers. The number of layers is chosen to be 4 since such models can extract features better than shallow neural networks (e.g., 3-layer NN) [Schmidhuber, 2015]. Neural Networks of other depths are planned to be tested in future work. The two hidden layers are using leaky ReLU activation functions [Maas et al., 2013], for which the alpha parameter representing the negative slope is tuned to 0.3, which is also the default value in Keras. The output layer is using the identity activation function. The size of each hidden layer is set to be the average of the size of its previous layer and that of the output layer. The number of training epochs is tuned to be 300 and batch size is 32, which agrees with the recommend default batch size in recent findings [Bengio, 2012, Masters and Luschi, 2018].

The first LSTM NN, simply denoted as LSTM in later figures, is similar to a vanilla LSTM NN, but with one additional hidden layer using the leaky ReLU activation function before the output layer using the identity activation function. Its first hidden layer is the LSTM layer, which requires 3-dimensional input unlike the previously mentioned regression and machine learning models. Instead of the instances $\times$ features training input matrix, the LSTM layer requires a 3-dimensional structure of instances $\times$ time steps $\times$ features as

input. Because of the addition of the time steps dimension, and to maintain the overall design of the input matrix V, this LSTM NN is trained with the weekly evolution of the same features defined in V. To construct the input structure, the instances in the training 2-dimensional input matrix V used in other models are simply grouped weekly for 4 weeks. As a result, the time steps becomes 4, and the number of features remains the same, but now the temporal evolution of those features for 4 weeks are represented. Similar to NN, the size of each hidden layer is set to the average of the size of its previous layer and that of the output layer. The number of training epochs is tuned to be 100. Both NN and LSTM also have the option of training to forecast 12 output features at once or considering one output feature at a time but train 12 separate networks. We have chosen the latter approach so that all the weights in a neural network may be devoted to forecasting one particular time step into the future.

The second LSTM NN is the Encoder-Decoder LSTM NN, simply denoted as ED LSTM in later figures. The number of neurons for the encoder LSTM unit and decoder LSTM unit is both tuned to be 150. The output layer, similar to the previous two neural networks, is using the identity activation function. The Encoder-Decoder LSTM NN is designed to forecast a sequence of output values (12 in this study) and its input sequence is best represented in the "time steps" dimension rather than in the "feature" dimension in the previous LSTM NN design. As a result, the 3-dimensional input structure is re-designed as follows: The number of time steps is 24, consisting of 12 seasonal components and 12 most recent observations. A feature is now re-defined to mean a single time series. For the univariate case, the time series of traffic flow and the time of the day are considered. The number of training epochs is tuned to be 50. The Encoder-Decoder LSTM NN also inherently considers all entire output sequence as a whole, so training one such network per sensor is preferred.

All three neural networks in Keras are trained using Adam [Kingma and Ba, 2014] using mean squared error as the loss function. To reduce overfitting, a random 20% slice of the

available training instances is reserved as validation set while the neural networks are trained with the remaining 80% of data. Any other parameter not explicitly stated is left to Keras' default value.

Multivariate Experiments

On a given highway, there can be many sensors. Intuitively, data from traffic sensors nearby are spatially dependent. Data from an upstream sensor can provide information on upcoming congestions while data from downstream sensors determines the rate of traffic flow down the road. In this multivariate traffic flow forecasting experiment, traffic sensors on a particular highway are first sorted by either longitude or latitude, depending on the direction of the highway; then they are divided into groups of 3, with a distance of at least 5 miles between any neighboring sensors in the group. Most PeMS sensors in the area are approximately half a mile apart. In a preliminary experiment in which the most closely located 3 sensors were grouped, the improvements of most multivariate models seemed marginal. This was likely because the sensors were simply located too closely together and could only cover a short segment of the highway. The data patterns were also likely to be extremely similar for the most closely located sensors.

The central sensor in each group is the focus of forecast, considering data from both the upstream sensor and the downstream sensor. In this experiment, both flow and speed from all 3 sensors are considered and features are generated similarly as in the univariate experiment for each of the 6 concurrent time series. The seasonal VARIMA $(1, 0, 1) \times (0, 1, 1)$ is used with matching orders from its univariate counterparts. However, no VARIMA equivalents are considered for the matching SARIMA2 models, for the following reasons: 1) the optimization of parameters of a high-order VARIMA model can be very computationally expensive; 2) though the univariate SARIMA2 models generally outperform SARIMA models, the SARIMA2 models still reside in the lowest tier of models in terms of accuracy, implying

that their matching VARIMA models are very likely to exhibit similar performance patterns. Therefore, time and computational resources are devoted to other models with better performance.

For other models, due to a large number of available features, a simple feature selection process is also conducted to extract 72 most useful features, which would be approximately one-third of all the available features. The feature selection is based on a repeated forward selection process that picks the next best feature which optimally improves the overall adjusted R^2 when fitting a linear regression model with the output feature in step 12, which is the hardest one to forecast. The selected input features are used with all regression and machine learning models except for Encoder-Decoder LSTM NN because its inputs are designed quite differently from the other models.

Besides the expanded training inputs, the other setups of the multivariate models are mostly kept the same with their univariate counterparts, with the following exceptions. In preliminary testings, the response surface model did not yield decent results in the multivariate experiment due to the increase in the number of input features, and as a result, an increase in quadratic order of the number of cross/interaction terms. Therefore, the response surface model is excluded from the multivariate experiments. The number of neurons in the hidden layer of ELM is re-tuned to be 5 times the number of input neurons plus one. For Encoder-Decoder LSTM NN, the sizes of the encoder and decoder LSTMs are also increased to 200 to accommodate inputs that are now larger in dimensions.

3.4.4 Forecasting Evaluations

The forecasting evaluation results are aggregated from all 373 sensors using a weighted scheme, since each sensor may have a slightly different number of observed (non-imputed) values in the evaluation set.

Forecasting Accuracies using 5-Minute Resolution Data

The first experiment is conducted using traffic data in 5-minute resolution, which are directly provided by PeMS. Figure 3.9 in particular illustrates the performance comparisons of the univariate models using the MAPE metric. The figure contains two graphs, and the lower graph simply zooms in on the top performers in the top graph for easier readability. The forecasting accuracies are plotted across 12 forecasting steps, equating to a forecasting horizon of 1 hour. Since the baseline exclusively relies on weekly historical data, its performance is independent of the forecasting horizon, e.g., the 4-week historical average value for this Monday at 8:00 AM remains the same whether the time of making a forecast is 7:00 AM (12-step ahead) or 7:55 AM (1-step ahead). Of course, all other models besides the simple baseline exhibit better forecasting performances for shorter a shorter forecasting horizon than a longer one.

The forecasting performances of the two SARIMA models are the lowest among all the models besides the baseline, and it is worth noting that the other models do take in a much greater number of input features. The next tier of forecasting models is a group of regression models. Linear regression is the least accurate in this tier, while quadratic regression and cubic regression both introduce non-linearity into the features and produce better accuracies. The response surface model is a quadratic regression model with cross/interactive terms and creates a reasonable boost in performance.

Among the machine learning models, SVR performs slightly better than the quadratic and cubic regression models but not as good as the response surface model. The ELM model, which has taken ideas from both neural networks and regression, performs better than the response surface model, which is the top performer within the regression family.

The top tier of forecasting models belongs to the family of neural networks. The 4-layer feedforward NN is slightly worse than the LSTM NN, particularly in the early steps. The Encoder-Decoder LSTM NN is the overall top performer among all models and across

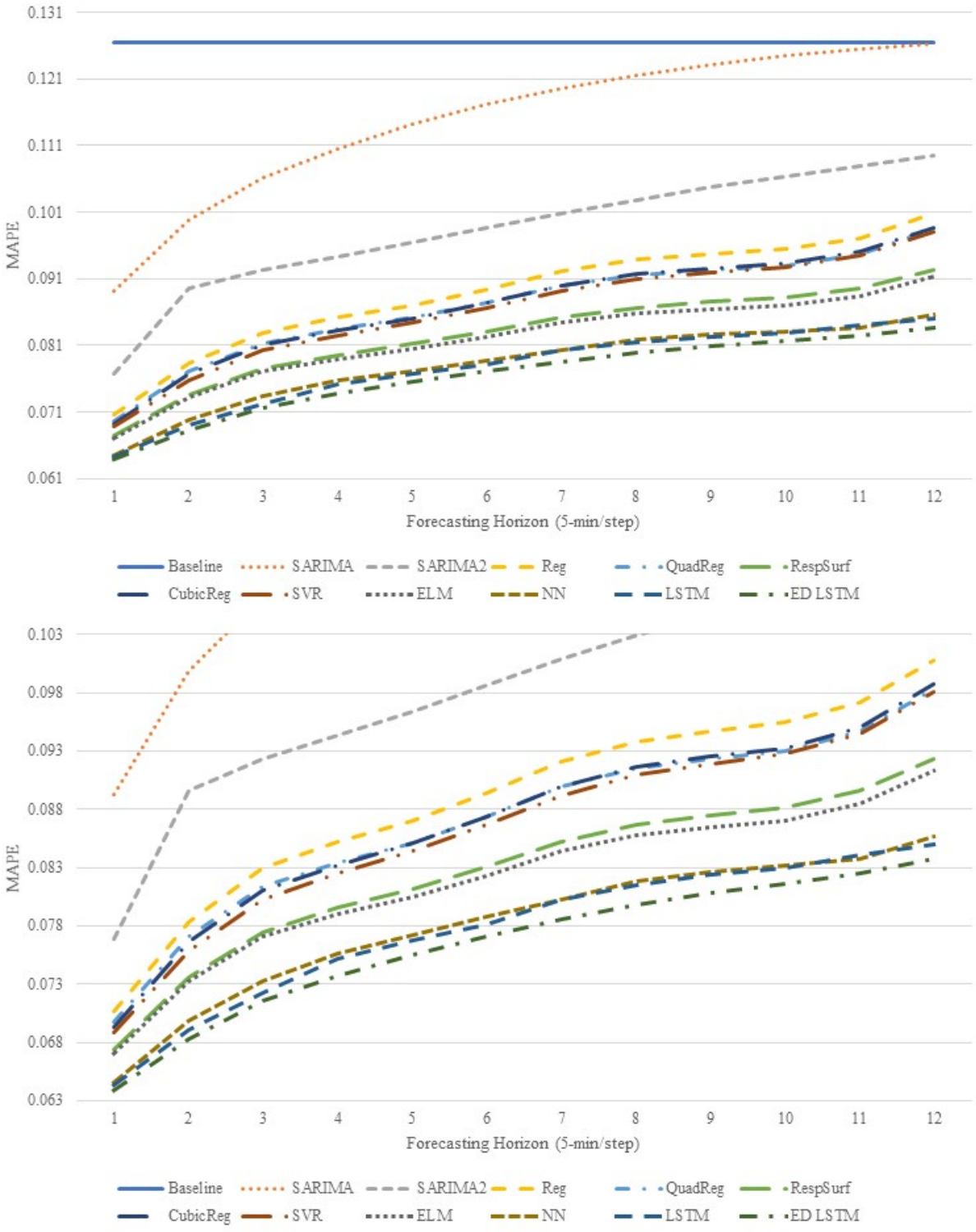

Figure 3.9: Univariate MAPE comparisons in 5-min resolution. Lower graph zooms in on the top performers.

all forecasting steps. It seems that the deep structure of Encoder-Decoder LSTM NN has captured the temporal dependency within the time series quite well. The cell state memory within the LSTM unit also seems to have helped to retain useful information.

The performance comparison using the MAPE metric of the forecasting models in the multivariate setting is shown in Figure 3.10. The relative standings of the models have mostly remained the same, with a few notable exceptions. Among the multivariate forecasting models, quadratic and cubic regression models have overtaken SVR in terms of performance. The multivariate SVR model can take advantage of spatial dependencies and improve upon its univariate counterpart, but the improvements are not as much as those of quadratic and cubic regression models. ELM has also further closed its gap with other types of neural networks. The standard feedforward NN has slightly improved its performance upon its univariate counterparts, but the other two types of LSTM Neural Networks seem to suffer from overfitting that results in slightly lower accuracies.

Figure 3.11 and Figure 3.12 are the performance comparisons in the NRMSE metric of the univariate and multivariate models, respectively. The univariate comparisons bear many similarities to the univariate MAPE comparisons in Figure 3.9. In the multivariate comparisons, the feedforward NN becomes the top performer and ELM achieves on-par performance with Encoder-Decoder LSTM NN.

Figure 3.13 and Figure 3.14 show the performance comparisons using the R^2 metric of the univariate and multivariate models, respectively. Once more, the univariate R^2 comparisons agree with those of the MAPE and NRMSE metrics. In the multivariate comparisons, the feedforward NN and ELM models are the top performers, with NN having a slight edge overall.

Table 3.2 shows the average improvements across all 12 steps of the multivariate models upon their univariate counterparts in the 3 evaluation metrics. The percentages are computed by first taking the difference between the univariate metric and the multivariate

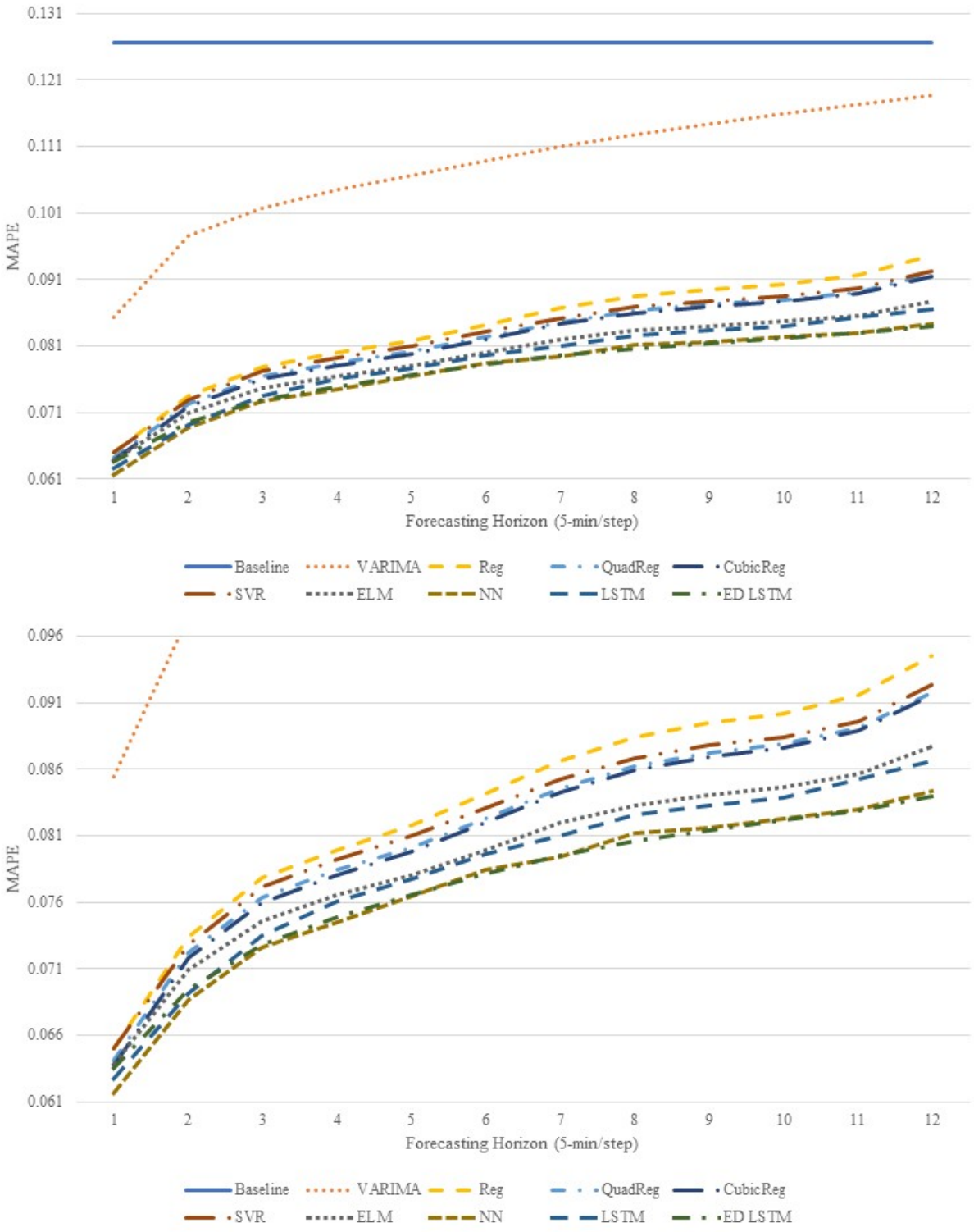

Figure 3.10: Multivariate MAPE comparisons in 5-min resolution. Lower graph zooms in on the top performers.

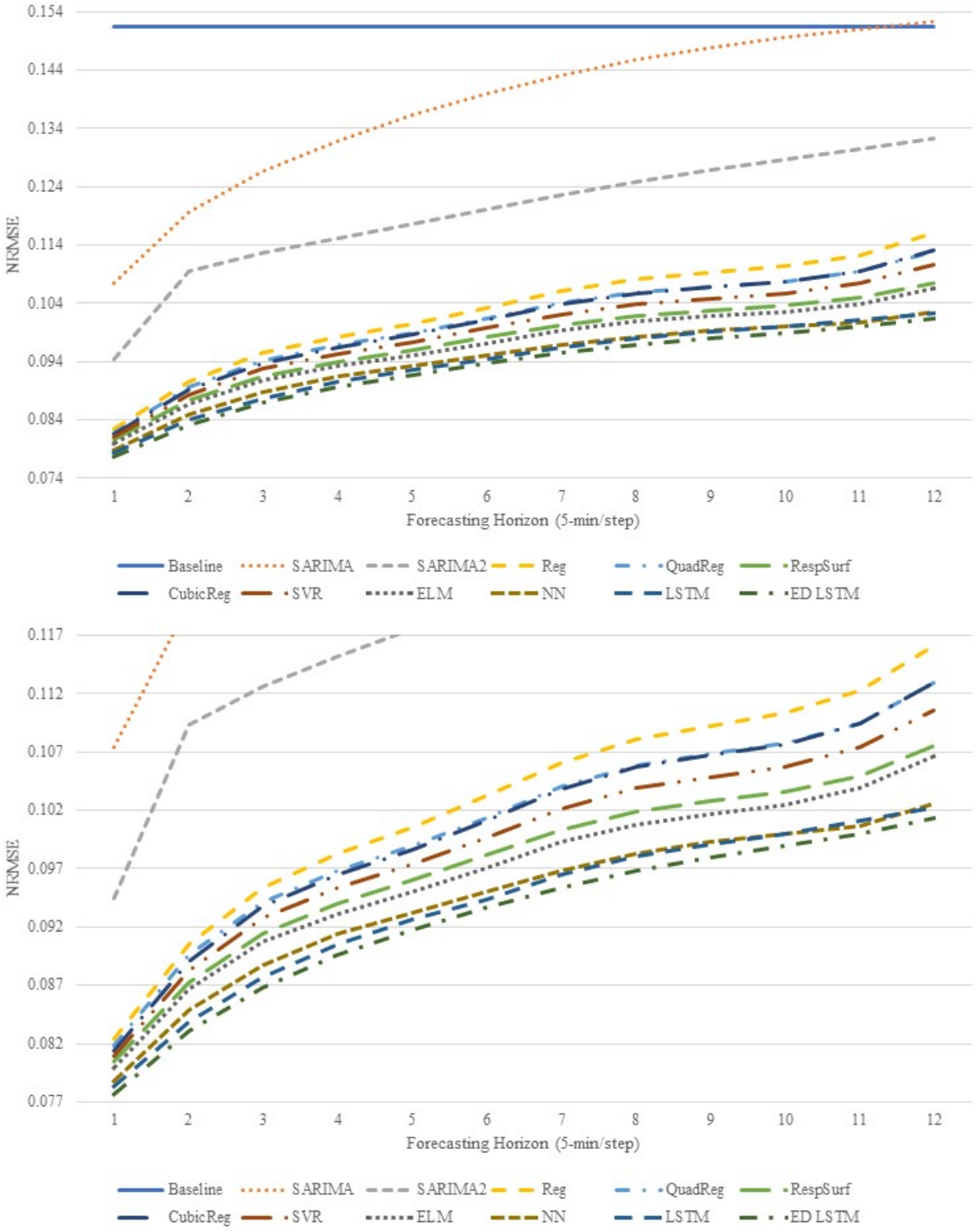

Figure 3.11: Univariate NRMSE comparisons in 5-min resolution. Lower graph zooms in on the top performers.

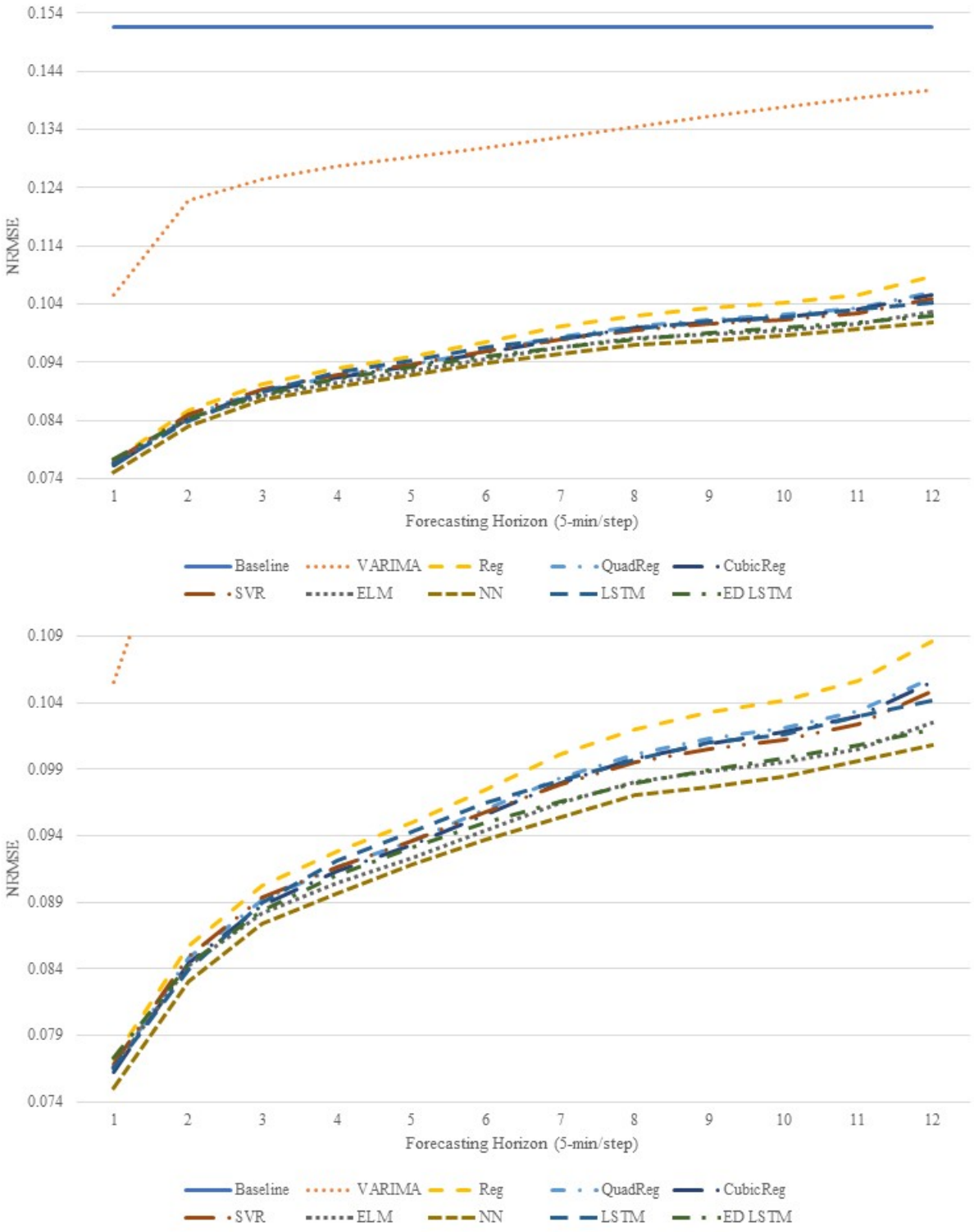

Figure 3.12: Multivariate NRMSE comparisons in 5-min resolution. Lower graph zooms in on the top performers.

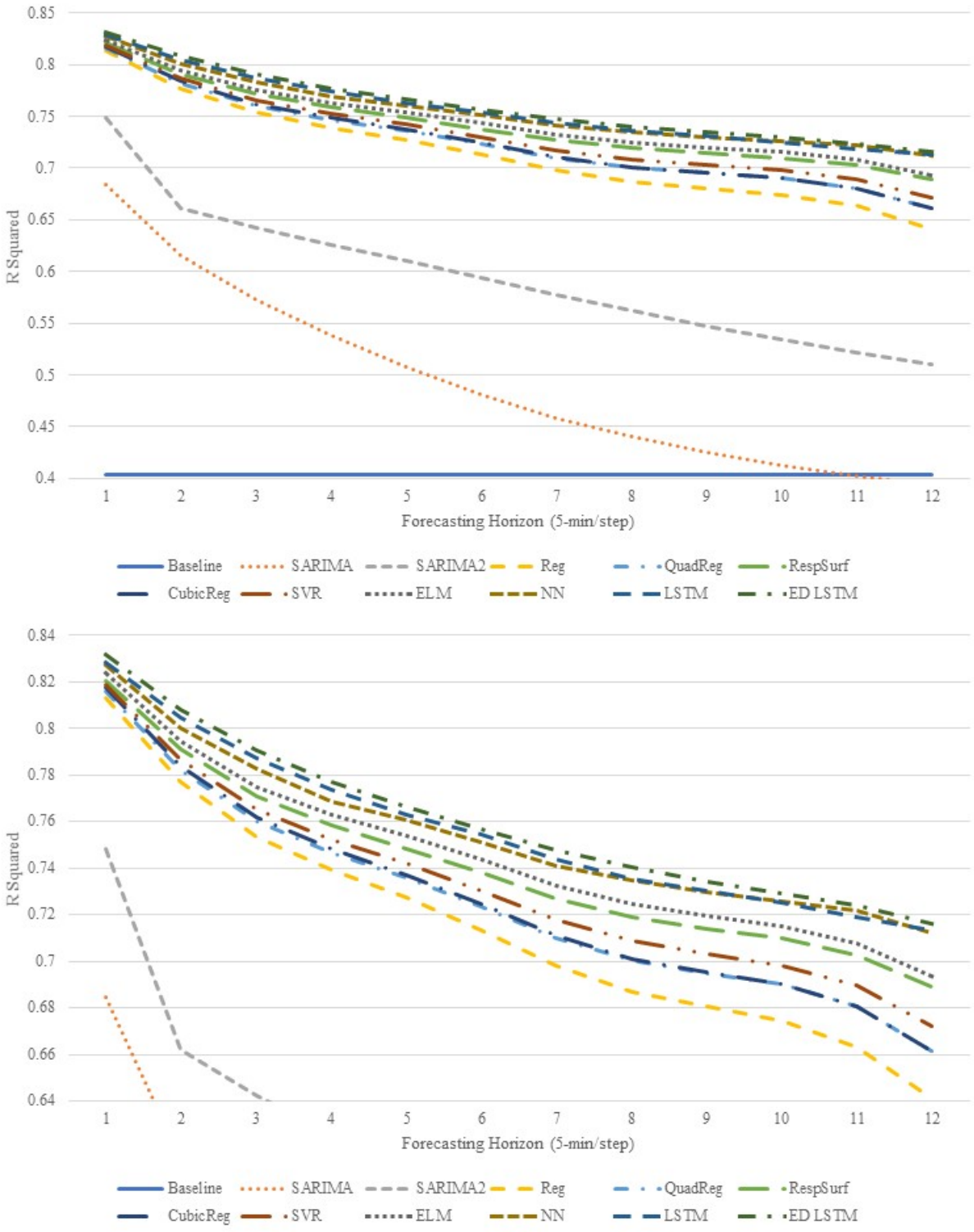

Figure 3.13: Univariate R^2 comparisons in 5-min resolution. Lower graph zooms in on the top performers.

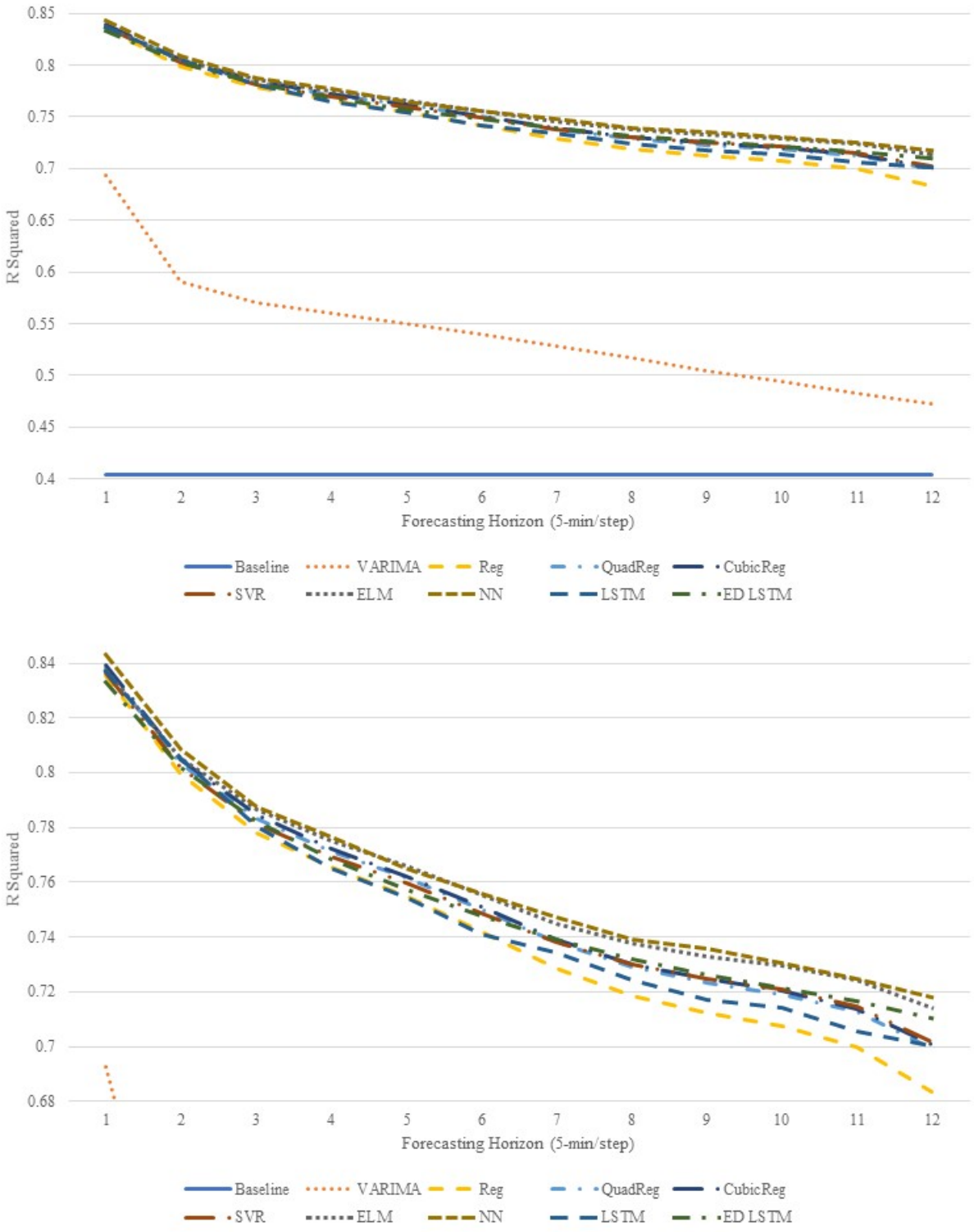

Figure 3.14: Multivariate R^2 comparisons in 5-min resolution. Lower graph zooms in on the top performers.

Table 3.2: Average Improvements of Multivariate Models over Univariate Models (5-min)

Model	MAPE	NRMSE	R^2
VARIMA	5.93%	5.16%	11.07%
Reg	6.08%	5.69%	4.26%
QuadReg	6.10%	5.52%	3.82%
CubicReg	6.45%	5.63%	3.84%
SVR	4.53%	4.22%	2.82%
ELM	3.23%	3.02%	1.84%
NN	1.35%	1.80%	0.84%
LSTM	-1.01%	-1.34%	-1.12%
ED LSTM	-1.01%	-1.12%	-0.94%

metric, and then divide by the univariate metric. An additional negative sign is applied to the R^2 metric calculations so that all positive percentages across the 3 metrics indicate improvements. It can be observed that generally, the time series models and regression models tend to have greater improvements than the machine learning models when given spatially dependent data from the upstream and downstream sensors. As for the two LSTM Neural Networks, the presence of data from two additional sensors seems to have also introduced more noise that the LSTM NNs, unfortunately, have unnecessarily fitted, even by using validation sets during training does not seem to completely prevent overfitting.

Forecasting Accuracies using 15-Minute Resolution Data

The second experiment is conducted by down-sampling the original 5-minute resolution data to 15-minute resolution, a common traffic data resolution found in the literature. The number of steps ahead to produce forecasts is still 12, which is equivalent to making forecasts up to 3 hours ahead.

Figure 3.15 and Figure 3.16 are the performance comparisons using the MAPE metric of the univariate and multivariate models, respectively. The SARIMA model has performed

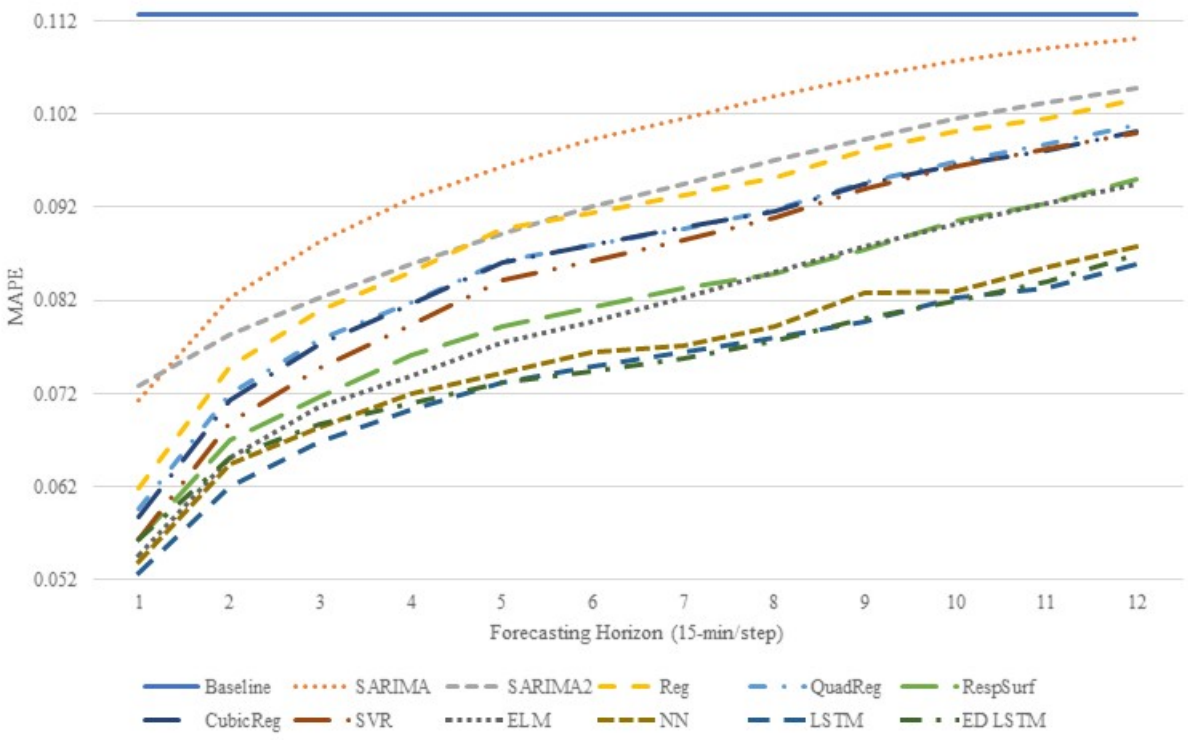

Figure 3.15: Univariate MAPE comparisons in 15-min resolution.

considerably better than its 5-minute data resolution counterpart, most likely due to the reduction of noise through down-sampling, though it still resides in the lowest tier of all forecasting models presented in the figure. The SARIMA2 model has also improved and its performance is only slightly worse than linear regression overall. The other models in Figure 3.15 have mostly maintained their relative standings when comparing with their 5-minute resolution counterparts. In the multivariate comparisons in Figure 3.16, Encoder-Decoder LSTM NN is the overall top performer, though NN and LSTM NN do perform slightly better in the first few early steps.

Figure 3.17 and Figure 3.18 are the NRMSE comparisons of the univariate and multivariate models, respectively. Figure 3.19 and Figure 3.20 are the R^2 comparisons of the

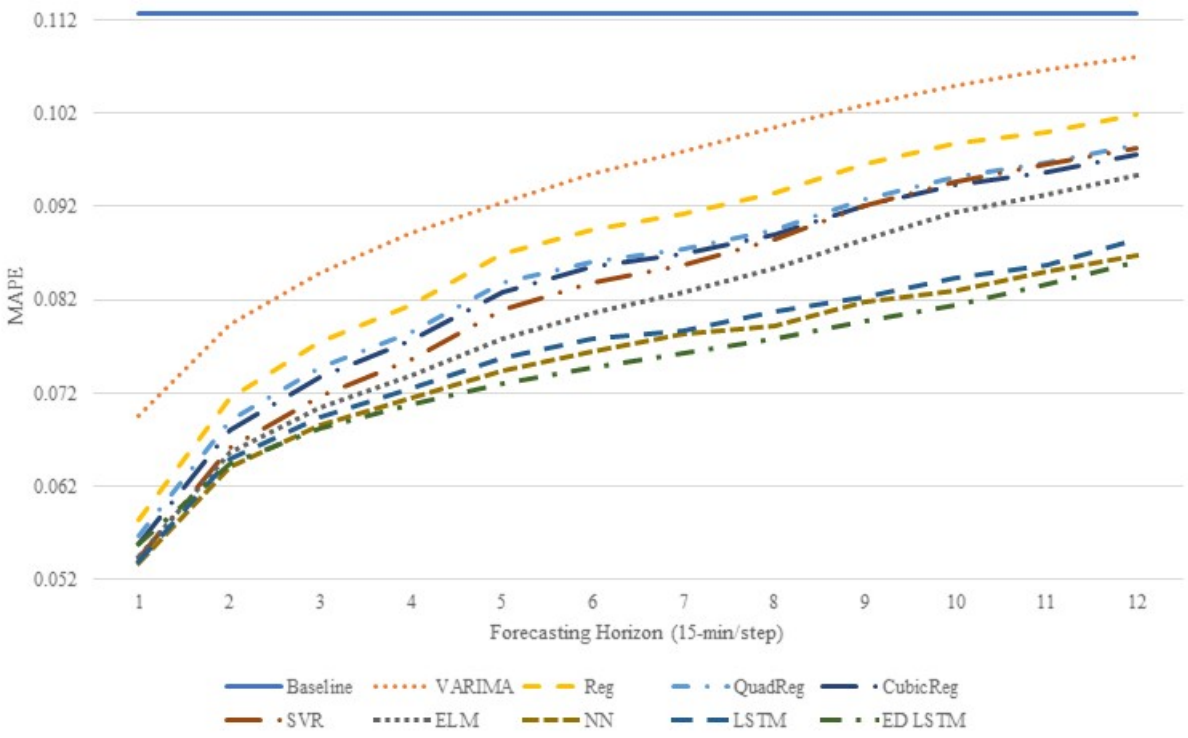

Figure 3.16: Multivariate MAPE comparisons in 15-min resolution.

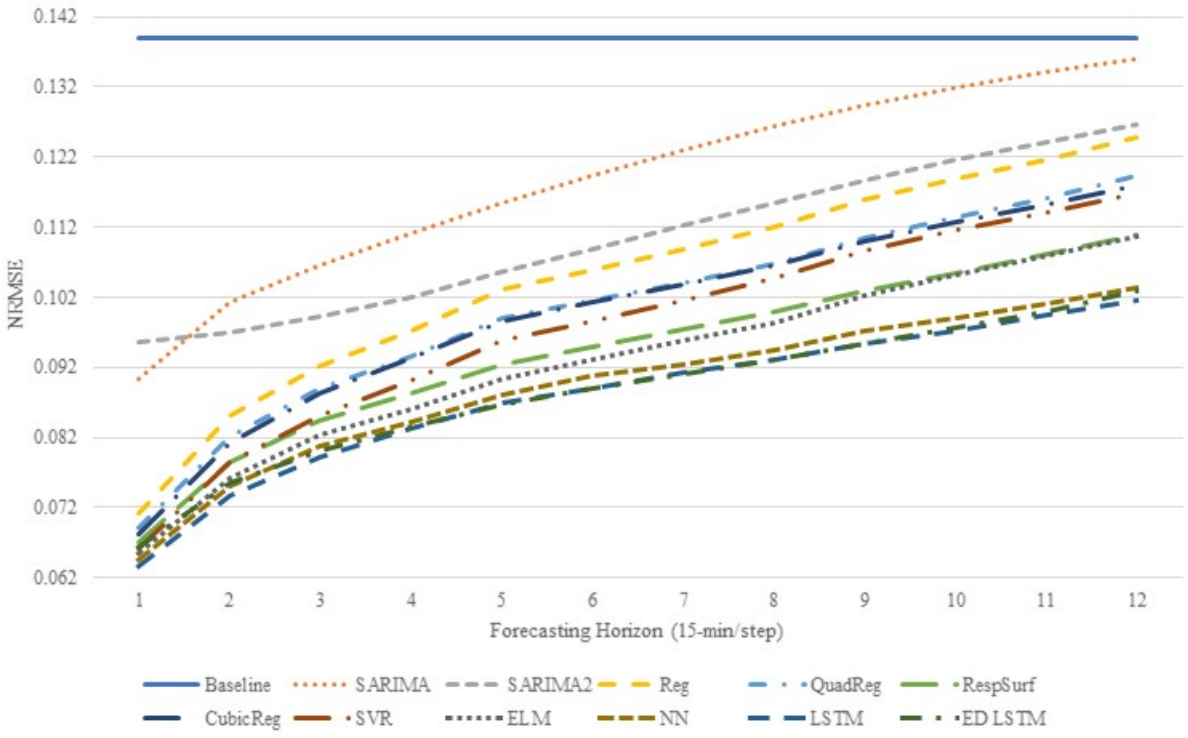

Figure 3.17: Univariate NRMSE comparisons in 15-min resolution.

univariate and multivariate models, respectively. Both sets of figures are consistent with the findings of the MAPE figures.

Table 3.3 provides the average improvements of the multivariate models over their univariate counterparts. Notably, the Encoder-Decoder LSTM NN does improve in all 3 metrics by using 15-minute resolution data, possibly due to the noise reduction effects of down-sampling. On the other hand, The LSTM NN still seems to overfit. It is likely because this model, in particular, considers selected features up to 4 weeks while other models only consider the seasonal component from the previous week. The additional data in two extra sensors from the past four weeks might have provided more noise than useful information.

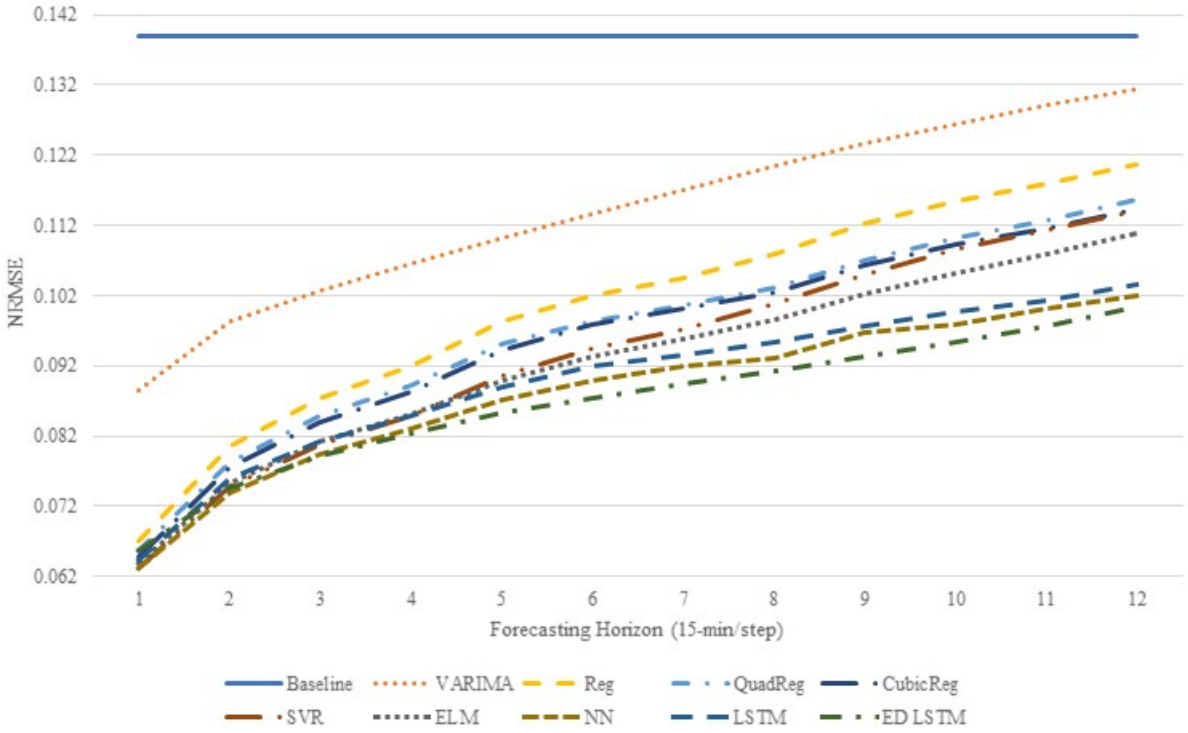

Figure 3.18: Multivariate NRMSE comparisons in 15-min resolution.

Table 3.3: Average Improvements of Multivariate Models over Univariate Models (15-min)

Model	MAPE	NRMSE	R^2
VARIMA	3.21%	3.93%	5.25%
Reg	2.82%	4.21%	3.55%
QuadReg	2.84%	3.72%	2.64%
CubicReg	3.48%	3.93%	2.69%
SVR	3.00%	4.03%	2.64%
ELM	-0.55%	0.47%	0.10%
NN	0.24%	1.19%	0.16%
LSTM	-3.29%	-2.34%	-1.80%
ED LSTM	0.36%	1.74%	0.81%

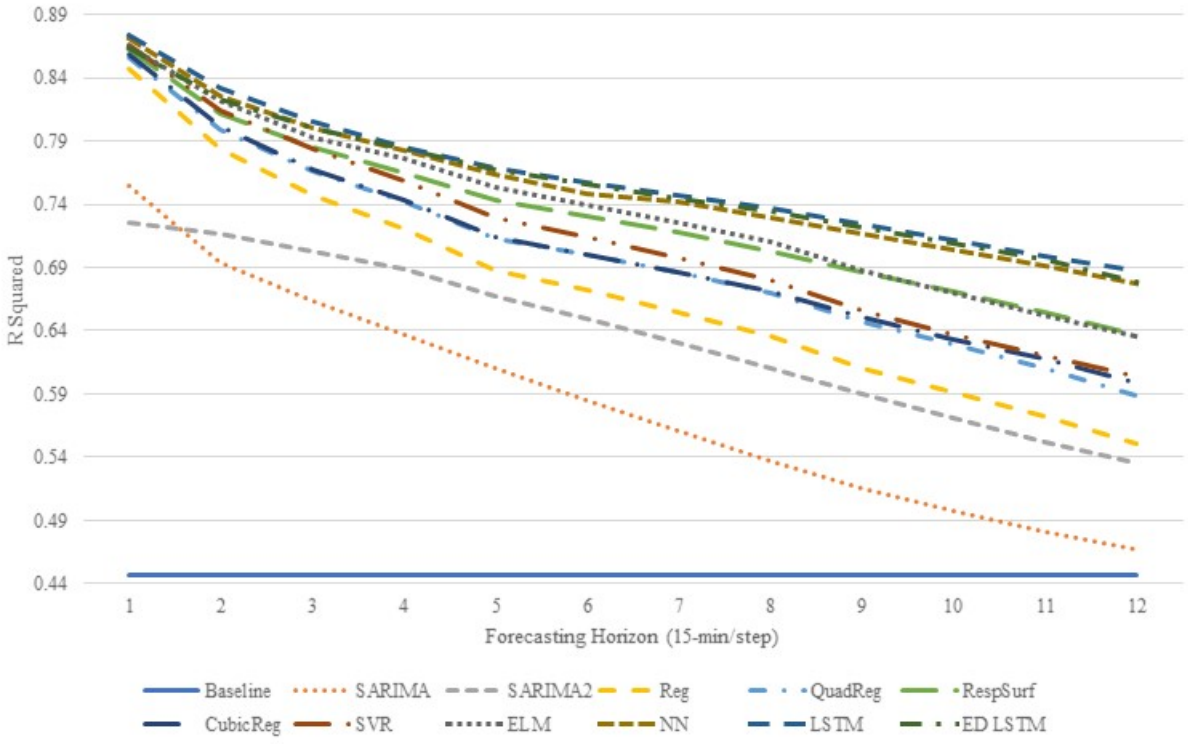

Figure 3.19: Univariate R^2 comparisons in 15-min resolution.

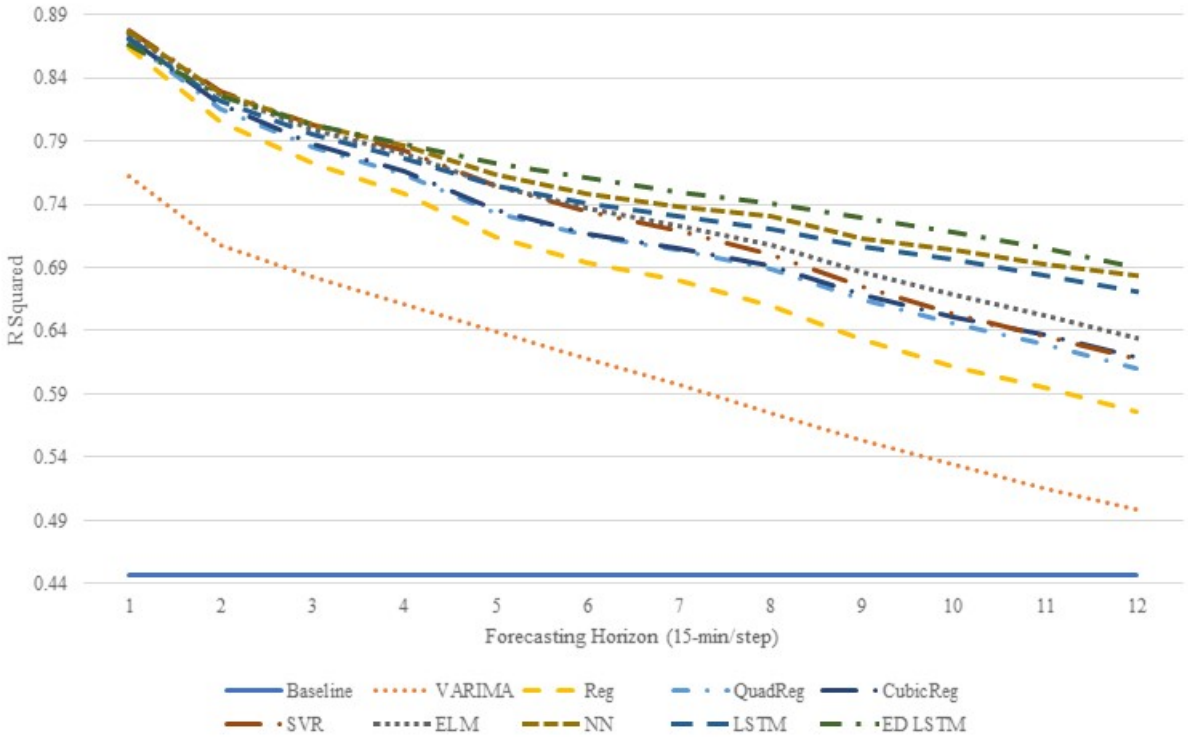

Figure 3.20: Multivariate R^2 comparisons in 15-min resolution.

Table 3.4: Average means and standard deviations using univariate 5-min resolution data.

Model	MAPE-m	MAPE-sd	NRMSE-m	NRMSE-sd	R^2-m	R^2-sd
Baseline	12.66%	7.39%	15.15%	3.02%	40.36%	23.84%
SARIMA	11.48%	5.84%	13.76%	3.04%	49.42%	22.36%
SARIMA2	9.84%	5.49%	11.96%	3.02%	59.47%	23.67%
Reg	8.90%	5.50%	10.27%	2.19%	71.39%	13.93%
QuadReg	8.70%	5.42%	10.08%	2.25%	72.51%	13.77%
CubicReg	8.70%	5.47%	10.06%	2.21%	72.61%	13.58%
SVR	8.63%	5.81%	9.91%	2.41%	73.20%	14.14%
RespSurf	8.26%	5.43%	9.74%	2.39%	74.07%	14.28%
ELM	8.19%	5.47%	9.64%	2.21%	74.55%	13.31%
NN	7.81%	5.38%	9.41%	2.27%	75.46%	13.70%
LSTM	7.76%	5.66%	9.37%	2.34%	75.65%	13.78%
ED LSTM	7.64%	5.59%	9.27%	2.28%	76.01%	13.81%

Model Rankings and Stability

Table 3.4 and Table 3.5 provide the average means ("-m") and standard deviations ("-sd"), respectively, across all datasets and all steps using the 5-minute resolution data. The standard deviations can be used to measure the level of stability of the models across a variety of datasets. Table 3.6 and Table 3.7 are similar tables for the 15-minute resolution data. The models in the tables are also ranked according to the mean MAPE values; the best performing models are placed at the bottom of the tables.

From the four tables, it may be observed that typically, regression models tend to reduce their standard deviations when presented with spatially dependent data in the multivariate experiment while the machine learning models tend to have greater standard deviations when given data from more sensors. The regression models appear to be more robust in guarding against noise when presented with more data while the machine learning models such as neural networks are more prone to be misled by the additional noise, though their forecasting accuracies do also tend to be higher.

Table 3.5: Average means and standard deviations using multivariate 5-min resolution data.

Model	MAPE-m	MAPE-sd	NRMSE-m	NRMSE-sd	R^2-m	R^2-sd
Baseline	12.66%	7.39%	15.15%	3.02%	40.36%	23.84%
VARIMA	10.79%	5.84%	13.01%	3.15%	54.20%	22.97%
Reg	8.36%	5.55%	9.69%	2.17%	74.38%	13.12%
SVR	8.24%	5.88%	9.49%	2.54%	75.24%	13.90%
QuadReg	8.17%	5.38%	9.52%	2.16%	75.24%	12.83%
CubicReg	8.14%	5.34%	9.49%	2.12%	75.36%	12.82%
ELM	7.93%	5.66%	9.35%	2.18%	75.91%	13.06%
LSTM	7.85%	5.44%	9.50%	2.38%	74.83%	14.53%
ED LSTM	7.72%	5.29%	9.38%	2.35%	75.30%	14.79%
NN	7.70%	5.56%	9.25%	2.22%	76.10%	13.67%

Table 3.6: Average means and standard deviations using univariate 15-min resolution data.

Model	MAPE-m	MAPE-sd	NRMSE-m	NRMSE-sd	R^2-m	R^2-sd
Baseline	11.27%	13.58%	13.89%	2.92%	44.64%	24.01%
SARIMA	9.74%	12.39%	11.87%	2.83%	58.34%	20.17%
SARIMA2	9.17%	12.34%	11.06%	2.74%	63.66%	18.81%
Reg	8.96%	12.53%	10.47%	2.32%	67.27%	15.69%
QuadReg	8.65%	12.72%	10.04%	2.38%	70.06%	15.09%
CubicReg	8.61%	12.76%	9.98%	2.37%	70.35%	15.16%
SVR	8.48%	12.65%	9.76%	2.36%	71.33%	14.59%
RespSurf	8.04%	12.52%	9.42%	2.56%	73.05%	16.14%
ELM	7.95%	12.42%	9.28%	2.49%	73.58%	16.60%
NN	7.54%	12.43%	8.92%	2.33%	75.43%	14.89%
ED LSTM	7.46%	11.41%	8.84%	2.33%	75.65%	15.54%
LSTM	7.38%	11.99%	8.78%	2.30%	76.06%	14.94%

Table 3.7: Average means and standard deviations using multivariate 15-min resolution data.

Model	MAPE-m	MAPE-sd	NRMSE-m	NRMSE-sd	R^2-m	R^2-sd
Baseline	11.27%	13.58%	13.89%	2.92%	44.64%	24.01%
VARIMA	9.43%	12.32%	11.40%	2.84%	61.22%	20.15%
Reg	8.73%	12.74%	10.05%	2.29%	69.61%	15.25%
QuadReg	8.41%	12.52%	9.67%	2.25%	71.88%	14.38%
CubicReg	8.33%	12.31%	9.59%	2.25%	72.22%	14.52%
SVR	8.24%	12.91%	9.38%	2.34%	73.19%	14.40%
ELM	8.00%	12.45%	9.25%	2.28%	73.69%	15.54%
LSTM	7.62%	12.90%	8.99%	2.36%	74.73%	15.92%
NN	7.52%	13.05%	8.82%	2.34%	75.56%	15.82%
ED LSTM	7.43%	13.24%	8.68%	2.31%	76.24%	15.71%

3.4.5 Discussions

After examining the forecasting models in a total of four experiments, univariate and multivariate forecasting using two different data resolutions, we make the following observations.

Effects of Data Resolution

The 5-minute resolution traffic data can be a little noisy. In experiments using 15-minute resolution data, the best MAPE for 1-step ahead forecast is a little over 5%. On the other hand, if using 5-minute resolution data, the best MAPE values for steps 1-3 are around 6% to 7%. In other words, it is easier to accurately forecast the number of vehicles that have passed a particular location in 15 minutes than to separately forecast three consecutive 5-minute periods. The smoothing effects of down-sampling can help models to maintain better forecasting accuracies for a longer forecasting horizon. Certain models, such as SARIMA, can be very sensitive to noise or sudden fluctuations in the data, especially if the number of parameters is to be kept relatively low. Depending on the applications, applying down-sampling to high-resolution traffic data can be beneficial; but if accurate high-resolution

forecasts are required, then it would be worthwhile to devote further research into effectively utilizing high-resolution data while minimizing the negative influence of noise.

Parameter Tuning and Overfitting

Most machine learning models require parameter tuning. Some researchers may spend months tuning their neural networks' parameters to achieve better results on a particular dataset, but if given a different problem or dataset, then the previously tuned parameters may not be effective. On the other hand, if given datasets of similar patterns, then the parameters tuned for one dataset may suffice for others. In this study, there are large amounts of data from hundreds of locations, so it is certainly not feasible to fine-tune a separate set of model parameters for data from each sensor.

Another interesting observation is that sometimes it can be difficult to tell if the neural networks have overfitted. For example, in Figure 3.10, the 3 multivariate neural networks are the top performers. Many in the deep learning community have also concluded the same for other types of problems. However, only by examining Figure 3.9 we can see that the 2 types of LSTM NNs can perform slightly better using only univariate data and the multivariate LSTM NNs have likely overfitted to the extra noise introduced by additional sensors. In the field of traffic forecasting, such comparisons between complex and deep models taking in large amounts of input data and relatively smaller models do not seem to be sufficiently addressed by the deep learning community. Nevertheless, even with some overfitting, the neural networks are still able to stay in the top tier of forecasters.

There are also other models that require little effort in parameter tuning and are much less likely to overfit. They are usually statistical models such as regression, as well as ELM, a type of neural network. Such models can be quickly and easily applied to a great variety of problems and have decent performances. For the traffic flow forecasting problem in this study, the neural networks that require effort in tuning parameters and can easily fall into the

trap of overfitting, are still able to produce the best accuracies. Though the other models, especially ELM and some regression models, are not that much behind in terms of accuracy but are much faster to train. The choice is once again for the users to make depending on what levels of accuracies are acceptable to them and how much time and effort they are willing to invest to tune the models.

Accuracy and Computational Cost Trade-Off

Another closely related issue is the trade-off between accuracy and computational cost. The various neural networks have the highest level of accuracy but are also the slowest to train. On the other hand, the regression models and ELM are lighting fast and can produce decent results. Of course, the neural networks may leverage GPU computing powers to decrease training time, but such computing resources may not be accessible to everyone. The ultra-low training times of the regression/ELM models may also be more suitable for real-time traffic forecasting, in which models may need to be repeated re-trained with the most up-to-date data. Since the nodes in the Sapelo2 cluster are often highly loaded, the actual timing results are not as reliable since they can fluctuate greatly depending on the load of the nodes. Therefore, only rough estimates of the computational costs of the models are given in Table 3.8, which also provides an overall summary of findings on the various forecasting models in this study.

3.5 Conclusion and Future Work

In this study, we focus on multi-step forecasting of traffic flow using large amounts of sensor data from the San Diego, California area. Various models are extensively evaluated in terms of accuracy, stableness, computational cost, and ease of use in terms of parameter tuning. The incorporation of spatially dependent data generally improves the performance of most

Table 3.8: Summary of Forecasting Models.

Model	Accuracy	Speed	Comments
SARIMA	Low	Fast	Highly sensitive to noise/sudden fluctuations in the data.
SARIMA2	Low	Fast	A little slower than SARIMA, but more accurate.
VARIMA	Low	Medium	MLE optimization can become slow for large numbers of parameters.
Reg	Medium	Very Fast	Fastest model, decent accuracy.
QuadReg	Medium	Very Fast	Improves upon accuracy of linear regression without adding too much computational costs.
RespSurf	High	Fast	Univariate only. The interactive/cross-terms can be effective in boosting accuracies, but too many such terms (e.g., from an increasing number of raw input features) can greatly increase computational costs and lower accuracies.
CubicReg	Medium	Very Fast	About the same level of accuracy as QuadReg but a bit slower.
SVR	Medium	Medium	Often outperforms QuadReg and CubicReg, but slower. Consistently outperformed by ELM, which is also much faster.
ELM	High	Fast	Highest accuracy besides the other neural networks, reasonably fast, and mimimal parameter tuning.
NN	Very High	Slow	High accuracy, much slower than previously mentioned models, parameters tuning can become very tedious, may use GPU to improve training speed if available.
LSTM	Very High	Slower	Most likely to suffer overfitting in multivariate experiments, possibly due to consideration of data that are too far back in time.
ED LSTM	Very High	Slowest	Often the most accurate model, always the slowest.

models when compared to those that rely on univariate temporal data alone. The down-sampling process also has an inherent smoothing effect on the data that help models to produce more accurate forecast in slightly larger intervals and for longer terms. The various types of neural networks, in particular, the Encoder-Decoder LSTM NN, produce the best accuracies overall, though they are also computationally the most expensive. The number of parameters in the neural networks is fairly large and does lead to better performance. The cell state memory in the LSTM unit also seems to contribute to superior accuracy. On the other hand, the Extreme Learning Machine is a fairly fast learner, requires little parameter tuning, and can produce great accuracies, only slightly lower than the other much slower neural networks.

For future work, we plan to conduct more detailed studies on the computational costs of the various models on dedicated machines with GPUs. We also plan to test the performance of a single neural network simultaneously producing forecasts for all steps and compare the performance to our current individual neural networks each responsible for one step. We are also interested in real-time traffic forecasting. Since many sensors located nearly are likely to exhibit similar traffic patterns, transfer learning can be used to effectively reduce the training time of the various neural networks.

Conflict of interest

The authors declare that they have no conflict of interest.

Chapter 4

Situation-Aware Vehicle Traffic Forecasting[1]

[1]Hao Peng, Nicholas Klepp, and John A. Miller. To be submitted to *IEEE Transactions on Big Data*.

Abstract

The dawn of the big data era has led to various breakthroughs in data science. The deep learning revolution, marked by its incredible achievements in image recognition, has drawn many to approach data science in a data-driven manner. In this process, the heavy reliance on the models to learn all knowledge from large quantities of data seems to have diminished the role of existing knowledge about the system and well-established theories. In particular, the vehicle traffic patterns, which are generally regular every week, can deviate from the norm in certain situations such as rain, accident, road work, etc. A purely data-driven approach would simply let the model handle all the different situations. However, when the data scientists are aware of such situations, they may take advantage of their knowledge about the situations and help guide the model building process. This work aims to illustrate how situation awareness can help data scientists to build more effective models in the field of vehicle traffic forecasting. The traffic affecting situations under consideration include holidays, special weather conditions, location awareness that facilitates transfer learning, and the awareness of potentially outdated data that prompt periodical retraining. In addition, we present a novel modeling technique, Quadratic Extreme Learning Machine, that generally improves upon the standard Extreme Learning Machine model while remaining relatively efficient. The Quadratic Extreme Learning Machine can be potentially used as an alternative to Neural Networks, which generally entails higher computational costs.

4.1 Introduction

In the big-data era, along with the rapid increase in volume, velocity, and variety of data, the needs for organizing and understanding the data are increasing. Furthermore, hundreds of modeling techniques that may be used to analyze the data need to be better understood. The recent breakthrough in deep learning has revolutionized fields such as image and speech

recognition. As a result, many data scientists are pursuing research in this direction and attempt to apply to various fields of study. The availability of large amounts of data has made it possible for data scientists to build and train deep learning models that take in great numbers of inputs. The rationale is to allow the models to learn anything and everything from the data, a common theme in a purely data-driven approach. However, in many fields of study, there are existing situations, conditions, knowledge, and theories that can affect the system of interest. Though models may learn much from the data, the learning can potentially be improved or made more efficient if the data scientists can leverage what they know to help guide and constrain the model building process. For example, in the field of vehicle traffic forecasting, a common data-driven approach would be to provide models with large amounts of traffic sensor data, train the models, and produce forecasts [Lv et al., 2015, Wu et al., 2018, Yang et al., 2019]. On the other hand, we know of potential situations that affect traffic patterns, such as holidays, weather conditions, accidents, lane closures, etc. Being aware of such situations should prompt data scientists to make smarter choices when designing their models to handle special conditions. The process should complement the popular data-driven approach that has gained popularity in recent years. However, in the most recent studies of traffic forecasting, many of which focus on deep learning models, traffic patterns affecting situations are not sufficiently addressed.

In this paper, we aim to illustrate how situation awareness can benefit data scientists to build better models. The traffic affecting situations under consideration include holidays, special weather conditions, location awareness that facilitates transfer learning, and the awareness of potentially outdated data that prompt periodical retraining. Data from a total of 1230 traffic sensors in the San Diego Bay Area (District 4) from the Caltrans Performance Measurement System (PeMS)[2] are used in this study. The total size of the

[2] `http://pems.dot.ca.gov/`

data is approximately 9.4 gigabytes. A map of the PeMS sensors considered in this study is illustrated in Figure 4.1.

Most of the models included in this study come from the ScalaTion project, which is an open-source framework under an MIT license for simulation, analytics, and optimization. More details about the project can be found in `http://cobweb.cs.uga.edu/~jam/scalation.html`.

The rest of this paper is organized as follows: Section 4.2 focuses on situation awareness and its role in data science while providing a focus on traffic forecasting. Related work is reviewed in Section 4.3. Section 4.4 provides an overview of the support provided by the ScalaTion big data framework for data science and forecasting. Illustrative examples of their use for vehicle traffic forecasting will be given in Section 4.5. More detailed results from ScalaTion models for traffic forecasting studies may be found in [Peng et al., 2018, Peng and Miller, 2019], while related traffic simulation studies may be found in [Bowman and Miller, 2016]. Finally, conclusions and future work are given in Section 4.6.

4.2 Situation Awareness in Data Science

The situations that affect a system of interest are generally specific to a particular field of study. They may be considered as simple forms of knowledge, which can be more formally organized in a knowledge base involving ontologies, Markov Logic Networks (MLN) [Richardson and Domingos, 2006], etc. As a starting point, the situations involve conditions and events that may prompt the data scientists to take helpful actions in the model building process. The next section briefly discusses related approaches to data science that could complement the data-driven approach while the subsequent sections discuss in detail the specific situations that affect typical traffic patterns and the appropriate actions data scientists should take.

Figure 4.1: Location of PeMS District 4 Sensors. Generated by gpsvisualizer.com.

4.2.1 Related Approaches to Data Science

Some situations could be more formally organized using semantic technologies (e.g., Markov Logic Networks) or represented using well-established scientific theories (e.g., laws of physics governing speed and motion of vehicles). Benefits include automated reasoning, more formal interpretations of the models and the results, etc. More concrete applications demonstrating the benefits of these approaches to data science are planned for future work.

The field of Semantic Data Science is beginning to emerge in the mid-2010s. For example, in 2014, the Department of Defense started an initiative on the topic `https://semanticommunity.info/Data_Science/Semantic_Data_Science_for_DoD`. In the paper, "Simulation and the Semantic Web," [Miller and Baramidze, 2005], the synergy between Semantic Web technology and simulation is explored. Due to the recent big data revolution, there are also growing interests in applying Semantic technology to data science/analytics.

Theory-Guided Data Science is a term coined by Faghmous and Kumar in 2014 [Faghmous and Kumar, 2014] and the concept is further expanded in later work in 2017 [Karpatne et al., 2017]. Many predictive analytics models can achieve high accuracy/low error rates, even if they provide no ability for interpretation or understanding. Worse, they may suggest mechanisms that are physically not realizable. Theory-Guided Data Science was defined by [Karpatne et al., 2017] to mean that portion of data science "where scientific theories are systematically integrated with data science models in the process of knowledge discovery." As discussed in [Karpatne et al., 2017] there have been some success stories in using Theory-Guided Data Science techniques. In [Miller et al., 2017], the support for theories in open science is discussed with a case study in the field of computational economics and finance.

4.2.2 Special Traffic Conditions

Typically traffic patterns on weekdays are very regular from week to week as most people would need to commute to work on weekdays. However, there are special scenarios that would cause traffic patterns to very much deviate from the norm. A holiday that falls on a weekday would have a very different pattern. During holidays, most people would not be commuting to work. They may choose to rest at home, travel to recreational activities, or go out of town to visit families. A special weather condition, such as rain, snow, and fog can also cause traffic patterns to deviate from the norm since people would either avoid driving or drive much slower than usual. An accident near the side of a road typically leads people to change to a neighboring lane and may slow down the overall traffic flow. Road work and lane closures may force drivers to take detours on other routes that they may not typically travel. The awareness of such situations can and should be used to help data scientists to build and train more effective models.

4.2.3 Transfer Learning

Being aware of the locations of the sensors and a correlation measure between data generated by two sensors can be used to facilitate transfer learning. Transfer learning is the idea that elements of one modeling study may be used in another related modeling study [Pan and Yang, 2009]. To do transfer learning with neural networks, two choices need to be made. The first choice is the which layers, or their associated weights and biases, need to be transfered. The second choice is which of the transferred layers are to be kept frozen, meaning no changes to the weights and biases. Transfer learning may result in improved accuracy of the model as well as reduced execution time [Pan and Yang, 2009, Goodfellow et al., 2016]. Reduction in execution time is important for the training of forecasting models that utilize Neural Networks because the training time can be very large. Improving this will potentially allow

for more frequent re-training. Preliminary testing suggests that for our traffic data, the best results can be produced when all the layers are transferred and none of them are kept frozen.

4.2.4 Re-training with More Recent Data

Traffic patterns, though typically possess a degree of regularity, may also gradually evolve as time passes. For example, a newly constructed highway may direct drivers to take the new route to work. As mentioned earlier, short term traffic pattern changes may also occur due to road work. The new construction/expansion of a major company/factory that employs many people will certainly cause increasing traffic flow towards the new workplace. As a result, the collected traffic data may become outdated due to a combination of obvious and unobserved factors. Being aware of this situation about traffic patterns, it would be helpful to adjust or re-train the models using the most up-to-date data This is rarely done in the recent deep learning studies on traffic forecasting because re-training a deep neural network can be very time-consuming. As an alternative, faster models can be used, including our proposed Quadratic Extreme Learning Machine model that is both efficient and possess a relatively high level of accuracy. The proposed model is discussed in more detail in Section 4.4.

4.3 Related Work

Most of the very recent work in the field of traffic forecasting focuses primarily on deep learning models using data-driven approaches. Our work aims to complement the existing literature by approaching the problem of traffic forecasting while remaining situation-aware. There have been some attempts to be more intelligent when building models while being aware of various situations that could affect traffic patterns.

There have been several studies attempting to handle special weather situations. In [Koesdwiady et al., 2016], weather information such as rain, temperature, humidity, etc. are given to a deep belief network. The additional weather data did yield better performance than using just traffic data alone to make forecasts. A total of 47 PeMS sensors from the San Francisco Bay Area are chosen along with 16 National Weather Service network stations. Other considered models include ARIMA and a standard feedforward Neural Network. Another study in [Jia et al., 2017], rainfall data are incorporated into models such as LSTM, ARIMA, Neural Networks, and Deep Belief Neural Networks. The best performing model was LSTM, and the additional rainfall information improved the MAPE by almost 1%. In [Zhang and Kabuka, 2018], a small-scale study was conducted using one PeMS sensor in the Santa Clara county. Precipitation, speed, and temperature data were given to a Gated Recurrent Unit, which performed better than a Gated Recurrent Unit without the weather data. It appears that typically in these studies, the weather data are simply given to the neural networks as inputs. As we demonstrate in our applications, such an approach may not be the best or most efficient way to handle such situations. For example, a feature containing rainfall data would contain mostly zeros. The imbalance in the feature may not be able to facilitate the most efficient learning from the data.

Not many researchers seem to have devoted much effort to studying holiday traffics, possibly due to the irregular traffic patterns and limited amounts of holiday traffic data. In [Chrobok et al., 2004], the holiday data were differentiated to predict traffic flow during special events such as a soccer game. Another work in [Cools et al., 2009] focused on holiday traffic prediction. Binary encoding was used to represent holidays and other days of the week. The study also focused on various types of roads and the authors concluded that the holiday effects are significant on commuter roads but not as much on leisure roads, which seem very reasonable and logical. A much more recent study in [Gao et al., 2019] focused

on predicting traffic speed on holidays. The work primarily focused on using LSTM to make predictions but did not consider other types of models.

A study in [Wang et al., 2016a] applied a transfer learning based technique to speed up the training process of deep neural networks, though the time range of the datasets was relatively small, spanning only a couple of weeks. Speed forecasts were made up to 50 minutes ahead. The transfer learning approach was to cluster various road segments based on speed and train models using data from the clusters. The goal was to make sure there were sufficient training instances by pulling all data from similar road segments. A recent study in [Deng et al., 2017] proposed an automated mechanism for both identifying various traffic situations (e.g., rush hour, accident, etc.) using clustering algorithms and pull the data from multiple sensors in the same situation to train models.

4.4 Support in the ScalaTion Big Data Framework

This section highlights ScalaTion's support for time-series analysis and forecasting. Our proposed Quadratic Extreme Learning Machine model is also presented.

4.4.1 Support for Time-Series Databases

ScalaTion provides support for Time-Series Databases [Bobade, 2018]. This makes it easy to collect multiple time-series datasets along with supporting related data. Support is also provided for pre-processing the data including time-based merging, outlier removal, and missing value imputation. Query languages are provided to make selections of data relevant to a modeling study easy. The ScalaTion's time-series databases is general-purpose working for non-time-series data as well. It is also among the fastest such databases being built as a main-memory, columnar database (see [Bobade, 2018] for a performance comparison).

Table 4.1: List of Modeling Techniques.

Technique	Description	Strength
AR	Autoregressive	Model Simplicity.
ARMA	AR, MA	Allows shocks to be introduced to the system.
ARIMA	AR, Integrated, MA	Support non-stationary time series.
SARIMA	Seasonal ARIMA	Can model seasonal time series.
SARIMAX	SARIMA with Exogenous	Allows exogenous variables.
VARIMA	Vector ARIMA	Can model multiple concurrent time series. Seasonality allowed.
Regression	Multiple Regression	Global minimization of MSE based on the given features.
PolyReg	Polynomial Regression	Allows nonlinearity into the system.
SVR	Support Vector Regression	Flexibility in the terms of the parameters and model equations.
ELM	Extreme Learning Machines	Efficient and accurate.
QuadELM	Quadratic ELM	Allows nonlinearity in the output layer. Typically more accurate than ELM.
NN	Neural Networks	Highly flexible in structure and great accuracy.
LSTM	Long Short-Term Memory	Specializes in modeling temporal dependencies in time series.

The database supports seamless integration with the over one hundred modeling techniques provided by ScalaTion .

4.4.2 Support for Forecasting

ScalaTion supports a large variety of modeling techniques for forecasting including the commonly used techniques as shown in Table 4.1. These forecasting techniques have been applied to traffic forecasting in the San Diego area using CalTrans Performance Measurement System (PeMS) [Peng and Miller, 2019]. Multi-hour ahead traffic forecast with low MAPE values are reported in [Peng et al., 2018, Peng and Miller, 2019]. Interesting trade-offs

emerge as the second tier modeling technique ELM executes much more quickly than the top tier consisting of 3 neural network models. Seeing the potential of ELM, we proceed to extend upon that model and formally present the Quadratic Extreme Learning Machine (QuadELM) model.

4.4.3 Quadratic Extreme Learning Machine

The Quadratic Extreme Learning Machine (QuadELM) model is considered to be a type of neural network that may be described as follows:

1. Input-to-Hidden: apply an activation function with fixed, random weights. This introduces non-linearity into the system. The size of the hidden layer typically needs to be tuned so that the hidden layer can sufficiently represent the transformed inputs.

2. Hidden-to-Output: use the quadratic regression model in which the hidden states are used as inputs. The parameters can be efficiently solved using well-established pseudo-inverse techniques.

Our proposed Quadratic Extreme Learning Machine model replaces the linear regression model in the output layer of the original ELM model [Huang et al., 2006] with the quadratic regression model, thus introducing another layer of nonlinearity in the output layer. In the traffic forecasting application presented in Section 4.5, the Quadratic Extreme Learning Machine model can further improve upon the accuracy of the standard ELM without adding too much to the computational costs.

4.5 Vehicle Traffic Forecasting

Traffic patterns on weekdays are typically very similar from week to week. In recent years, many traffic forecasting related studies focused on deep neural networks [Lv et al., 2015,

Shao and Soong, 2016, Wu et al., 2018, Yang et al., 2019]. A typical trend in the deep learning community is to learn all knowledge by using deep neural networks. However, various situations can cause traffic patterns to deviate from the norm and they would require additional considerations. Currently, there seem to be relatively little efforts devoted to allowing the awareness of traffic affecting situations into the modeling process as many data scientists elect to task the neural networks to do all the learning. A purely data-driven approach to traffic forecasting has several weaknesses that do not seem to be sufficiently addressed in the literature:

1. Slow training times of deep neural networks, which make re-training on most recently available data infeasible (e.g., in the context of real-time forecasting).

2. A single model may be able to handle the typical traffic patterns very well, but can be lacking when encountering special and rare traffic conditions (e.g., due to a special weather condition). In fact, many recent deep learning traffic forecasting studies do not consider such special cases at all.

3. Simultaneous consideration of data from large numbers of traffic sensors, which may exhibit distinct traffic patterns depending on the locations, may not be able to best capture the distinct traffic conditions of certain sensors. For example, a traffic sensor located near a popular interchange may exhibit special traffic patterns that are different from the rest of the highway; but if all the sensor data on the highway are trained together, then the traffic data from the sensor near the popular interchange may not be well represented in the training batches.

Instead of relying on purely data-driven approaches, better traffic forecasts may be produced if the data scientist can remain aware of traffic affecting situations while building the models. Various situations can arise, including the following:

- Weather

- Accidents
- Lane closures
- Holidays
- Sporting Events
- Locations of Sensors

As an example of remaining situation-aware during the modeling process, awareness of the locations of various traffic sensors and correlation among them may help us deduce that closely located traffic sensors are likely to share similar traffic patterns, for which transfer learning may be used for neural networks to significantly reduce training times. Depending on the specific situation, an effective traffic forecasting system should be able to

- Select models
- Select appropriate features
- Add relevant features
- Use selected data for training

Figure 4.2 provides the vision for such a traffic forecasting system. This study primarily focuses on leveraging various traffic affecting situations to help guide the model building and training process while tasks such as the construction of a knowledge base and apply automated reasoning to determine the appropriate actions in a given situation are planned for future work. Several case studies that captures some of the previously mentioned ideas are presented in this section. Traffic data come from the PeMS system. This study focuses on District 4, which is the San Francisco/Bay Area of California. Forecasts are produced on 1230 Mainline (ML) traffic sensors. A total of 9.4 gigabytes of traffic data are collected

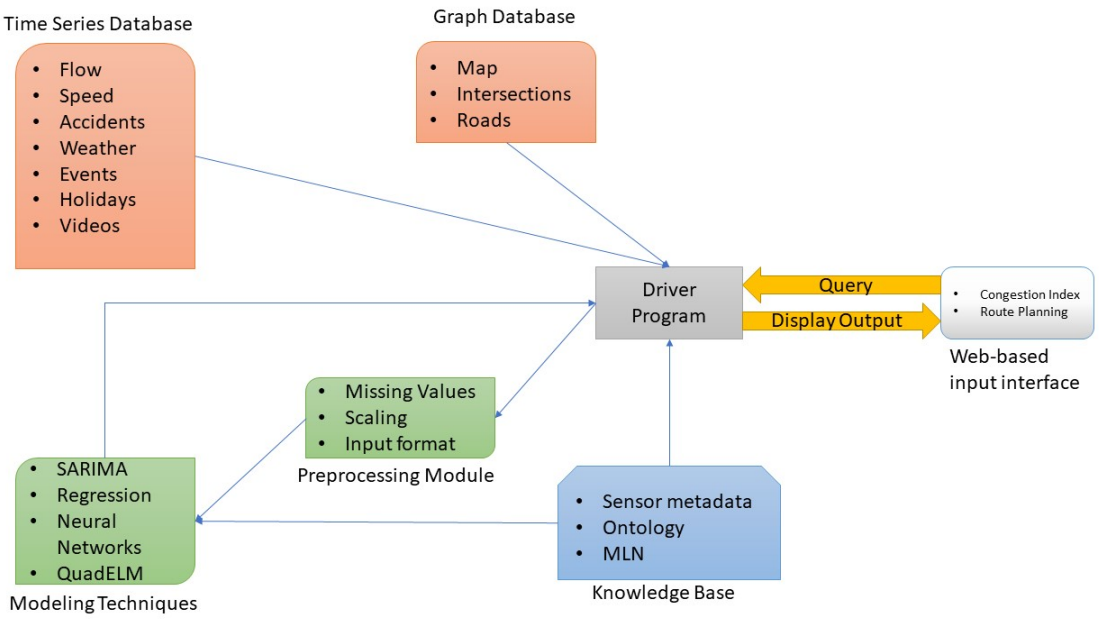

Figure 4.2: Vision of the Traffic Forecasting System.

from these sensors. Raw data are in 5-minute resolution. In this study, the data are further aggregated into 15-minute resolution. The forecasting performance is evaluated using 3 metrics: Mean Absolute Percentage Error (MAPE), Normalized Root Mean Squared Error (NRMSE), which divides RMSE by the mean of the time series, and the R^2 metric. Two baselines are considered: weekly historical average from the past four weeks (BaselineHA) and random walk (BaselineRW), which uses the current observation as forecasts for the future. Experiments are conducted on the Sapelo2 cluster from Georgia Advanced Computing Resource Center[3].

[3] `https://gacrc.uga.edu/`

4.5.1 Traffic Forecasting Study on Holidays

Holidays can introduce special traffic patterns since most people will not be commuting to work but may have other special travel plans, such as a short vacation or visiting families. A purely-data driven approach may elect to include a column representing holidays in the training inputs. To illustrate the effects of forecasting performance, various efficient models are used. The training data include the entire year of 2017, excluding weekends, and the testing data are from 2018, again excluding weekends and only daytime 7:00 AM to 7:00 PM traffic data are evaluated. Figure 4.3 illustrates the performance comparison of various efficient models for the entire year of 2018 using the MAPE metric. Our Quadratic Extreme Learning Machine (ELM) model is the overall top performer. The SARIMA model is the SARIMA $(1, 0, 1) \times (0, 1, 1)$ model using weekly seasonality, a common model found in the literature [Shekhar and Williams, 2008, Lippi et al., 2013, Peng et al., 2018]. The inputs to the other models include the most recent 12 traffic flow observations, 12 flow observations from the previous week, the most recent 4 traffic speed observations, day of the week, time of the day, and whether the day is a holiday (0 if not a holiday, normalized day of the year if it is a holiday). The ELM and QuadELM models are tuned to use the *tanh* activation function in the hidden layer, which is of size 8 and 6 times the size of the input layer, respectively.

Figure 4.4 and Figure 4.5 represent the performance evaluation using the NRMSE and R^2 metric. The relative performance of the models mostly agrees with those using the MAPE metric. One notable difference is the ELM model, which performs better when considering the NRMSE and R^2 metrics.

The interesting question is that since the models do take consideration of the holiday information, could this be sufficient for producing reliable forecasts during holidays? The same trained models can be evaluated only by their performance during holidays. Figure 4.6 illustrates the performance of the models evaluated on the holidays of 2018 only using the MAPE metric. The historical average baseline is not performing too well as expected.

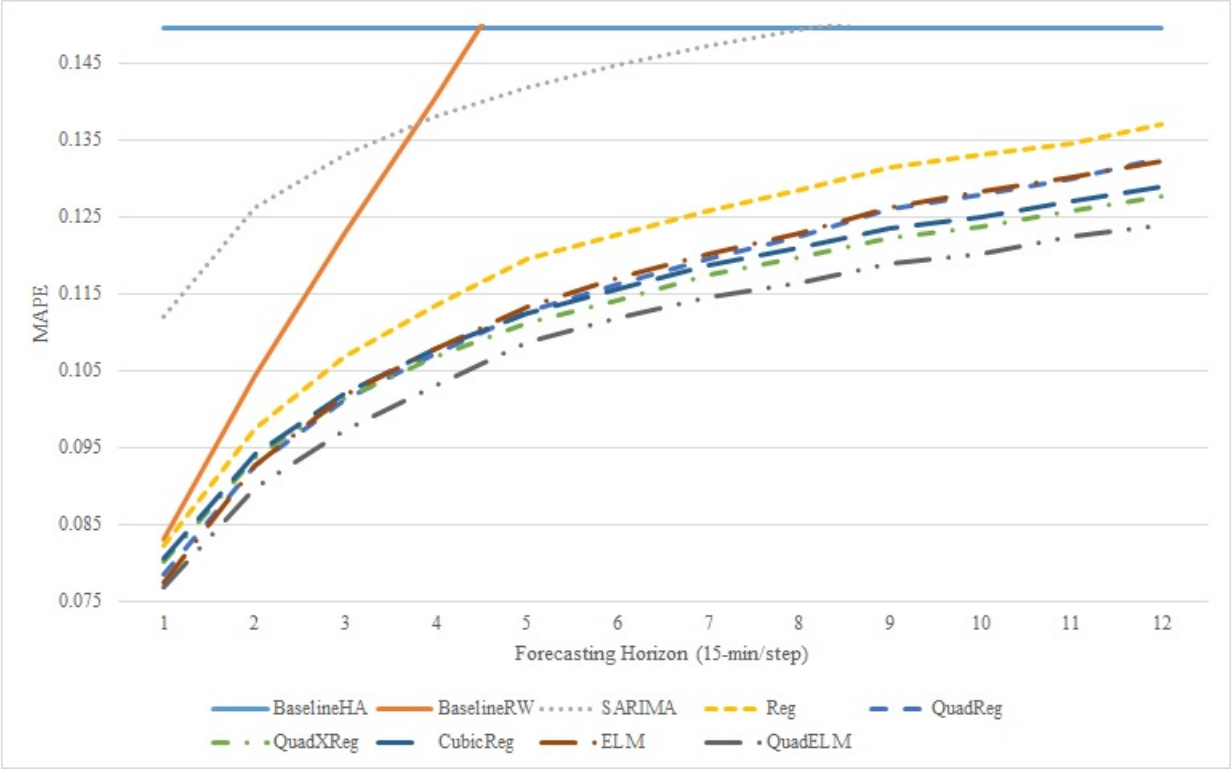

Figure 4.3: MAPE of Forecasts from 2018 in District 4.

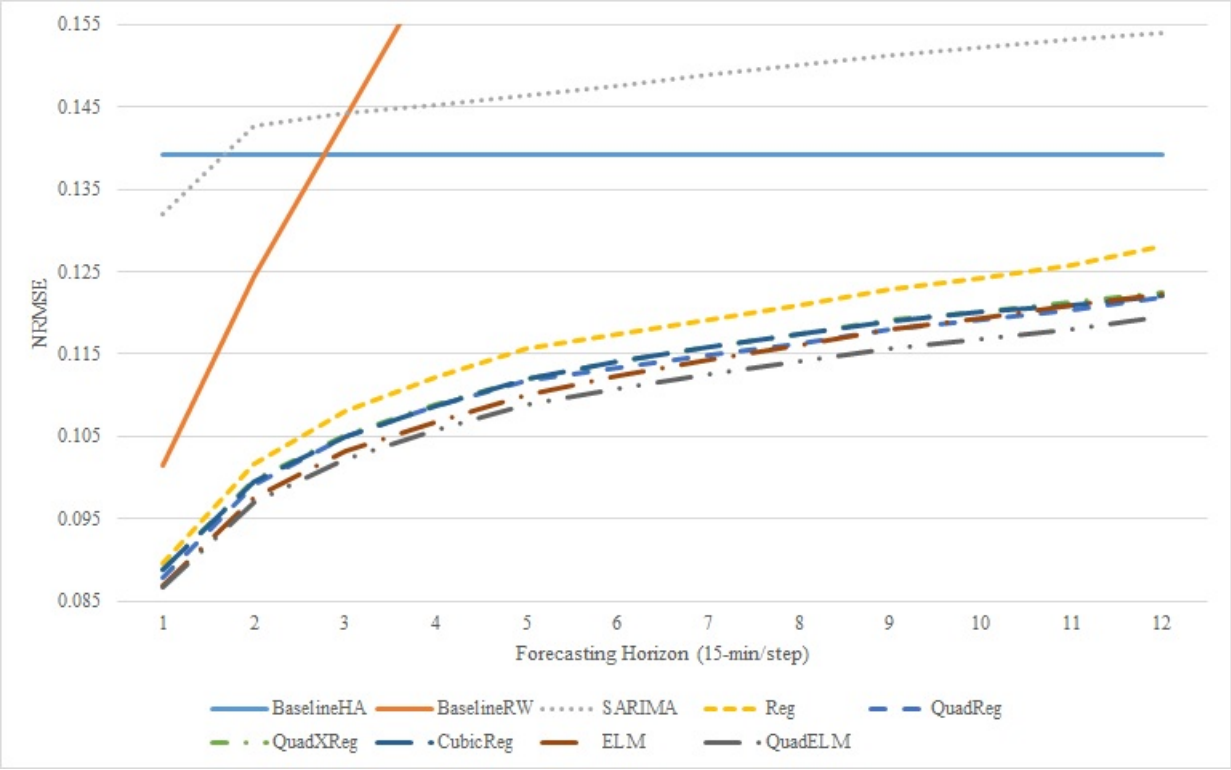

Figure 4.4: NRMSE of Forecasts from 2018 in District 4.

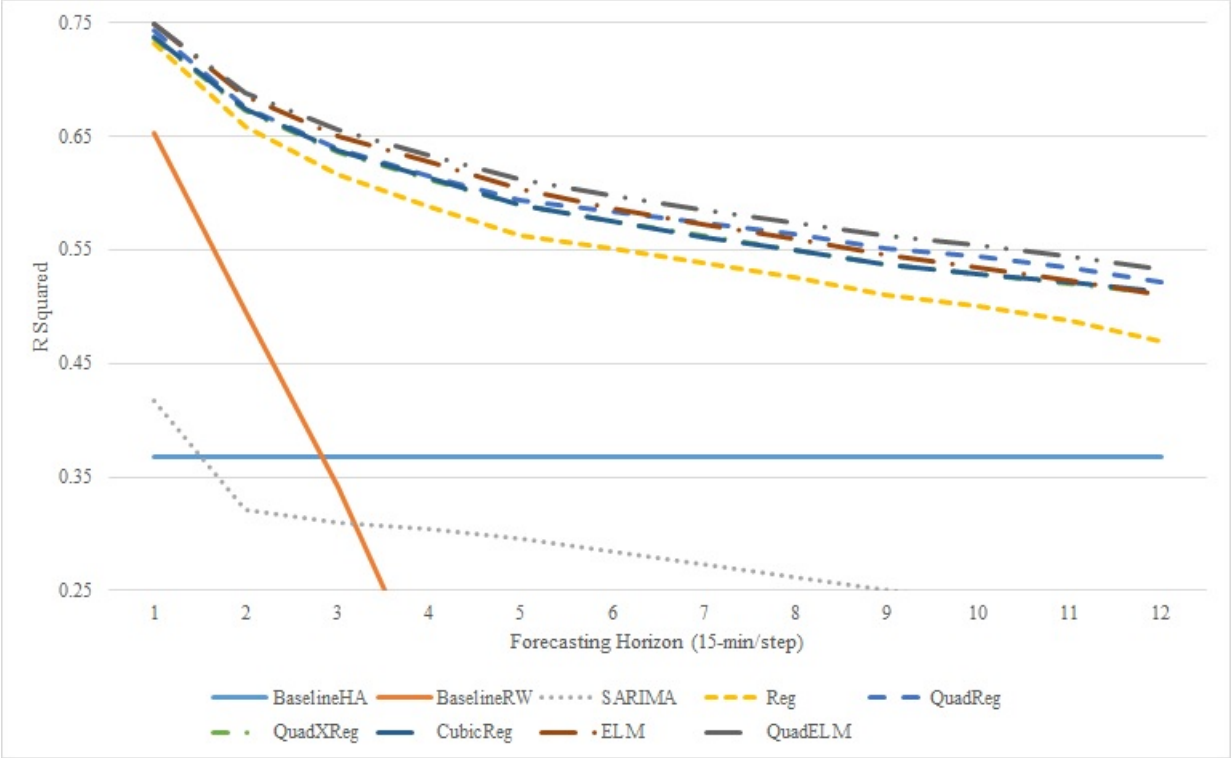

Figure 4.5: R^2 of Forecasts from 2018 in District 4.

The MAPE values of other models are also generally much worse than their overall 2018 performance as in Figure 4.3. The random walk baseline, surprisingly, performs relatively well. This could be indicative that the level of traffic pattern fluctuations is not as great as on a regular weekday.

Figure 4.7 and Figure 4.8 represent the performance of the models on holidays of 2018 using the NRMSE metric and the R^2 metric, respectively. The models typically perform slightly worse than the random walk baseline in the early steps but do catch up for longer steps.

It appears that even though the holiday information is available to the models, they may not be able to produce reliable forecasts. This is likely because holidays are generally rare

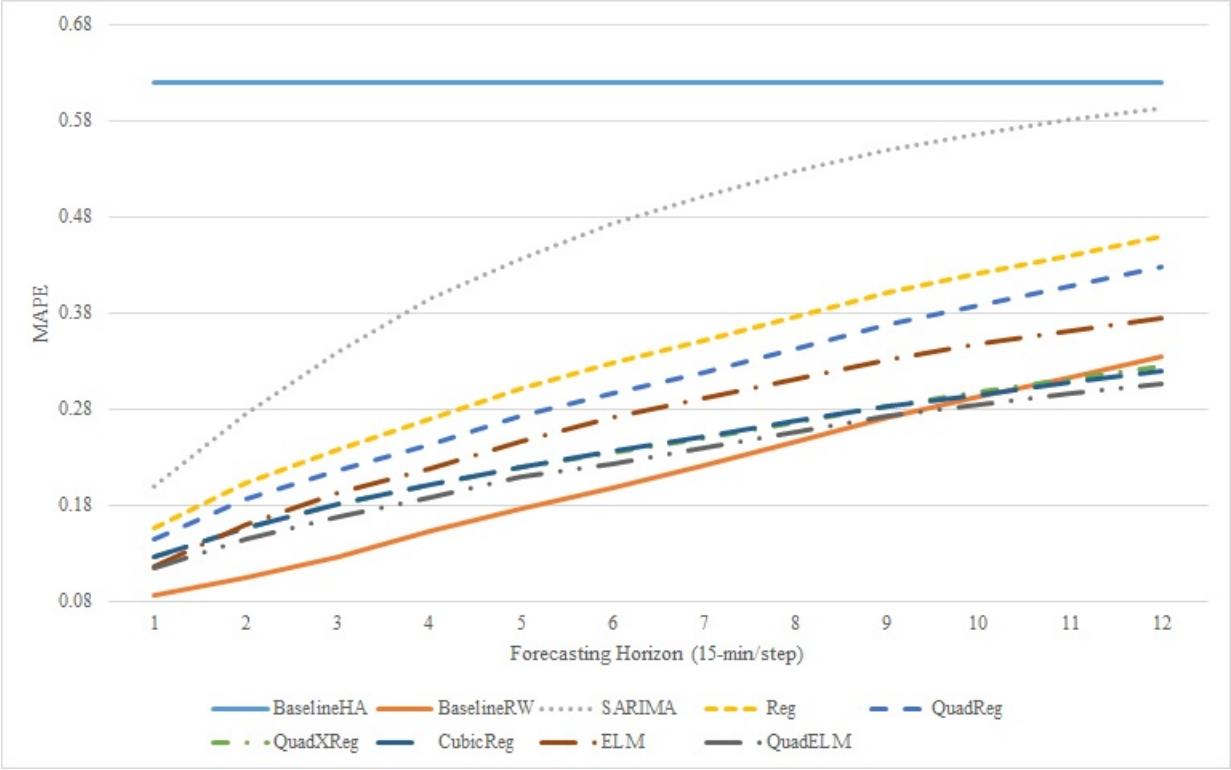

Figure 4.6: MAPE of Models from Holidays of 2018 in District 4.

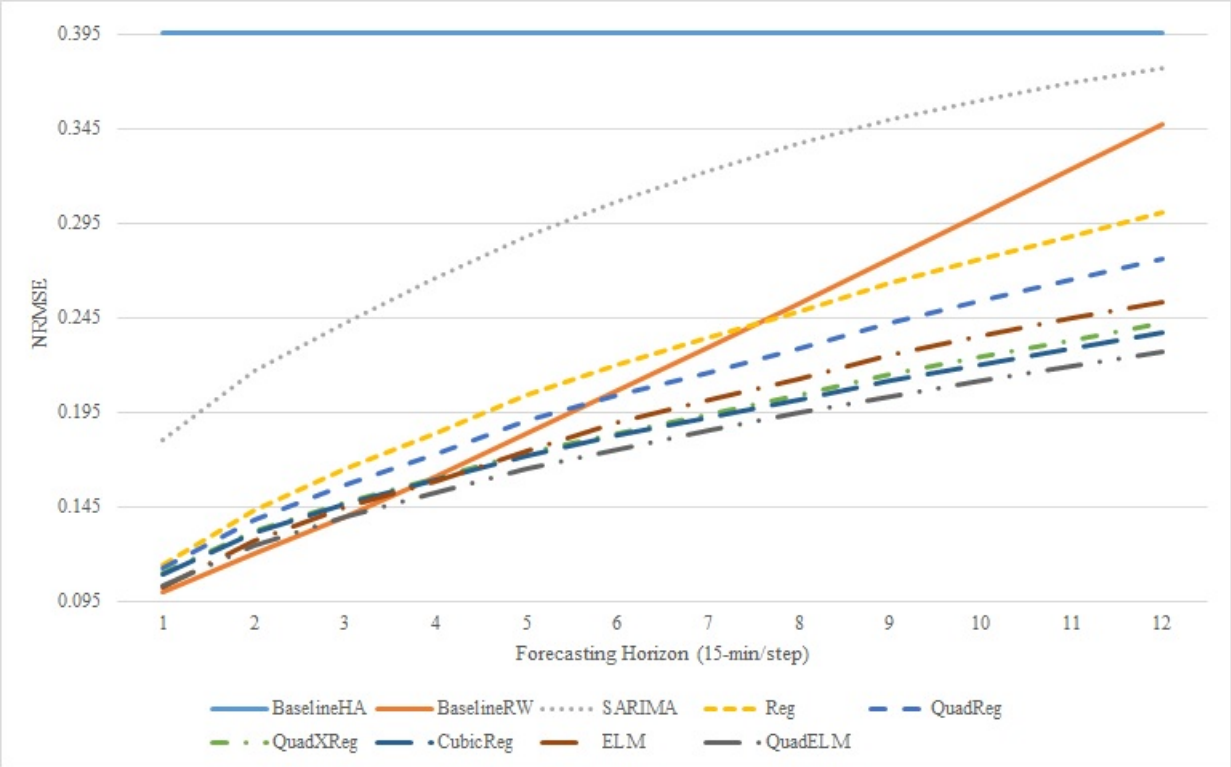

Figure 4.7: NRMSE of Models from Holidays of 2018 in District 4.

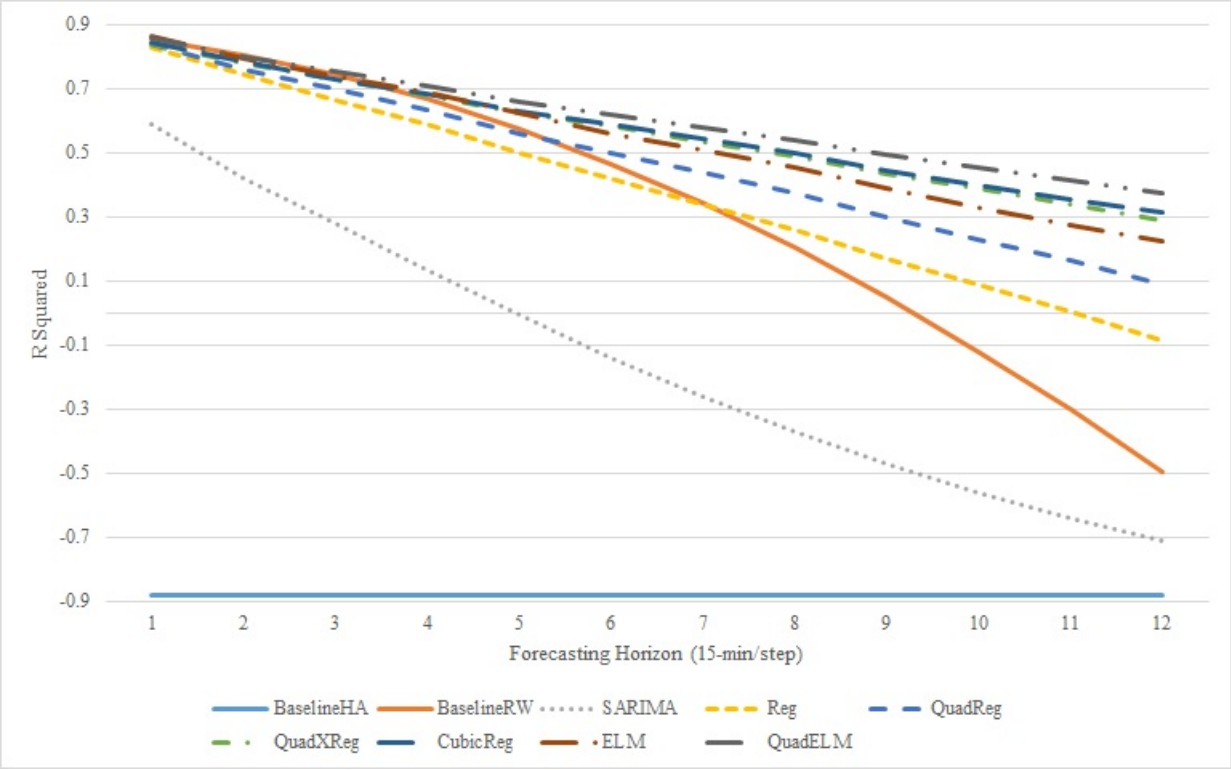

Figure 4.8: R^2 of Models from Holidays of 2018 in District 4.

occurrences in a year, and the highly unbalanced holiday input data consisting of mostly 0's may not be able to guide the model to sufficiently learn the special traffic patterns.

To create a more intelligent traffic forecasting system due to our awareness of holidays, some adjustments can be made to the modeling process:

- Remove the seasonal components of the inputs, since the weekly seasonality most likely will not apply to a holiday.
- Train separate models by using holiday data only.

The performance evaluations of the revised models are illustrated in Figure 4.9, Figure 4.10, and Figure 4.11, based on the 3 evaluation metrics. There is a very small number of training datasets, 21 in particular, for which the Cubic Regression model was having difficulty learning the parameters when training only on holiday data. It is likely due to the small number of training instances while trying to maintain a large number of parameters. These datasets are therefore removed from considerations. To maintain fair comparisons, they were also removed from considerations in earlier figures: Figure 4.6, Figure 4.7, and Figure 4.8.

The historical average baseline and the ARIMA model are omitted from the figures due to poor performance. The size of the hidden layer of ELM has been reduced from 8 times the size of the input layer to 3 times the size. Similarly, the size of the hidden layer of QuadELM has been reduced from 6 times the size of the input layer to the same size as the input layer. The reductions are necessary due to the reduction of both the number of training instances and the number of features available in the model. From the performance results, it seems that quadratic regression, in two out of the three metrics, is the best performing model for forecasting traffic during holidays. Linear regression and ELM are the next best in performance, though the linear regression model does appear to be the top-performing model if judging by the MAPE metric alone. More complex models, in terms of parameters, such as QuadXRegression, CubicRegression, and QuadELM, did not do too well in this scenario.

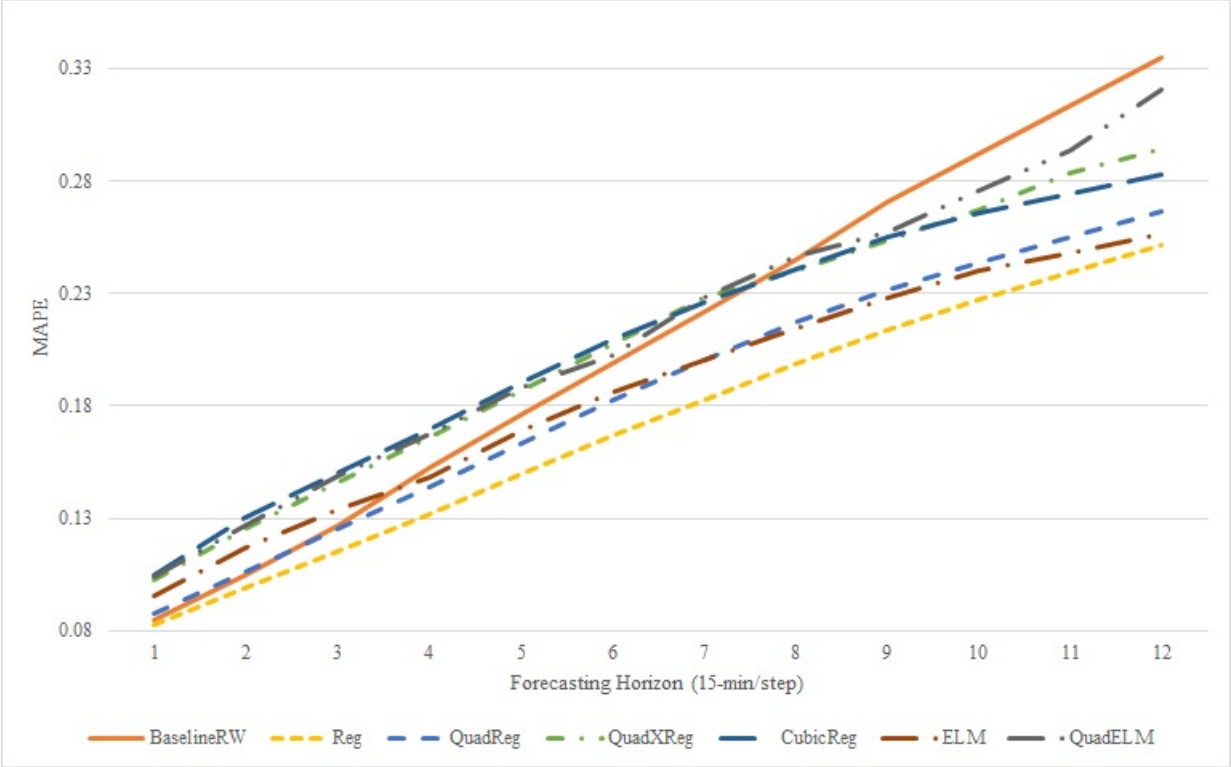

Figure 4.9: MAPE of Models Trained with Holiday Data Only.

They could require more training data to obtain better fits. It is interesting to see that the simpler models tend to do better in this experiment, likely due to the reduced number of features and training instances to work with.

To visualize the improvement from our situation-aware models over the standard models, a simple metric such as the percentage of improvement can be used. The percentage of improvement metric can be calculated, using MAPE as an example, as follows:

$$\frac{MAPE_{std} - MAPE_{sa}}{MAPE_{std}} \times 100\% \tag{4.1}$$

where *std* represent the standard models and *sw* represent situation-aware models. The percentage of improvement of the NRMSE metric is calculated similarly while that of the

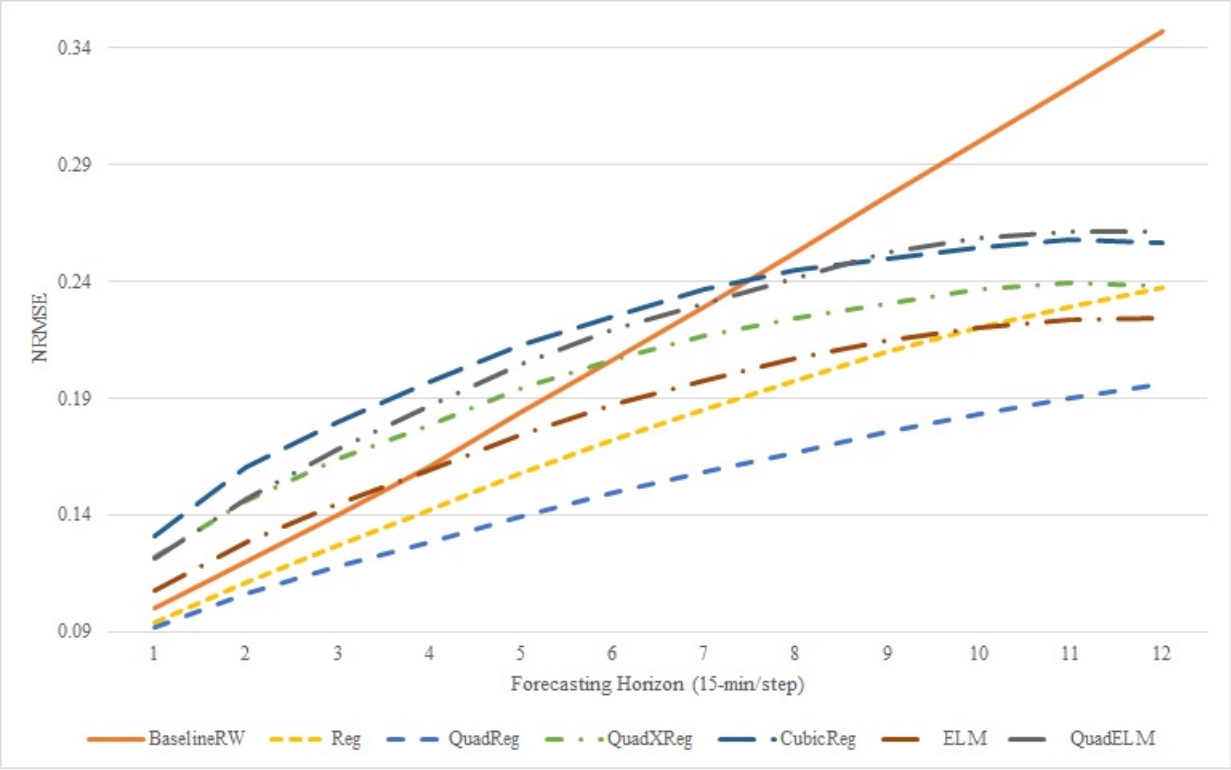

Figure 4.10: NRMSE of Models Trained with Holiday Data Only.

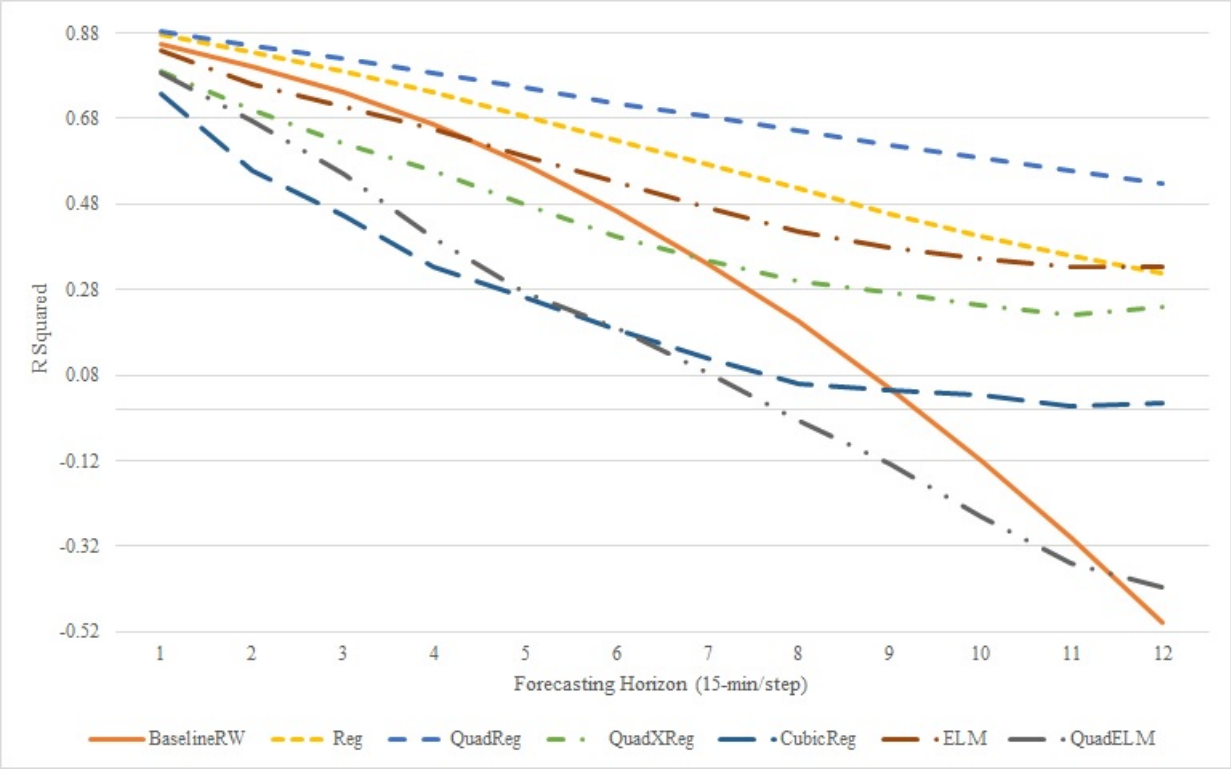

Figure 4.11: R^2 of Models Trained with Holiday Data Only.

R^2 metric has an extra negative sign prepended to the equation so that a positive value will indicate improvements across all 3 evaluation metrics. However, in some cases for the R^2 metric, the denominator can potentially become close to 0 or negative, leading to unreliable resulting percentages. To resolve this issue in certain cases, the following simple difference formulas can be used.

$$(R^2_{sa} - R^2_{std}) \times 100\% \tag{4.2}$$

In cases where the simple difference of R^2 is used, it will be explicitly stated. Otherwise, it will be assumed to be a percentage of improvement.

Figure 4.12, Figure 4.13, and Figure 4.14 represented the improvements of the situation-aware models over the standard models, which were trained with the feature column represented holidays. Figure 4.14 illustrates the simple difference improvements.

4.5.2 Transfer Learning

In general, neural networks can take time to train, especially for complex network structures trained with large amounts of data. In traffic forecasting, we know that traffic conditions from closely located sensors are likely to be very similar. Instead of training neural networks individually per sensor, or simply give all the sensor data to a large neural network to train, it may be better to take advantage of transfer learning. To determine the similarity of traffic patterns between 2 sensors, the correlation metric is used. The awareness of the locations of the sensors and the correlation between two sensors should be helpful in this regard.

For the number of epochs, preliminary testing suggests that at around 175 epochs the learning of Neural Networks and Convolutional Neural Networks have mostly plateaued. Therefore, the number of training epochs of the initial sensor in the transfer learning group for Neural Networks and Convolutional Neural Networks is set to be 200. The Encoder-Decoder Gated Recurrent Unit is slightly different. The value of the loss function, MSE in

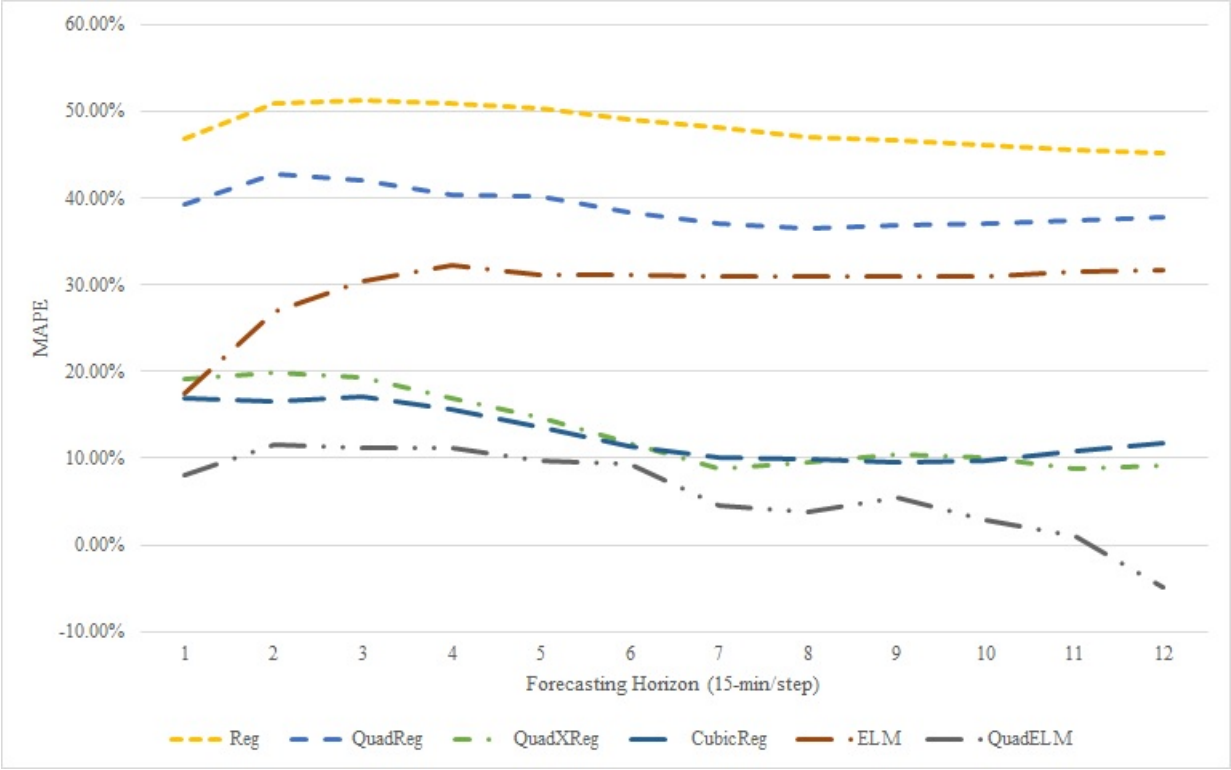

Figure 4.12: MAPE Percentage of Improvement on Holidays by Using Situation-Aware Models.

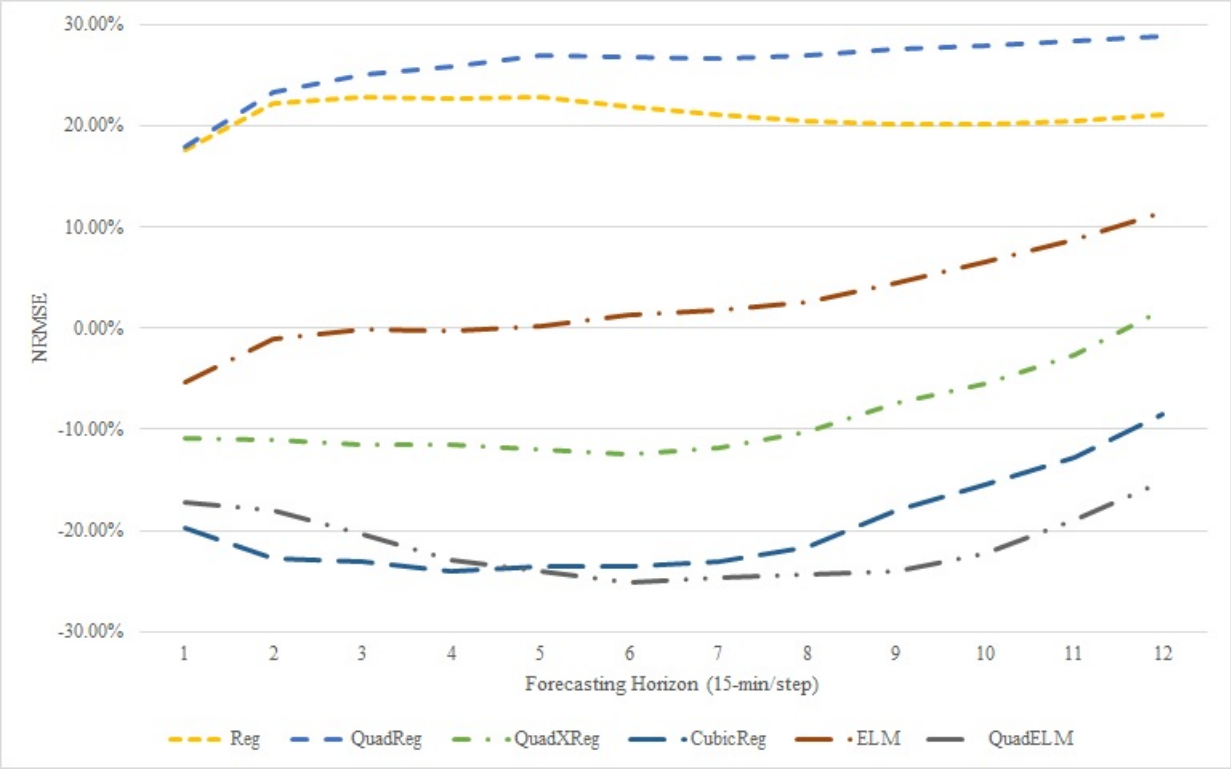

Figure 4.13: NRMSE Percentage of Improvement on Holidays by Using Situation-Aware Models.

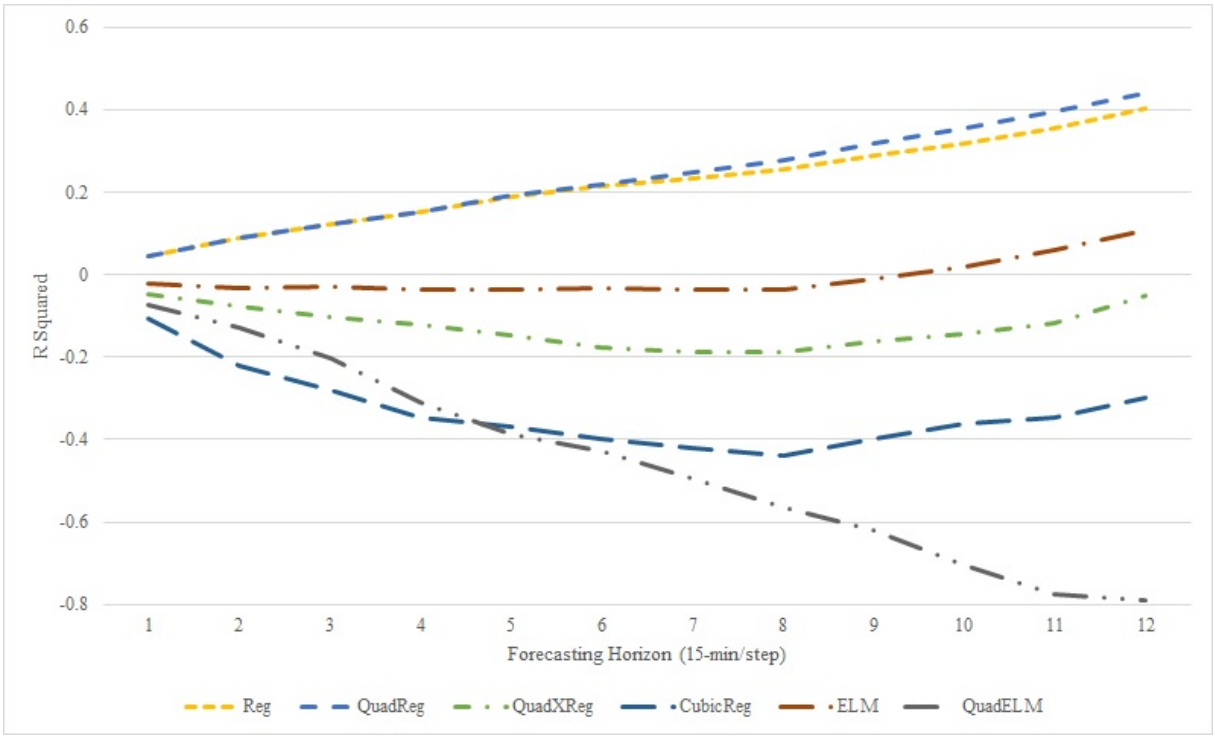

Figure 4.14: R^2 Simple Difference Improvement on Holidays by Using Situation-Aware Models.

this case, is continually decreasing, but during initial testings, going over around 50 epochs actually yield worse results on the testing sets even when the MSE of both the training and validation sets are still decreasing. This has come to us as an expected phenomenon. As a result, the number of epochs for Encoder-Decoder Gated Recurrent Unit is set to 50. Furthermore, a patience value of 30 is used to allow early stopping based on the MSE of the validation set. With transfer learning, Neural Networks and Convolutional Neural Networks are only trained for 40 epochs and Encoder-Decoder Gated Recurrent Unit is only trained 20 epochs. By applying transfer learning, the potential savings in training times per sensor can be 80% for Neural Networks and Convolutional Neural Networks and 60% for Encoder-Decoder Gated Recurrent Unit. The various neural network models tested are provided by Keras [Chollet et al., 2015] using the TensorFlow [Abadi et al., 2015] backend.

However, before doing transfer learning, one may still make the argument that traffic patterns are generally similar in a geographical location and the awareness of the locations and correlations may not be necessary. Furthermore, perhaps transfer learning may be detrimental since the inherited weights can be stuck in a local minimum. To address this concern, a sample sensor is chosen to received learned weights from a closely located and highly correlated sensor. Another random sensor in District 4 is also chosen to initialize the weights. Furthermore, a neural network model is also trained without transfer learning, meaning the weights are initialized randomly. Figure 4.15, Figure 4.16, and Figure 4.17 illustrate the results of transfer learning from a nearby correlated sensors vs. a random sensor and the effects of not using transfer learning.

It can be seen from the figures that though the nearby located and correlated sensor demonstrate better performance for transfer learning, the random sensor also did reasonably well. It is likely that because both sensors are located on major interstates in the same geographical region, there are sufficient traffic pattern similarities. Though the results of the transfer learning models would certainly be more interpretable if the source of the weights

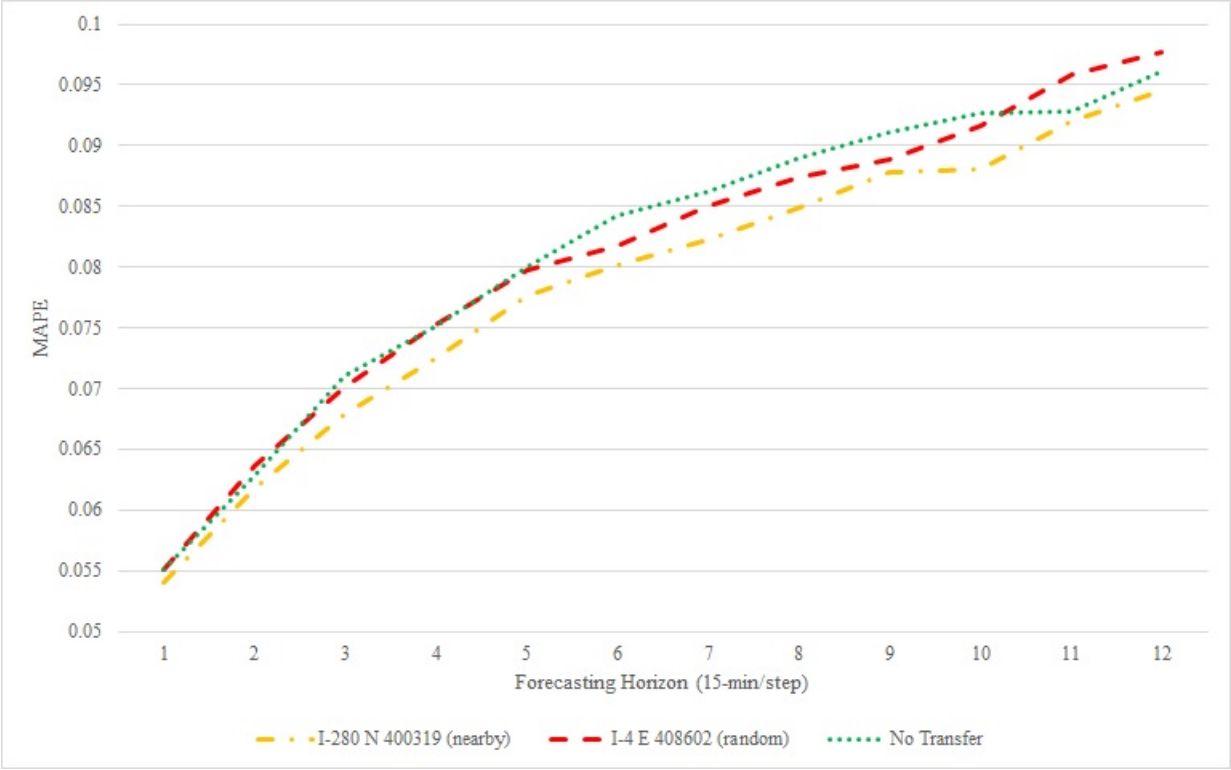

Figure 4.15: MAPE of Sensor 403318 on I-280 N using Transfer Learning.

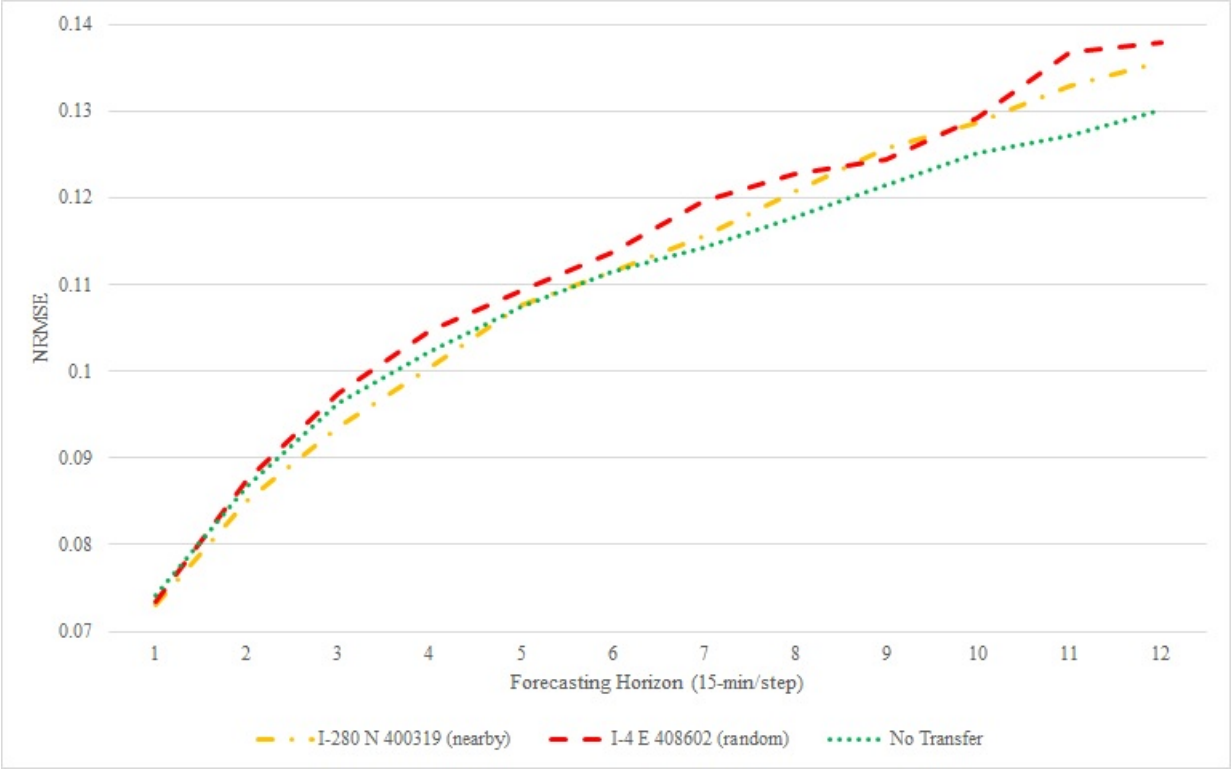

Figure 4.16: NRMSE of Sensor 403318 on I-280 N using Transfer Learning.

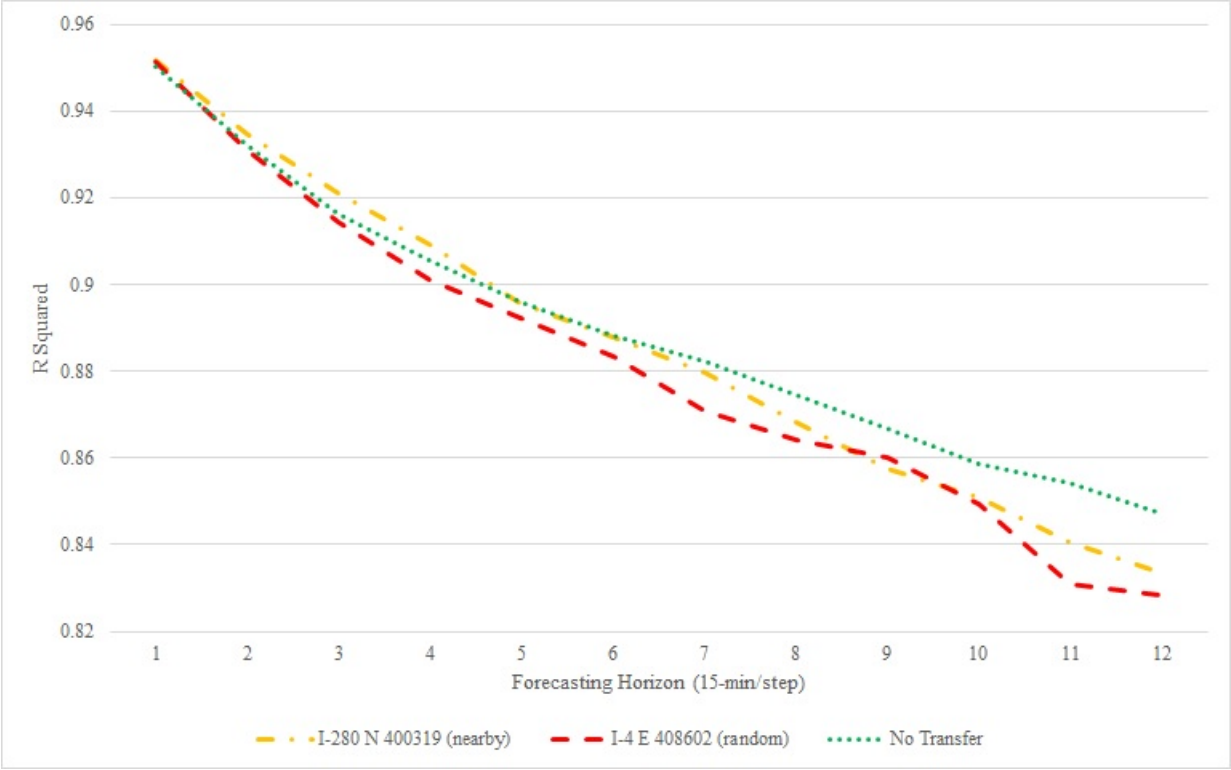

Figure 4.17: R^2 of Sensor 403318 on I-280 N using Transfer Learning.

and bias are from a nearby correlated sensor of the same highway. Comparing with the performance without using transfer learning at all, the neural network trained from randomly initialized weights performed quite similarly with the neural network that is trained using transfer learning from a nearby correlated sensor. In fact, in some steps and metrics, such as MAPE and early steps of NRMSE and R^2, transfer learning actually leads to better results.

To proceed with the transfer learning process, initially, all the sensors from a given highway in a single direction are sorted by their post miles, representing their relative locations on the highway. Then starting from the initial sensor where the traffic is flowing from, check its correlation with subsequently located sensors using 2017 data only. If the correlation is greater than a particular threshold, 80% in our case, then traffic patterns from the two sensors are considered to be sufficiently similar. If at any point the threshold falls below the threshold, then a new group of sensors is to be created starting from that sensor. During the training phase, the starting sensor in each group is to be trained first. Then each subsequent sensor would start training, using much fewer epochs, by initializing weights and biases from the finished state of the models for the previous sensor. The performance evaluations using the 3 metrics are illustrated in Figure 4.18, Figure 4.19, and Figure 4.20.

Overall, the Neural Network (NN) model, which consists of 4 feed-forward layers, performs the best. Its inputs are the same as those used in the models from the holiday experiment. The sizes of the two hidden layers are set to 90% and 80% of the size of the input layer while the output layer is of size 12, representing forecasts from all 12 steps ahead.

The Encoder-Decoder Gated Recurrent Unit (EDGRU) did not perform as well, which is rather unexpected since some of our earlier results obtained using data from the San Diego area demonstrates that the Encoder-Decoder LSTM, which is very similar to EDGRU, was obtain to yield decent, top tier forecasting performance. One possible reason could be the lack of flexibility in its input structures by requiring the additional time steps dimension. Because of the additional time steps dimension, the inputs to EDGRU has to be adjusted.

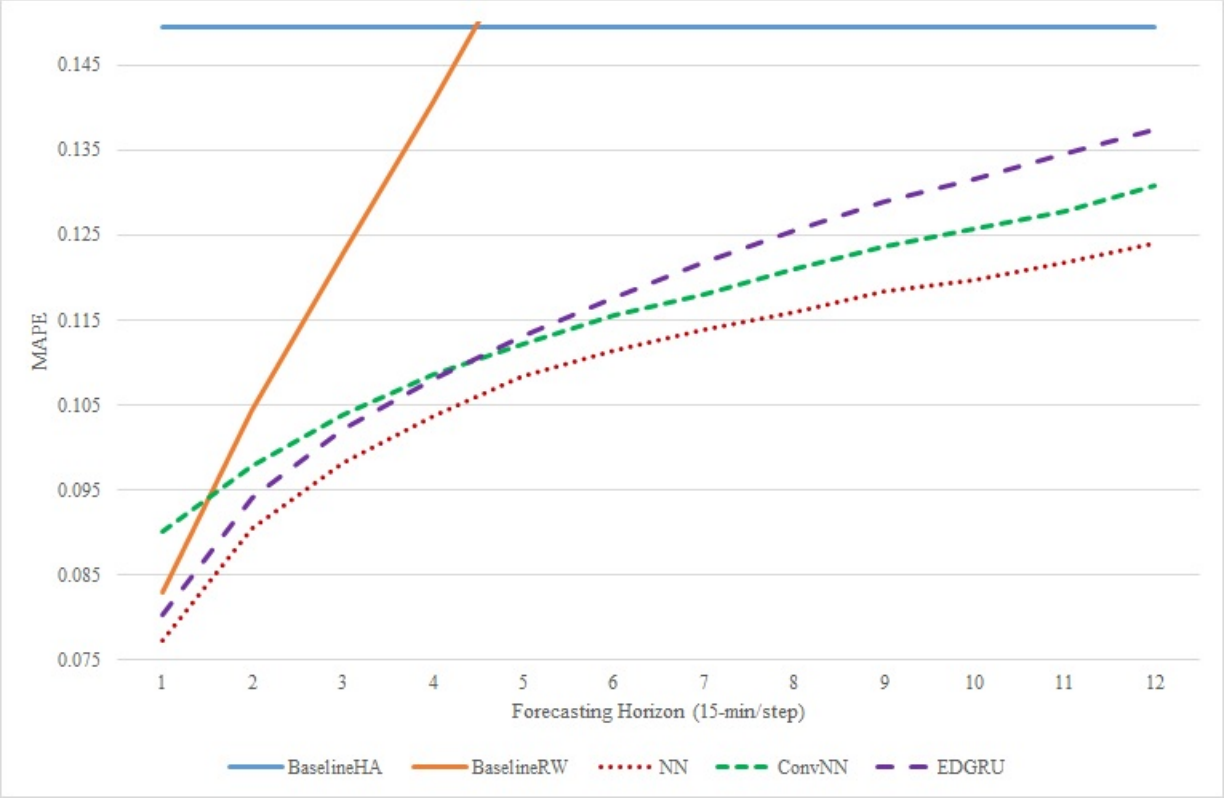

Figure 4.18: MAPE of Neural Network Models using Transfer Learning.

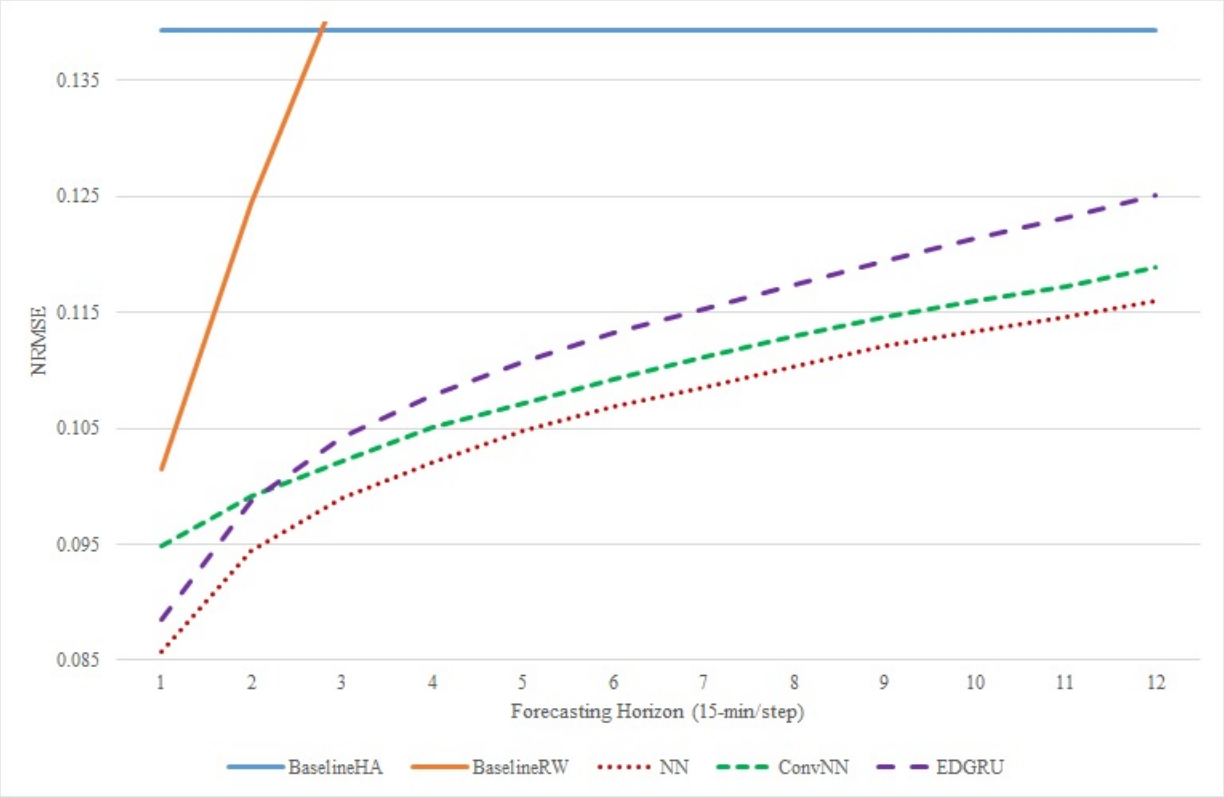

Figure 4.19: NRMSE of Neural Network Models using Transfer Learning.

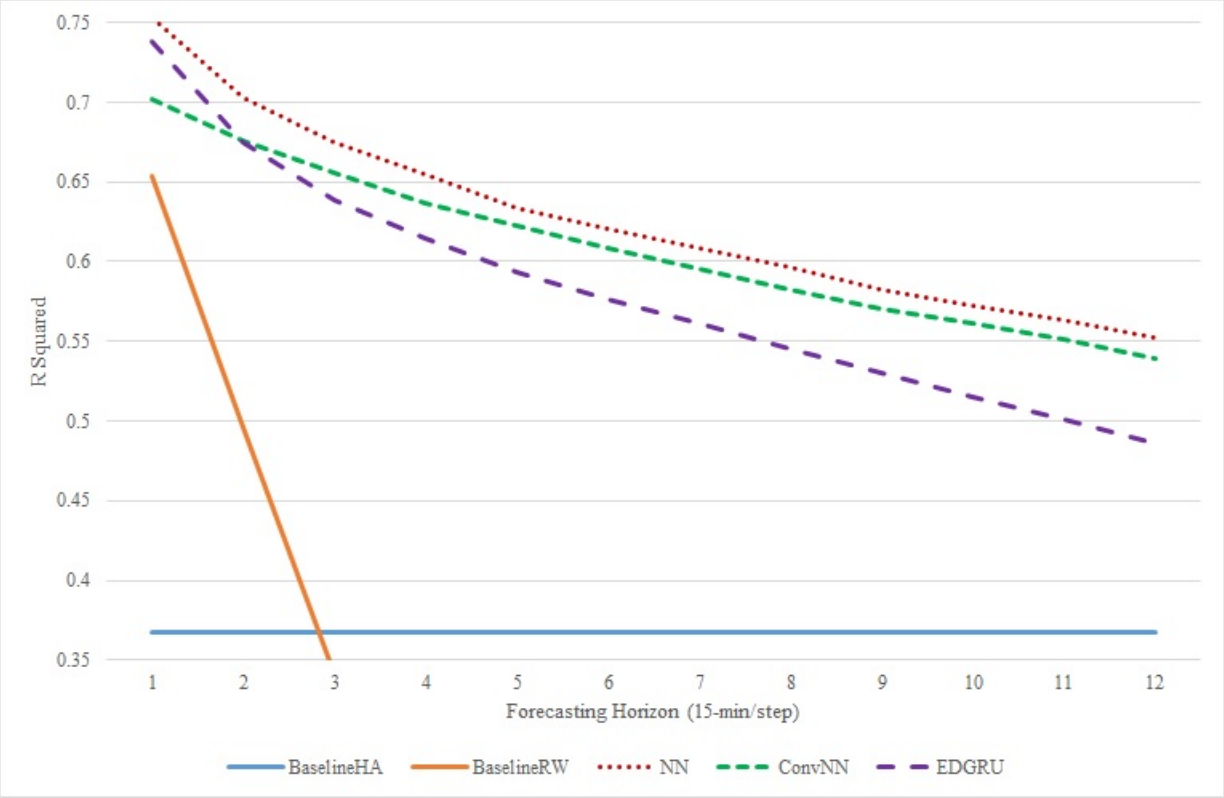

Figure 4.20: R^2 of Neural Network Models using Transfer Learning.

The number of time steps is set to 24, consisting of 12 seasonal steps and 12 most recent lags. The time series used as inputs include historical average, traffic flow, traffic speed, and time of the day. Some inputs must be omitted, such as the day of the week, which would contain a sequence of the same value for all 24 steps; and holiday, which would be a sequence consisting of all zeros most of the time. EDGRU also tends to overfit on a particular dataset, even though the training and validation MSE are still decreasing, but the forecasting performance can become poorer. The validation set does not seem to be able to completely guard EDGRU against overfitting, which is also unexpected since the validation set serves the purpose of guarding the model against overfitting. It is also possible that the test set consisting of the entire year of 2018 may be too far into the future and the trained parameters for EDGRU are not as up to date. Convolutional Neural Networks (ConvNN) provides reasonable performance, typically slightly worse than NN. The inputs to ConvNN are identical to those used in EDGRU.

After the completion of transfer learning for the neural network models, their performance graphs may be merged with earlier figures consisting of only the efficient models, namely Figure 4.3, Figure 4.4, and Figure 4.5. Figure 4.21, Figure 4.22, and Figure 4.23 illustrate the performance comparisons among all our available models.

It would be interesting to compare the performance of the top performers from the group of efficient models and the best neural network model. Figure 4.24 contains a graphs of the percentage of improvement of QuadELM over NN. It can be seen that if judging by the MAPE metric, the performance of QuadELM and NN are actually quite similar, with QuadELM take the lead in the first 4 steps and slightly loses to NN in the remaining steps until step 12. The NRMSE and R^2 metrics, on the other hand, both show the NN performs better, only slightly in the initial steps and then around 3% for later steps. It is worth noting that the NRMSE and R^2 metrics are inherently more similar because they both contain the Sum of Squared Error (SSE) component in the definition of the metrics. Besides, the QuadELM

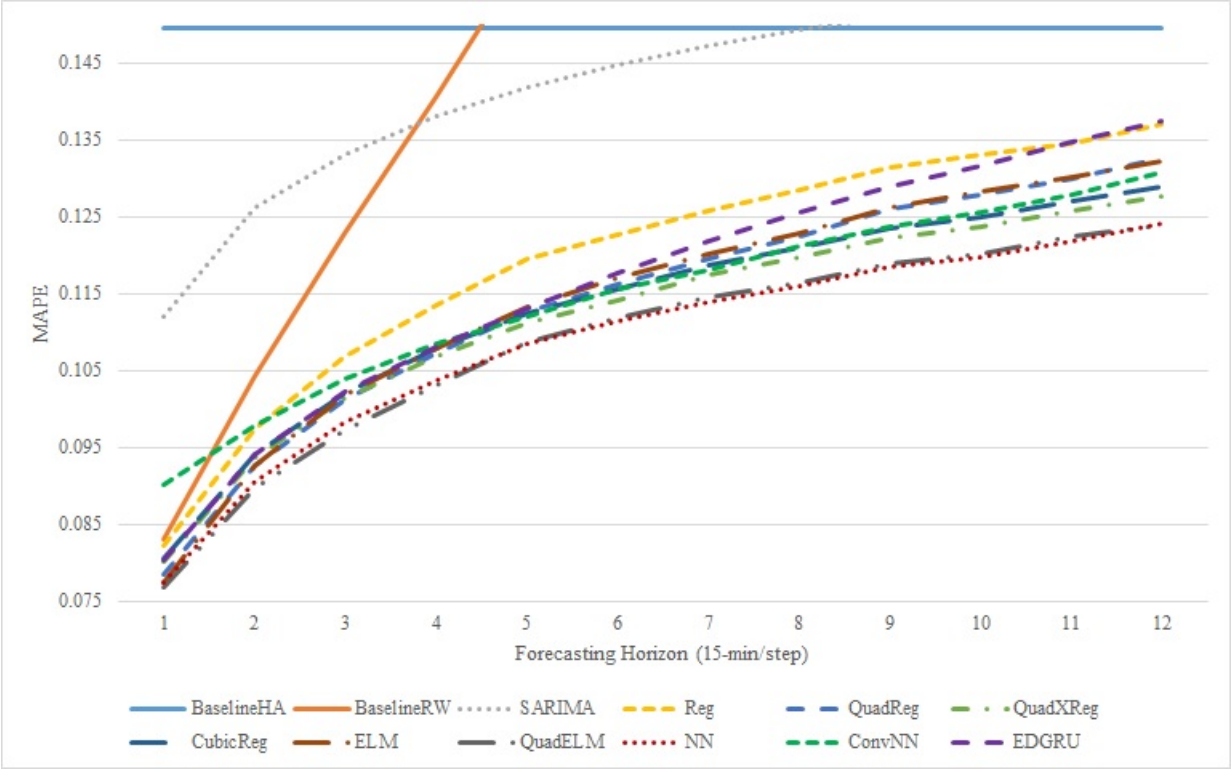

Figure 4.21: MAPE of Forecasts for All Models from 2018 in District 4.

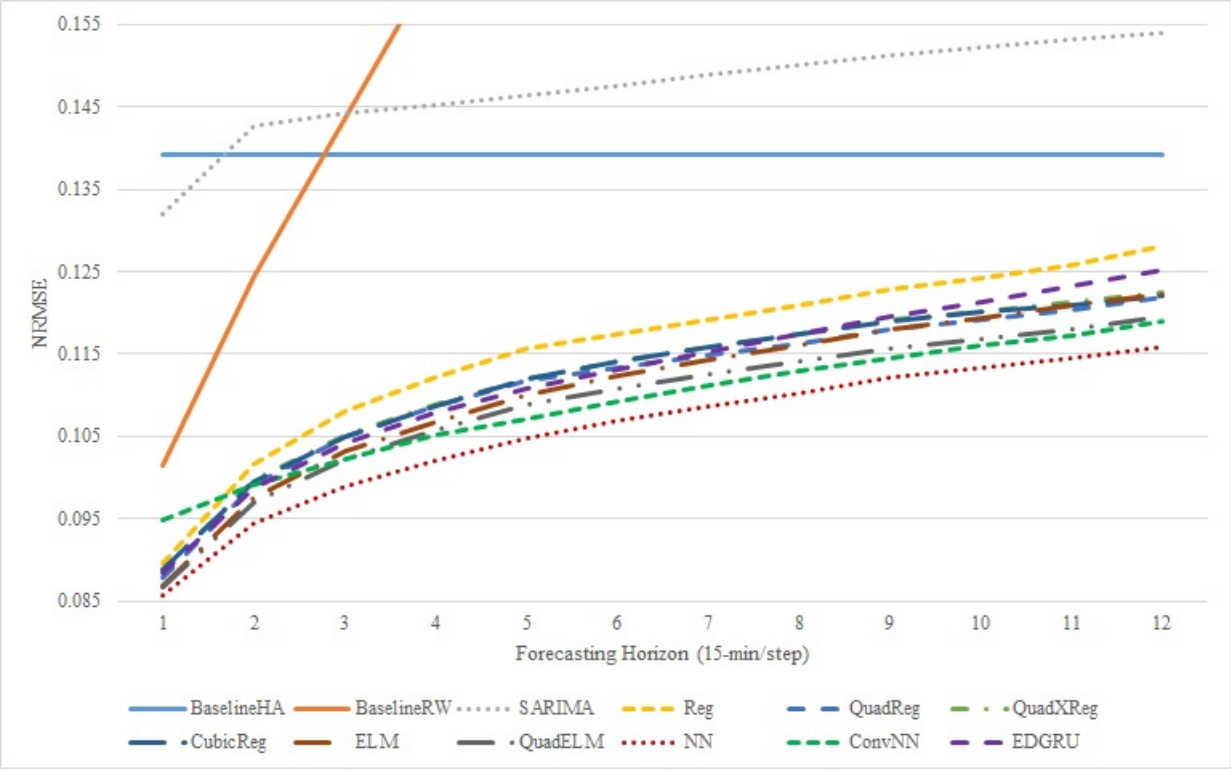

Figure 4.22: NRMSE of Forecasts for All Models from 2018 in District 4.

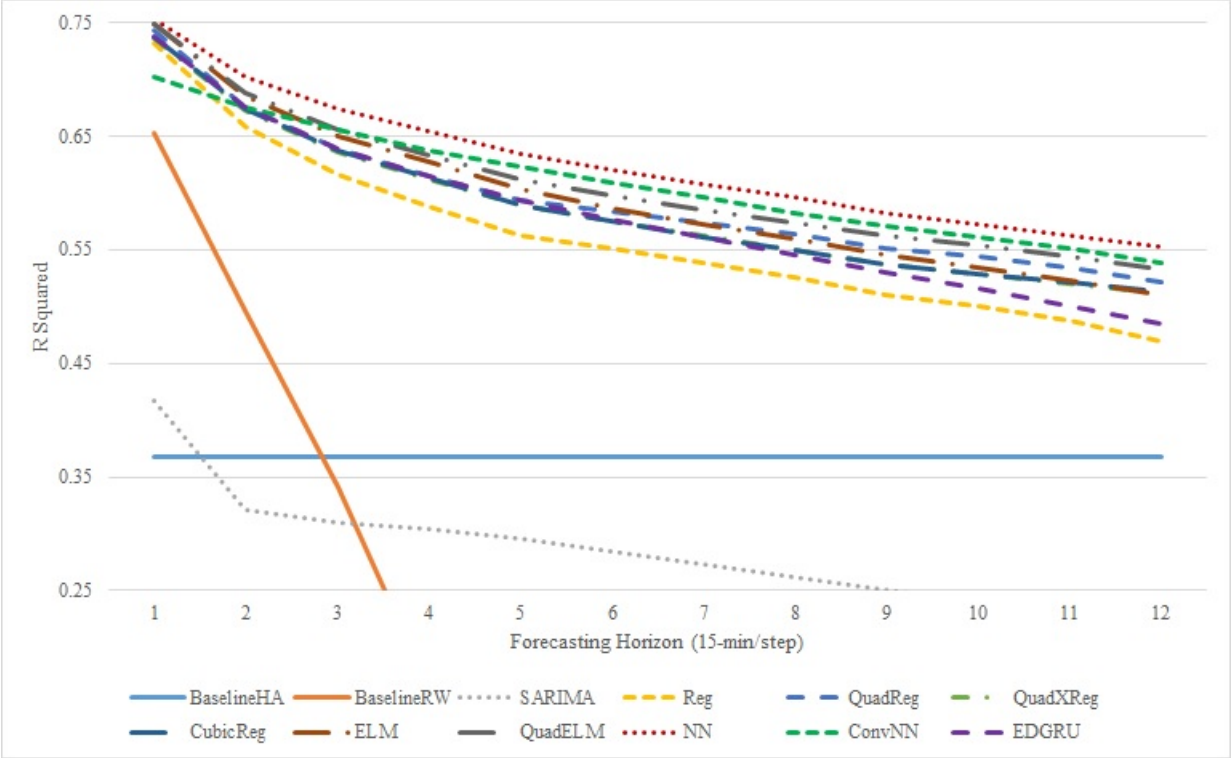

Figure 4.23: R^2 of Forecasts for All Models from 2018 in District 4.

is generally much less computationally expensive than the Neural Networks. Depending on the availability of computational resources, QuadELM may be more preferable than Neural Networks in certain situations.

The performance of ELM and Neural Networks can also be compared. Figure 4.25 illustrates the improvement of ELM over NN. Since all percentage values are negative, ELM is generally not as accurate as NN. When using NN as a benchmark, the performance difference between QuadELM and ELM are indeed significant.

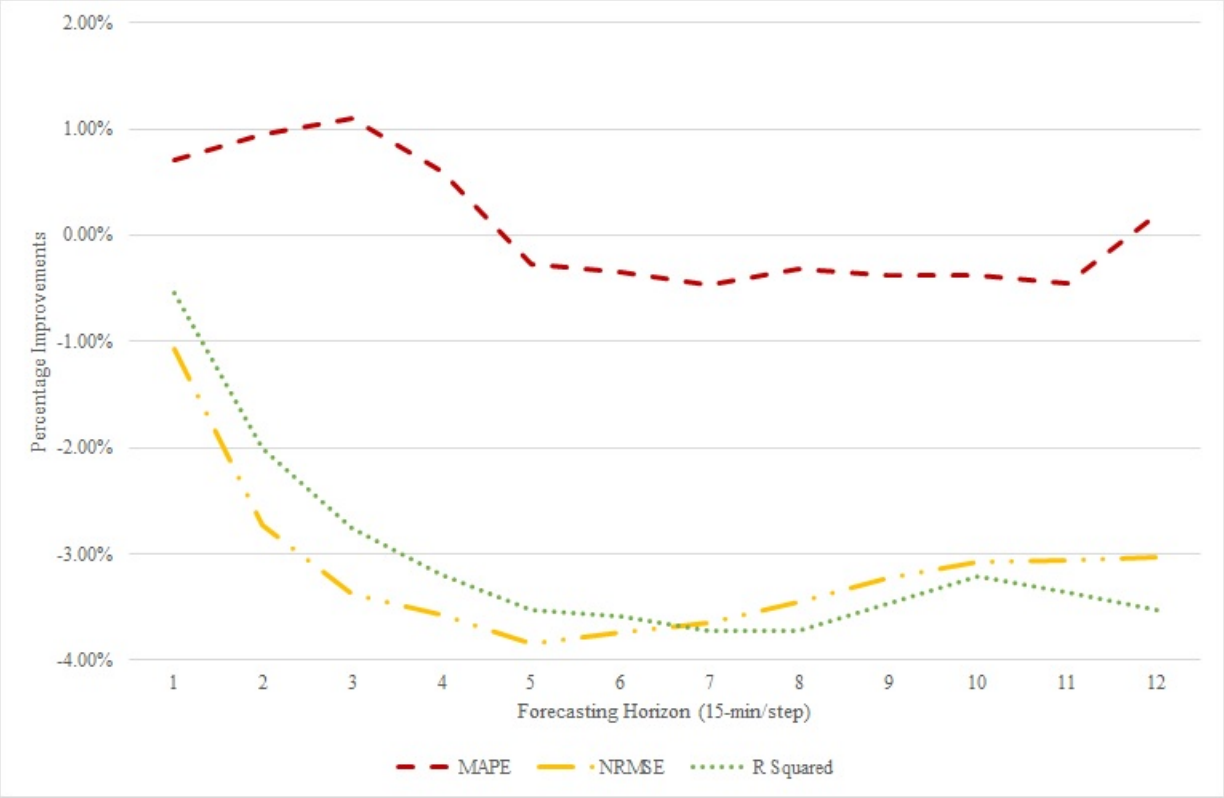

Figure 4.24: Percentage of Improvement of QuadELM over NN.

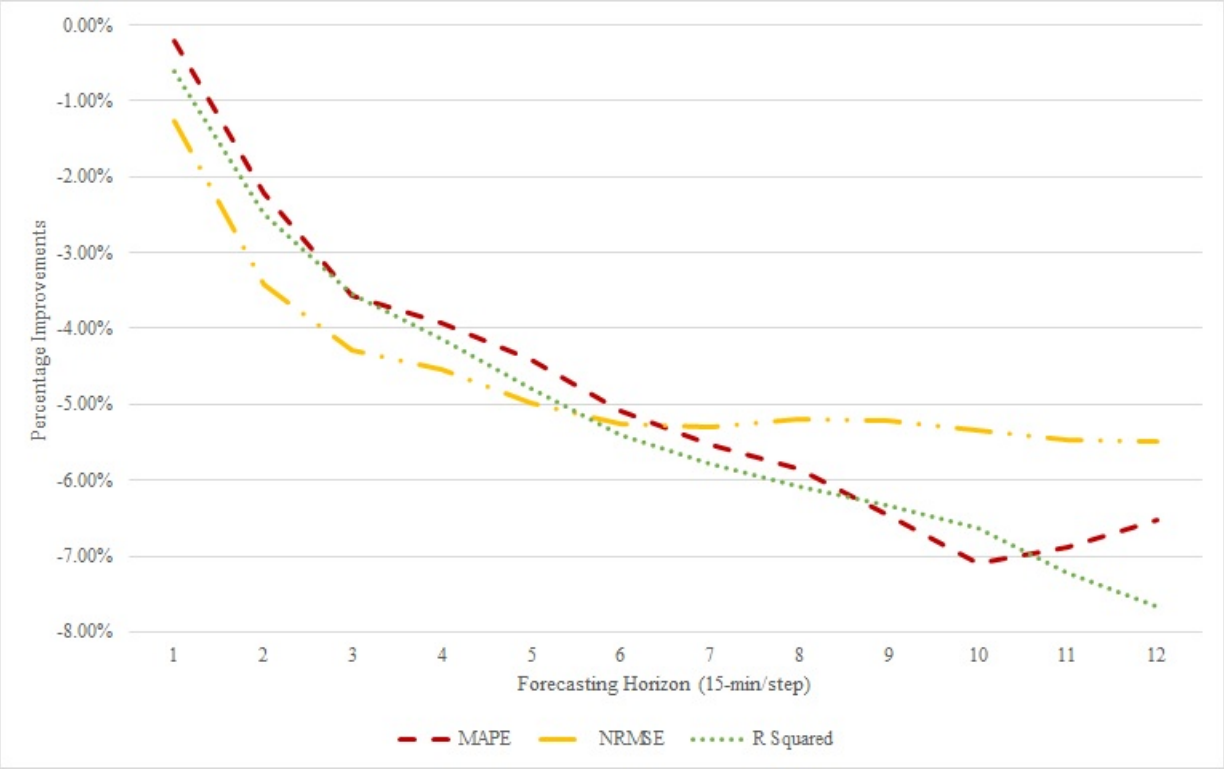

Figure 4.25: Percentage of Improvement of ELM over NN.

4.5.3 Periodic Re-training with Up to Date Data

Though traffic patterns, for the most part, possess similar weekly patterns at a given location, there can still be situations that cause traffic patterns to shift. Extended periods on road work or expansion of a major company that creates thousands of new employment opportunities are examples of such situations. There may also be other unobserved situations that cause the training data to gradually become outdated as time passes. Therefore, it would be useful to periodically retrain the models so that the most up-to-date traffic patterns can be learned. In recent studies on traffic forecasting, most of which focus on deep learning, this is generally not a considered option due to the high computational costs of the deep learning models. To explore this option, several very efficient models were chosen and tested. The performance degradation is first studied to see how well the models can forecast traffic as time passes throughout the testing set, which includes the entire year of 2018. Figure 4.26 illustrates the performance degradation of a sample sensor throughout the months of 2018. It can be observed that the general trend of the metrics are declining as time passes in 2018.

The sliding window, representing the size of the training set, is tuned to be 52 weeks, roughly equivalent of a year worth of data. The re-training frequency is set to 2 weeks. Figure 4.27, Figure 4.28 and Figure 4.29 illustrate the forecasting performance in District 4 for the entire year of 2018 using periodically re-trained models. The figures are intended to be compared with Figure 4.3, Figure 4.4 and Figure 4.5, for MAPE, NRMSE and R^2, respectively.

For more straight-forward comparisons, the percentage of improvement metrics are calculated for the models. Figure 4.30, Figure 4.31 and Figure 4.32 illustrate the percentage of improvement of the periodically re-trained models over their one time trained counterparts.

From the figures, models generally show improvements when they are periodically re-trained using the most up-to-date data. The ELM and QuadELM models show greater potentials in their ability to take advantage of the most up-to-date data than linear and

Figure 4.26: Degradation in Performance of Sensor 400001 on I-101 N. Step 12 Forecasts by QuadELM are Used.

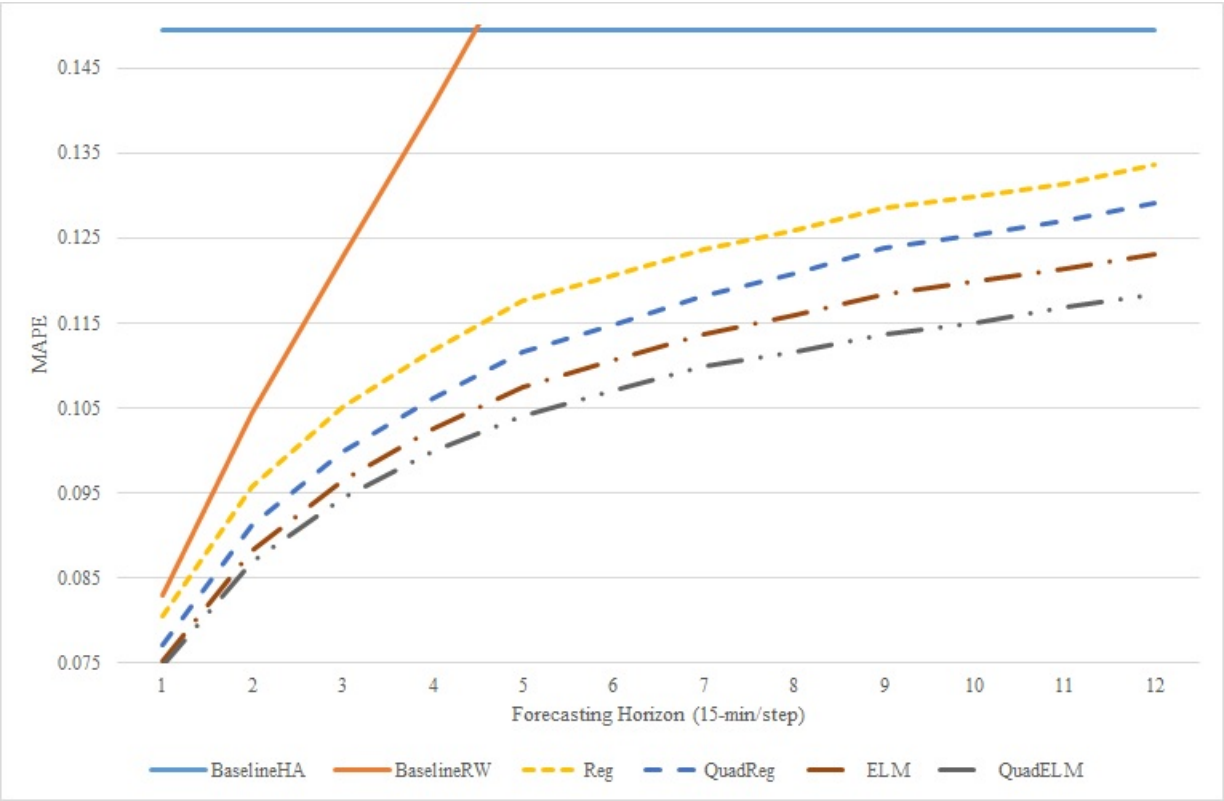

Figure 4.27: MAPE of Forecasts using Periodically Re-trained Models.

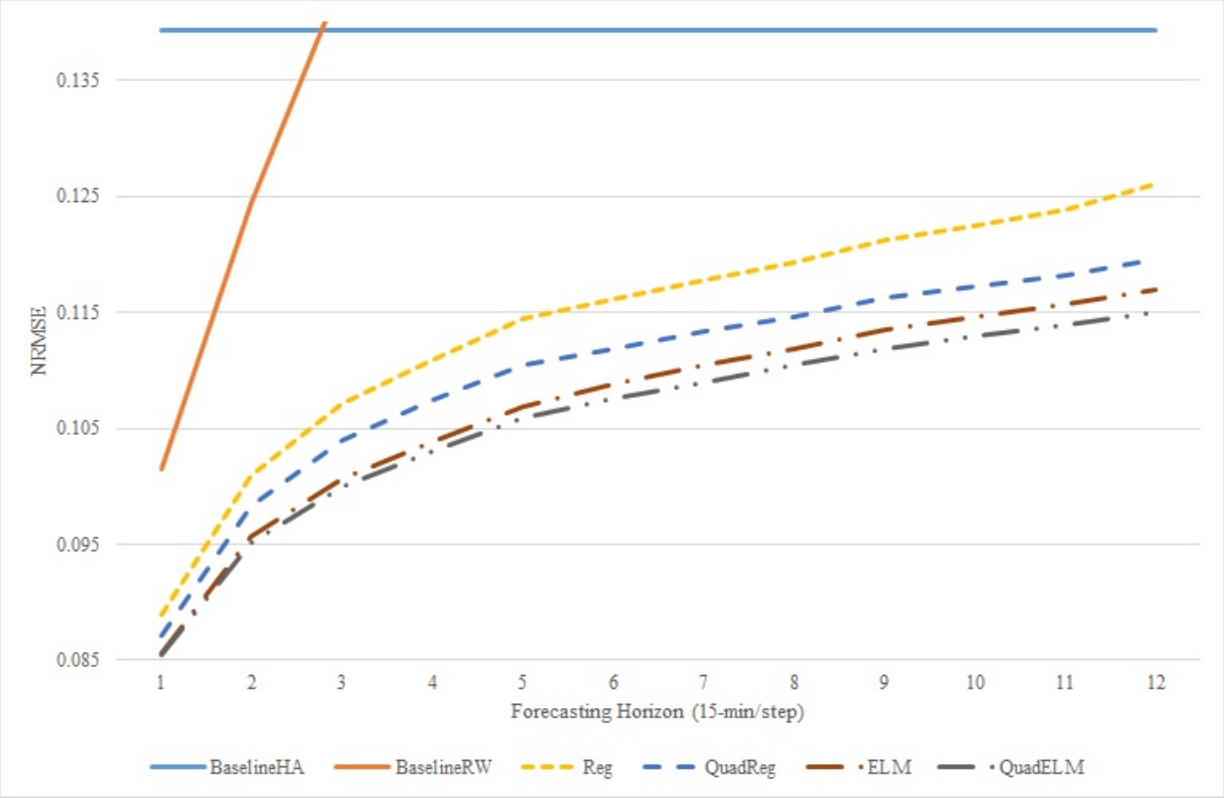

Figure 4.28: NRMSE of Forecasts using Periodically Re-trained Models.

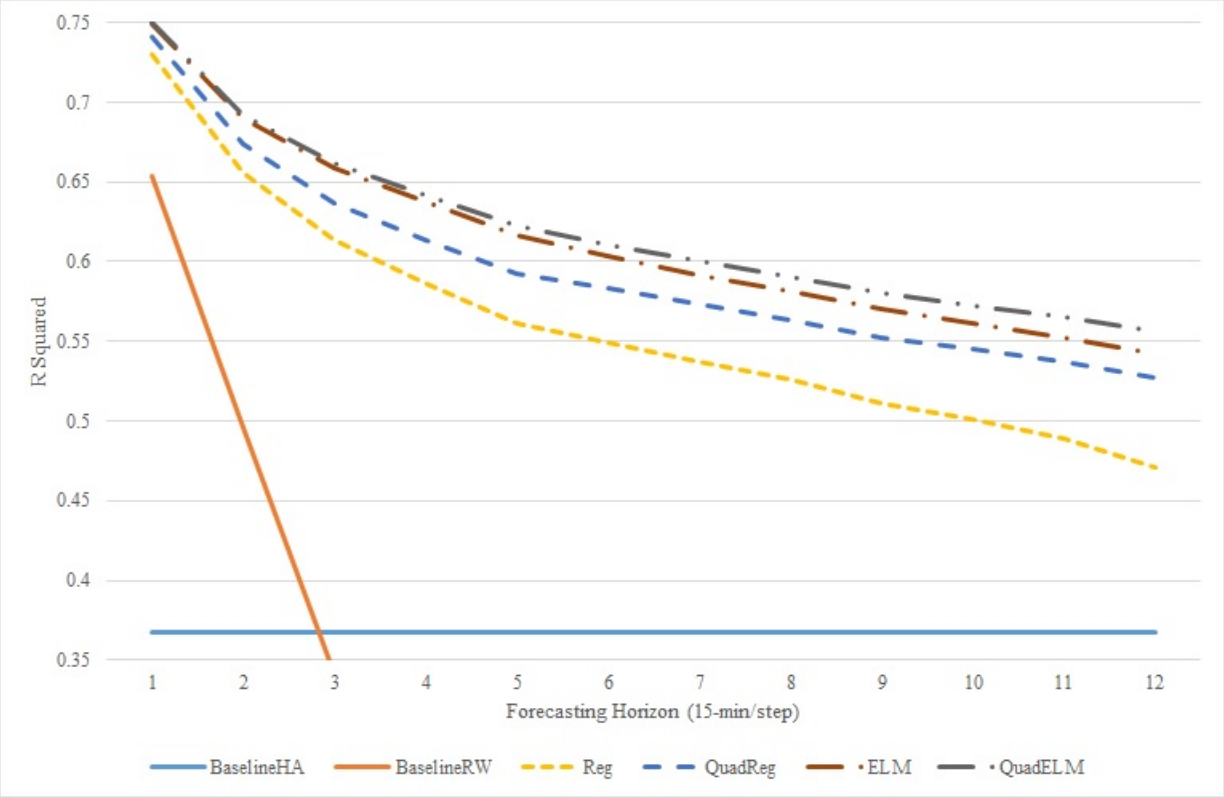

Figure 4.29: R^2 of Forecasts using Periodically Re-trained Models.

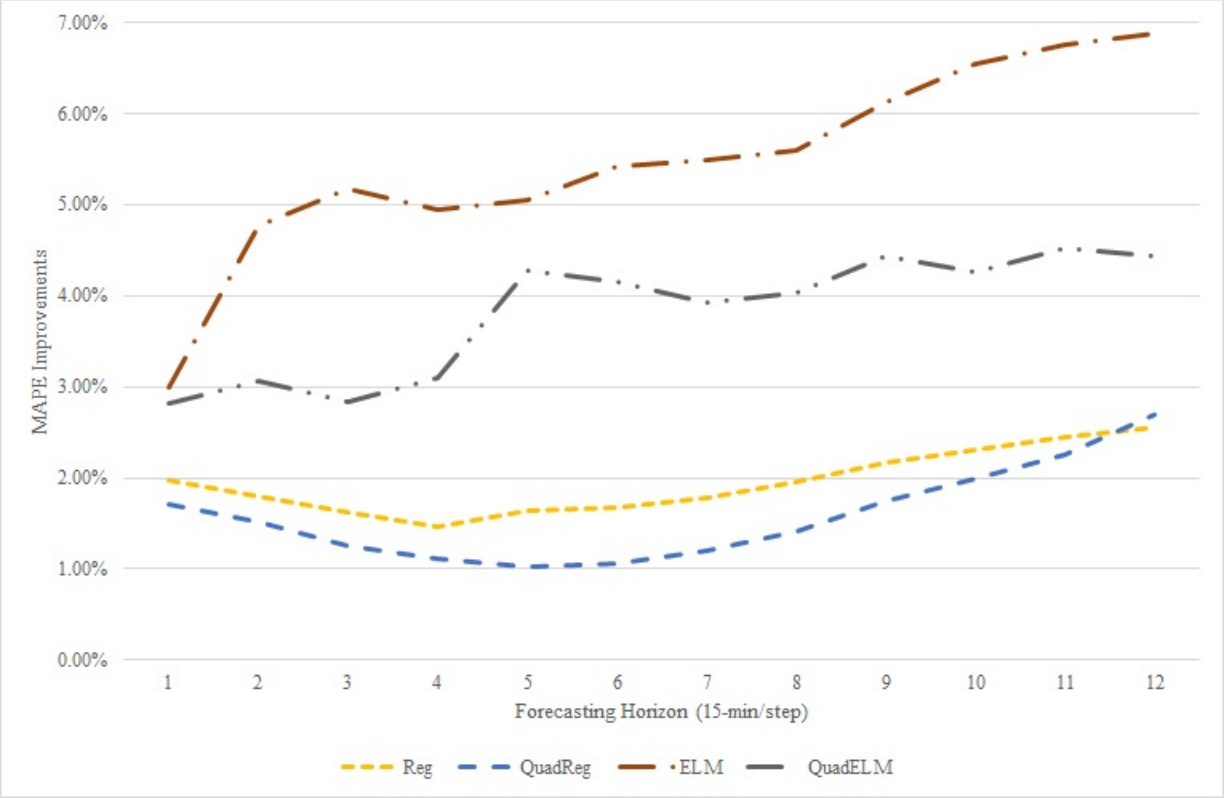

Figure 4.30: MAPE Improvement using Periodically Re-trained Models.

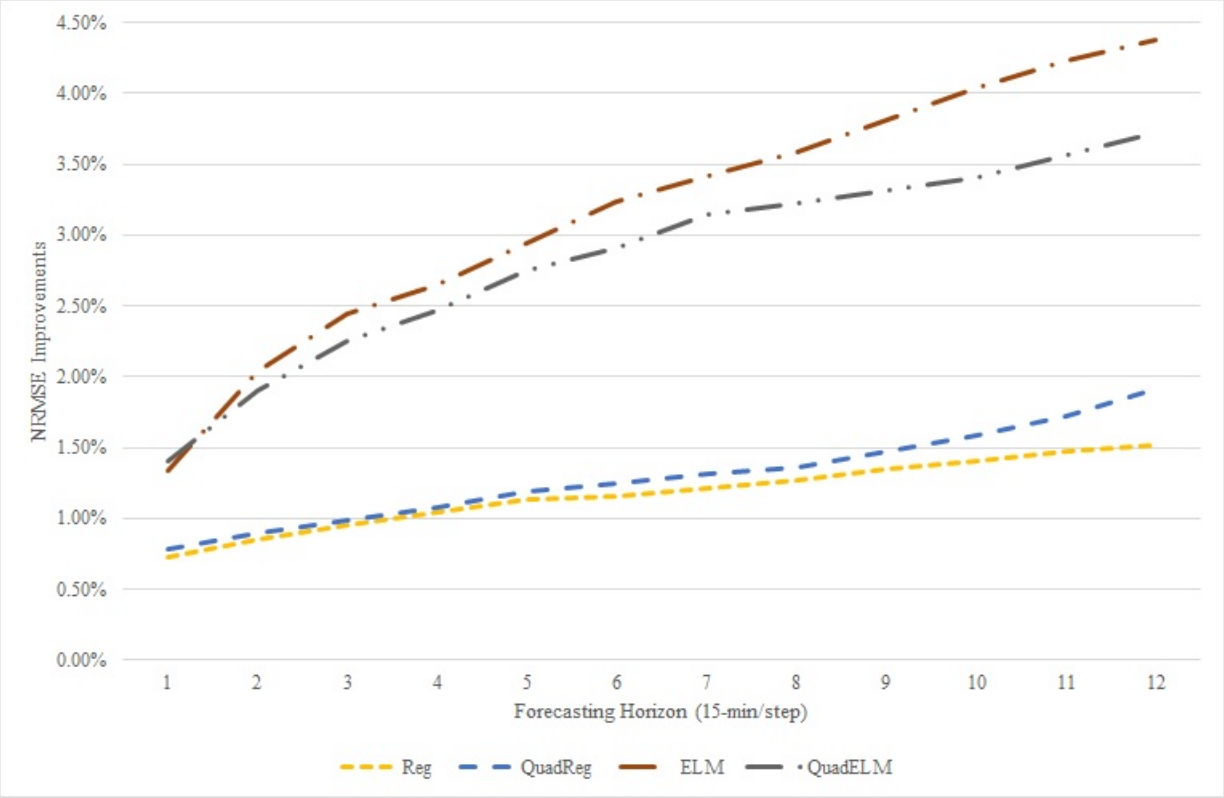

Figure 4.31: NRMSE Improvement using Periodically Re-trained Models.

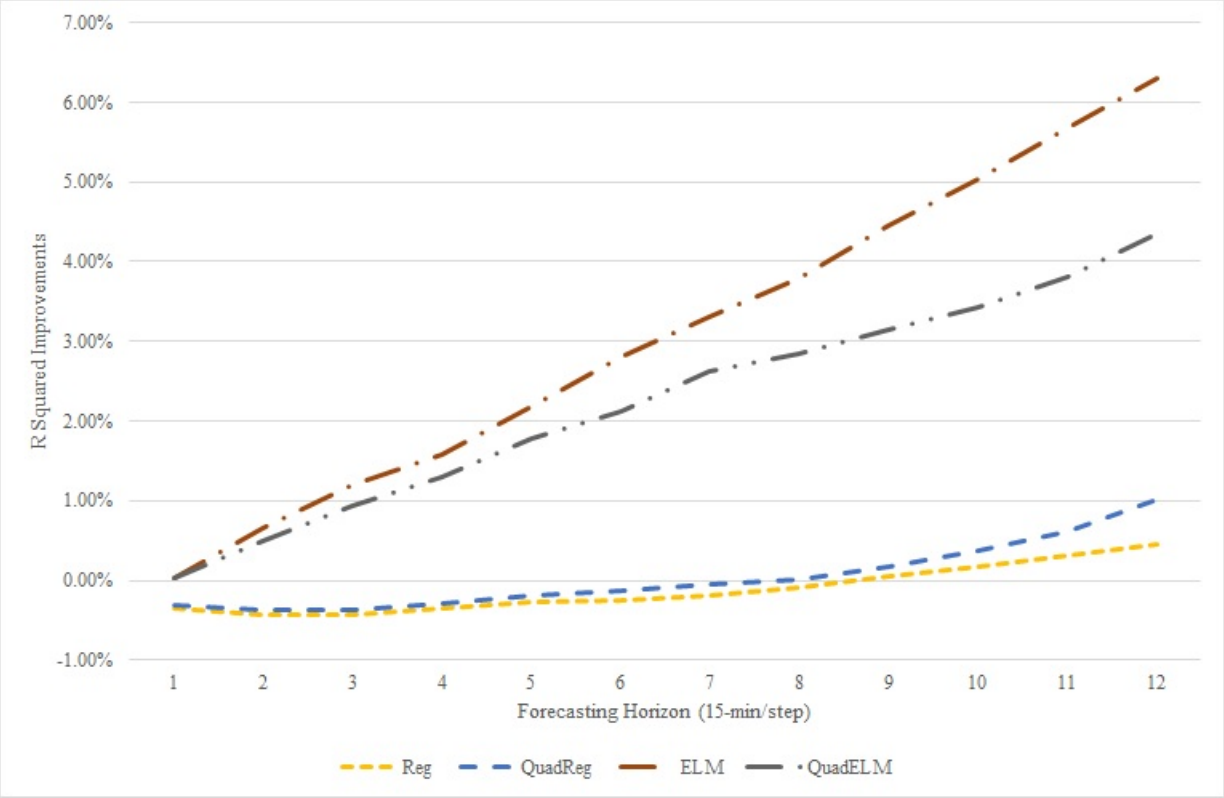

Figure 4.32: R^2 Improvement using Periodically Re-trained Models.

quadratic regression models. A very interesting observation is that the percentage of improvement generally increases with the forecasting step. In other words, the most up-to-date training data generally help more when trying to make longer-term forecasts than the immediate short term. A possible explanation for this scenario is that in the immediate short term, say 1-step ahead, the forecasting problem is generally the easiest when comparing against longer steps ahead. As a result, there is not much room for improvement. On the other hand, the longer-term forecasting problem is relatively more difficult, and training using the most up-to-date data helps to allow the models to capture the potentially evolving traffic patterns. It is also worthy to note that even though ELM shows greater improvements than QuadELM, the performance of QuadELM is still the best.

4.5.4 Traffic Forecasting in Low Visibility Conditions

Another important factor that may cause traffic patterns to deviate from the norm is a special weather condition. Rainfall and fog are likely to cause drivers to slow down to ensure safety. To consider special weather conditions, a total of 14 Automated Surface Observing System (ASOS)[4] sensors near the San Francisco Bay Area are used. To ensure the accuracy of the data, each PeMS sensor is paired with its closest ASOS sensor and the distance between the two sensors must be less than 5 miles. As a result, a total of 447 PeMS sensors meet the qualification and are considered in this case study.

The weather attribute of interest is visibility. Visibility data are available in 5-minute resolution, while the rainfall data are of hourly resolution. Also, fog is a pretty common occurrence in the San Francisco Bay Area, therefore it would be important to consider such a factor. Initially, several models are evaluated only during low visibility conditions (3 miles or less) without any inputs from the ASOS sensors. Figure 4.33, Figure 4.34 and Figure 4.35 illustrate the performance of the forecasting models in the 3 metrics. The historical

[4] `https://www.weather.gov/asos/`

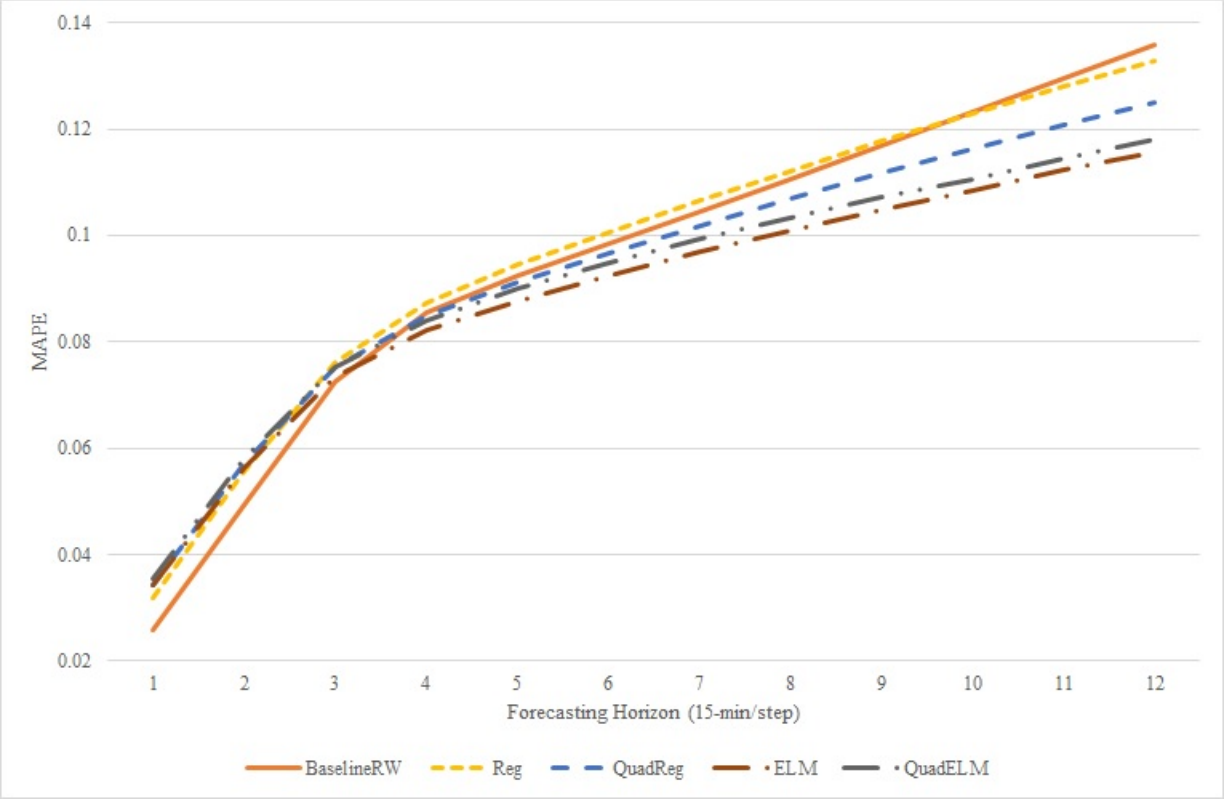

Figure 4.33: MAPE in Low Visibility Conditions without ASOS data.

average baselines, unsurprising, performs very poorly and are excluded from the graphs. The historical average values for MAPE, NRMSE and R^2 are approximately 0.332, 0.374 and -3.36, respectively.

There are a couple of noteworthy observations. The random walk baseline is very effective in the immediate short terms. It is likely because, in low visibility conditions, all drivers will slow down their vehicles to a somewhat uniform speed and carefully follow the car in front from a safe distance. The result is likely a near-constant traffic flow in the immediate short term. Another interesting observation is that unlikely the holiday case study, the models, without any inputs from ASOS sensors, actually perform somewhat decently. This is likely because the models have access to speed data from the PeMS sensors already, and low

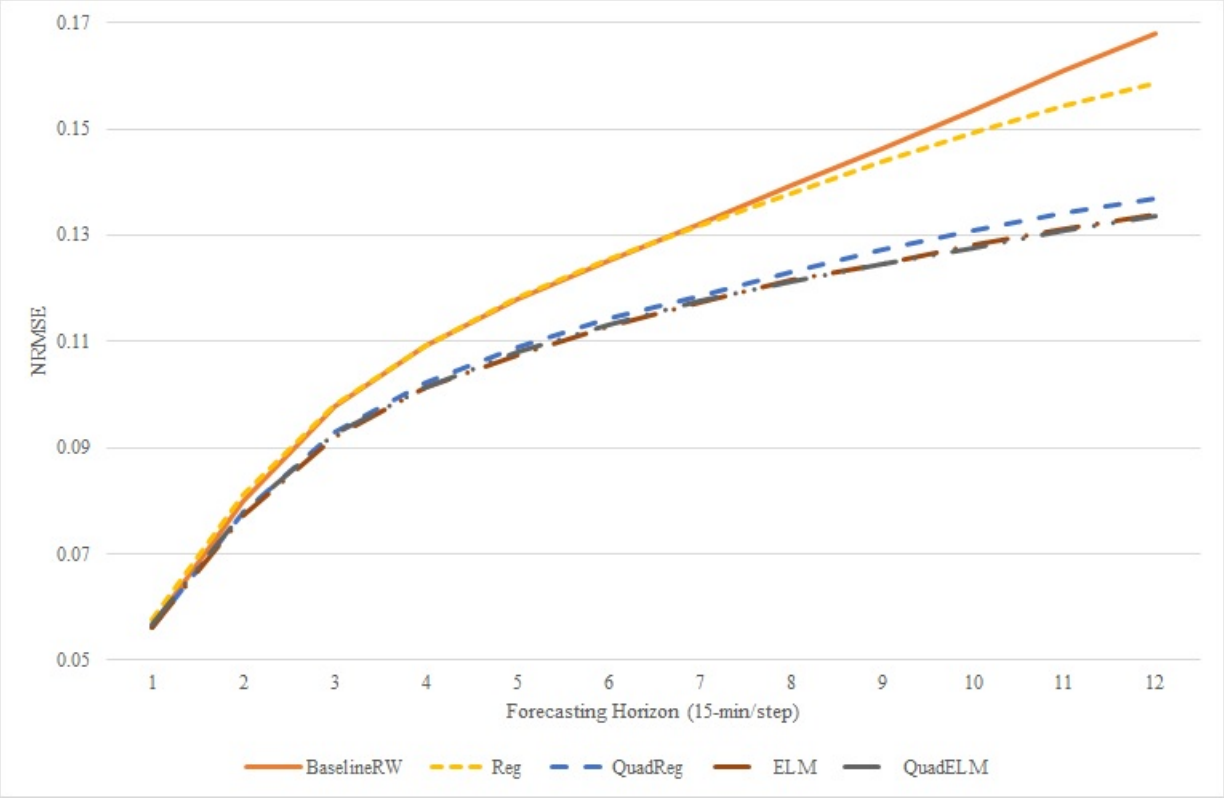

Figure 4.34: NRMSE in Low Visibility Conditions without ASOS data.

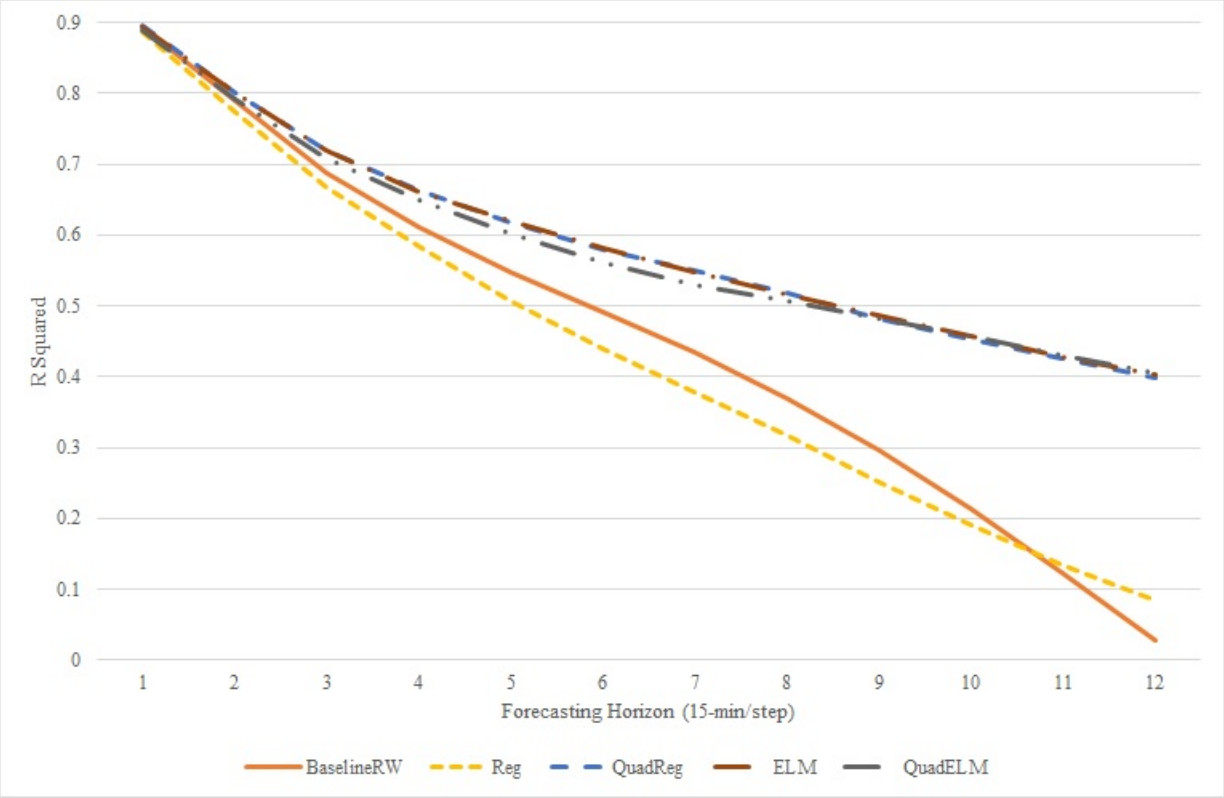

Figure 4.35: R^2 in Low Visibility Conditions without ASOS data.

visibility will cause a reduction in speed, thus the information about the weather condition is indirectly given to the models through speed data. This is different from the holiday scenario, for which the speed data, during holidays, are likely to remain high throughout the day.

In an attempt to improve forecasting accuracy in low visibility conditions, the following adjustments to the models are made. Remove the seasonal components, as they are not likely to help with the short term forecasting. This may potentially cause the models to lose accuracy in longer terms, depending on how soon the visibility restores to normal. Also, the models are now trained with instances where the visibility is less than 10 miles, which is considered to be perfect visibility in the ASOS datasets. A preliminary attempt to train the models using data with less than or equal to 3 miles of visibility did not yield good results, possibly due to a lack of training data. Figure 4.36, Figure 4.37 and Figure 4.38 show the performance of the models trained with consideration of the ASOS visibility data.

Overall, it appears that the quadratic regression model performs the best in this scenario. The linear regression model also appears to show the greatest improvements, though still at the low tier of all the models. The ELM and QuadELM models closely follow quadratic regression but appeared to suffer losses in accuracy in longer terms. The sizes of their hidden layers are also reduced just like in the holiday study. For easier comparison of the effects of training with instances in which the visibility is less than perfect, the percentage of improvement graphs are included in Figure 4.39, Figure 4.40 and Figure 4.41. Figure 4.41, in particular, computes improvement scores using the simple difference formula in Equation 4.2.

The figures show that overall, the simpler models, linear and quadratic regression, benefit the most from working with selected training instances and features. The ELM model generally did not benefit. The QuadELM model can benefit a bit from this setup in the short term but loses accuracy in the longer terms. It is possible that ELM and QuadELM

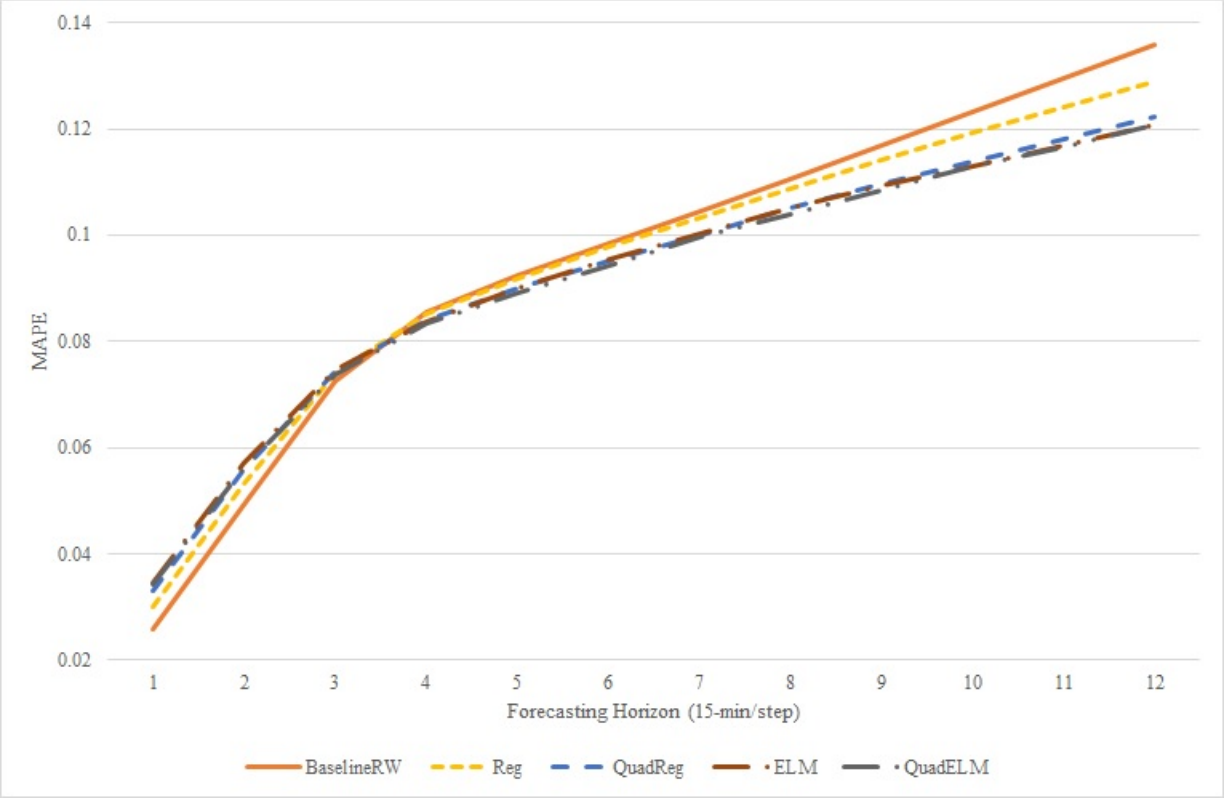

Figure 4.36: MAPE of Models Trained with Consideration of ASOS Data.

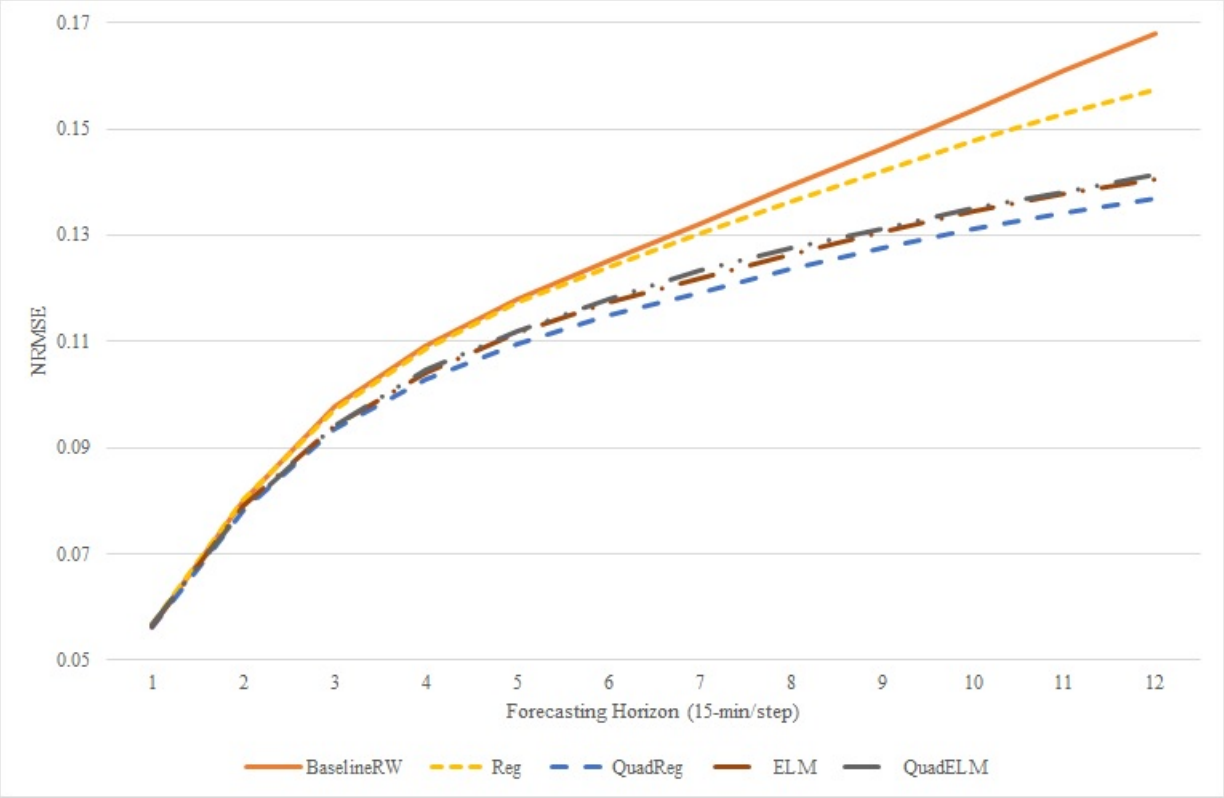

Figure 4.37: NRMSE of Models Trained with Consideration of ASOS Data.

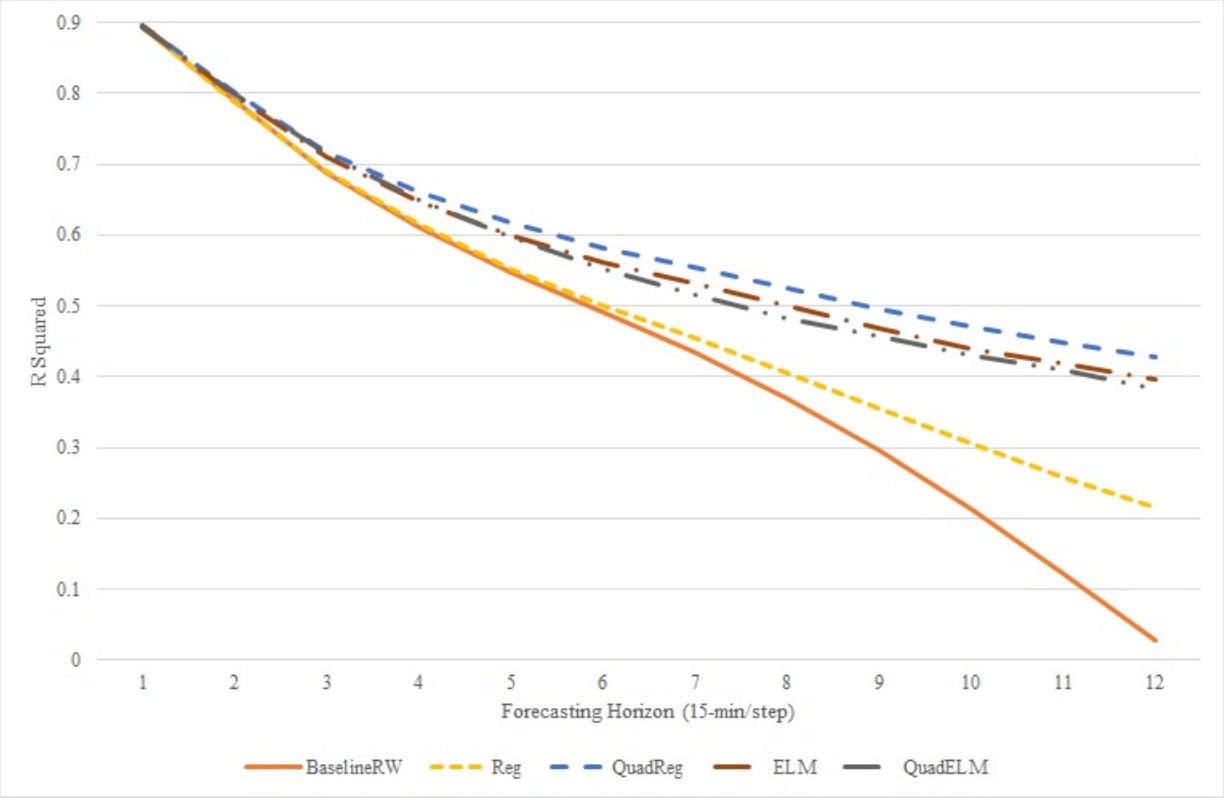

Figure 4.38: R^2 of Models Trained with Consideration of ASOS Data.

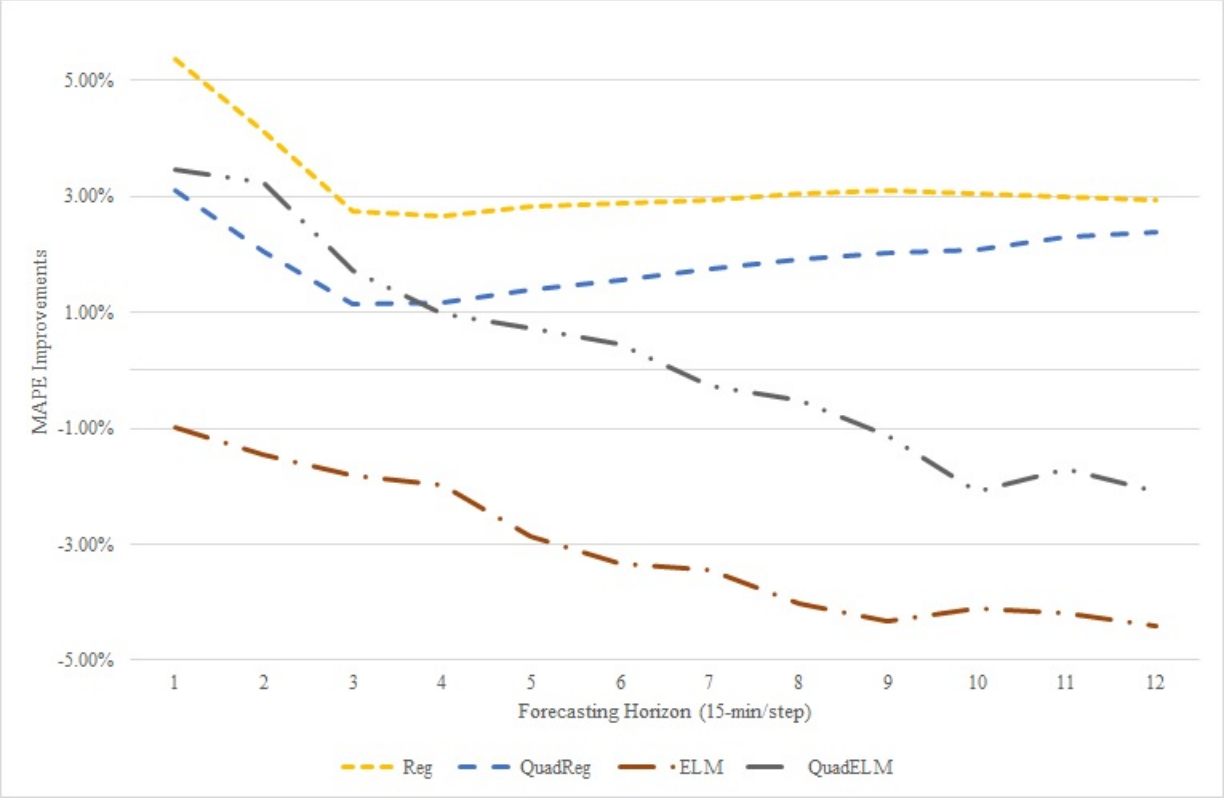

Figure 4.39: MAPE Percentage of Improvement Due to Consideration of ASOS Data.

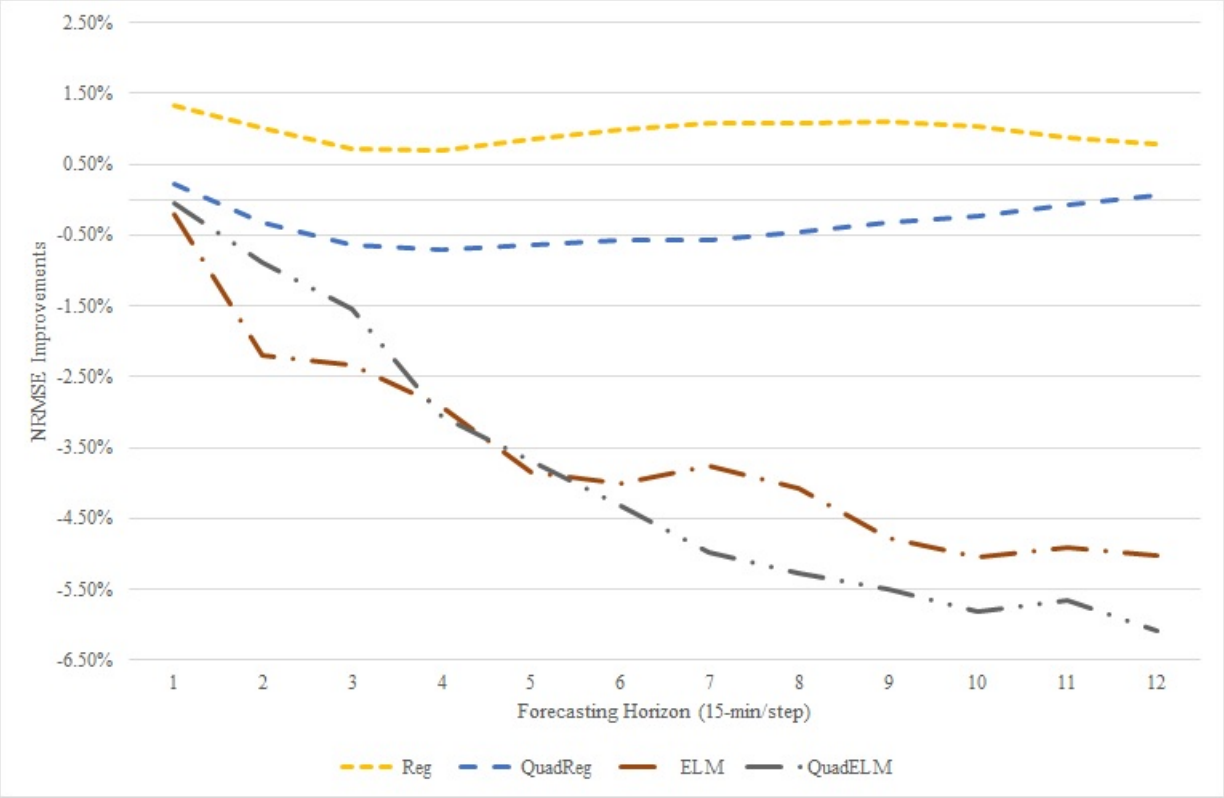

Figure 4.40: NRMSE Percentage of Improvement Due to Consideration of ASOS Data.

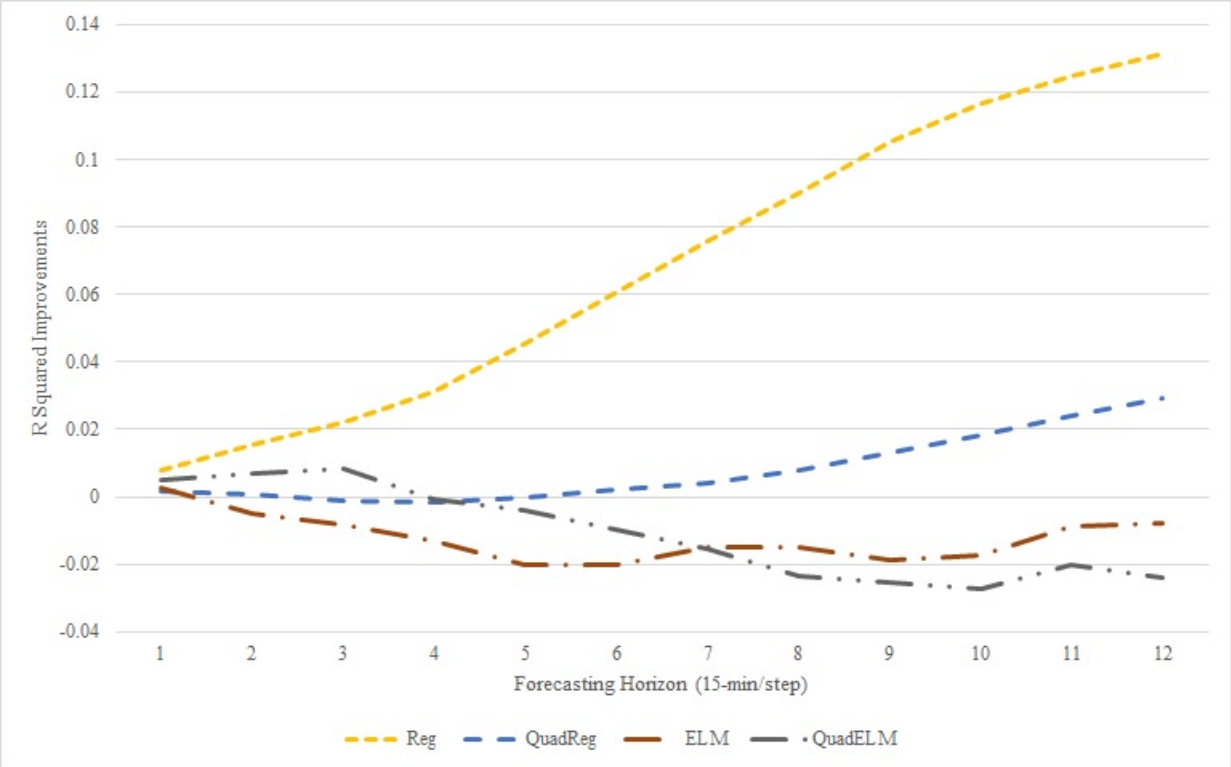

Figure 4.41: R^2 Simple Difference Improvement Due to Consideration of ASOS Data.

simply need more training data in low visibility conditions. Potentially multiple years of training data may be downloaded to see if there improvements on the ELM and QuadELM model.

4.6 Conclusions and Future Work

In this study we have demonstrated the advantages of being aware of various traffic affecting situations can indeed help with the modeling process. In particular, four experiments are conducted to illustrates the effectiveness of doing big data analytics while remaining situation-aware. The awareness of holidays has led to better forecasting models designed to handle the rare situation. The awareness of the locations and correlations among the sensors has facilitated effective transfer learning. The awareness of the need to use up-to-date training data has produced improved results by various models. The awareness of special weather conditions has led to the preference of simpler models in such situations. We have also presented the Quadratic Extreme Learning Machine model that is efficient, generally improves upon the standard Extreme Learning Machine model, and can potentially be an alternative to Neural Networks. Future work includes the expansion of the ScalaTion framework to better support the use of semantic technology and theories in data science.

Chapter 5

Conclusion

In summary, we have made the following contributions in this work:

- An extensive literature review is provided, focusing on both the various models used in traffic forecasting and a chronological overview of the developments in this field.
- Using big data, the performance of various statistical and machine learning models traffic flow forecasting is evaluated.
- The impacts of incorporating spatially dependent data into multivariate forecasting models are studied and compared against the univariate cases.
- The performance of multi-step forecasts and the impacts of varying data resolutions are discussed.
- The trade-offs/pros and cons of the models in terms of accuracy, stability, computational cost, and ease of use are explored.
- A situation-aware approach to forecasting traffic is discussed. Being aware of situations such as holidays, special weather conditions, and the locations of sensors can

be effectively used to guide the model building process and complement the popular data-driven modeling approach.

- The Quadratic Extreme Learning Machine model is presented. It generally improves upon the standard Extreme Learning Machine while remaining relatively efficient. Its performance can be competitive with Neural Networks.

For related future work, more support for the theory-driven data science and the inclusion of additional modeling techniques in the ScalaTion project are planned.

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Abdel-Aty and Radwan, 2000] Abdel-Aty, M. A. and Radwan, A. E. (2000). Modeling traffic accident occurrence and involvement. *Accident Analysis & Prevention*, 32(5):633–642.

[Ahmed and Cook, 1979] Ahmed, M. S. and Cook, A. R. (1979). *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*. Number 722.

[Ballard, 1987] Ballard, D. H. (1987). Modular learning in neural networks. In *AAAI*, pages 279–284.

[Barceló et al., 2010] Barceló, J., Montero, L., Marqués, L., and Carmona, C. (2010). Travel time forecasting and dynamic origin-destination estimation for freeways based on blue-

tooth traffic monitoring. *Transportation Research Record: Journal of the Transportation Research Board*, (2175):19–27.

[Bengio, 2012] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bobade, 2018] Bobade, S. U. (2018). The scalation time series database: Support for big data analytics. Master's thesis, University of Georgia.

[Bowman and Miller, 2016] Bowman, C. N. and Miller, J. A. (2016). Modeling traffic flow using simulation and big data analytics. In *Proceedings of the 2016 Winter Simulation Conference*, pages 1206–1217. IEEE Press.

[Box and Jenkins, 1970] Box, G. E. and Jenkins, G. M. (1970). Time Series Analysis Forecasting and Control. Technical report, DTIC Document.

[Brown, 1956] Brown, R. (1956). *Exponential Smoothing for Predicting Demand*. Little.

[Castro-Neto et al., 2009] Castro-Neto, M., Jeong, Y.-S., Jeong, M.-K., and Han, L. D. (2009). Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, 36(3):6164–6173.

[Chan et al., 2012] Chan, K. Y., Dillon, T. S., Singh, J., and Chang, E. (2012). Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and levenberg–marquardt algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):644–654.

[Chandra and Al-Deek, 2008] Chandra, S. and Al-Deek, H. (2008). Cross-correlation analysis and multivariate prediction of spatial time series of freeway traffic speeds. *Transportation Research Record: Journal of the Transportation Research Board*, (2061):64–76.

[Chandra and Al-Deek, 2009] Chandra, S. R. and Al-Deek, H. (2009). Predictions of freeway traffic speeds and volumes using vector autoregressive models. *Journal of Intelligent Transportation Systems*, 13(2):53–72.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Chen and Chien, 2001] Chen, M. and Chien, S. (2001). Dynamic freeway travel-time prediction with probe vehicle data: Link based versus path based. *Transportation Research Record: Journal of the Transportation Research Board*, (1768):157–161.

[Cheng et al., 2018] Cheng, X., Khomtchouk, B., Matloff, N., and Mohanty, P. (2018). Polynomial regression as an alternative to neural nets. *arXiv preprint arXiv:1806.06850*.

[Chien and Kuchipudi, 2003] Chien, S. I.-J. and Kuchipudi, C. M. (2003). Dynamic travel time prediction with real-time and historic data. *Journal of transportation engineering*, 129(6):608–616.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras. `https://keras.io`.

[Chrobok et al., 2004] Chrobok, R., Kaumann, O., Wahle, J., and Schreckenberg, M. (2004). Different methods of traffic forecast based on real data. *European Journal of Operational Research*, 155(3):558–568.

[Cools et al., 2009] Cools, M., Moons, E., and Wets, G. (2009). Investigating the variability in daily traffic counts through use of arimax and sarimax models: assessing the effect of holidays on two site locations. *Transportation Research Record: Journal of the Transportation Research Board*, 2136:57–66.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[Davis et al., 1990] Davis, G. A., Nihan, N. L., Hamed, M. M., and Jacobson, L. N. (1990). Adaptive forecasting of freeway traffic congestion. *Transportation Research Record*, (1287).

[Deng et al., 2017] Deng, D., Shahabi, C., Demiryurek, U., and Zhu, L. (2017). Situation aware multi-task learning for traffic prediction. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 81–90. IEEE.

[Ding et al., 2011] Ding, Q. Y., Wang, X. F., Zhang, X. Y., and Sun, Z. Q. (2011). Forecasting traffic volume with space-time arima model. In *Advanced Materials Research*, volume 156, pages 979–983. Trans Tech Publ.

[Drucker et al., 1997] Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., and Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161.

[Duan et al., 2016] Duan, P., Mao, G., Zhang, C., and Wang, S. (2016). Starima-based traffic prediction with time-varying lags. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 1610–1615. IEEE.

[Dunne and Ghosh, 2013] Dunne, S. and Ghosh, B. (2013). Weather Adaptive Traffic Prediction Using Neurowavelet Models. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):370–379.

[El-Basyouny and Sayed, 2009] El-Basyouny, K. and Sayed, T. (2009). Collision prediction models using multivariate poisson-lognormal regression. *Accident Analysis & Prevention*, 41(4):820–828.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Faghmous and Kumar, 2014] Faghmous, J. H. and Kumar, V. (2014). A big data guide to understanding climate change: The case for theory-guided data science. *Big data*, 2(3):155–163.

[Fei et al., 2011] Fei, X., Lu, C.-C., and Liu, K. (2011). A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction. *Transportation Research Part C: Emerging Technologies*, 19(6):1306–1318.

[Fernández et al., 2007] Fernández, S., Graves, A., and Schmidhuber, J. (2007). An application of recurrent neural networks to discriminative keyword spotting. In *International Conference on Artificial Neural Networks*, pages 220–229. Springer.

[Francis, 1961] Francis, J. G. (1961). The qr transformation a unitary analogue to the lr transformationpart 1. *The Computer Journal*, 4(3):265–271.

[Galton, 1886] Galton, F. (1886). Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263.

[Gao et al., 2019] Gao, Z., Yang, X., Zhang, J., Lu, H., Xu, R., and Diao, W. (2019). Redundancy-reducing and holiday speed prediction based on highway traffic speed data. *IEEE Access*, 7:31535–31546.

[Gardner et al., 1980] Gardner, G., Harvey, A. C., and Phillips, G. D. (1980). Algorithm as 154: An algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of kalman filtering. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(3):311–322.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Guo et al., 2014] Guo, J., Huang, W., and Williams, B. M. (2014). Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, 43:50–64.

[Habtemichael and Cetin, 2016] Habtemichael, F. G. and Cetin, M. (2016). Short-term traffic flow rate forecasting based on identifying similar traffic patterns. *Transportation Research Part C: Emerging Technologies*, 66:61–78.

[Hadayeghi et al., 2010] Hadayeghi, A., Shalaby, A. S., and Persaud, B. N. (2010). Development of planning level transportation safety tools using geographically weighted poisson regression. *Accident Analysis & Prevention*, 42(2):676–688.

[Hamed et al., 1995] Hamed, M. M., Al-Masaeid, H. R., and Said, Z. M. B. (1995). Short-term prediction of traffic volume in urban arterials. *Journal of Transportation Engineering*, 121(3):249–254.

[Hochreiter, 1991] Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91:1.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Holt Charles, 1957] Holt Charles, C. (1957). Forecasting trends and seasonal by exponentially weighted averages. *International Journal of Forecasting*, 20(1):5–10.

[Hong et al., 2011a] Hong, W.-C., Dong, Y., Zheng, F., and Lai, C.-Y. (2011a). Forecasting urban traffic flow by svr with continuous aco. *Applied Mathematical Modelling*, 35(3):1282–1291.

[Hong et al., 2011b] Hong, W.-C., Dong, Y., Zheng, F., and Wei, S. Y. (2011b). Hybrid evolutionary algorithms in a svr traffic flow forecasting model. *Applied Mathematics and Computation*, 217(15):6733–6747.

[Huang et al., 2006] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501.

[Hyndman and Athanasopoulos, 2014] Hyndman, R. J. and Athanasopoulos, G. (2014). *Forecasting: principles and practice*. OTexts.

[Hyndman and Athanasopoulos, 2018] Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

[Hyndman et al., 2019] Hyndman, R. J., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., and Razbash, S. (2019). Package forecast.

[Hyndman et al., 2007] Hyndman, R. J., Khandakar, Y., et al. (2007). *Automatic time series forecasting: the forecast package for R*. Number 6/07. Monash University, Department of Econometrics and Business Statistics.

[Ishak et al., 2003] Ishak, S., Kotha, P., and Alecsandru, C. (2003). Optimization of dynamic neural network performance for short-term traffic prediction. *Transportation Research Record: Journal of the Transportation Research Board*, (1836):45–56.

[Jia et al., 2017] Jia, Y., Wu, J., and Xu, M. (2017). Traffic Flow Prediction with Rainfall Impact Using a Deep Learning Method. *Journal of Advanced Transportation*, 2017.

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics.

[Kalman et al., 1960] Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

[Kamarianakis and Prastacos, 2005] Kamarianakis, Y. and Prastacos, P. (2005). Space–time modeling of traffic flow. *Computers & Geosciences*, 31(2):119–133.

[Karlaftis and Vlahogianni, 2011] Karlaftis, M. G. and Vlahogianni, E. I. (2011). Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, 19(3):387–399.

[Karpatne et al., 2017] Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V. (2017). Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Koesdwiady et al., 2016] Koesdwiady, A., Soua, R., and Karray, F. (2016). Improving Traffic Flow Prediction With Weather Information in Connected Cars: A Deep Learning Approach. *IEEE Transactions on Vehicular Technology*, 65(12):9508–9517.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kublanovskaya, 1962] Kublanovskaya, V. N. (1962). On some algorithms for the solution of the complete eigenvalue problem. *USSR Computational Mathematics and Mathematical Physics*, 1(3):637–657.

[Levin and Tsao, 1980] Levin, M. and Tsao, Y.-D. (1980). On forecasting freeway occupancies and volumes (abridgment). *Transportation Research Record*, (773).

[Li et al., 2013] Li, Z., Wang, W., Liu, P., Bigham, J. M., and Ragland, D. R. (2013). Using geographically weighted poisson regression for county-level crash modeling in california. *Safety science*, 58:89–97.

[Lin et al., 2009] Lin, S.-L., Huang, H.-Q., Zhu, D.-Q., and Wang, T.-Z. (2009). The application of space-time arima model on traffic flow forecasting. In *Machine Learning and Cybernetics, 2009 International Conference on*, volume 6, pages 3408–3412. IEEE.

[Lingras and Mountford, 2001] Lingras, P. and Mountford, P. (2001). Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 290–299. Springer.

[Lippi et al., 2013] Lippi, M., Bertini, M., and Frasconi, P. (2013). Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):871–882.

[Liu et al., 2006] Liu, H., van Zuylen, H., van Lint, H., and Salomons, M. (2006). Predicting urban arterial travel time with state-space neural networks and kalman filters. *Transportation Research Record: Journal of the Transportation Research Board*, (1968):99–108.

[Lv et al., 2015] Lv, Y., Duan, Y., Kang, W., Li, Z., and Wang, F.-Y. (2015). Traffic Flow Prediction with Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873.

[Ma et al., 2008] Ma, J., Kockelman, K. M., and Damien, P. (2008). A multivariate poisson-lognormal regression model for prediction of crash counts by severity, using bayesian methods. *Accident Analysis & Prevention*, 40(3):964–975.

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

[Maas et al., 2013] Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

[Masters and Luschi, 2018] Masters, D. and Luschi, C. (2018). Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*.

[McElhoe, 1966] McElhoe, B. A. (1966). An assessment of the navigation and course corrections for a manned flyby of mars or venus. *IEEE Transactions on Aerospace and Electronic Systems*, (4):613–623.

[Miller, 2018] Miller, J. A. (2018). Introduction to scalation.

[Miller and Baramidze, 2005] Miller, J. A. and Baramidze, G. (2005). Simulation and the semantic web. In *Proceedings of the 37th conference on Winter simulation*, pages 2371–2377. Winter Simulation Conference.

[Miller et al., 2017] Miller, J. A., Peng, H., and Cotterell, M. E. (2017). Adding support for theory in open science big data. In *Big Data (BigData Congress), 2017 IEEE International Congress on*, pages 251–255. IEEE.

[Min and Wynter, 2011] Min, W. and Wynter, L. (2011). Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616.

[Min et al., 2009] Min, X., Hu, J., Chen, Q., Zhang, T., and Zhang, Y. (2009). Short-term traffic flow forecasting of urban network based on dynamic starima model. In *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pages 1–6. IEEE.

[Min et al., 2010] Min, X., Hu, J., and Zhang, Z. (2010). Urban traffic network modeling and short-term traffic flow forecasting based on gstarima model. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1535–1540. IEEE.

[Mori et al., 2015] Mori, U., Mendiburu, A., Álvarez, M., and Lozano, J. A. (2015). A review of travel time estimation and forecasting for advanced traveller information systems. *Transportmetrica A: Transport Science*, 11(2):119–157.

[Mozer, 1995] Mozer, M. C. (1995). A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications*, page 137.

[Neilson et al., 2019] Neilson, A., Daniel, B., Tjandra, S., et al. (2019). Systematic review of the literature on big data in the transportation domain: Concepts and applications. *Big Data Research*.

[Okawa et al., 2017] Okawa, M., Kim, H., and Toda, H. (2017). Online traffic flow prediction using convolved bilinear poisson regression. In *Mobile Data Management (MDM), 2017 18th IEEE International Conference on*, pages 134–143. IEEE.

[Okutani and Stephanedes, 1984] Okutani, I. and Stephanedes, Y. J. (1984). Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological*, 18(1):1–11.

[Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Peng et al., 2018] Peng, H., Bobade, S. U., Cotterell, M. E., and Miller, J. A. (2018). Forecasting traffic flow: Short term, long term, and when it rains. In *International Conference on Big Data*, pages 57–71. Springer.

[Peng and Miller, 2019] Peng, H. and Miller, J. A. (2019). Multi-step short term traffic flow forecasting using temporal and spatial data. In *International Conference on Big Data*, pages 110–124. Springer.

[Richardson and Domingos, 2006] Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine learning*, 62(1-2):107–136.

[Robinson and Fallside, 1987] Robinson, A. and Fallside, F. (1987). *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Sak et al., 2015] Sak, H., Senior, A., Rao, K., Beaufays, F., and Schalkwyk, J. (2015). Google voice search: faster and more accurate. *Google Research blog*.

[Salamanis et al., 2015] Salamanis, A., Meladianos, P., Kehagias, D., and Tzovaras, D. (2015). Evaluating the effect of time series segmentation on starima-based traffic prediction model. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 2225–2230. IEEE.

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

[Schwarz et al., 1978] Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

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

[Shekhar and Williams, 2008] Shekhar, S. and Williams, B. (2008). Adaptive Seasonal Time Series Models for Forecasting Short-Term Traffic Flow. *Transportation Research Record: Journal of the Transportation Research Board*, (2024):116–125.

[Sims, 1980] Sims, C. A. (1980). Macroeconomics and reality. *Econometrica: journal of the Econometric Society*, pages 1–48.

[Smith et al., 1962] Smith, G. L., Schmidt, S. F., and McGee, L. A. (1962). *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration.

[Smola and Schölkopf, 2004] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

[Stathopoulos and Karlaftis, 2003] Stathopoulos, A. and Karlaftis, M. G. (2003). A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C: Emerging Technologies*, 11(2):121–135.

[Sun et al., 2003] Sun, H., Liu, H. X., Xiao, H., He, R. R., and Ran, B. (2003). Short Term Traffic Forecasting Using the Local Linear Regression Model. In *82nd Annual Meeting of the Transportation Research Board, Washington, DC*.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Tan et al., 2009] Tan, M.-C., Wong, S. C., Xu, J.-M., Guan, Z.-R., and Zhang, P. (2009). An aggregation approach to short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):60–69.

[Tang et al., 2017] Tang, J., Liu, F., Zou, Y., Zhang, W., and Wang, Y. (2017). An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2340–2350.

[Tsirigotis et al., 2012] Tsirigotis, L., Vlahogianni, E. I., and Karlaftis, M. G. (2012). Does Information on Weather Affect the Performance of Short-Term Traffic Forecasting Models? *International Journal of Intelligent Transportation Systems Research*, 10(1):1–10.

[Van Der Voort et al., 1996] Van Der Voort, M., Dougherty, M., and Watson, S. (1996). Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies*, 4(5):307–318.

[van Hinsbergen et al., 2009] van Hinsbergen, C. I., Van Lint, J., and Van Zuylen, H. (2009). Bayesian training and committees of state-space neural networks for online travel time prediction. *Transportation Research Record*, 2105(1):118–126.

[Van Lint et al., 2002] Van Lint, J., Hoogendoorn, S., and Van Zuylen, H. (2002). Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks. *Transportation Research Record: Journal of the Transportation Research Board*, (1811):30–39.

[Van Lint et al., 2005] Van Lint, J., Hoogendoorn, S., and van Zuylen, H. J. (2005). Accurate freeway travel time prediction with state-space neural networks under missing data. *Transportation Research Part C: Emerging Technologies*, 13(5-6):347–369.

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE.

[Vlahogianni and Karlaftis, 2013] Vlahogianni, E. and Karlaftis, M. (2013). Testing and comparing neural network and statistical approaches for predicting transportation time series. *Transportation Research Record: Journal of the Transportation Research Board*, (2399):9–22.

[Vlahogianni et al., 2004] Vlahogianni, E. I., Golias, J. C., and Karlaftis, M. G. (2004). Short-term traffic forecasting: Overview of objectives and methods. *Transport reviews*, 24(5):533–557.

[Vlahogianni et al., 2014] Vlahogianni, E. I., Karlaftis, M. G., and Golias, J. C. (2014). Short-term traffic forecasting: Where we are and where were going. *Transportation Research Part C: Emerging Technologies*, 43:3–19.

[Waibel et al., 1990] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1990). Phoneme recognition using time-delay neural networks. In *Readings in speech recognition*, pages 393–404. Elsevier.

[Wang et al., 2016a] Wang, J., Gu, Q., Wu, J., Liu, G., and Xiong, Z. (2016a). Traffic speed prediction and congestion source exploration: A deep learning method. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 499–508. IEEE.

[Wang et al., 2016b] Wang, J., Tsapakis, I., and Zhong, C. (2016b). A space–time delay neural network model for travel time prediction. *Engineering Applications of Artificial Intelligence*, 52:145–160.

[Werbos, 1988] Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356.

[Werbos et al., 1990] Werbos, P. J. et al. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[Whittaker et al., 1997] Whittaker, J., Garside, S., and Lindveld, K. (1997). Tracking and predicting a network traffic process. *International Journal of Forecasting*, 13(1):51–61.

[Williams and Hoel, 2003] Williams, B. M. and Hoel, L. A. (2003). Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. *Journal of transportation engineering*, 129(6):664–672.

[Wu et al., 2014] Wu, C.-J., Schreiter, T., and Horowitz, R. (2014). Multiple-clustering armax-based predictor and its application to freeway traffic flow prediction. In *American Control Conference (ACC), 2014*, pages 4397–4403. IEEE.

[Wu and Tan, 2016] Wu, Y. and Tan, H. (2016). Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv preprint arXiv:1612.01022*.

[Wu et al., 2018] Wu, Y., Tan, H., Qin, L., Ran, B., and Jiang, Z. (2018). A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C: Emerging Technologies*, 90:166–180.

[Yang et al., 2019] Yang, B., Sun, S., Li, J., Lin, X., and Tian, Y. (2019). Traffic flow prediction using lstm with feature enhancement. *Neurocomputing*, 332:320–327.

[Yu et al., 2017] Yu, H., Wu, Z., Wang, S., Wang, Y., and Ma, X. (2017). Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors*, 17(7):1501.

[Yue et al., 2010] Yue, H., Jones, E. G., and Revesz, P. (2010). Local polynomial regression models for average traffic speed estimation and forecasting in linear constraint databases. In *Temporal Representation and Reasoning (TIME), 2010 17th International Symposium on*, pages 154–161. IEEE.

[Zen and Sak, 2015] Zen, H. and Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4470–4474. IEEE.

[Zeng et al., 2008] Zeng, D., Xu, J., Gu, J., Liu, L., and Xu, G. (2008). Short term traffic flow prediction using hybrid arima and ann models. In *Power Electronics and Intelligent Transportation System, 2008. PEITS'08. Workshop on*, pages 621–625. IEEE.

[Zhang and Kabuka, 2018] Zhang, D. and Kabuka, M. R. (2018). Combining Weather Condition Data to Predict Traffic Flow: A GRU Based Deep Learning Approach. *IET Intelligent Transport Systems*.

[Zhang et al., 2014] Zhang, Y., Zhang, Y., and Haghani, A. (2014). A hybrid short-term traffic flow forecasting method based on spectral analysis and statistical volatility model. *Transportation Research Part C: Emerging Technologies*, 43:65–78.

[Zhao et al., 2017] Zhao, Z., Chen, W., Wu, X., Chen, P. C., and Liu, J. (2017). Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75.

[Zhong et al., 2005] Zhong, M., Sharma, S., and Lingras, P. (2005). Short-term traffic prediction on different types of roads with genetically designed regression and time delay neural network models. *Journal of computing in civil engineering*, 19(1):94–103.

[Zou et al., 2015] Zou, Y., Hua, X., Zhang, Y., and Wang, Y. (2015). Hybrid short-term freeway speed prediction methods based on periodic analysis. *Canadian Journal of Civil Engineering*, 42(8):570–582.