Traffic speed prediction using big data enabled deep learning

by

Shuo Wang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Civil Engineering (Transportation Engineering)

Program of Study Committee:
Anuj Sharma, Co-Major Professor
Soumik Sarkar, Co-Major Professor
Peter Savolainen
Jing Dong
Chinmay Hegde

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

Copyright © Shuo Wang, 2018. All rights reserved.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Dr. Sharma for his support and guidance all these years; my co-major professor, Dr. Sarkar; and my committee members—Dr. Dong, Dr. Savolainen and Dr. Hegde—for their guidance throughout this research.

I would also like to thank my friends and colleagues who helped me during the research work at InTrans.

In addition, I would like to express my appreciation to my parents for their support. And special thanks to my wife, Tingting, who is always supporting me in my research and my life.

ABSTRACT

The objective of the proposed study is to predict traffic speeds at a route level so that the traffic management has a chance to operate proactively. A distributed file system and parallel computing platform is used to store the big data sets of statewide traffic and weather data in a fault-tolerant way and process the big data in a timely manner. Traffic speed prediction problem is studied at two levels, and two deep networks are proposed accordingly: a fully convolutional deep network for long-term speed prediction and a hybrid long short-term memory (LSTM) network for short-term speed prediction. The fully convolutional deep network utilizes both weather information and historical traffic speeds to make long-term traffic speed predictions, and a trained model can be transferred to predict traffic speed at any spatial-temporal scale. The hybrid LSTM network utilizes the previous traffic speeds on the current day as well as historical traffic speeds to make short-term speed predictions, and a trained model can be used to predict speeds at any timestamps ahead in a streaming fashion. The proposed long-term and short-term traffic speed prediction models can be combined as a multilayer decision supporting system to provide traffic management an opportunity to operate proactively.

CHAPTER 1. INTRODUCTION

1.1 Background

Transportation systems aim to move people and goods safely and efficiently; thus, the system mobility becomes a great concern for both agencies and road users. In urban traffic systems, congestion is a major harm to mobility, which can also result in a total of $78 billion of societal and energy costs in the United States (TTI, 2007). One way to alleviate the congestion is to provide accurate and reliable traffic information and predictions, which can help both traffic planners and travelers to change plans for an alternative route in advance. Therefore, there is a need for research on how to produce an accurate understanding and reliable forecasting of traffic conditions, especially traffic speed.

With lots of intelligent transportation system (ITS) technologies being applied, large-scale, multisourced and high-resolution traffic data are available to researchers. In Iowa, enormous data can be obtained statewide from multiple sources, including roadside radar sensors, the private sector, a road asset management system (RAMS), and a road weather information system (RWIS). How to understand those massive data in an efficient way becomes a critical research task.

In recent decades, the emergence of big data technology and successful implementation of neural network algorithms provide the opportunities to conduct data-driven research in transportation. Moreover, the innovation of a parallel computing platform makes graphics processing unit (GPU)-accelerated computing available, which brings deep learning algorithm into reality. Some deep learning algorithms, like convolutional neural network (CNN) and long short-term memory (LSTM), have the powerful ability to explore massive, heterogeneous data and make predictions. All these advances in artificial intelligence provide researchers the chance to learn the features from traffic data in a fast and deep way. Thus, predicting the traffic speed in both the short and long term through massive, heterogeneous traffic data could be improved by using big data and deep learning approach.

1.2 Research Motivation

Traffic speed predictions studied in this dissertation are to provide agencies and road users an estimation of roadway traffic speed in the coming future by using deep networks trained on large-scale historical data.

In traffic operations and management, heatmaps have been widely used to visualize traffic speeds at a route level. A traffic speed heatmap puts speed values in a two-dimensional array. The 2-D array typically arranges roadway sensors or monitored roadway segments along the vertical axis, expands the time along the horizontal axis, and colors each cell from green to red by the value of traffic speeds. A typical traffic speed heatmap is illustrated in Figure 1.1.

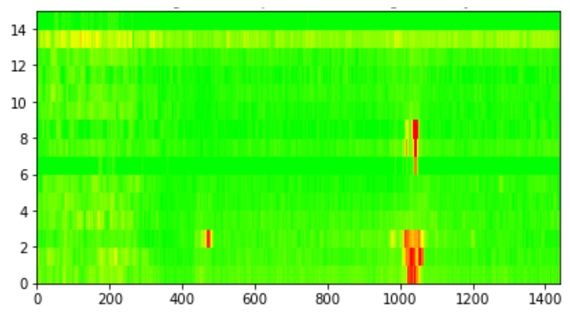


Figure 1.1 Sample speed heatmap

As shown in the illustration, traffic speed heatmaps are plots that can provide meaningful traffic speed visualization both spatially and temporally. Because of the matrix nature, the colorful 2-D plots of traffic speeds on a route level can be viewed as images. In a live traffic monitoring system, these traffic speed images are rolling along the temporal axis and the displayed images are always behind the current timestamps because of data processing and communication latency. In this context, traffic speed prediction is basically sneak-peeking the traffic speed images beyond the current timestamp.

From the traffic operation perspective, public agencies like the Department of Transportation (DOT) would be interested in traffic speed predictions at different temporal scales.

- **Long-term:** A good traffic speed estimation of the next day could make DOT operators well prepared and operational strategies can be planned in advance for the potential upcoming congestions.
- **Short-term:** A good traffic speed estimation of the next few minutes could give traffic operation officers an opportunity to prepare for the low-speed events proactively. Traffic incident management (TIM) may take this opportunity to relocate highway helpers in advance.

Thus, a good traffic speed prediction could provide traffic operations extra information as reference and potentially improve the reaction speed of TIM for incidents and enhance the mobility and safety of the roadway system overall.

Agencies would also be interested in traffic speed predictions at different spatial scales. The future traffic speed estimate at a certain sensor location would be useful for traffic monitoring at a targeted area, but a speed prediction of a roadway network can provide a better sense of a whole picture and thus be more useful.

Roadway users can benefit from the traffic speed prediction as well. The estimated travel speed in the next day can help drivers plan their travel time accordingly and choose an appropriate detour if applicable, and the network traffic loads can be more balanced. Similarly, an estimation of the travel speed in the next few minutes can keep the drivers both on the road and before leaving better informed and help them make better adjustments from both user optimal and system optimal perspectives, thus the overall traffic congestion level on the network can be alleviated.

1.3 Problem Statement

There are several challenges in predicting traffic speeds. First is efficiently managing the large-scale, high-resolution, and heterogeneous data. Traditional databases could deal with a certain amount of multivariate data but require excessive time and memory, which is a main obstacle in data-driven research.

The second is the high computational complexity requested by predicting traffic speeds. Traffic speeds on a network have both spatial and temporal correlations that make them challenging to predict. Although there have been lots of prediction models available, the traditional methods still lack the capability to deal with high computational complexity.

To provide accurate predictions to travelers and planners, both short-term and long-term predictions should be considered. Current methods tend to treat speed data as a time-series and have less prediction power, especially in long-term prediction. They also have limited ability in exploring heterogeneous data in a scalable manner. This research aims to solve these problems by answering the following research questions:

1. How to apply big data technologies to efficiently integrate massive traffic data?
2. How to predict short-term speeds at a route level?
3. How to predict long-term speeds at a route level?

1.4 Research Objective

According to the different use cases and different technologies required, this dissertation discusses the whole traffic prediction in two different temporal scales: (1) long-term prediction, which estimates traffic speeds in the next few hours or even the next day; and (2) short-term prediction, which estimates traffic speed in the next few minutes. In both long-term and short-term prediction cases, this dissertation applies deep neural network models that can be trained on historical data collected from various sensors with no manual labeling needed and estimate future traffic speeds at a route level. The research objectives are the following:

1. Develop database to store, integrate, and analyze big traffic data.

In this research, the raw streaming traffic data from roadside radar sensors (Wavetronix) and weather data provided by Iowa Environmental Mesonet (IEM) are downloaded and stored in a distributed way. A parallel processing method will be introduced and applied to those data.

2. Predict long-term speed to provide big picture traffic conditions to traffic planners.

The long-term speed prediction discussed in this dissertation is predicting the traffic speeds of the next few hours or the next day at a route level. The prediction model should take advantage of the large amount of historical data. The pretrained model should provide high transferability for deploying on different roadway networks and predict speeds in the desired time scale.

3. Predict short-term speed to provide advance traveler information.

The short-term speed prediction discussed in this dissertation is predicting the traffic speeds of the next few minutes at a route level. Unlike the long-term traffic speed prediction that usually is done once in a while, the short-term traffic speed prediction should be done in a streaming fashion.

1.5 Dissertation Organization

This dissertation consists of six chapters including an introduction. Chapter 2 presents a comprehensive literature review on current predictive analysis and artificial intelligence in transportation. Chapter 3 describes the database built on multiple data sources using big data technology. According to the complexity level of the methodology used, the long-term prediction will be discussed first then the short-term prediction will follow. Chapter 4 presents the prediction of long-term traffic speed at a route level using a fully convolutional deep network. Chapter 5 presents the prediction of short-term traffic speed at a route level using a hybrid LSTM network. Chapter 6 concludes the research by summarizing the findings and outlining the future work.

CHAPTER 2. LITERATURE REVIEW

This chapter presents a comprehensive literature review, focusing on the research related to traffic speed prediction and deep learning in transportation. Currently, the methods used for traffic speed prediction can be put into two general categories: statistical modeling and artificial intelligence (AI). Based on the categories, this section will review two kinds of past research on prediction and provide the background of methods employed in this study.

2.1 Statistic Modeling in Traffic Prediction

Statistical analysis has been the major method in the traffic speed prediction field for a long time, and it is still being widely used. Several kinds of methods, including fundamental traffic flow theory, time-series models, and Kalman Filter (KF), are discussed.

2.1.1 Traffic Flow Theory

Previously, many transportation researchers relied on the flow theory to explore different stages of traffic flow and make predictions. Most of them were focusing on volume prediction by estimating the origin-destination (OD) matrix on the network (Camus et al., 1994; Crittin and Bierlaire, 2002). There are also many studies using flow theory that were implemented in simulations. Besides traffic volume estimation, researchers also tried to represent the full traffic evolution with volume, density, and speed. Daganzo (1994) proposed a cell transmission model (CTM) to represent traffic evolution and further predict it over time and space.

Treating traffic flow as a stochastic process, some researchers have explored and modeled the stochastic characteristics in traffic flow to predict short-term speed. Qi and Ishak (2013) focused on urban freeways during peak hours and estimated the speed transition probabilities, from which the expected values were extracted and fitted using exponential models. However, this method tends to be affected by study location and time period, which makes it less scalable.

2.1.2 Time Series Model

Since the traffic data is time variant, time-series modeling has been widely used in traffic data analysis. One typical model, autoregressive integrated moving average (ARIMA), assuming stationarity and constant variance, is often adopted in traffic prediction research. Hamed et al. (1995) and Lee and Fambro (1999) have used the ARIMA model to predict traffic volume. Further, Williams et al. (1998) and Williams and Hoel (2003) explored the seasonality by using the seasonal ARIMA (SARIMA) model. Other researchers also tried to combine generalized autoregressive conditional heteroscedasticity (GARCH) with the ARIMA model to deal with the volatility in traffic conditions (Kamarianakis et al., 2005; Chen et al., 2011).

Several methods developed on the ARIMA model for speed prediction were also explored. Cetin and Comert (2006) have proposed an adaptive ARIMA model to accommodate regime change in traffic (such as free flow regime, congested regime, etc.). They implemented expectation maximization (EM) and cumulative sum (CUSUM) methods to detect the change in the mean of process, then applied the ARIMA model with adaptive parameters. Wang et al. (2014) combined empirical model decomposition (EMD) with ARIMA and created a hybrid EMD-ARIMA model. Emperical model decomposition decomposed the traffic data into intrinsic model functions (IMFs), then ARIMA models were applied on each of the IMFs, and finally the components were reconstructed to the predicted traffic speed.

To capture the spatial relationship existing in traffic flow, many researchers have been working on multivariate time-series analysis. Williams (2001) used the ARIMAX model to incorporate upstream volume as a transfer function input(s). Kamarianakis and Prastacos (2003) compared univariate and multivariate models for traffic prediction. Mainly they applied the ARIMA model, the vector autoregressive moving average (VARMA) model, and the space-time ARIMA (STARIMA) model to predict the speed on major arterials. The forecasting results showed a better performance in the multivariate model, which can cope with the interdependencies between speed from neighboring detectors. Pavlyuk (2017) also compared several multivariate models, such as VARMA, error correction model (VECM), STARIMA, and the multivariate autoregressive space state (MARSS) model for speed prediction. This study found the multivariate model can capture the spatial and temporal relationship in traffic flow and suggested simultaneous modeling on volume, speed, and occupancy to improve predictions.

Similar work in traffic speed predictions using multivariate models has also been done. Chandra and Al-Deek (2008, 2009) have used the vector autoregressive (VAR) model to account for the spatial relationship with taking two upstream and two downstream locations into consideration. And Zou et al. (2015) have proposed a hybrid model to estimate the speed series by periodic trend and residual part, with the assumption that speeds have a daily periodic trend on work days. They used the trigonometric regression function to estimate the periodic component and tried space time (ST), VAR, and ARIMA models to estimate the residual. The results showed a better performance in the hybrid ST model with different prediction steps.

Besides the ARIMA model, other multivariate methods were also explored. Ghosh et al. (2009) have proposed the structural time-series model (STM) with regard to multivariate data. This model separated components in a time-series and was used on signalized intersections to predict traffic volume. Another improvement in multivariate time-series analysis is made by Szeto et al. (2009). They embedded CTM in the SARIMA model to achieve volume prediction on a signalized network.

Time series models are widely used in traffic prediction; however, they are still constrained by the assumption of stationary process and the linear combination of previous observations. Moreover, compared to traffic volume, traffic speed has less obvious patterns and can be impacted by latent factors, which makes it hard to be predicted by traditional time-series analysis.

2.1.3 Kalman Filter

The KF utilizes the observed measurements and current estimated state to generate the estimation of a future state with statistical noises. This state space-based model can cope with multivariate data; thus, it has been applied in many traffic prediction studies. Okutani and Stephanedes (1984) first used KF to predict short-term traffic volume and found a better performance than the benchmark model UTCS-2. Further, Xie et al. (2007) improved the volume prediction by using discrete wavelet decomposition to denoise the raw data and get multilevel data series, then applying KF to estimate the volume. The results showed the proposed wavelet KF outperformed the direct KF. Another improvement on volume prediction was made by Guo et al. (2014). They proposed an adaptive KF to update the process variance by using the observation errors and state estimation errors to fine-tune the variances and implement the Kalman recursion.

Some researchers also focused on travel time prediction. They have used KF to estimate freeway travel time using different data sources such as toll tags (Chien and Kuchipudi, 2003; Chien et al., 2003), GPS data (Yang, 2005), and inductive loops (Xia, 2006). Further, Kuchipudi and Chien (2003).provided the flexibility in choosing path-based or link-based travel time estimation according to the traffic condition based on their previous research. Lint (2008) also proposed an extended KF (EKF) enabling online learning to predict freeway travel time. Other applications have also been done on network-level traffic prediction embedding KF (Whittaker et al., 1997; Wang et al., 2006).

Although there are lots of studies in traffic flow prediction using KF, few applications can be found in traffic speed prediction. Yang et al. (2004) have proposed an adaptive recursive least-squares method to predict traffic speed online. They used an autoregressive (AR) model to generate an offline estimate for the transition coefficient matrix and noise covariance matrix, and they made predictions with KF online. One problem in this study is the spatial relationship has not been explored, which exists in traffic propagation.

The KF can deal with multivariate time series and update the state continuously; however, it requires the knowledge of state transition and noise covariance, which are hard to determine in traffic flow.

2.2 Artificial Intelligence in Transportation

By moving toward AI, many more techniques and tools are revealed in front of traffic researchers and engineers. The adequate data also made it possible to implement machine learning and data mining techniques to make short-term and long-term predictions of traffic flow. This section reviews several shallow machine learning algorithms and the application in traffic flow prediction, such as k-nearest neighbor (kNN), support vector machine (SVM), and artificial neural network (ANN). With the emergence of deep learning, this powerful tool can also be used to predict traffic. Thus, this section will also emphasize deep learning algorithm applications in transportation and how they inspire this study.

2.2.1 Nearest Neighbor Algorithm

In data mining domain, a nonparametric, pattern matching technique, kNN algorithm, is commonly used. This method has also been adopted in transportation research. Smith and Demetsky (1996) used the nearest neighbor model to predict traffic volume in multiple intervals in advance on freeway segments. It had the advantages in scalability and ability of predicting relative long-term volume. Other than volume, Handley et al. (1998) have also proposed the kNN model to predict the travel time at multiple locations on a freeway. They examined and selected three nearest neighbor, different predictive features and normalization schemes, and from the results, they suggested that speed prediction is less accurate than volume during peak hours. Further, a hybrid model integrating geographic information system (GIS) and nonparametric regression has been developed to improve the prediction of travel time (You and Kim, 2000). In regard to multivariate traffic characteristics, Clark (2003) has predicted the traffic state by using the kNN model. The method was applied on one highway location with one month of data; it turned out to be a good prediction in flow and occupancy, but it did not perform well in speed prediction.

In previous literature, compared to traffic speed, the pattern of traffic volume—such as the peak hour pattern and seasonal pattern—seems easier to capture. Thus, this method is more often used in traffic volume prediction. Researchers have conducted many studies on predicting traffic flow (Smith et al., 2002; Zhang et al., 2013; Zhong and Ling, 2015). Efforts have also been made for a real-time system (Oswald et al. 2000; Smith and Oswald, 2003). Researchers have also tried different input variable structures to improve the prediction. Kindzerske and Ni (2007) have used a composite approach in the nearest neighbor search to predict traffic conditions. They tended to not use the whole network as input; instead, they only chose two upstream and two downstream sensors to form a local network to make predictions in order to minimize the error.

There is little research with a focus on traffic speed prediction using kNN. One was done by Yildirim and Cataltepe (2008), which used both kNN and SVM to predict traffic speed in 5-minute intervals. The results indicated SVM has a better performance than kNN. The other speed prediction study conducted by Chen et al. (2014) integrated Gaussian process regression with kNN. They used kNN to extract data features and input them to Gaussian process regression to achieve a reliable prediction.

Overall, this method is easy to implement and transfer using simple features extracted from historical data; however, it does not outperform the more advanced machine learning techniques in terms of prediction accuracy.

2.2.2 Support Vector Machine

The SVM method or support vector regression (SVR) is one of the most currently used machine learning techniques in traffic flow prediction. Wu et al. (2004) have used SVR to predict travel time on a highway network with three different kernel functions examined. The results performed better than the baseline prediction the authors selected. Other researchers also have applied SVM on travel time prediction and compared the root mean square error (RMSE) with other baseline models (Yu, Yang and Yao, 2017; Vanajakshi and Rilett, 2007).

Since SVM is a powerful tool in traffic prediction, there are many researchers who have added extra techniques to improve the model performance from solely using SVM. Asif et al. (2014) clustered the traffic data by unsupervised learning methods based on their spatiotemporal patterns before applying SVR, and it benefits the prediction results. One advantage of SVR is that it allows a kernel function to implicitly boost the input features up to a higher dimension, but choosing the right kernel function can be a trial-and-error process. Wang and Shi (2013) constructed a new kernel function in SVR using a wavelet function to capture the nonstationary characteristics of the short-term traffic speed data and report better performance over traditional SVR. Although SVR with kernel function overcomes the linear constraints, the model performance is highly relied on for the engineered input features.

Furthermore, there are many variants of SVR, such as least squares SVM (Zhang and Liu, 2009), v-SVM (Zhang and Xie, 2007), online-SVR (Castro-Neto et al., 2009), and seasonal SVR (Hong, 2011), which focused on different aspect of traffic flow prediction, such as travel time, traffic volume, etc. Also, to capture the spatial and temporal relationship in traffic flow, Li et al. (2016) have combined the ARIMA model and SVR to generate a hybrid strategy to predict highway volume in a more accurate and stable way. Besides those SVR variants with focus on prediction, Gopi et al. (2013) have proposed a Bayesian SVR to focus on the error variation of predicted traffic speeds.

Support vector machine can be a powerful tool in prediction; however, it mainly relies on the pattern of historical data and has shortcomings in exploring the temporal relationship, which is a nature of traffic speed data.

2.2.3 Artificial Neural Network

One great aspect of the neural network (NN) is that it can mimic any function by stacking tons of parameters in a simple way, and the useful features are automatically extracted through the training process. This method has been applied in traffic prediction increasingly in recent years.

One typical NN often refers to the back propagation NN (BPNN). The idea for back propagation is that inputs are fed forward to the network and then errors between output and target are propagated backwards through the network; meanwhile the weights are updated by some optimization technique. The BPNN was applied in traffic volume prediction as early as the late 90s (Kwon and Stephanedes, 1994; Smith and Demetsky, 1994; Yun et al., 1998). In terms of traffic speed prediction, Huang and Ran (2003) have investigated the traffic speed under adverse weather using BPNN for a single link. They used weather and traffic variables as inputs and built a fully connected NN. Lee et al. (2007) took the time of day and day of week into account. Other researchers also used the ANN to predict traffic speed and compared it with traditional ARIMA or a pattern matching model (Lee et al., 2006; Fabritiis et al., 2008; Park et al., 2011; Ye et al., 2012; Habtie et al., 2017).

Based upon the NN concept, much improvement has also been accomplished. One improvement is focusing on the network structure optimization. Researchers used a genetic algorithm to determine layer connection or nodes (Abdulhai et al., 1999; Lingras and Mountford, 2001; Wang et al., 2005; Vlahogianni et al., 2005). Some researchers also focused on preprocessing the traffic data to achieve a better prediction performance. Park and Rilett (1998) explored two clustering techniques—Kohonen self-organizing feature maps and fuzzy c-means model—to extract features from data before they were fed it into the NN. Jiang and Adeli (2005), Xie and Zhang (2006), and Boto-Giralda et al. (2010) used wavelet transform to denoise the data for network training and made predictions on traffic volume.

Inside the NN, some researchers also tried to replace the sigmoid function in hidden layers with radial basis function (RBF). They have employed this model to predict traffic volume and compared it with other methods (Amin et al., 1998; Park et al., 1998; Xie and Zhang, 2006; Zheng et al. 2006; Chen, 2017; Li et al. 2017). This kind of NN has a simple structure and is fast to learn. But the performance relies on the center and width parameter estimation of the RBF.

Different NNs could be applied on different traffic conditions. Park and Rilett (1998) proposed a modular NN by first clustering the data and then building a modular NN for each classification. This approach could estimate the entire function separately without the requirement of prior knowledge like other function approximation schemes. Similarly, Chen et al. (2001) used a self-organizing map (SOM) to cluster the traffic data and then used BPNN for each cluster. Lee (2009) used the k-means clustering method before predicting the travel time using ANN. This concept has also been adopted by other researchers (Yin et al., 2002; Coufal and Turunen, 2003; Tang et al., 2017). They tended to cluster the input data by fuzzy approach and apply the NN to each group with a similar traffic pattern.

To cope with the nature of time dependency in traffic data, many researchers have improved the NN structure with a recurrent feature. Some recurrent neural networks (RNNs) with context units to remember the previous output have been applied, such as the recurrent Jordan networks to forecast traffic volume (Yasdi, 1999), the recurrent state-space NN to predict travel time (Lint et al., 2002), a time-lagged recurrent network to predict short-term traffic speed (Dia, 2001). Ishak et al. (2003) explored three different networks for speed prediction—Jordan-Elman networks, partially recurrent networks, and time-lagged feedforward networks—and tried to optimize the network performance by using different input settings. Similar works have also been done by Zeng and Zhang (2013) and Jiang et al. (2016), which compared different RNN topologies on traffic speed and travel time prediction.

Some hybrid models have been explored by embedding statistical modeling in ANNs (Zeng et al., 2008; Moretti et al., 2015). Zheng et al. (2006) have also tried to combine two types of NN (BPNN and RBFNN) by using the Bayesian approach. The results showed it performed better than the single predictor. In addition, Alecsandru and Ishak (2004) have explored various topologies of the NN model to test the performance in traffic flow prediction.

The advantage in the ANN family algorithm is the capability in handling complex, nonlinear relationships in multivariate data. In terms of time-series prediction, however, the traditional NN is limited by short-term memory, which neglects the long-term trend that exists in a speed sequence. Also, a traditional ANN also has a limited structure due to the computational power constraint before GPU-accelerated computing was in place.

2.2.4 Deep Learning

2.2.4.1 Deep Learning Trend in Transportation Research

The deep learning concept was introduced back in the 80s and developed increasingly since the breakthrough in 2006 made the training fast and effective (Hinton and Salakhutdinov, 2006; Hinton et al., 2006). Transportation researchers have also adopted the deep learning algorithm in recent years. Some advanced deep models like the CNN and LSTM have been used for traffic prediction. Table 2.1 lists the number of literatures in deep learning (with CNN and LSTM highlighted) in the general transportation area and the traffic speed prediction with deep learning.

Table 2.1 Number of literatures in deep learning and speed prediction*

Key Words	**Google Scholar**	**TRID*** **
Deep Learning	111**	46
CNN	44**	41
LSTM	22**	15
Traffic Speed Prediction / w. Deep Learning	110 / 6	31 / 1
Traffic Speed Forecasting / w. Deep Learning	27 / 1	14 / 0

* As of 3/10/2018
** With additional key words "traffic" or "transportation"
*** TRID is an integrated database that combines the records from the Transportation Research Information Services (TRIS) database and International Transport Research Documentation (ITRD) Database. TRID provides access to more than one million records of transportation research worldwide.

Figure 2.1 also shows the trend of deep learning in transportation research. Notably, the peak is the last Transportation Research Board (TRB) annual meeting, with many researchers presenting their studies. Deep learning has gained more and more attention, which motivates this study in speed prediction as well.

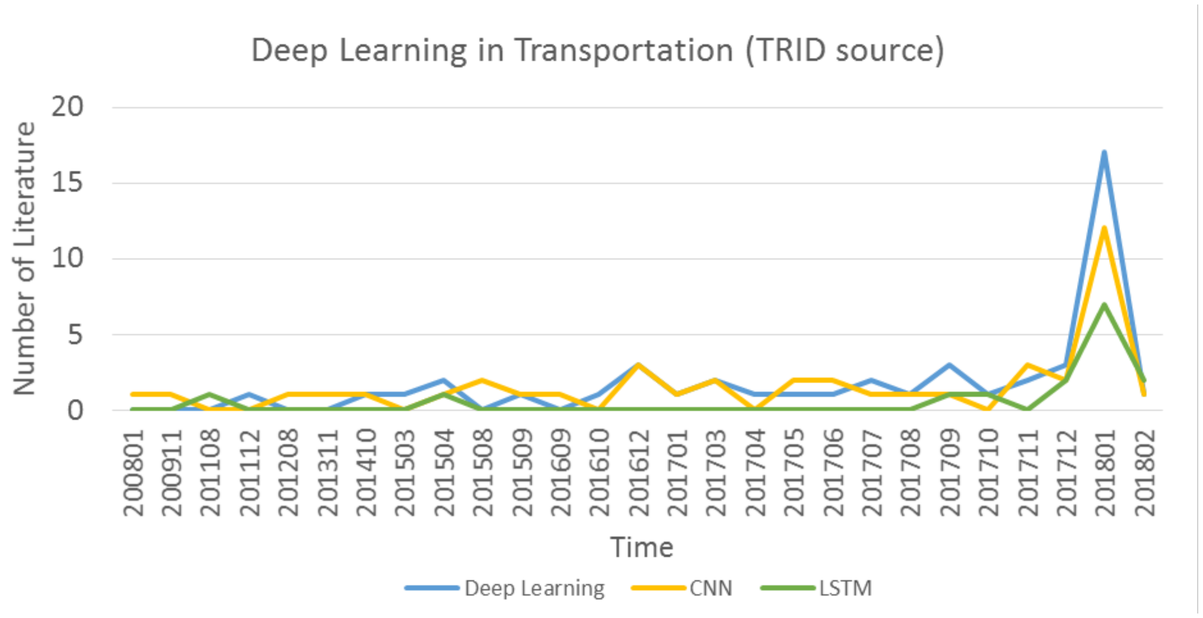


Figure 2.1 Deep learning trend in transportation research

2.2.4.2 Deep Belief Network

In traffic flow prediction, the first application is made by Huang et al. (2014) using deep belief networks (DBNs) to predict traffic volume from a single location and multiple locations (with multitask learning). Unlike the deep neural network (DNN), DBN has undirected connections in layers, which can be treated as stacked restricted Boltzmann machines (RBMs) and trained layer by layer as unsupervised. Huang et al. (2014) utilized DBN architecture and further added a multitask regression layer to predict volume with supervision. The results outperformed shallow machine learning models. Similar studies have also been done. Jia et al. (2016) used DBN to predict traffic speed; Tan et al. (2016) compared different RBM settings in the DBN for volume prediction. In addition, Lv et al. (2015) used stacked auto-encoders (SAE) with unsupervised greedy layer-wise training to predict traffic flow.

2.2.4.3 Deep Neural Network

Some researchers have worked on developing traditional ANN into deep learning. A simple DNN was applied by Yi et al. (2017) predicting the traffic performance index from speed data. Another structure adopted from image processing is CNN, which has the advantages in dealing with spatial relationship as it maintains the spatial correlation through convolution. By treating multiple location speed sequences as images, Ma et al. (2017) applied deep CNN to predict the speed image in the next 10 and 20 minutes. However, one important problem in this approach is that the network is not fully convolutional. The last several fully connected layers may potentially lose the structural dependencies maintained throughout the previous CNN layers, and the full connection limits the model inputs to a fixed size; thus a pretrained model cannot be directly applied on another roadway segment.

2.2.4.4 Deep Recurrent Neural Network

Due to the nature of time dependency in traffic data, some studies have been conducted with focuses on recurrent model structure. Ma et al. (2015a) combined RBM with RNN to predict a binary congestion condition on the roadway network. Wang et al. (2016) used CNN with an error-feedback recurrent layer to predict traffic speed. After using the convolution layer to extract the features, they applied a recurrent layer containing both regular neurons and error-feedback neurons to capture the incidents that can cause speed pattern change.

Although RNN aims to capture the pattern in time series, in practice it has a big problem—it is not good at capturing long-term dependencies. To overcome the long-term memory loss, a new structure has been developed. Long short-term memory has been introduced by Hochreiter and Schmidhuber (1997) and increasingly used in time-series prediction. Ma et al. (2015b) applied an LSTM model that is a special type of RNN for traffic speed prediction. With the speed, volume, and occupancy of one sensor as input and the speed in the next 2 minutes as output, the author demonstrated the efficiency of DNN, but a lot more potentials of DNN have not been excavated. Chen et al. (2016) used LSTM to classify and predict the categorized traffic conditions (congested, slow, free flow). Duan et al. (2016) trained LSTM models for each roadway segment using travel time series data, and they predicted a travel time vector four steps ahead. The LSTM model achieved relatively higher accuracy in first step prediction. However, they predicted each location on the same highway independently; thus the spatial dependencies are not utilized. Fu et al. (2016) applied both LSTM and a gated recurrent unites (GRU) model to predict traffic volume in 5-minute intervals. The GRU has a simpler structure than LSTM, which can potentially reduce the computation time. They randomly selected 50 locations from a traffic network and used previous 30-minute data to predict the next 5-minute volume. The results showed LSTM and GRU outperformed the traditional ARIMA model. Jia et al. (2017) integrated rainfall data to predict traffic speed using DBN and LSTM. To take the spatial correlation into account, Zhao et al. (2017) included the data from other locations as the input of the LSTM unit at the target location. They used traffic volume data in 5-minute intervals and predicted up to a 60-minute volume. In order to capture the backward temporal dependency, Cui et al. (2018) used a bidirectional LSTM model. They used speed data in 5-minute intervals and tested different numbers of steps in spatial and temporal input. The results indicated stacking one bidirectional LSTM layer and one unidirectional LSTM layer performed the best in the experiments compared to other architectures. One limitation is the lack of historical speed information as inputs so that the seasonal or periodical characteristics of traffic may not be captured. In addition, "backward temporal dependency" in the prediction problem itself may be arguably a nonvalid term, so that the need of bidirectional LSTM is questionable.

2.2.4.5 Hybrid Deep Learning

With the advance of CNN in spatial feature extraction and LSTM in temporal feature extraction, some studies have been conducted to learn the feature from spatiotemporal correlated traffic data by assembling two models. Wu and Tan (2016) combined CNN and LSTM to predict traffic flow. They proposed the model with one CNN layer, two LSTM layers, and one fully connected layer. The results showed a lower mean absolute error (MAE) compared to the single LSTM model. Similar work has also been done by Yu et al. (2017). They used CNN to learn the spatial features and LSTM to learn the temporal features. One improvement occurred when preparing the training data—they represented the road network in grid to retain the structure. Liu et al. (2018) also used CNN to extract features and ensemble LSTM to predict the travel time. They tested different settings of model architecture such as tuning the number of LSTM layers, number of DNN layers, etc. They proposed one CNN layer plus two LSTM layers plus two DNN layers is the best structure in the predictions on different horizons and using different sliding inputs.

Deep learning is prevailing in transportation research as we are getting large-scale, high-resolution, and multisource traffic data. Deep learning has the advantages in exploring dynamic and implicit correlation in data with less assumptions and prior knowledge. But deep networks need to be driven by a large amount of data and the impact of a certain feature could not be easily explained.

2.3 Summary

Numerous studies have been done in traffic prediction, from exploration in fundamental traffic flow theory, time series modeling, to artificial intelligence applications. Prediction is still a tough task to complete. In past research, most traffic prediction focuses on volume prediction rather than speed prediction. One reason could be traffic speed has less obvious trends (peak hour, weekday/weekend) than traffic volume. Also, volume is determined by traffic demand and supply; however, speed can be less sensitive to volume if demand is served. On the other hand, speed can be sensitive to other factors such as weather condition, incident occurrence, etc., that sometimes are unobserved. These properties in traffic speed result in a harder prediction.

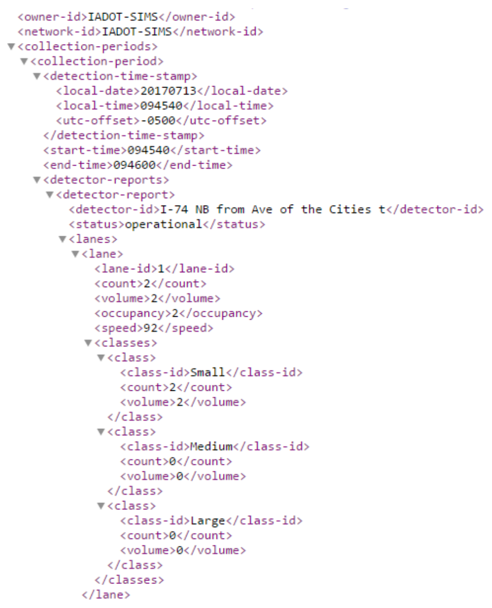
Among all the reviewed studies, a true long-term speed prediction like daily speed profile prediction is still not investigated. Forecasting daily speed profiles can benefit traffic planners in preparation for any weather hazards impacting traffic or for congestion relief measures planning.

CHAPTER 3. BIG DATA IN TRAFFIC DATABASE DEVELOPMENT

High-volume, high-resolution, heterogeneous traffic data can be obtained or accessed today by many transportation agencies. How to store, manage, and utilize these data is important. With the advent of big data technology, a solution is developed to efficiently manage the traffic database. In this chapter, the database developed for statewide traffic data management using big data techniques is discussed. The contents include (a) data acquisition, (b) big data storage and management, and (c) data preprocessing, particularly for this speed prediction research.

3.1 Data Acquisition

The Iowa DOT deploys more than 500 roadside radar sensors and 700 cameras, including both permanent and temporary versions statewide. Permanent sensors and cameras are typically located within major metropolitan areas in the state, while the temporary versions are commonly used at locations where a work zone is present. Traffic data collected by Wavetronix radar sensors stream to a web server, and the ITS vendor manages and disseminates them to the Iowa DOT via secured uniform resource locators (URLs). This implementation gives us the capability to perform real-time processing and application. Data are accessible in a not well-structured extensible markup language (XML) format, which requires a parsing program using more flexible languages than structured query languages (SQLs) in traditional database management. A sample of raw data we received is illustrated in Figure 3.1.

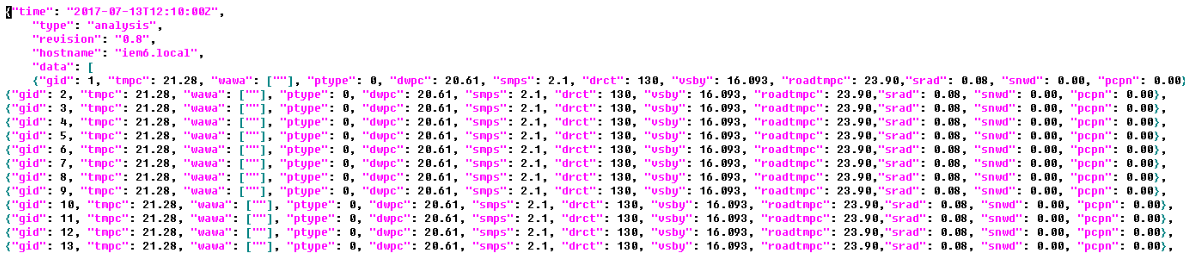


```
 <owner-id>IADOT-SIMS</owner-id>
 <network-id>IADOT-SIMS</network-id>
▼<collection-periods>
  ▼<collection-period>
    ▼<detection-time-stamp>
       <local-date>20170713</local-date>
       <local-time>094540</local-time>
       <utc-offset>-0500</utc-offset>
     </detection-time-stamp>
     <start-time>094540</start-time>
     <end-time>094600</end-time>
    ▼<detector-reports>
      ▼<detector-report>
         <detector-id>I-74 NB from Ave of the Cities t</detector-id>
         <status>operational</status>
        ▼<lanes>
          ▼<lane>
             <lane-id>1</lane-id>
             <count>2</count>
             <volume>2</volume>
             <occupancy>2</occupancy>
             <speed>92</speed>
            ▼<classes>
              ▼<class>
                 <class-id>Small</class-id>
                 <count>2</count>
                 <volume>2</volume>
               </class>
              ▼<class>
                 <class-id>Medium</class-id>
                 <count>0</count>
                 <volume>0</volume>
               </class>
              ▼<class>
                 <class-id>Large</class-id>
                 <count>0</count>
                 <volume>0</volume>
               </class>
             </classes>
           </lane>
```

Figure 3.1   Sample raw data from Wavetronix sensor

Traffic data collected include traffic volume, average speed, sensor occupancy, vehicle classification, and sensor status (operational, failed, off) every 20 seconds. Such high resolution and large-scale data requires big data tools to store and parallel processing to manage.

Along with Wavetronix data, weather information collected from the IEM in the observing networks is also available. Different from sensor data that are streaming, weather data are stored on a server every five minutes by the IEM team. A sample of raw weather data is illustrated in Figure 3.2. We download and parse the json file into a comma-separated value (CSV) file and store it on our local machine. Since the weather data are in five-minute resolution and cover the whole state of Iowa with small grids (rectangular longitude and latitude grids at a resolution of 0.01 degrees in both directions), it results in an enormously large amount of data that cannot be accessed by any traditional tools (more than 140 gigabytes [GB] per month). The weather information contains variables like temperature, precipitation, wind speed, etc. Selected variables are used for traffic speed prediction, which will be discussed in section 3.3.



```
{"time": "2017-07-13T12:10:00Z",
    "type": "analysis",
    "revision": "0.8",
    "hostname": "iem6.local",
    "data": [
    {"gid": 1, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 2, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 3, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 4, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 5, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 6, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 7, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 8, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 9, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 10, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 11, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 12, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
{"gid": 13, "tmpc": 21.28, "wawa": [""], "ptype": 0, "dwpc": 20.61, "smps": 2.1, "drct": 130, "vsby": 16.093, "roadtmpc": 23.90,"srad": 0.08, "snwd": 0.00, "pcpn": 0.00},
```

Figure 3.2   Sample raw weather data from IEM

Since the access methods for traffic and weather data are different, different data acquisition techniques and programs are implemented. Traffic data are downloaded and migrated to our file system through a real-time processing Java program and weather data are downloaded and parsed in Java in a batch processing. The traffic data acquisition program is written in multithread fashion. It is running on the high-performance cluster (HPC) to download the XML from the webpage every 20 seconds. The program allows the data downloading processes to be executed independently in parallel. This handles the potential timing-out issues in the web service connection and ensures that the data can be downloaded smoothly. The downloaded XML data is further parsed and then appended into a CSV formatted file for better data structure and storage saving.

3.2 Big Data Storage and Management

The streaming traffic data accumulate to more than 15 GB monthly and batch weather data are more than 140 GB monthly. An enormous amount of data requires a large-capacity, fault-tolerant, and fast-processing database. The database should hold all the data from multiple sources as their raw format so that no information is lost and new analysis can be potentially applied on any historical data at any time point. These directly lead to a big data tool for data storage and management.

3.2.1 Distributed File System (DFS)

A DFS is a model in which components located on networked computers communicate and coordinate their actions by passing messages. In other words, a DFS is a cluster of computers in which each computer within the cluster interacts and coordinates with each other to achieve a common goal as a whole. An illustration of the DFS structure is shown in Figure 3.3.

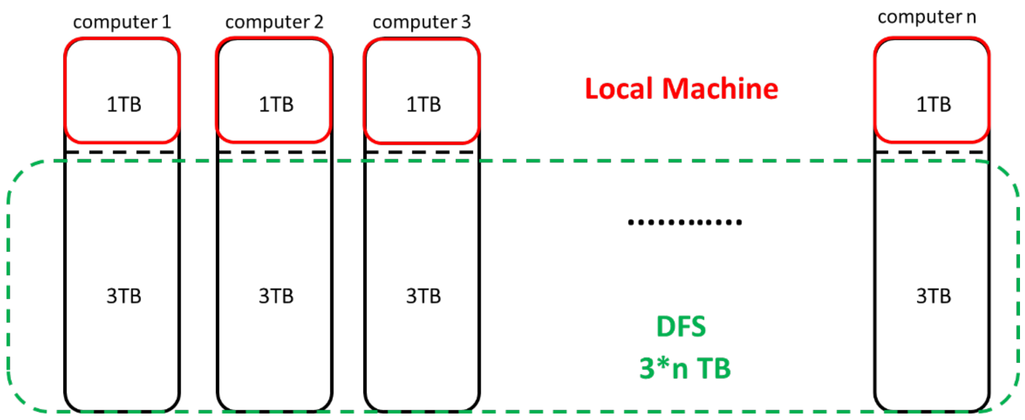


Figure 3.3 DFS structure

A DFS can be easily expanded by adding more computers into the cluster so that it is scalable and can handle big data sets. On a DFS, distributed processing can be applied on big data sets, which normal file systems can hardly handle. In distributed processing, a certain data processing job can be split into multiple components and all the components are executed in parallel by different processing units. With this distributed processing fashion, not only can a big data set be processed with limited amounts of memory, but the time it takes to process a certain job is significantly reduced.

3.2.2 Hadoop Distributed File System (HDFS)

There are various software frameworks used to operate a DFS and run distributed processing jobs on very large data sets. In our case, Apache Hadoop was selected because it is open source and can be installed on computer clusters built from commodity hardware. In an HDFS, a big dataset is split into multiple smaller chunks and each chunk is duplicated several times with each duplicate being stored on a different computer within the cluster. An illustration is shown in Figure 3.4. This way of splitting makes it possible to store a big data set with a size larger than the storage on a single computer and the duplication makes the system robust to hardware failures.

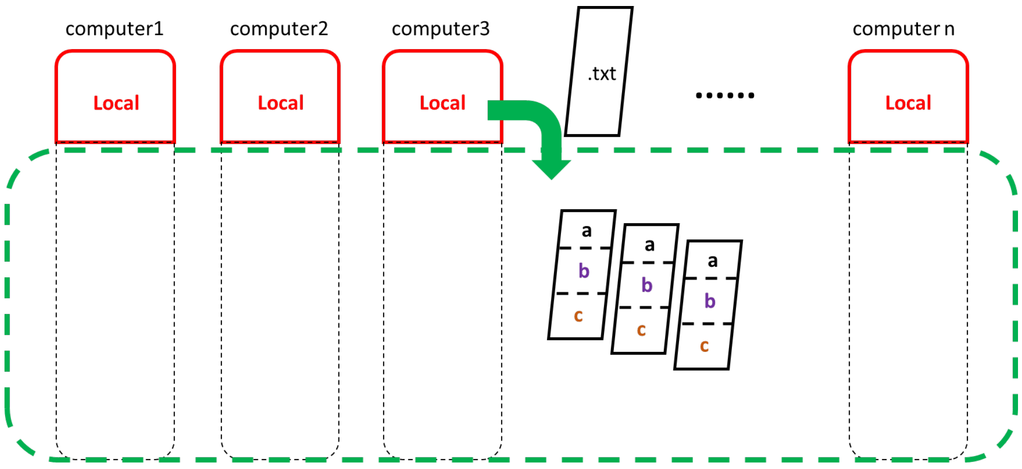


Figure 3.4   HDFS storage strategy

The HDFS itself controls the level of splitting and duplication to optimize the system performance and handles the addressing between chunks behind the scenes so the user is not bothered by these lower-level communications when using the HDFS. In this study, 14 computers with 6 terabyte (TB) storage each were built as an HPC. Half the storage from each computer was configured into the HDFS. The remaining half storage was preserved for local usage. The final configured HPC consists of 14 computers with 3 TB local storage on each and a 42 TB HDFS. An illustration is shown in Figure 3.5.

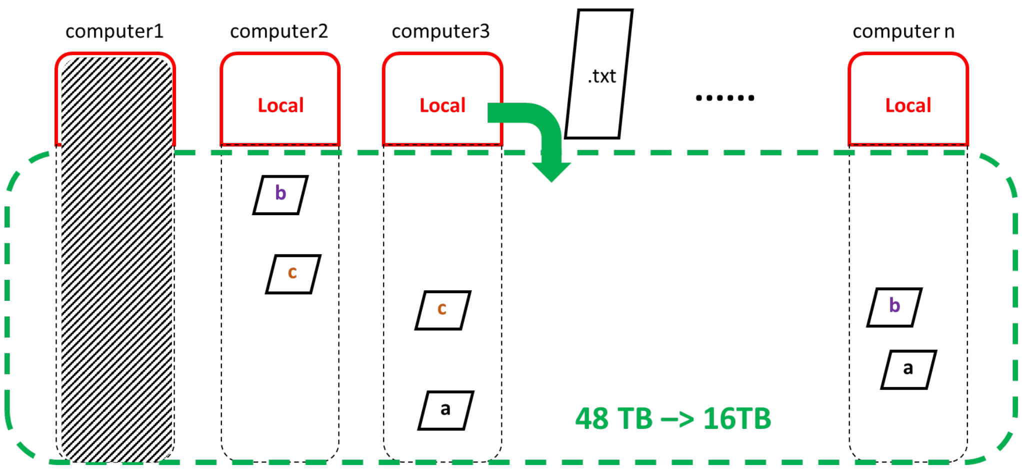


Figure 3.5 HDFS splitting and duplication

3.2.3 MapReduce Programming

MapReduce is the basic framework for distributed processing in an HDFS. It parcels out work to various processing units called nodes within the cluster, then organizes and reduces the results from each node into a cohesive answer to a query. MapReduce utilizes the key value pair to distribute the data as programmed. Different data but with the same key will dump into one reducer to process. A brief process inside MapReduce programing is shown in Figure 3.6. MapReduce works very efficiently in many frequently performed jobs of traffic data processing such as filtering, grouping, data aggregation, etc.

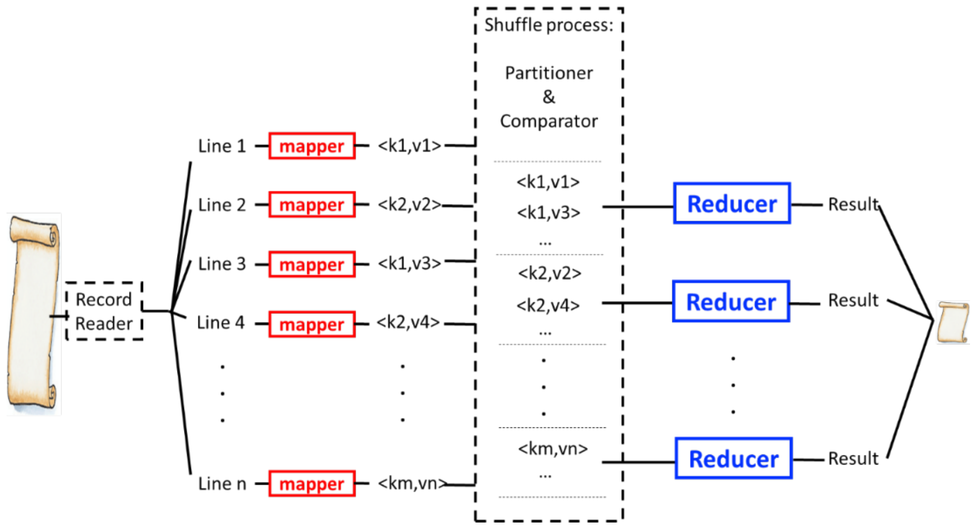


Figure 3.6 MapReduce programming process

There are more than 5 million records (rows) of traffic data in the raw CSV file of a single day. EXCEL is not capable of processing data of a single day because of the 1 million row limit, and MATLAB, R, or other traditionally used tools can be annoyingly slow. As a comparison, the distributed computing under the MapReduce framework is super-fast and scalable to larger datasets. In the HDFS, a sequence of filtering, grouping, and aggregating tasks can be done on one-day data in several seconds, on one-week data in a minute, and on one-year data in approximately half an hour.

In our study, the 20s raw data needed to be re-aggregated into larger time bins for model training. MapReduce programs were written in Java and executed to process the big datasets stored in the HDFS.

3.3 Preprocessing for Speed Prediction Analysis

For the speed prediction case study, the eastbound traffic data collected by 15 Wavetronix sensors on Interstate 235 (I-235) were used. Figure 3.7 displays the locations of the 15 traffic sensors. The 15 sensors covered an 11-mile-long corridor located in the center of the Des Moines metropolitan area. To cover this area, weather information from 15 corresponding grids were used. Traffic and weather data collected from September 2015 to the end of 2016, 447 days in total, were used for both the long-term and short-term traffic speed prediction case study.

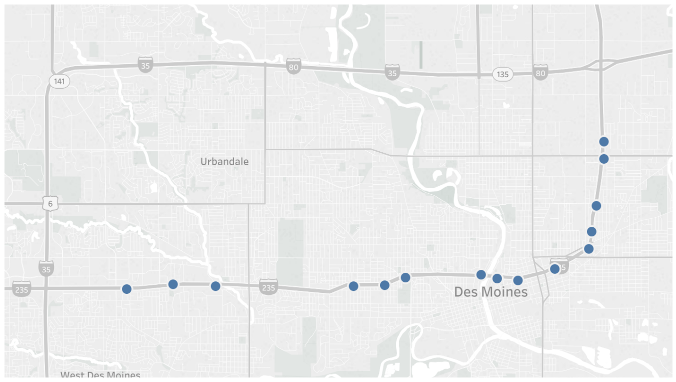


Figure 3.7 Locations of Wavetronix sensors along I-235 eastbound

The traffic and weather data are arranged in image-like 2-D arrays by time and location as described before. Because of connection failures or sensor errors, there are missing data as well as wrong data involved in the raw data collected directly from sensors. Therefore, data filtering and smoothing are needed before any analysis. Figure 3.8 shows a speed calendar plot of the raw traffic speeds re-aggregated in 1-minute intervals in March 2016.

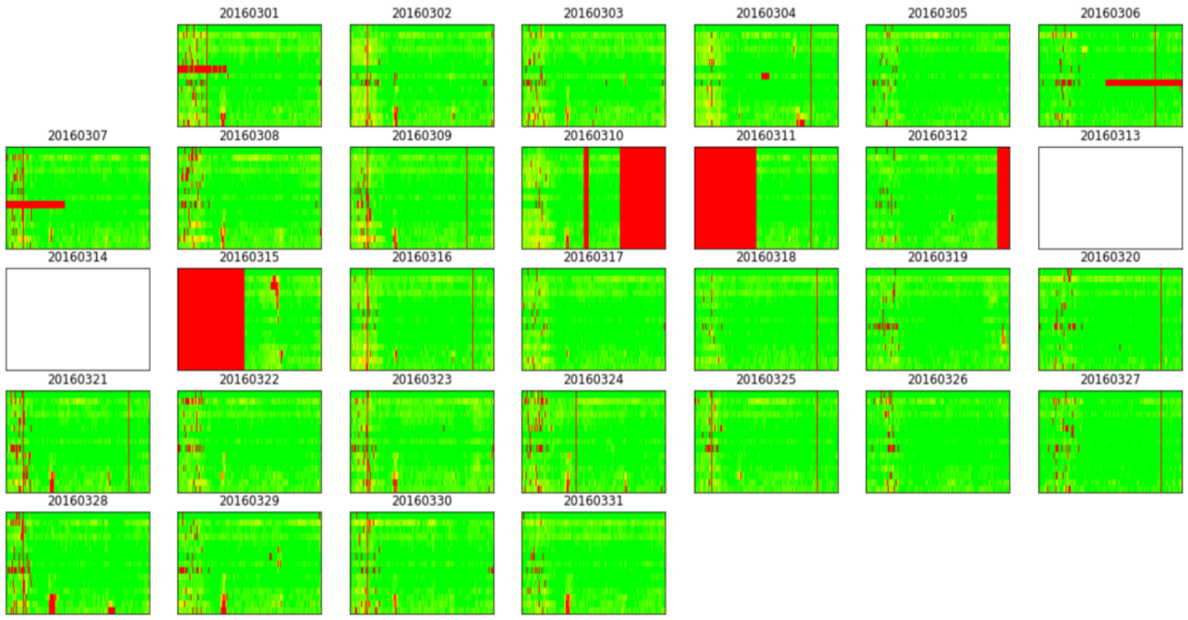


Figure 3.8 Raw speed data in 1-minute aggregation

Each subplot is a 1-day traffic speed heatmap with the 15 sensors ordered by location along the vertical axis and time of day along the horizontal axis. Each traffic speed heatmap is a 15x1440 2-D array. The red patches spreading a whole column are missing data usually caused by connection failures. The narrow red bands spreading horizontally are missing data due to a single sensor malfunction or temporal shut-down. Some other sparse red dots are possibly caused by sensor errors. A day is left blank if the whole day's data is completely missing. The data filtering and smoothing is performed in two steps.

1. Fill the missing data by the average of data from the same sensor, same time of day, and same day of week in other weeks.
2. Smooth the data of each sensor by moving the average with a 5-minute window size.

Note that the method for filling missing data used here may not be the best strategy. In addition to averaging by sensor location, time of day and day of week, averaging by more detailed weather conditions such as by snow versus no snow may better retain the dynamics in data and provided better modeling results. But considering the fact that the missing data is relatively a small potion (5.6%) and the conditional averaging strategy can go very deep itself involving study on significant impact factors, the data smoothing method used here sticks to the simple strategy.

The speed calendar plot of March 2016 after data filter and smoothing is shown in Figure 3.9. Besides traffic speeds, traffic information includes vehicle count and sensor occupancy as well. Weather information contains 9 variables listed in Table 3.1.

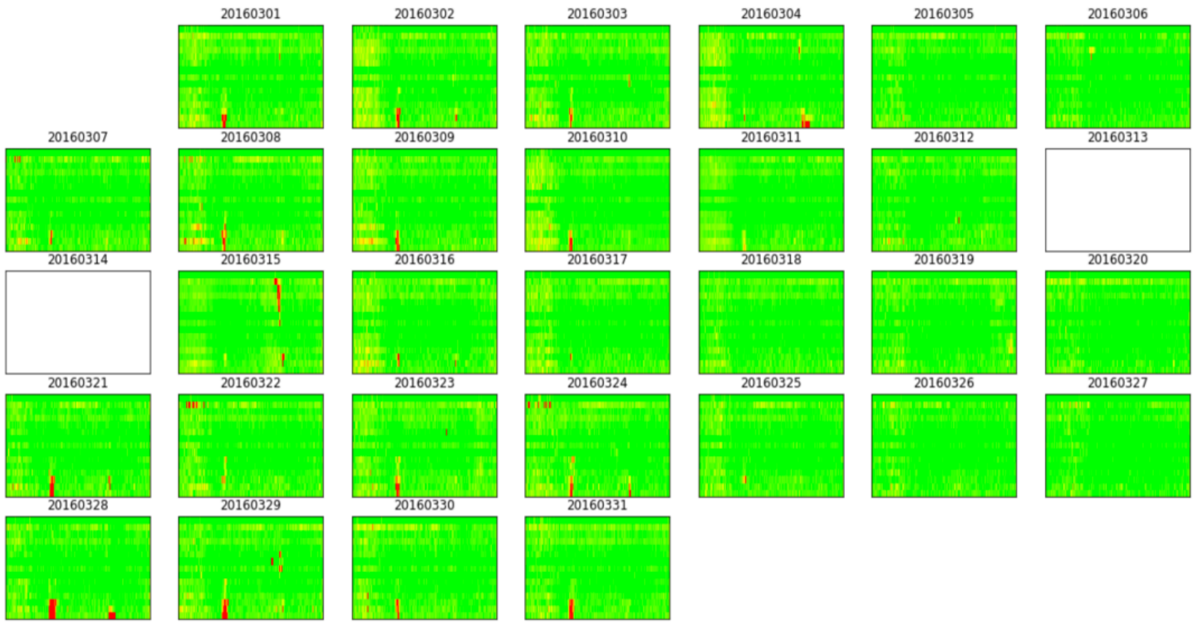


Figure 3.9 Smoothed speed data in 1-minute aggregation

Table 3.1 Weather variables description

Variable	**Description**	**Measure Details**
tmpc	Temperature	Two-meter above ground level air temperature. This value would be over a typical landscape for the location and not necessarily concrete, except in very urban areas. Units are Celsius.
dwpc	Dew point temperature	Two meters above ground level dew point temperature. As with "mpc," the same landscape assumptions apply. Units are Celsius.
smps	Wind speed	Ten meters above ground level wind speed. This speed does not include gusts, but is averaged over a couple-of-minutes period. Units are meters per second.
drct	Direction	Wind direction, where the wind is blowing from, at ten meters above ground level. Units are degrees from North.
vsby	Visibility	Horizontal visibility from automated sensors. Units are kilometers.
roadtmpc	Road temperature	Pavement surface temperature derived from available RWIS reports. These reports include both bridge and approach deck temperatures. Units are Celsius.
srad	Solar radiation	Photoactive global solar radiation, sometimes called "shortwave down". Units are watts per meter squared.
snwd	Snow depth	Snowfall depth analyzed once per day at approximately 7 AM local time. If the reported snowfall depth was zero at 7 AM and it started snowing at noon, this field would still be zero until it updated the next day at 7 AM. Units are millimeters.
pcpn	Precipitation	5-minute precipitation accumulation ending at the time of analysis. This is liquid equivalent. Snow and sleet are melted to derive this value. Units are millimeters accumulated in 5 minutes.

There are 5.6% of traffic data is missing and 0.0% of weather data is missing. Table 3.2 shows the summary statistics of the data after the preprocessing in the total 447 days.

Table 3.2 Statistic summary for all traffic and weather data

All Days	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Traffic Data				
Speed	3.93	90.75	65.61	6.07
Volume	0	165.6	27.67	22.93
Occupancy	0	38.73	2.26	3.25
Weather Data				
tmpc	-25	35.6	10.28	11.07
dwpc	-29	28	4.82	10.23
smps	0	17.5	3.99	2.63
drct	0	360	183.96	102.11
vsby	0	16.09	13.9	4.97
roadtmpc	-20.4	51.3	13.89	13.05
srad	0	960	133.23	216.69
snwd	0	152.4	6.87	22.31
pcpn	-36	1315.2	0.86	11.78

Three days are selected for testing purposes in both long-term and short-term predictions.

1. Test day 1: 12-02-2016 Friday. Nonrecurrent congestion on I-235 west end during PM peak hours. A comparison of average traffic speeds against test day 1 is shown in Figure 3.10. The data summary statistics of test day 1 are shown in Table 3.3.
2. Test day 2: 12-05-2016 Monday. Recurrent congestions on I-235 west end during both AM and PM peak hours and nonrecurrent congestion on I-235 east end during PM peak hours. A comparison of average traffic speeds against test day 2 is shown in Figure 3.11. The data summary statistics of test day 2 are shown in Table 3.4.

3. Test day 3: 12-16-2016 Friday. Nonrecurrent congestion on I-235 west end during AM peak hours and big congestion on the whole segment during PM peak hours. A comparison of average traffic speeds against test day 3 is shown in Figure 3.12. The data summary statistics of test day 3 are shown in Table 3.5.

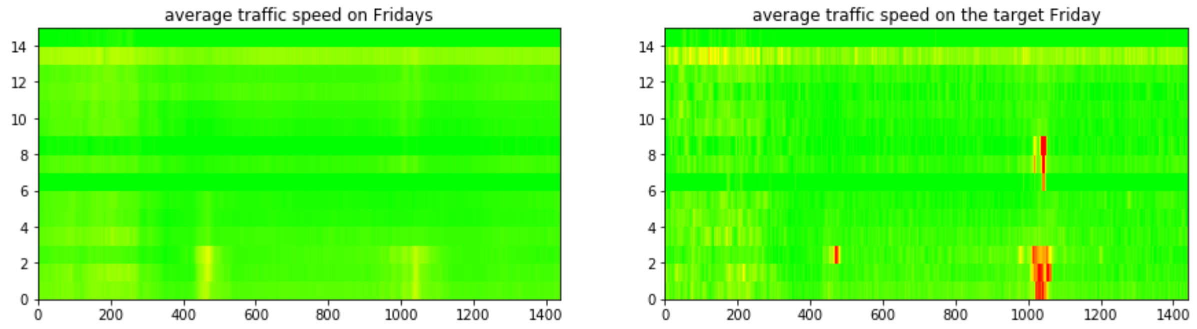


Figure 3.10 Speed data for test day 1.

Table 3.3 Statistic summary for test day 1

Test Day 1	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Traffic Data				
Speed	27.12	85.03	65.04	5.81
Volume	0	117.8	16.59	22.77
Occupancy	0	34.7	1.36	2.4
Weather Data				
tmpc	0	28	21.15	6.44
dwpc	0	20	15.98	4.44
smps	0	5.7	2.71	1.47
drct	0	300	150.28	45.39
vsby	0	16.09	14.96	4.03
roadtmpc	0	37.2	25.34	8.37
srad	0	593.4	123.32	189.1
snwd	0	0	0	0
pcpn	0	240	0.39	6.55

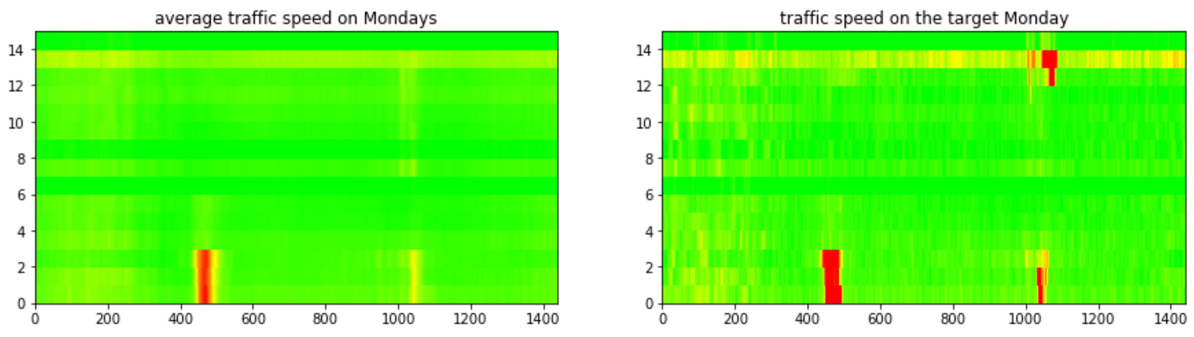


Figure 3.11 Speed data for test day 2

Table 3.4 Statistic summary for test day 2

Test Day 2	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Traffic Data				
Speed	22.38	83.49	65.32	6.01
Volume	0	121.6	28.83	24.22
Occupancy	0	16.68	2.12	1.73
Weather Data				
tmpc	0	25.6	20.15	5.63
dwpc	0	20	16.95	4.64
smps	0	5.1	1.98	1.55
drct	0	360	145.55	62.95
vsby	0	16.09	14.94	4.02
roadtmpc	0	31.2	22.5	6.65
srad	0	227.5	58.5	77.7
snwd	0	0	0	0
pcpn	0	823.2	2.75	29.32

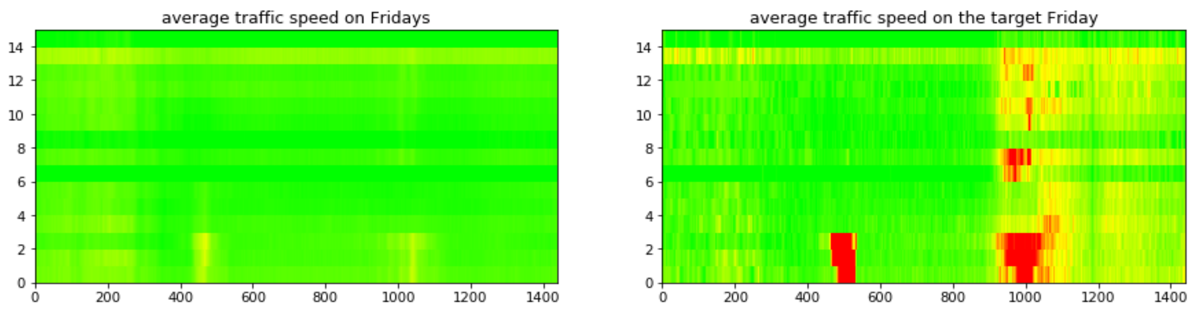


Figure 3.12 Speed data for test day 3

Table 3.5 Statistic summary for test day 3

Test Day 3	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Traffic Data				
Speed	21.11	83.9	65.05	6.7
Volume	0	126.8	30.23	24.53
Occupancy	0	13.3	2.14	1.68
Weather Data				
tmpc	0	26.1	20.27	5.82
dwpc	0	20	14.27	4.21
smps	0	6.2	2.79	1.45
drct	0	200	124.09	42.36
vsby	0	16.09	15.02	4.01
roadtmpc	0	31.4	23.18	6.91
srad	0	628.1	93.09	158.79
snwd	0	0	0	0
pcpn	0	14.4	0	0.22

CHAPTER 4. LONG-TERM SPEED PREDICTION USING ADVANCED CONVOLUTIONAL NEURAL NETWORK

4.1 Introduction

This chapter discusses the methodology for long-term speed prediction at a route level. Compared to the "short-term" in the next chapter, "long-term" here means "next few hours" or "the next day."

It has been explained earlier that traffic speed heatmaps of a roadway network over a certain period of time can be viewed as images (see section 1.2). The long-term speed prediction discussed here is predicting the traffic speed image of the next few hours or the next day as the output. Various studies have pointed out that weather is a big impact factor to traffic speeds and traffic incidents have a higher likelihood during adverse weather conditions. In addition, comparing to future traffic speeds, future weather information is easier to predict and the predicted weather information has been made widely available. Therefore, the predicted weather information will be used as input to predict future traffic speeds. Traffic also follows patterns. The traffic patterns will not change much from week to week on a certain roadway network. Considering that, the historical traffic speeds on the same roadway network should be used as input to predict future traffic speeds as well. Similar to the speed image as output, each piece of the input information can also be organized spatially and temporally in 2-D matrices and thus be viewed as images. The long-term traffic speed prediction then becomes a problem that needs to convert the image-like inputs to an image-like output. The problem can be formularized as the following.

$$Speed_{output} = f(Input_1, Input_2, \dots, Input_K) \quad (4.1)$$

where

$$Speed_{output} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & & s_{1n} \\ s_{21} & s_{22} & s_{23} & \cdots & s_{2n} \\ s_{31} & s_{32} & s_{33} & & s_{3n} \\ & \vdots & & \ddots & \vdots \\ s_{m1} & s_{m2} & s_{m3} & \cdots & s_{mn} \end{bmatrix}$$

$m = number\ of\ locations\ in\ the\ roadway\ networks$

$n = length\ of\ time\ window\ to\ predict$

$$Input_k = \begin{bmatrix} a_{11} & a_{12} & a_{13} & & a_{1n_k} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n_k} \\ a_{31} & a_{32} & a_{33} & & a_{3n_k} \\ & \vdots & & \ddots & \vdots \\ a_{m_k1} & a_{m_k2} & a_{m_k3} & \cdots & a_{m_kn_k} \end{bmatrix}, k = 1,2, \ldots, K$$

In this chapter, a fully convolutional deep network with encoder and decoder is used as the function f in the equations above to deal with long-term traffic speed prediction.

4.2 Methodology

To clearly explain the proposed fully convolutional deep network with encoder and decoder used in this chapter, the discussion needs to start from basic ANNs, then talk about the advantage of CNNs, then move to advanced uses of CNN, and finally explain the proposed network in details.

4.2.1 ANN

4.2.1.1 The Basic Building Block

There are many model families in the deep learning world. But no matter whether it is a fully connected network, CNN, or other complicated network such as variational auto-encoder generative adversarial networks (VAE-GAN) (Larsen et al., 2015), they all are rooted from the very basic ANNs. Before the term "artificial neural network" became dominant in today's modeling field, researchers were all very familiar with the statistical models such as linear regression and logistic regression. Actually, the linear regression model and logistic regression model can be viewed as ANNs with a single node as well. A single node of ANN is shown as a graph in Figure 4.1.

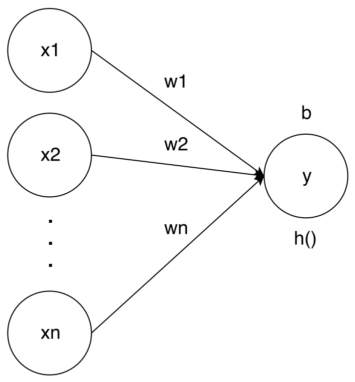


Figure 4.1 One-node ANN structure

As a formula, the output y can be written as:

$$y = h(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b) \tag{4.2}$$

or written in matrix format as:

$$y = h(WX + b) \tag{4.3}$$

where

$W = [w_1 \quad w_2 \quad w_3 \quad \ldots \quad w_n]$ *are the learnable weights*

b *is the learnable bias*

$X^T = [x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_n]$ *are the inputs*

h *is called the activation function*

y *is the output*

Now, if we recall the formula of linear regression:

$$y = WX + b \tag{4.4}$$

it can easily be related to ANN and treated as a one node network in ANN context with a special activation function:

$$h(x) = x \tag{4.5}$$

Similarly, if we recall the formula of logistic regression:

$$y = \frac{\exp(WX+b)}{1+\exp(WX+b)} = \frac{1}{1+\exp(-(WX+b))} \tag{4.6}$$

it can be viewed as a one node network in ANN context as well with a well-known sigmoid function as the activation function:

$$h(x) = sigmoid(x) = \frac{1}{1+\exp(-x)} \tag{4.7}$$

4.2.1.2 Understand the Capability of ANN

Now both linear regression and logistic regression can be graphically viewed as a one-node ANN shown in Figure 4.1, and the regular ANN is just a stack of lots of connected nodes similar to what is shown in Figure 4.2. Figure 4.2 shows a regular ANN with one output layer and three hidden layers and where each hidden layer has 10 nodes.

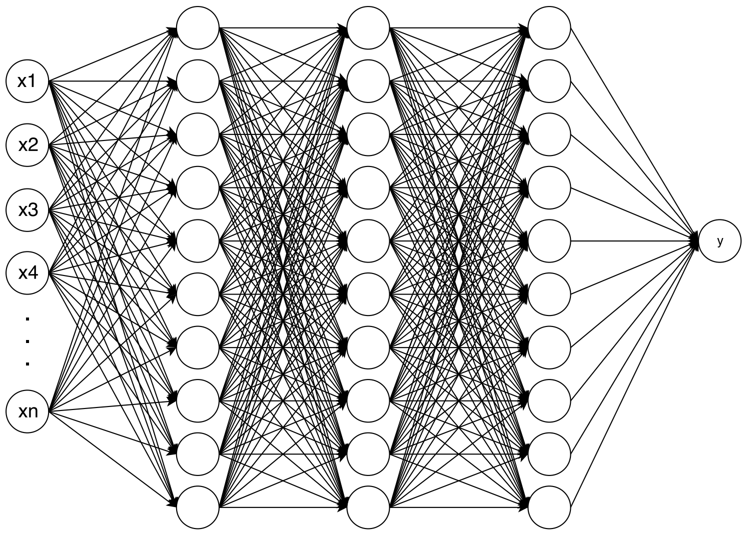


Figure 4.2 Fully connected deep ANN

In this typical ANN shown in Figure 4.2, each node functions exactly the same as the signal node case illustrated in Figure 4.1. If this deep ANN structure was used for modeling a problem instead of logistic regression, writing out all the actual formula becomes impracticable.

To better understand the capability of ANNs, we still use the three-layer ANN shown in Figure 4.2 vs. logistic regression as a comparison. Suppose in both cases we model the same problem and to be specific suppose we have ten inputs ($n = 10$), $x_1, x_2, \dots, x_{10}$, and one output, y. When modeling the mapping function f using logistic regression, we basically use $10 \times 1 = 10$ parameters, $w_1, w_2, \dots, w_{10}$, to capture the dependencies between the inputs and the output. As a comparison, when using the three-layer ANN shown in Figure 4.2, we use $10 \times 10 + 10 \times 10 + 10 \times 10 + 10 \times 1 = 320$ parameters to capture the dynamics. Not only do we use more parameters to model the problem, but also, because of the multiple layers and the nonlinear activation function in each layer, the network can capture higher level nonlinear dynamics than logistic regression can do. In the ANN context, we can easily build a larger and more powerful network by adding more layers or adding more nodes in each layer. In addition, the output is not necessarily restricted to one y, we can add more nodes in the output layer so that the network can provide multivariate outputs $y_1, y_2, \dots, y_m$.

4.2.1.3 How to Train an ANN

An artificial neural network is developed upon very simple building blocks and the theory behind it is that a stack of lots of simple functions can mimic any complex functions. By using simple nonlinear activation functions and multiple layers, the ANN network can model highly nonlinear dynamics. An ANN network is parameterized. Some existing successfully deployed networks contain millions of parameters. Those large networks are extremely powerful, but meanwhile how to find the right values for those millions of parameters (can also be called weights) is essential and difficult. Ultimately, finding the right values for those millions of parameters in ANN is nothing but an optimization problem and it is fundamentally exactly the same as finding the weights in a linear regression or a logistic regression.

4.2.1.3.1 Gradient descent

Again, let's talk about the general method of solving an optimization problem, and then illustrate using linear regression and logistic regression, and finally talk about ANNs. Most of the problems, if not all, can be modeled as optimization problems. For any problem, we always want to achieve some goals. To mathematically and strategically achieve the goal, we need to formularize some objective value regarding our goal and try to minimize or maximize it—this is an optimization problem. An optimization problem can be generally formularized as:

$$x = \underset{x}{\operatorname{argmin}} J(x) \tag{4.8}$$

where

$x\ is\ the\ parameter\ weights$

$J\ is\ the\ objective\ function, usually\ is\ a\ loss\ function$

The general method to find the optimal x so that the loss function $J(x)$ can be minimized is gradient descent. Gradient descent comes from a simple and intuitive ideal:

If I can find the direction in which changing x increases the value of $J(x)$, I can change x in the opposite direction to make $J(x)$ smaller. Then I can keep changing x in that opposite direction so that $J(x)$ can be minimized.

The direction "in which changing x increase[s] the value of $J(x)$" is called gradient, and the method "changing x in the opposite direction to make $J(x)$ smaller" is called gradient descent. Gradient descent is an iteratively updating process and can be generally formularized as:

$$x^{r+1} = x^r + \alpha^r d^r, r = 0,1,2, \ldots \tag{4.9}$$

where

$x^r \text{ is the weights at step } r$

$d^r \text{ is the weight changing direction}$

$\nabla f(x^r) d^r < 0, \text{if } \nabla f(x^r) d^r \neq 0$

$\nabla f(x^r) \text{ is the gradient with respect to } x^r$

$\alpha^r \text{ is a positive step size}$

Different strategies of choosing the descent direction d^r and the updating step size α^r yield to different solvers, and they are all variants of gradient descent. Now after the gradient decent has been formularized here, let’s recap how to solve linear regression and logistic regression. We all know the solution for linear regression is called “least squares,” which can be translated as “use mean square errors (MSE) as the loss function and minimize it.” The MSE loss function of linear regression is a quadratic function and to minimize a quadratic function there is a closed-form solution by setting the gradient to zero. The closed-form solution is essentially a one-step gradient descent update using the Quasi-Newton method. For logistic regression, the well-known solution “maximum likelihood” can be translated as “use negative log-likelihood as the loss function and minimize it.” Compared to linear regression, logistic regression is more difficult to solve because its objective function, negative log-likelihood, has a higher nonlinearity than a quadratic function such as MSE. Although logistic regression cannot be solved by a one-step gradient descent update, it can be solved by iterative gradient descent. And iterative gradient descent is guaranteed to find the global optimal solution because negative log-likelihood is a convex function. The aforementioned Quasi-Newton method is the most widely used strategy for solving logistic regression, which uses the inverse of the Hessian matrix as the descent direction as shown below.

$$x^{r+1} = x^r + \alpha^r (\nabla^2 f(x^r))^{-1} \nabla f(x^r), r = 0,1,2, \dots \qquad (4.10)$$

where $\nabla^2 f(x^r)$ is the second order derivitive in matrix format, aka the Hessian matrix.

Since the inversion of the Hessian matrix is hard to compute, some methods use an estimation of the Hessian matrix to make the computation more efficient. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and the limited-memory BFGS (L-BFGS) algorithm are two solvers falling in that category and are widely used in software packages such as NLogit.

4.2.1.3.2 Backpropagation and learning rate

As in linear regression and logistic regression, finding the parameter weights of ANN is also an optimization problem, also needs to set a loss function, and also needs to use gradient descent to minimize it. Given an input, there are two major steps to perform one iteration of weights update:

1. Calculate the gradient for all learnable weights in all layers.
2. Update the weights based on the calculated gradient.

To perform step 1, backpropagation comes into play. Backpropagation is nothing other than using the chain rule to calculate derivatives for a composite function. Consider a very simple ANN structure with one hidden layer and one node in the hidden layer as shown below in Figure 4.3.

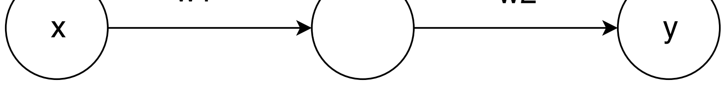


Figure 4.3 Simplified ANN for backpropagation illustration

Given input x and target y, the loss can be calculated as:

$$loss = J(h_2(h_1(x, W_1), W_2), y) \tag{4.11}$$

where

$J\ is\ the\ loss\ function$

$h_1\ is\ the\ activation\ function\ for\ hidden\ layer$

$h_2\ is\ the\ activation\ function\ for\ ouptut\ layer$

$W_1, W_2\ are\ learnable\ weights$

Then using the chain rule, the gradient with respect to W_1 and W_2 and be calculated as:

$$\frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial h_2}\frac{\partial h_2}{\partial W_2}$$

$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_1}$$

As demonstrated, the gradient can be calculated step by step from the output layer backwards to the first hidden layer. This method of gradient calculation is called backpropagation.

After getting the gradient, to perform step 2 we need to choose an updating direction and an updating step size for gradient descent as described earlier. Usually the updating direction is simply set as the opposite direction of gradient (the negative gradient direction). Gradient descent using the negative gradient direction is often called steepest gradient descent. The updating step size is also known as the learning rate. There are various ways of choosing the learning rate. A more aggressive strategy with a larger learning rate lowers the loss faster but may overshoot near the optimal solution so that it is harder to converge. A more conservative strategy with a smaller learning rate lowers the loss slower but can better converge. A smarter way of setting the learning rate is called learning rate decay, which makes the learning rate a decreasing function with respect to the iteration number r (as shown in Equation 4.9).

4.2.1.3.3 Activation function and batch

Activation functions provide the nonlinearity of the ANNs. We can choose different activation functions for each node in the networks or we can choose the same activation function for all nodes. Different activation functions provide different output value ranges and different gradient descent characteristics. As an example, the sigmoid function that is used by logistic regression takes any real-number input and outputs real-number value between 0 and 1. The rectified linear unit (ReLU) is the most widely used activation function in the current ANN meta. The ReLU is nothing but a max function:

$$f(x) = \max(0, x) \tag{4.12}$$

The ReLU has a huge computational advantage when performing backpropagation on a large ANN because of its simplicity. Meanwhile its nonlinear nature still provides ANN the capability to model highly nonlinear dynamics.

How to effectively train the weights is the key to develop a good ANN. Besides using ReLU and other activation functions, putting data samples in batch to train is another strategy from which training can benefit. Usually when training a model given a training dataset, the objective is to minimize the overall loss on all training samples. The loss function is defined on the entire training set X:

$$loss_{all} = J(W|X_{all}) \tag{4.13}$$

Then the gradient descent will guarantee reduction of the overall loss at each iteration. But in an ANN application, since there are so many weights in the network, usually it needs at least one order higher number of training samples than the number of weights to effectively train it. In that case, the very large training dataset usually cannot be fit in the memory all at once. Besides learning the training samples all at once, another way is to learn the training samples one at a time, called stochastic gradient descent. The stochastic gradient descent calculated based on the loss function given a random training sample is shown as:

$$loss_i = J(W|X_i) \tag{4.14}$$

Stochastic gradient descent assumes by updating the weights after seeing the training samples one at a time the overall loss can also be minimized. This assumption is true in the long run, but it is not guaranteed to decrease the overall loss at every iteration. Therefore, stochastic gradient descent saves memory and computation at each update iteration, but it takes many more iterations and much longer time to reach an optimal solution than regular gradient descent. Neither learning the training samples all at once nor learning the training samples one at a time, batch gradient descent is somewhere in the middle. By learning a group of samples or a batch of samples one at a time, batch gradient descent needs less computation and memory than regular gradient descent and trains faster than stochastic gradient descent.

4.2.1.4 How to select the trained weights of an ANN

Because of the multilayer structure and nonlinear activation function in between, a typical ANN network is a highly nonlinear and nonconvex function. The nonconvexity of ANN means it is not guaranteed to reach the global optimal solution when we perform gradient descent. In addition, because the available training dataset is just an estimation of the actual population, even the global optimal solution on the training dataset is not guaranteed to perform well on the unseen data. Third, because ANN can perform as an ultra-high dimensional function, which can be an overshoot of the actual dynamics underneath the problem, an ANN over-fitted on the training dataset can perform poorly on the unseen data. Because of the aforementioned reasons, it is important to decide when the ANN has been trained at the most appropriate point.

The good or bad of an ANN is decided by how well it can perform on the unseen data, not on the training data. In order to find the model weight with the best generalizability, the dataset is usually split into three subsets: training set, validation set, and test set. Here is model selection strategy during training:

- Train ANN weights using the training set.
- Select the weights with the best performance on the validation set.
- The final model performance is reported on the test set.

4.2.2 CNN

The CNN is a special kind of network in the ANN family. The major advantage of the CNN is that it is better at capturing local features while maintaining the dimensional structure of the input data than the regular fully connected ANNs. The CNN also requires less parameters than the regular ANNs. The advantages of the CNN come from its two main features: convolution and weights sharing. Figure 4.4 shows a regular fully connected layer and a one-dimensional (1-D) CNN layer side by side.

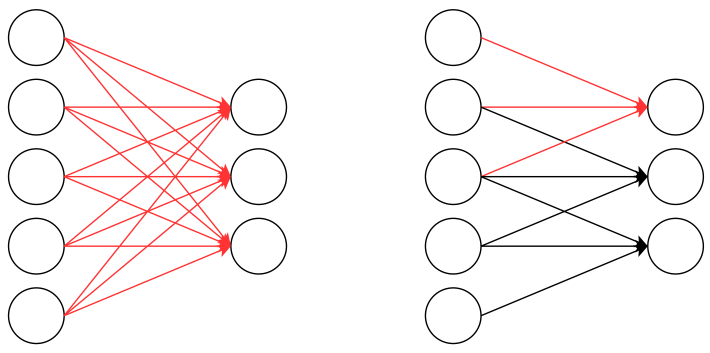


Figure 4.4 Fully connected layer and CNN layer

On the left is a regular fully connected layer and on the right is a 1-D convolutional layer. A fully connect layer connects every node in one layer to every node in another layer. The specific fully connected layer shown in Figure 4.4 has $5 \times 3 = 15$ connections so that is fifteen parameters. A CNN layer applies a convolution operation to the input and passes the results to the output. The specific CNN layer shown in Figure 4.4 applies a convolution operation using one filter (also can be called a kernel) with size = 3 and passes the results to the output. This CNN layer uses only three parameters. Figure 4.5 is another example of a 2-D CNN layer. The specific 2-D CNN layer applies a convolution operation using one filter with size = 3 x 3 (and stride = 1) on a 10 x 10 input sample and outputs an 8 x 8 feature map. A fully connected layer mapping 10 x 10 input to 8 x 8 output takes 10 x 10 x 8 x 8 = 6400 parameters, whereas this CNN layer only takes 3 x 3 = 9 parameters.

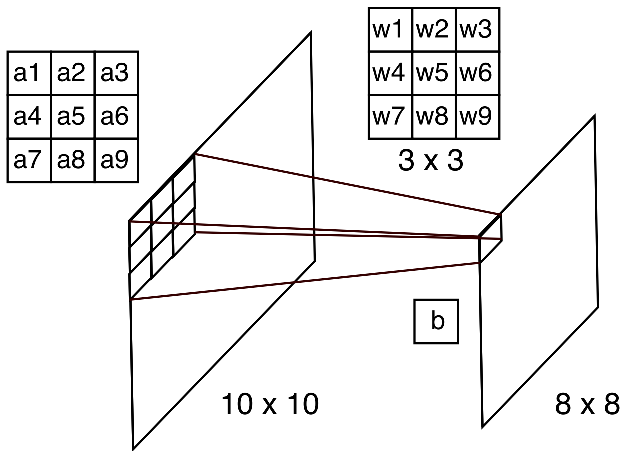


Figure 4.5 Two-dimensional CNN layer

As shown in both Figure 4.4 and Figure 4.5, in CNN context, convolution is an operation that slides a certain filter (or kernel) though the input data and performs an operation at each location. The sliding can use step size = 1 (both Figure 4.4 and Figure 4.5) or any other step sizes. The operation performed at each location is a sum of elemental-wise matrix multiplication. At the specific location in Figure 4.5, the operation is:

$$b = \sum_{i=1}^{9} w_i a_i$$

Convolutions are computationally calculated as matrix multiplication, the same as the fully connection. Using the example in Figure 4.4, the computation of the fully connection can be written as:

$$[a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5]\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \\ w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \end{bmatrix} = [b_1 \quad b_2 \quad b_3] \tag{4.15}$$

The computation of the convolution can be written as:

$$[a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5]\begin{bmatrix} w_1 & 0 & 0 \\ w_2 & w_1 & 0 \\ w_3 & w_2 & w_1 \\ 0 & w_3 & w_2 \\ 0 & 0 & w_3 \end{bmatrix} = [b_1 \quad b_2 \quad b_3] \tag{4.16}$$

By comparing Equation 4.15 and Equation 4.16, it is also clear that convolutions use less parameters by weights sharing.

Convolution extracts features from local information while maintaining the dimensional structure from the input data. By sliding a same set of filters, the weights are shared by local information at different locations and a CNN layer uses less parameter than a regular fully connected layer. Convolutional neural networks are good at processing structured information. Images are naturally spatially structured data and the CNN is currently dominant in the image processing field. Traffic data that contains spatial and temporal information is another good fit for CNNs.

4.2.3 Proposed CNN

As described in section 4.1, the predicted weather information and the historical traffic speeds on the same roadway network will be used as input to predict future traffic speeds. As described in section 3.3, there are nine variables in weather data; the data of each variable is arranged in a 2-D array in which one dimension stands for location and the other dimension represents time. The nine 2-D arrays are stacked together as a three-dimensional (3-D) volume in which the third dimension represents different variables and is called a channel dimension. Traffic speeds on the same time/same day in the past six weeks are used as historical traffic speeds. The traffic speeds on one day are arranged in a 2-D array in which one dimension strands for location and the other dimension represents time. Six of them are stacked together as a 3-D volume in which the third dimension represents weeks. A fully convolutional deep network is proposed to take in both the predicted weather information and the historical speed information and predict the traffic speed correspondingly. The whole structure of the proposed network for long-term speed prediction is shown in Figure 4.6.

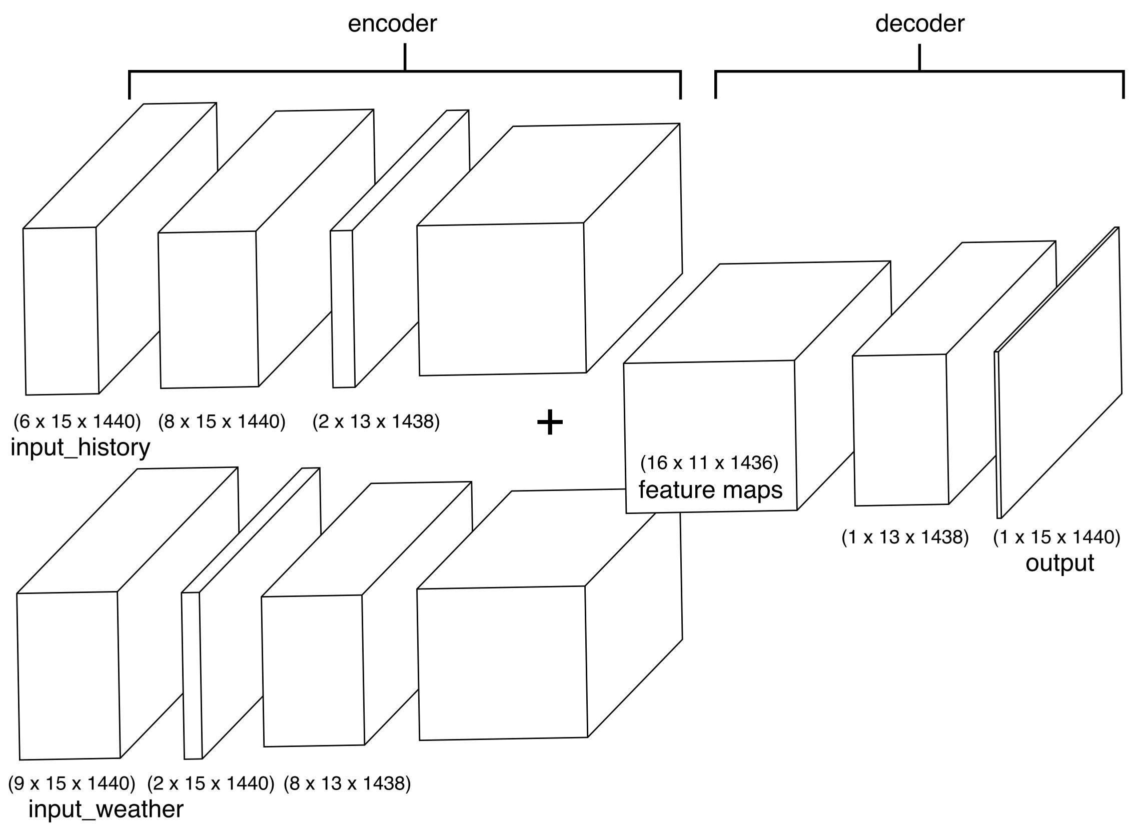


Figure 4.6 Proposed CNN

This fully convolutional deep network has an “encoder-decoder” structure where two input sources are mapped and merged to a feature space and then the output is reconstructed from the extracted features. Because the spatial-temporal dimension is fixed given a target roadway network and a predicted time range, the input information is not shift-invariant and thus the proposed fully convolutional deep network uses no pooling layers in the feature extraction layers and no un-pooling layers in the output reconstruction layers as a normal CNN-based encoder and decoder will do. Still, since the proposed network deploys the concept of extracting features and then reconstructing from features, “encoder” and “decoder” are used here to depict the model structure.

The details in proposed CNN architecture are the following:

1. Two different CNN encoders are used to extract features from weather data and historical speed data separately.
2. The extracted features maps from two sources have the same size and are merged together.
3. The merged feature map is then up-sampled by a CNN decoder to reconstruct the predicted speed.

The ideas behind the design of this network structure include the following:

1. The network should be deep to capture the highly nonlinear dynamics.
2. The network should use CNN layers to capture spatial-temporal dependencies.
3. The network should only use CNN layers (fully convolutional) so that the network is independent of the size of input and a pretrained network can be applied on roadway networks with any number of sensors and predict traffic speed in any length of time window. The networks should use no fully

connected layers also because the spatial-temporal structure should be retained from end to end.

4. The network should treat the two input sources separately so that if one of the sources is missing in some practical use cases a pretrained model still contains valuable weights.
5. The network should not contain too many parameters so that it can be trainable with a limited dataset.

All the CNN layers in this section use stride = 1 and no padding.

4.2.3.1 Encoder for Predicted Weather

The detailed structure of the encoder for predicted weather is shown in Figure 4.7. From one-day predicted weather data containing nine variables as input, this encoder extract features three CNN layers:

1. 2-D CNN layer using two filters with size = 9 x 1 x 1
2. 2-D CNN layer using eight filters with size = 2 x 3 x 3
3. 2-D CNN layer using sixteen filters with size = 8 x 3 x 3

The number of total parameters contained in CNN layers is 2 x 9 x 1 x 1 + 8 x 2 x 3 x 3 + 16 x 8 x 3 x 3 = 1314.

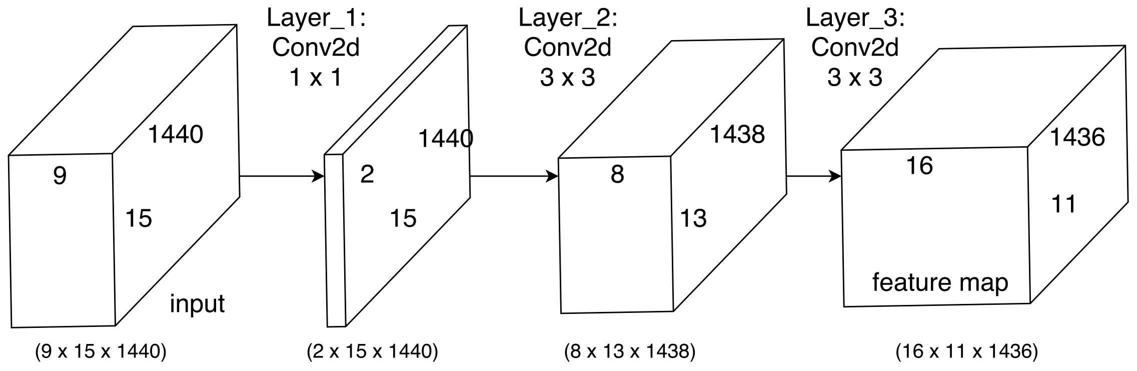


Figure 4.7 Structure of encoder for predicted weather

Inspired by the original GoogLeNet paper (Szegedy et al., 2014), the 1-by-1 convolution layer is used before the relatively more expensive 3-by-3 convolution layer to reduce dimensionality in the channel dimension while providing depth and nonlinearity in the network. As shown in Figure 4.7, suppose we jump from the input with size = 9 x 15 x 1440 to the second feature map with size = 8 x 13 x 1438 using 3-by-3 convolution and skipping the 1-by-1 convolution; the number of parameters will be 9 x 3 x 3 x 8 = 648. By adding the 1-by-1 convolution in the middle, the dimensionality of the channel dimension reduces the 3-by-3 convolution and the number of parameters decreases to 9 x 1 x 1 x 2 + 8 x 2 x 3 x 3 = 162.

Following the 1-by-1 CNN layer, two sequential 3-by-3 CNN layers extract the feature map from local information while maintaining the spatial-temporal structure.

4.2.3.2 Encoder for Historical Speed

The detailed structure of the encoder for historical speed is shown in Figure 4.8. From six days of historical speed as input, this encoder extract features using three CNN layers:

1. 3-D CNN layer using two filters with size = 3 x 3 x 3
2. 2-D CNN layer using two filters with size = 8 x 1 x 1
3. 2-D CNN layer using sixteen filters with size = 2 x 3 x 3

The number of total parameters contained in CNN layers is 2 x 3 x 3 x 3 + 2 x 8 x 1 x 1 + 16 x 2 x 3 x 3 = 358.

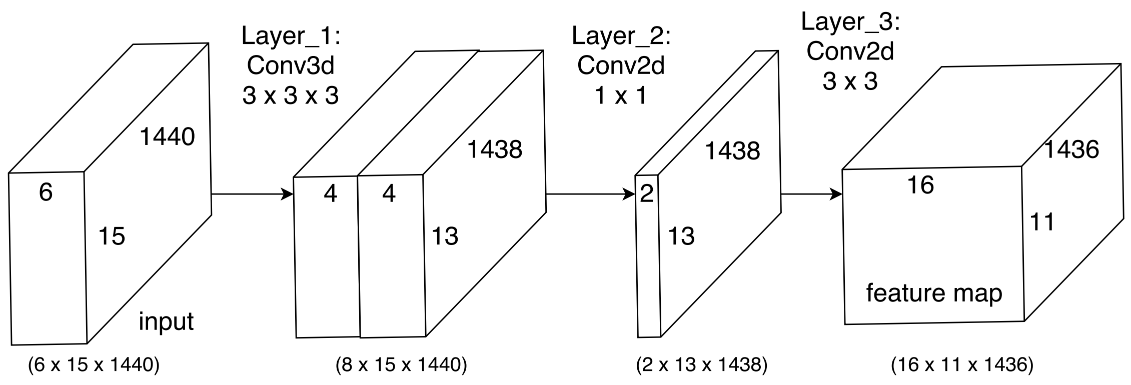


Figure 4.8 Structure of encoder for historical speed

The major difference between the 3-D input data volumes from two sources is that the third dimension of predicted weather is a channel dimension that contains no structural information (the order of channel is not meaningful), whereas the third dimension of historical data contains temporal order. The advantage of convolution against fully connection applies to every dimension. In order to capture the temporal dynamics in the third dimension of historical speeds, convolution should replace fully connect in this dimension as well. Therefore, a 3-D convolutional layer is used. Two 3-by-3-by-3 filters extract two feature maps and the channel dimension is then removed by stacking the two maps together to remain a total of three dimensions.

Following the 3-D CNN layer, again a 1-by-1 convolution is applied before the 3-by-3 convolution to reduce dimensionality in the channel dimension while providing depth and nonlinearity in the network as explained in section 4.2.3.1. Finally a 3-by-3 CNN layer extracts the feature maps from local information while maintaining the spatial-temporal structure. The feature maps extracted from the historical speed encoder have the same size as the ones extracted from the predicted weather encoder.

4.2.3.3 Decoder

After merging the extracted feature maps from two input sources, the merged feature maps are sent to the decoder to reconstruct the predicted speed. The detailed structure of the decoder is shown in Figure 4.9. This decoder reconstructs predicted speeds using two CNN layers.

1. 2-D transposed CNN layer using eight filters with size = 16 x 3 x 3
2. 2-D transposed CNN layer using one filter with size = 8 x 3 x 3

The number of total parameters contained in CNN layers is 8 x 16 x 3 x 3 + 1 x 8 x 3 x 3 = 1224.

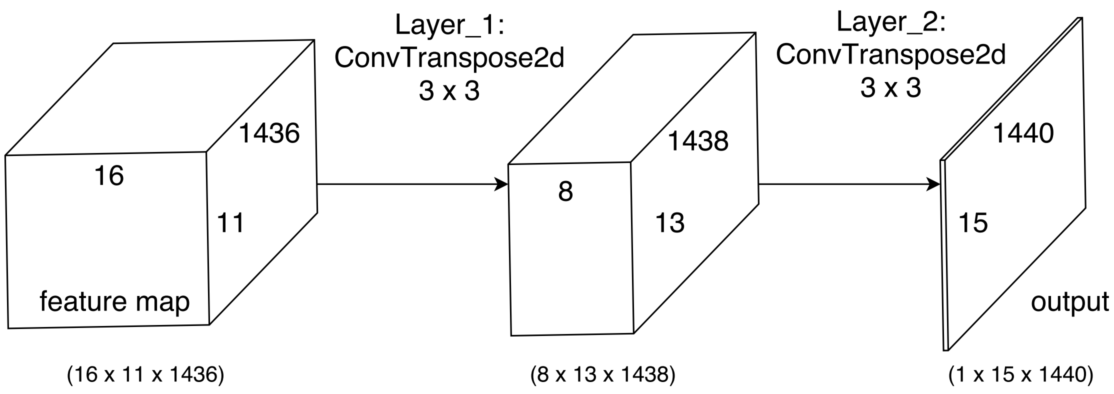


Figure 4.9 Structure of decoder

Transposed convolutions generally arise from the desire to use a transformation going in the opposite direction of a normal convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution. Transposed convolutions are used here as decoding layers to project feature maps to back to the same dimensional space as the input. As described in Equation 4.16, convolutions are essentially matrix multiplications so they are transposed convolutions. Because the weights shape is transposed when going from the right-hand side to the left-hand side of a CNN computation (see Equation 4.16), this operation is called transposed convolution.

4.3 Case Study

4.3.1 Model Input

As described in section 3.3 and section 4.1, both weather information and historical speeds are used as the model input to predict long-term speeds. All nine variables in weather information of the predicted day are used. Historical speeds of the same day in the past six weeks are used. Data are arranged in 2-D arrays with the 15 sensor locations along the vertical axis and the 1,440 minutes in the day along the horizontal axis. Nine weather variables are stacked along the third axis and six historical days of traffic speeds are stacked along the third axis. After the data processing describe above, each of the 447 samples are arranged as below:

1. Weather input: a 9-by-15-by-1440 3-D array
2. Historical speed input: a 6-by-15-by-1440 3-D array
3. Speed output label: a 15-by-1440 2-D array

4.3.2 Model Training

Model training is the key step in deep learning. A common belief is that even an arbitrary deep network has a very high modeling power, but how much potential we can get from it depends on how we train it. Another common belief is that to effectively train a deep model, we need one order higher number of training samples compared to the number of parameters in the deep network.

As described in section 4.2.3, the proposed fully connected deep network contains about 2,900 parameters, but in this study we only have 447 training samples. A data augmentation method here is used to boost the number of training samples. One big advantage of the fully convolutional network is that it accepts any input size and the output size will depend on the input it receives. With that being said, the model can be trained on any size of data samples; for example, it can be trained on two-hour patches as well as one-day patches. Following that idea, the 447 one-day training samples are processed into 447 x ((1440 – 120) / 10 + 1) = 59,451 two-hour patches using a sliding window with step size = 10 minutes. After the data patching, we have about 59,000 training samples that is one order more than the 2,900 parameters.

The ReLU as the nonlinear activation function and batch normalization as the anti-overfitting strategy are applied at each layer of the fully convolutional deep network. Eighty percent and 20% of the samples are used for training and validation separately while the three test days described in section 3.3 are reserved for the testing purpose only. The RMSE is used as the loss function and batch gradient descent is performed using adaptive motion estimation (Adam) optimizer where the learning rate and moving direction are adaptively changed during training. Data samples are put in batches with size = 200.

Given the designed "encoder-decoder" network structure described in the previous section, there are other details that may differ the model performance. For example, the number of CNN filters used in each layer determines the number of features extracted and the number of weights as well. With more filters, the model extracts more features from the input data and potentially has higher prediction power. Meanwhile, more filters bring more weights and the model becomes harder to train given limited train data. The model structure is designed to deal with two input sources separately and has the ability to disable one or the other as needed. The fully convolutional deep network is tested with different number of filters in every layer following the regulation that the number of filters changes exponentially in consecutive layers. The model is also tested with only weather information as input, only historical traffic data as input and both sources as input. After all those experiments, The model with proposed number of filters shown in the previous section and with both sources as input provides the best performance. The final selected fully convolutional deep network is trained on an NVIDIA TITAN Xp GPU for 5,000 minutes.

4.3.3 Testing Results and Discussion

The trained fully convolutional deep network is tested on the three selected test days described in section 3.3. Each testing is performed by three experiments as follows:

1. Use one-day patch as input, predict the whole day speeds, and compare with the one-day patch ground truth.
2. Test on two two-hour patches: 7:00–9:00AM during AM peak hours and 4:30–6:30PM during PM peak hours, and compare with the two-hour patch ground truth.
3. Predict two-hour speeds every 10 minutes, then stitch all predicted two-hour patches into a one-day prediction and compare with the one-day patch ground truth.

The first and second experiments are designed to demonstrate that the fully convolutional deep network can be used on any size of input to predict the speeds accordingly. And the third experiment is designed to connect to two input scales to demonstrate that the fully convolutional deep network learns structural dependencies from input, so that even though the model is trained on two-hour patches it has the transferability to predict the speeds at any scale. The testing results of the three test days are shown in Figure 4.10, Figure 4.11 and Figure 4.12, respectively. Each figure is organized as follows:

- Row 1 shows the whole day ground truth for experiment 1 and experiment 3.

- Row 2 shows the prediction results from experiment 1.
- Row 3 shows the ground truth for experiment 2.
- Row 4 shows the prediction results from experiment 2.
- Row 5 shows the prediction results from experiment 3.

Several observations can be made from the testing results:

1. The fully convolutional deep network is capable of using information from both historical speeds and weather data to make predictions. By comparing test day 2 against test days 1 and 3, the model predicts different traffic patterns based on historical traffic conditions on Mondays and Fridays. By comparing test day 1 against test day 3, the model predicts different overall traffic conditions based on the variation in the weather of the two test days even though there is not a very significant difference between the weather data summary statistics of the two test days.
2. The model can predict traffic speeds at any scales. By comparing the results from experiment 1 and experiment 2 in each test case, the model is able to capture the dynamics and predict similar traffic conditions at different temporal scales. By comparing the results from experiment 1 and experiment 3 in each test case, even though the model is trained on two-hour patches, at inference it can take a whole-day patch as input and directly predict the whole-day speeds at even a higher accuracy level than predicting and stitching two-hour patches. With that being said, the fully convolutional deep network has high transferability.
3. Long-term prediction is hard and the traffic speeds are impacted by many stochastic factors. Human factors and other stochastic factors that happened on

the actual day can cause big fluctuations in the traffic condition during peak hours and only utilizing historical traffic speeds and weather information as input sources could not predict the nonrecurrent congestions well enough. Those fluctuations usually are caused by short-term dependencies and will be discussed in the next chapter.

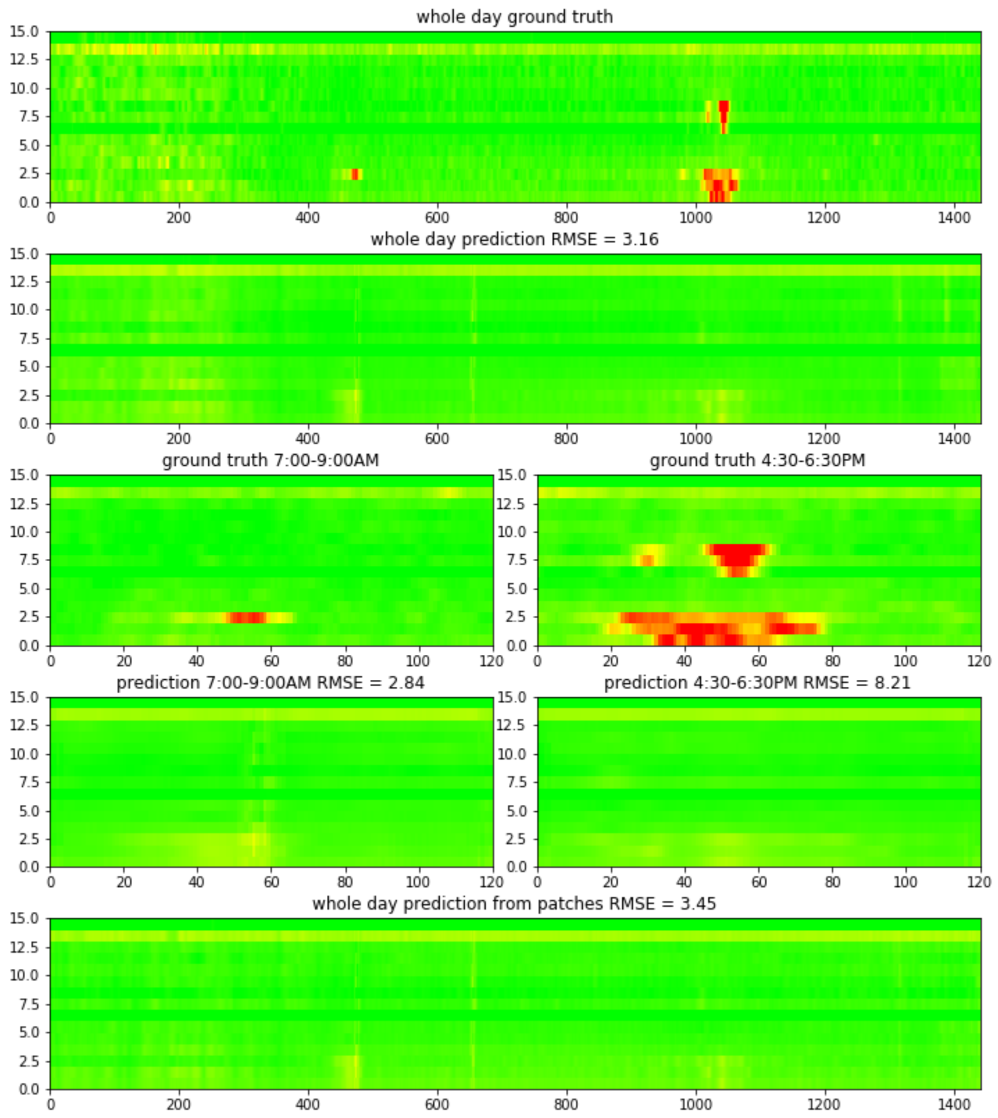


Figure 4.10 Model results on test day 1, 12-02-2016 Friday

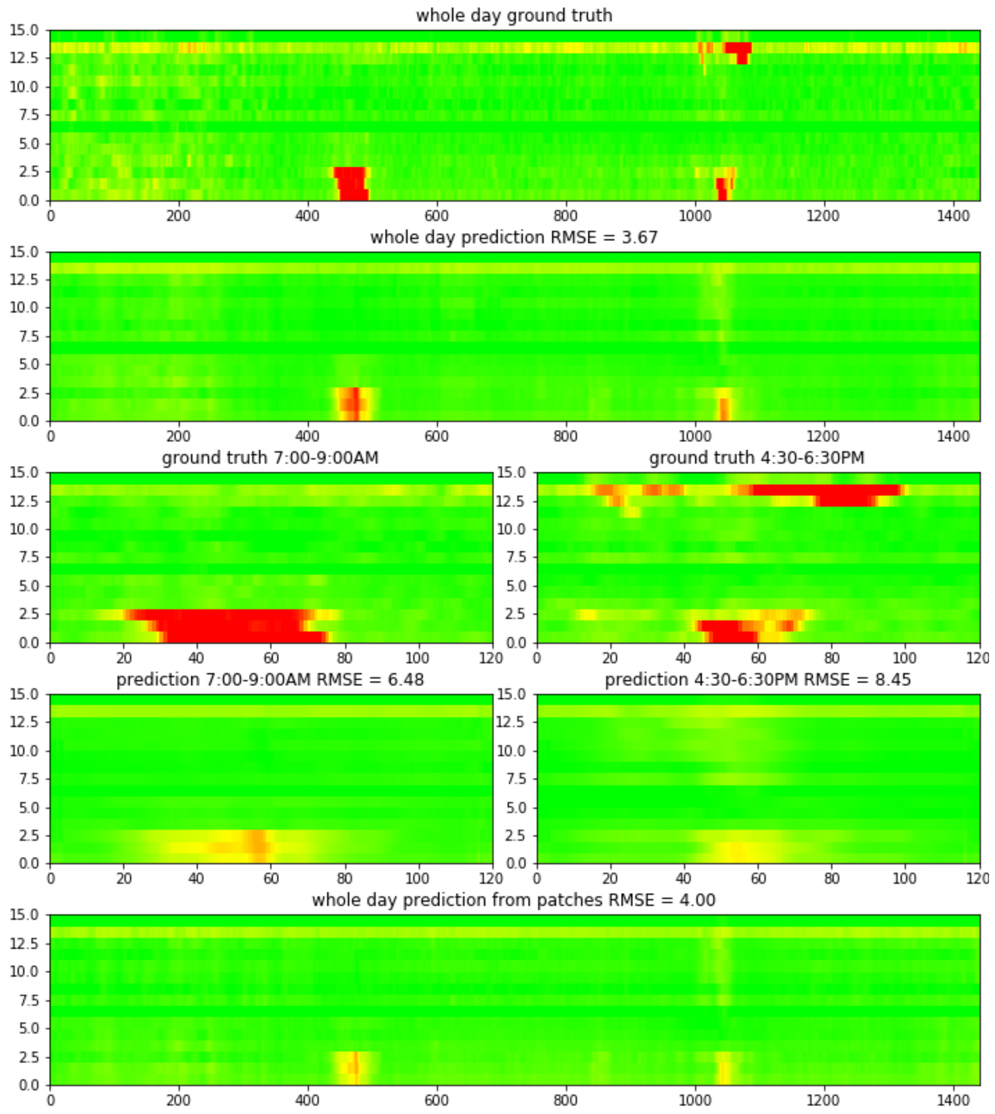


Figure 4.11 Model results on test day 2, 12-05-2016 Monday

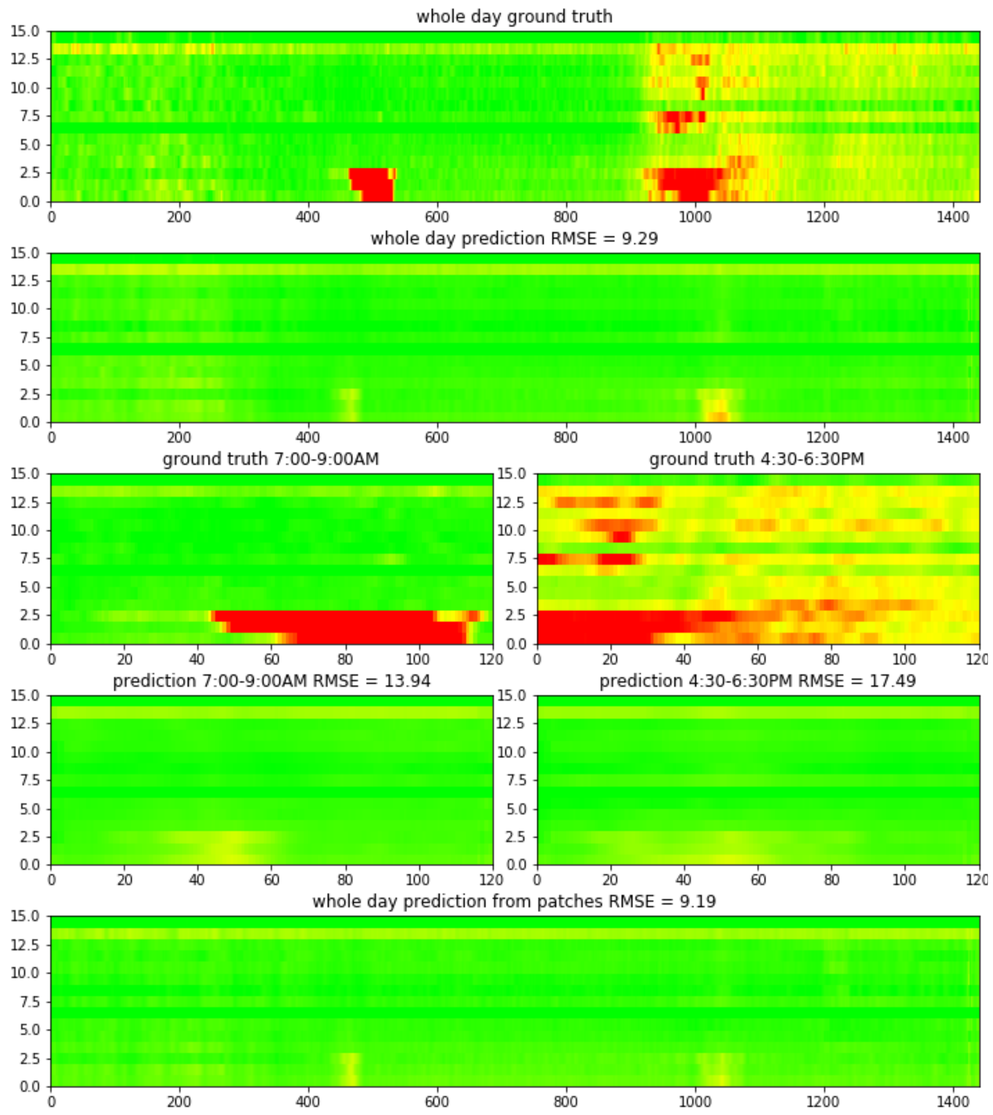


Figure 4.12 Model results on test day 3, 12-16-2016 Friday

CHAPTER 5. SHORT-TERM SPEED PREDICTION USING HYBRID DEEP NEURAL NETWORK

5.1 Introduction

This chapter focuses on short-term prediction at a route level using a hybrid LSTM network. Corresponding to the "long-term" in the previous chapter, "short-term" refers to prediction for the next few minutes.

It has been explained in section 2.1 that traffic speed heatmaps of a roadway network over a certain period of time can be viewed as images. The short-term speed prediction discussed here is predicting the traffic speed image of the next few minutes as the output. Unlike the long-term traffic speed prediction that usually is done once in a while—e.g., the prediction is done once a day when predicting the traffic speed of the next day—the short-term traffic speed prediction should be done in a streaming fashion. The raw traffic information collected by various sensors usually comes in by a fixed time interval. Then the prepossessed data for analysis will keep the same pace or may be aggregated to a larger interval. In whatever time interval, the data collected from sensors come in every interval and in a streaming fashion. Thus the short-term speed prediction should provide the speed estimation several intervals ahead and also in a streaming fashion. Suppose the data come in every one minute and the short-term speed prediction is expected to provide the speed estimation five minutes in the future; then every time a new data point comes in, the prediction program should also roll one time interval ahead and predict the speeds five minutes ahead related to the current timestamp.

In the previous chapter, long-term traffic speed prediction leverages predicted weather information as one of the major inputs. Undeniably, weather has an impact on traffic conditions at a large time scale, but it may not be a very dominant and responsive indicator in the short-term speed prediction case. There are various factors including weather that may affect the actual traffic speed. Some of the factors are able to be captured by sensors, some of them are not, and some of them may even be impossible to measure. But eventually all those impacts will be reflected on the real traffic speeds. As a kind of time series data, the past evolution of traffic speeds is intuitively the best predictor of its future states. Therefore, the traffic speeds of the near past timestamps are used as input. Traffic also follows patterns. The traffic patterns will not change much from week to week on a certain roadway network at the same time of day. Considering that, the historical traffic speeds on the same roadway network at the same time of day should be used as input to predict future traffic speeds as well. Here we can view the traffic speeds as 2-D matrices as well, where rows represent locations and columns stand for time. The short-term traffic speed prediction has then become a problem that predicts the next column in the matrix, using previous columns as well as the same column in the same day in the last several weeks as input. The problem can be formularized as the following.

$$Speed_{d,t} = f(Speed_{d,t-1}, \dots, Speed_{d,t-T}, Speed_{d-7*1,t}, \dots, Speed_{d-7*W,t})$$

where

$$Speed_{d,t} = \begin{bmatrix} s_{1t} \\ s_{2t} \\ s_{3t} \\ \vdots \\ s_{mt} \end{bmatrix}_d$$

$m = number\ of\ locations\ in\ the\ roadway\ networks$

$d = day\ number$

$t = timestamp\ number$

In this chapter, an (LSTM-based network is used as the function f in the equations above to deal with short-term traffic speed prediction.

5.2 Methodology

To clearly explain the proposed LSTM-based network used in this chapter, the discussion needs to start from regular recurrent neural networks, transits to the advantages of LSTM, and finally explain the proposed network in details.

5.2.1 RNN

In the short-term traffic speed prediction problem, when we try to predict what is going on next we would intuitively not only use the information at the current timestamp. Instead, we would consider the information in past timestamps as well to predict what is going on next. Suppose we are currently at timestamp t and want to predict traffic speeds at timestamp $t + 1$. In order to take information from the previous timestamps into consideration, there are two ways to model this problem.

1. A model takes input data from the previous N timestamps, including the current timestamp all at once, and predicts traffic speed at time $t + 1$ as shown in Figure 5.1.
2. A model only takes input data from one timestamp at a time and makes an output. At each timestamp, the model takes the data from that timestamp as well as the output from the previous timestamp to make an output. The model reads in data from all previous timestamps in order and finally makes a prediction for timestamp $t + 1$ as shown in Figure 5.2.

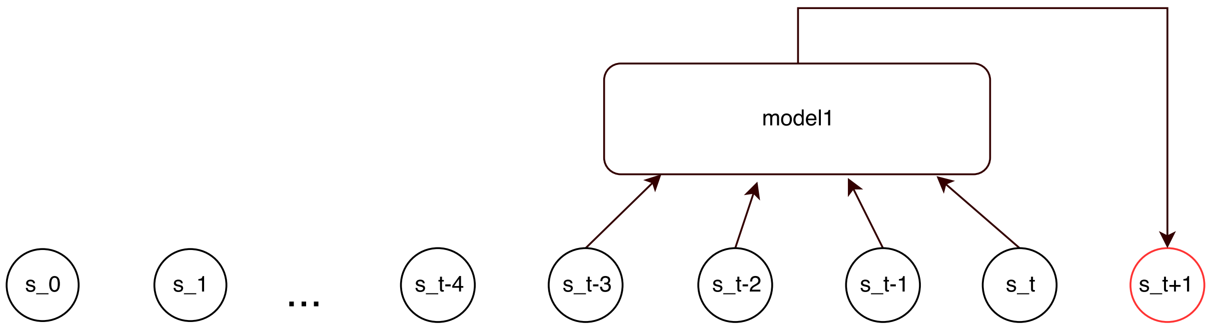


Figure 5.1 Traditional NN structure for sequence prediction

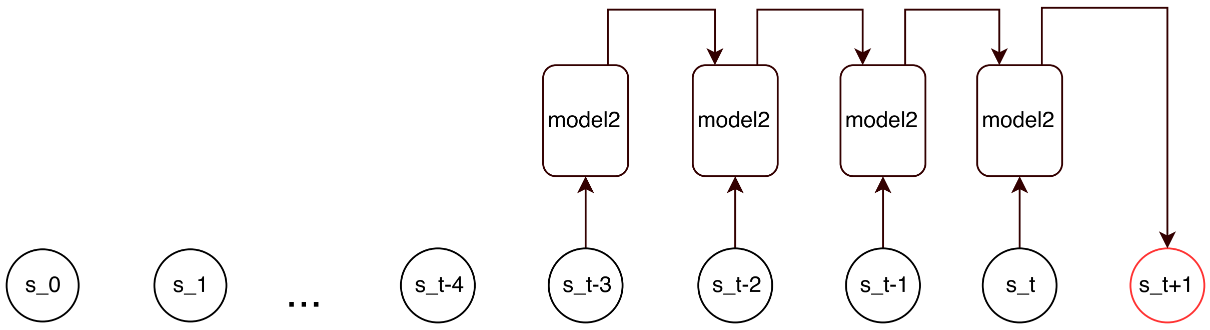


Figure 5.2 Recurrent NN structure for sequence prediction

By comparing the two strategies, the first one has limitations such as the following:

- The window size N is prefixed.
- The temporal dependencies have to be handled inside the model.

The second strategy has several advantages when handling a time series data such as the following:

- There is no fixed window size; the number of previous timestamps used to predict the next one is flexible.
- The temporal dependencies are naturally handled by the recurrent model structure.

An ANN having a recurrent structure as shown in Figure 5.2 is called an RNN. Recurrent neural network models have a chain-like structure and they are the natural architecture of ANN to use data in sequences.

5.2.2 LSTM

Long-short term memory networks belong to the RNN family. The major problem of the regular RNNs is that they are not good at modeling the long-term dependency. The reason is that, as shown in Figure 5.2, the regular RNN passes the previous information only through the output-input connections and at each pass the information is nonlinearly transformed. In theory the information has been successfully passed on, but in reality people find that the long-term dependencies are lost after several nonlinear passes. Because of that, the regular RNNs are all "short-memory."

The LSTM addresses this long-term dependency loss problem by adding a "green channel" for the "memories" to pass through. This "green channel" is called cell state and when "memories" passing through, only linear transformations are applied along the way. Each LSTM cell utilizes the input from the current timestamp, the output from the previous timestamp, and the "memories" passed from the "green channel" to make an output, and then it linearly modifies the "memories" and keeps passing them along the "green channel." In such a way, the long-term dependencies are well preserved. The structure is shown in Figure 5.3.

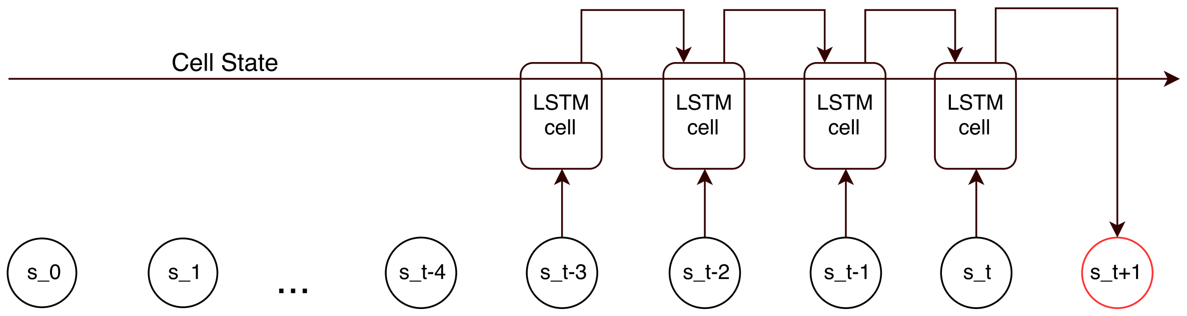


Figure 5.3 LSTM NN structure

In a short-term traffic speed prediction problem, a driver would probably take a very long-term dependency into consideration. For example, at 4:50pm a driver is planning a travel at 5:00pm, which is usually when the PM peak starts. The driver would consider what the traffic conditions currently and in the past hours are. The driver may also check whether it is more congested during the morning peak today to estimate the traffic condition at 5:00pm. Preferably, we may want to look through the traffic conditions from the start of the day until the current time sample to predict what is going to happen next. The LSTM is a great model architecture to model data in time sequences with long-term dependencies.

5.2.3 Proposed Hybrid LSTM

As described in section 5.1, the traffic information from the previous timestamps and the historical traffic speeds on the predicted timestamp from the previous weeks will be used as input to predict traffic speeds in the next few minutes. A hybrid LSTM network is proposed to take in both the input sources and predict the traffic speed correspondingly. The whole structure of the proposed network for short-term speed prediction is shown in Figure 5.4.

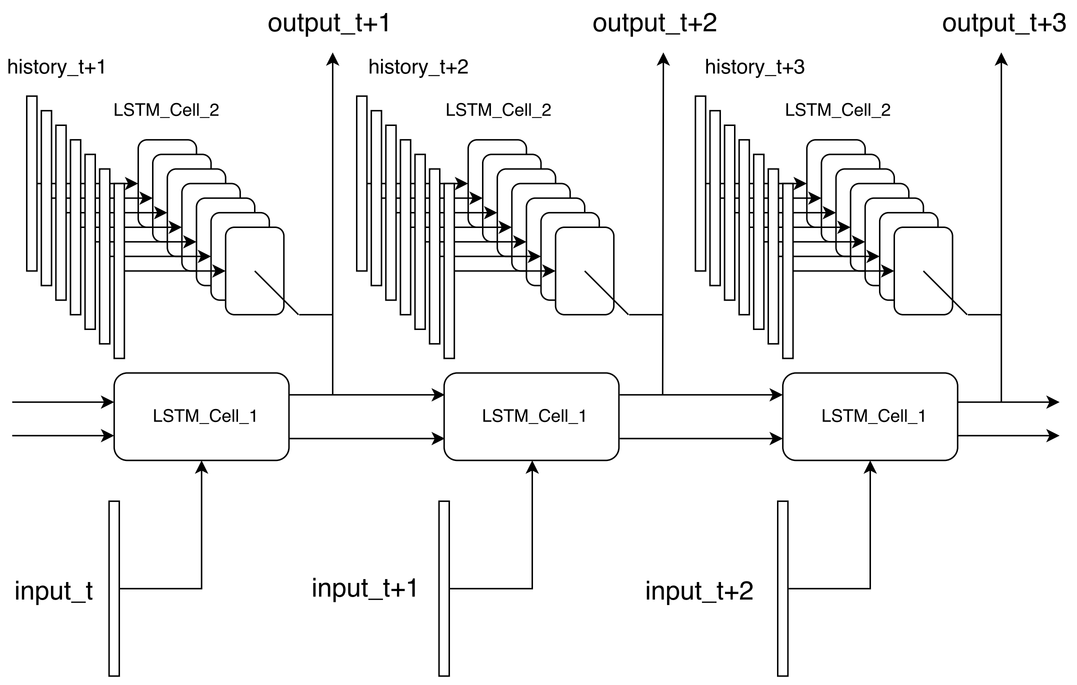


Figure 5.4 Proposed hybrid LSTM model structure

This hybrid LSTM network predicts traffic speed at timestamp $t + 1$ using features extracted from two input sources:

1. The information from the start of the current day to timestamp t. The input information can have traffic speed as the only variable or it can contain any more variables that are available.
2. The traffic speeds at timestamp $t + 1$ of the same day in the past six weeks.

At timestamp t, steps to predict traffic speeds at time $t + 1$ include the following:

1. One LSTM cell takes features extracted from timestamp t, the outputted feature maps from timestamp $t - 1$, and the cell state and outputs feature maps.
2. Another LSTM cell goes through the sequence of features in the past six weeks and outputs another feature map.
3. The feature maps extracted from two sources are combined and the combined feature maps are outputted as predicted speed at time $t + 1$ by a fully connected layer.

At time t, to predict speed at time $t + n$, first predict speed as time $t + 1$, then use speed at $t + 1$ as input to predict speed at $t + 2$, and then keep looping until you get the predicted speed as $t + n$. In this mode, the first input source contains traffic speed as the only variable.

The features from both of the aforementioned two sources are extracted by a 1-D convolutional filter with size = 5 as shown in Figure 5.5. Then the extracted features are passed to LSTM cells for prediction.

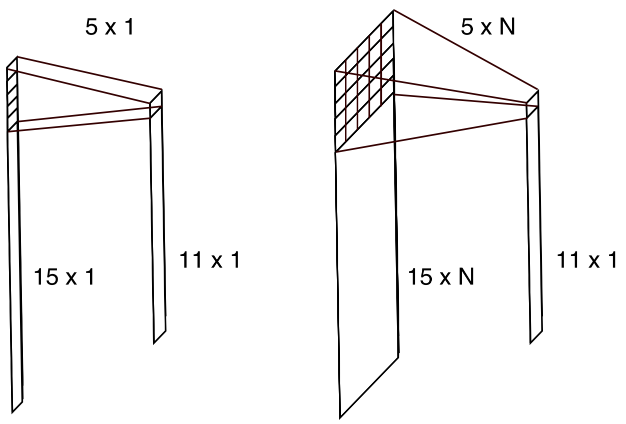


Figure 5.5 1-D convolutional filter for feature extraction for LSTM cell input

5.3 Case Study

5.3.1 Model Input

As described in section 3.3 and section 5.1, both the traffic information from the previous timestamps and the historical traffic speeds from the previous days are used as input to predict traffic speeds in the next few minutes. Only traffic speeds are used as input here to give the model the ability to predict any timestamps ahead as described in section 5.2.3. Historical speeds on the predicted timestamp of the same day from the previous six weeks are used. Data of each timestamp are arranged in 1-D arrays ordered by the 15 sensor locations. At each timestamp t, the data are structured as follows:

1. Current speed input: a size = 15 1-D array.
2. Historical speed input: a 6-by-15 2-D array.
3. Speed output label: a size = 15 1-D array. The actual speeds at time $t + 1$ are used as labels.

5.3.2 Model Training

There is so much flexibility in training an LSTM model. There are two LSTM cells in the proposed hybrid LSTM network described in section 5.2.3: the current speed LSTM cell and the historical speed LSTM cell; in this section we call them LSTM cell 1 and LSTM cell 2, respectively. For LSTM cell 1, the training method used here is using the actual speed at timestamp $t + 1$ as the label for training sample at timestamp t and the "memory" is reset at every start of day. The LSTM cell 2 lives outside of the LSTM cell 1 timeline. At each timestamp on LSTM cell 1's timeline, LSTM cell 2 goes through all six of its own timestamps to provide features from historical information. This specific training strategy teaches the hybrid LSTM to remember everything that has happened from the start of the current day, as well as the historical speeds, and to predict the next timestamp at its best.

The ReLU as the nonlinear activation function and batch normalization as the anti-overfitting strategy are applied at the input layer and the output layer of both LSTM cells. Eighty percent and 20% of the data are used for training and validation separately whereas the three test days described in section 3.3 are reserved for the testing purpose only. The RMSE is used as the loss function and batch gradient descent is performed using adaptive motion estimation (Adam) optimizer where the learning rate and moving direction are adaptively changed during training. Data samples are put in batches with size = 1,000. The hybrid LSTM network is trained on an NVIDIA TITAN Xp GPU for 5,000 minutes.

5.3.3 Testing Results and Discussion

The trained hybrid LSTM network is tested on the three selected test days described in section 3.3. Each testing is performed by two experiments as follows:

1. Use only the current speed LSTM cell with historical speed LSTM cell disabled to predict speeds in six difference time scales: 1-minute ahead, 5-minute ahead, 10-minute ahead, 15-minute ahead, 30-minute ahead, and 60-minute ahead.

2. Use both LSTM cells to predict speed in six different time scales: 1-minute ahead, 5-minute ahead, 10-minute ahead, 15-minute ahead, 30-minute ahead, and 60-minute ahead.

The two experiments are designed to compare the different model behaviors with and without historical information. For each experiment, six different time scales are tested to demonstrate the flexibility of the model in terms of prediction range as well as the model prediction accuracy. The testing results of the three test days are shown in Figure 5.6, Figure 5.7, and Figure 5.8, respectively. Each figure is organized as follows:

- Column 1 shows the results from experiment 1.
- Column 2 shows the results from experiment 2.
- Row 1 shows the ground truth.
- Rows 2–7 show results at the six different prediction time scales.

Several observations can be made from the testing results:

1. The short-term hybrid LSTM model can predict more accurate traffic speeds than the long-term model. Traffic conditions have trends and also are sensitive to the near past stochastic factors. A combination of short-term prediction and long-term prediction can predict more comprehensive multilayer information.
2. The hybrid LSTM network can predict traffic speed at different prediction time scales. As described in section 5.2.3, the model can keep rolling ahead by using the prediction as input to predict the next timestamp. The results show that although the model is trained to predict only the next one timestamp, it can predict further. The network's prediction accuracy decreases as the prediction

range gets longer. The short-term prediction model works better when predicting speed less then fifteen minutes ahead.

3. The hybrid LSTM network shows different behavior with and without historical input. Using only the previous information from the current day as input, the model tends to carry the current traffic speed and predict the future speed near it. Although the overall RMSE is lower than the RMSE when using historical information, the predicted congestion is some kind of a shifted version of observed congestion. When using historical information, the overall prediction, although having a higher RMSE, has a better value in terms of predicting the start and end of a congestion event.

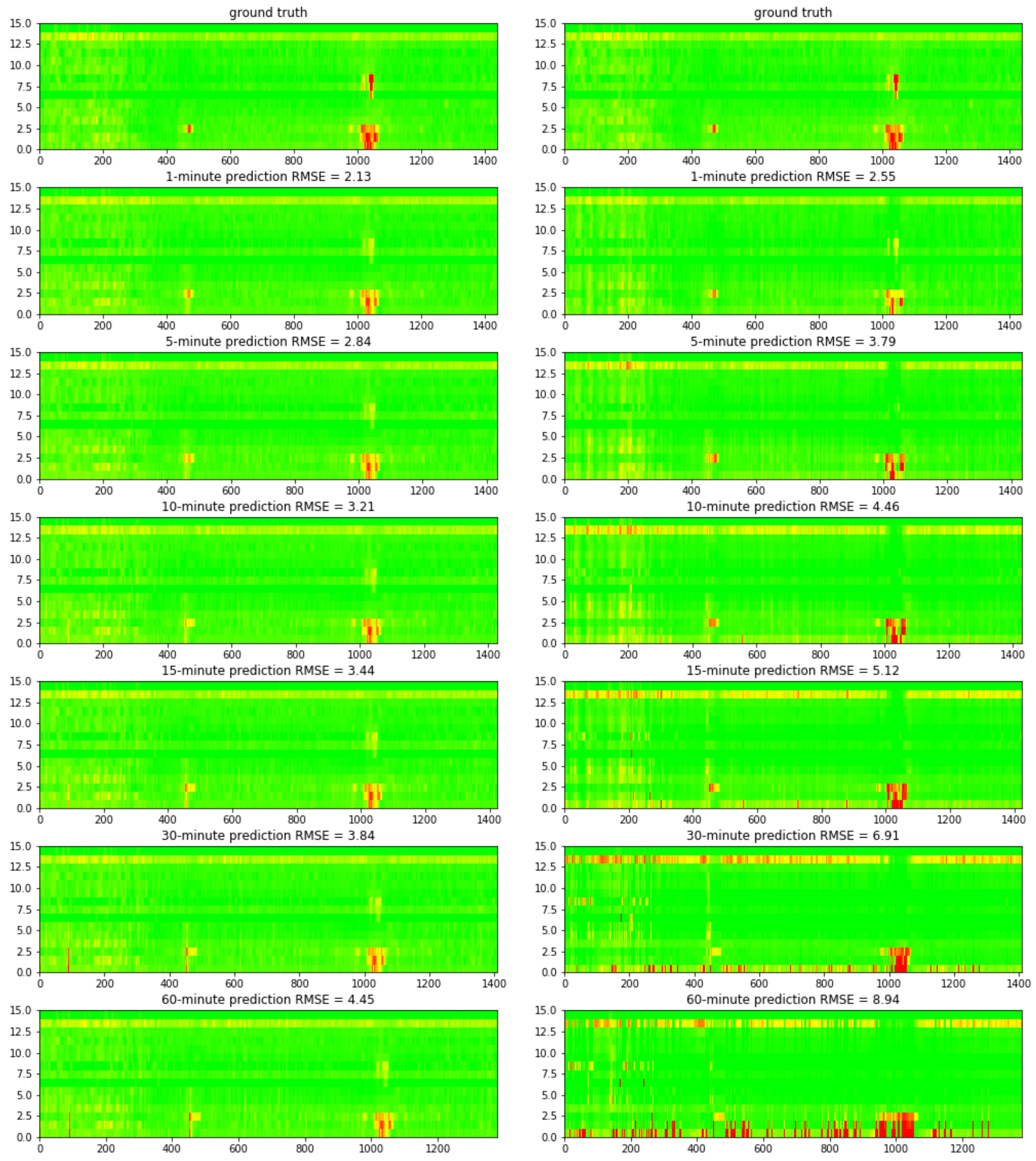


Figure 5.6   Model results on test day 1, 12-02-2016 Friday

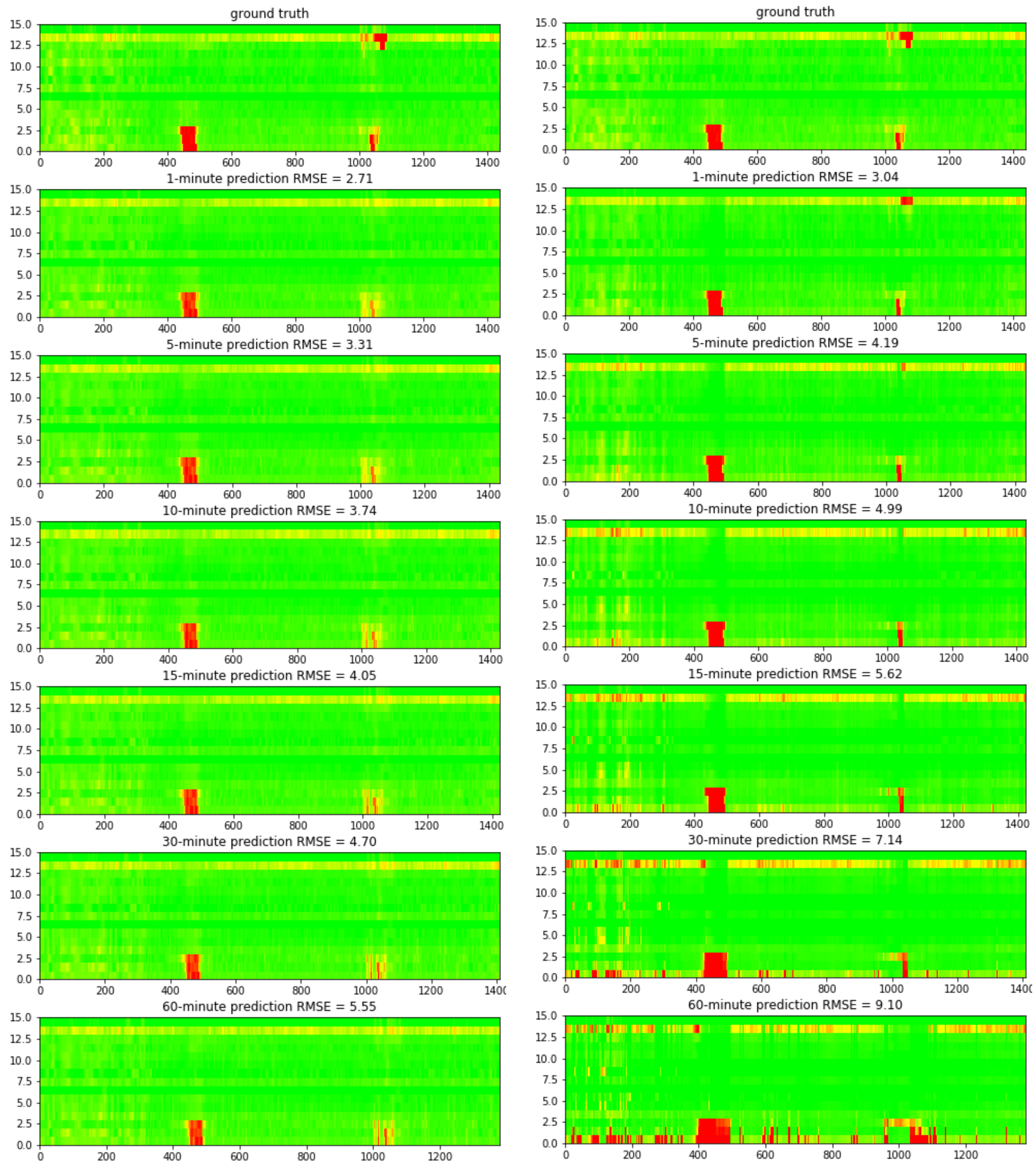


Figure 5.7 Model results on test day 2, 12-05-2016 Monday

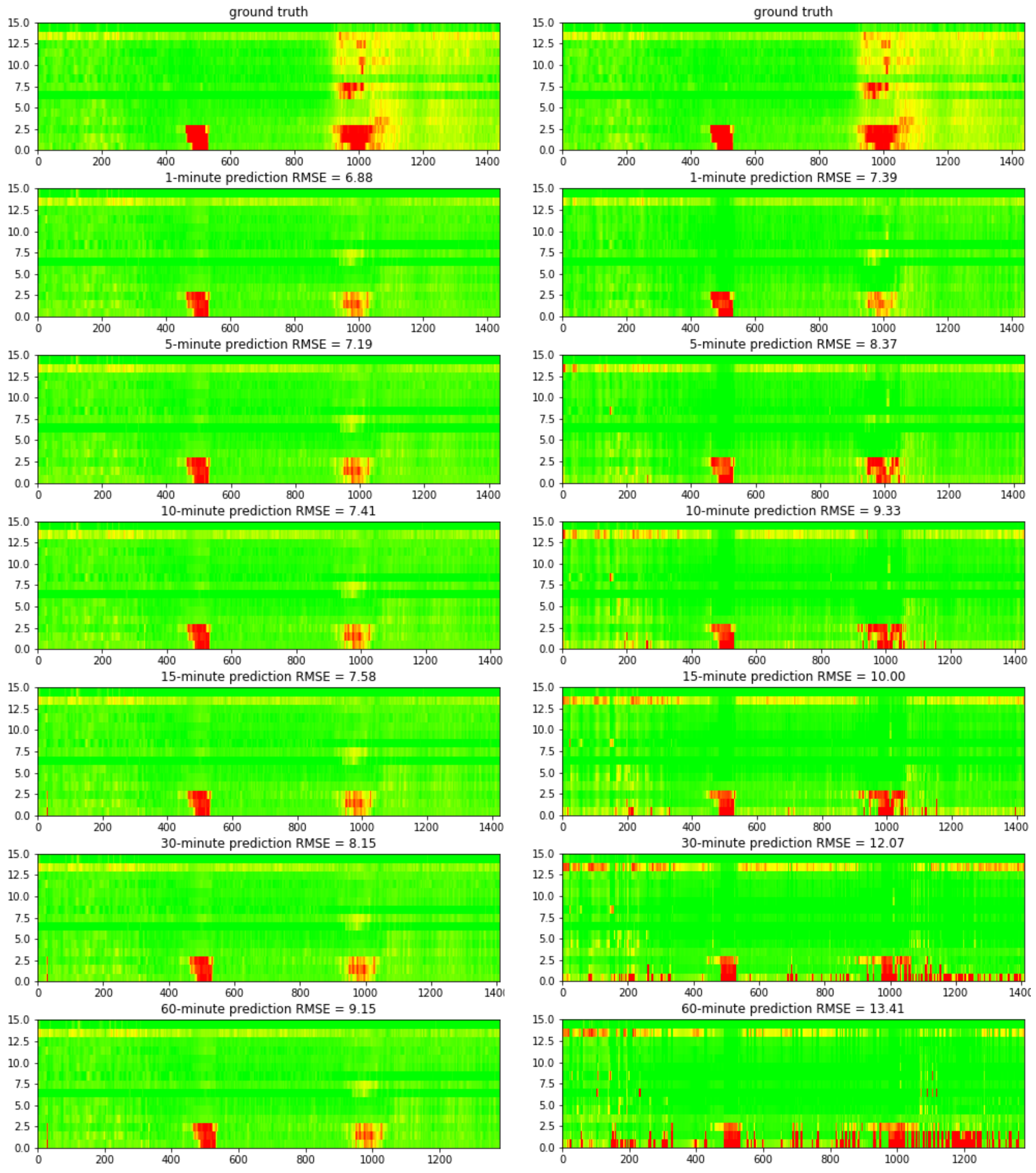


Figure 5.8 Model results on test day 3, 12-16-2016 Friday

CHAPTER 6. CONCLUSIONS

The objective of the proposed study is to provide traffic management the predicted traffic speeds at a route level to operate proactively.

Traffic speed prediction analysis using deep learning can benefit from the large amount of historical data. The past observations are the ground truth predictions at the historical timestamps so that although a supervised method is used to train a deep network for traffic speed prediction, there is no manual labeling needed. The large amount of historical data are all we need. Traffic information collected from RTMSs and weather data provided by IEM are the two sources used in this study. To accommodate large amount of statewide data and to provide data preprocessing in a timely manner, HDFS is used as a large-capacity fault-tolerant distributed file system and MapReduce is adopted as a parallel computing framework in this study.

Data collected from fifteen locations on eastbound I-235 in Des Moines metropolitan area from September, 2015 to the end of 2016 totaling 447 days are used in this study. Due to random connection failure and sensor errors, about 5.6% of traffic data is missing. The missing data are filled with average value by sensor location, time of day and day of week throughout the study period. Data is then smoothed by a moving average method with a window length of five minutes.

A fully convolutional deep network is developed to make long-term traffic speed prediction from both weather information and historical traffic speeds. CNN layers are used so that the network is able to effectively extract features from the spatial-temporal structure in the input data and predict traffic speed at a route level. Only CNN layers are used so that the network is independent on the spatial-temporal dimension of the input data so that a pretrained model can be transferred on traffic speed prediction at difference scales. The proposed fully convolutional deep network is optimized by using both 3-D CNN layer and one-by-one CNN layers so that the entire model is deep while the number of weights is reduced. Different number of filters are tested and the model structure with the best performance are proposed. The testing experiments on three test days demonstrate the proposed model can effectively capture the dynamics from both two input sources to make predictions. The testing experiments also demonstrate that a model trained on two-hour patches can make an accurate whole day prediction directly from a whole day input.

A hybrid LSTM network is developed to make short-term traffic speed predictions from both historical traffic speeds and traffic speeds in the current day. LSTM structure is used so that the long-term memory can be well maintained while the timestamp keeps rolling forward. The LSTM memory will not be reset until the end of the day so that the observed traffic speeds in the current day from the beginning of the day are taken into account to predict the traffic speeds at the next time stamp. The impact of historical input is tested and the testing cases demonstrate that with the impact of historical input, the model tends to have lower latency in terms of predicting recurrent congestions. The recurrent LSTM model structure gives the model ability to predict arbitrary number of timestamps ahead by using the previous prediction as input and rolling forward. The testing experiments demonstrates that a model trained to predict only one timestamp ahead can effectively predict multiple timestamps ahead using the aforementioned mechanism.

The long-term traffic prediction model and short-term traffic prediction model are superior in different aspects when comparing with each other. The short-term prediction model provides much higher accuracy than the long-term prediction model but it can only predict traffic speed in a very short time scale. The long-term prediction model can provide hours to days of buffer time between the current timestamp and the predicted timestamp but it compromises on the prediction accuracy. A combination of the two models can take advantage of the strength from both aspects and provide an effective multilayer decision supporting system and both models are compatible to become a live application.

There are also many aspects where the study can be improved in the future. Data is the fundamental ingredients of deep learning. The more validate data the better a deep network can be trained. In this study, only data from one route is used due to practical limitations. The adopted distributed file system and parallel computing framework is scalable to larger amount of data. Thus in the future when more data is available the deep models can potentially gain higher prediction power.

The proposed fully convolutional deep network structure can be applied to any spatial-temporal dimension. This study applies the model to a fixed route and demonstrates it can be applied on multiple temporal scales. Future studies can dig deeper in applying the model in different spatial scales. Following the experiment design in this study, one use case in exploring the spatial transferability could be applying a model pre-trained on a short route to a longer route. One hidden assumption when apply the same model to different spatial scales is that the geospatial dependencies do not change at those different spatial scales. Sometimes this assumption does not hold because the consecutive sensors at different locations can be differently spaced and the segment type may change from regular to merging to diverging to weaving segments. One way to give the model a better ability to learn the varying geospatial dependencies is to parameterize the CNN filter shape, such as dilated CNN or deformable CNN filter.

A different training strategy may significantly improve the performance of a deep network. The training strategy used in this study for the hybrid LSTM model is to train it to predict traffic speed at only the next timestamp. Although the testing experiments demonstrate that the model trained in such a way can effectively predict multiple timestamps ahead, but a different training strategy such as training the model to predict a certain number of timestamps ahead may improve the prediction power. A different loss function could also bring different behavior to a model. The future study may use a weighted RMSE as the loss function giving higher weighted to peak hours or amplifying the error when the actual traffic speed is low so that the model may better predict congestions.

REFERENCES

Abdulhai, B., Porwal, H., & Recker, W. (1999). Short term freeway traffic flow prediction using genetically-optimized time-delay-based neural networks. https://escholarship.org/uc/item/4t05p2mp

Alecsandru, C., & Ishak, S. (2004). Hybrid model-based and memory-based traffic prediction system. *Transportation Research Record: Journal of the Transportation Research Board*, (1879), 59–70.

Amin, S. M., Rodin, E. Y., Liu, A.-P., Rink, K., & García-Ortiz, A. (1998). Traffic prediction and management via RBF neural nets and semantic control. *Computer-Aided Civil and Infrastructure Engineering*, *13*(5), 315–327. https://doi.org/10.1111/0885-9507.00110

Asif, M. T., Dauwels, J., Goh, C. Y., Oran, A., Fathi, E., Xu, M., Dhanya, M. M., Mitrovic, N., & Jaillet, P. (2014). Spatiotemporal patterns in large-scale traffic speed prediction. *IEEE Transactions on Intelligent Transportation Systems*, *15*(2), 794–804. https://doi.org/10.1109/TITS.2013.2290285

Boto-Giralda, D., Díaz-Pernas, F. J., González-Ortega, D., Díez-Higuera, J. F., Antón-Rodríguez, M., Martínez-Zarzuela, M., & Torre-Díez, I. (2010). Wavelet-based denoising for traffic volume time series forecasting with self-organizing neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *25*(7), 530–545. https://doi.org/10.1111/j.1467-8667.2010.00668.x

Camus, R., Cantarella, G. E., & Inaudi, D. (1994). O-D prediction for real-time traffic management. *IFAC Proceedings Volumes*, *27*(12), 707–711. https://doi.org/10.1016/S1474-6670(17)47555-0

Castro-Neto, M., Jeong, Y.-S., Jeong, M.-K., & Han, L. D. (2009). Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, *36*(3, Part 2), 6164–6173. https://doi.org/10.1016/j.eswa.2008.07.069

Cetin, M., & Comert, G. (2006). Short-term traffic flow prediction with regime switching models. *Transportation Research Record: Journal of the Transportation Research Board*, (1965), 23–31.

Chandra, S., & 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. https://doi.org/10.3141/2061-08

Chandra, S. R., & Al-Deek, H. (2009). Predictions of freeway traffic speeds and volumes using vector autoregressive models. *Journal of Intelligent Transportation Systems*, *13*(2), 53–72. https://doi.org/10.1080/15472450902858368

Chen, C., Hu, J., Meng, Q., & Zhang, Y. (2011). Short-time traffic flow prediction with ARIMA-GARCH model. In *2011 IEEE Intelligent Vehicles Symposium (IV)* (pp. 607–612). https://doi.org/10.1109/IVS.2011.5940418

Chen, D. (2017). Research on traffic flow prediction in the big data environment based on the improved RBF neural network. *IEEE Transactions on Industrial Informatics*, *13*(4), 2000–2008. https://doi.org/10.1109/TII.2017.2682855

Chen, H., Grant-Muller, S., Mussone, L., & Montgomery, F. (2001). A study of hybrid neural network approaches and the effects of missing data on traffic forecasting. *Neural Computing & Applications*, *10*(3), 277–286. https://doi.org/10.1007/s521-001-8054-3

Chen, X. Y., Pao, H. K., & Lee, Y. J. (2014). Efficient traffic speed forecasting based on massive heterogeneous historical data. In *2014 IEEE International Conference on Big Data (Big Data)* (pp. 10–17). https://doi.org/10.1109/BigData.2014.7004425

Chen, Y., Lv, Y., Li, Z., & Wang, F. (2016). Long short-term memory model for traffic congestion prediction with online open data. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 132–137. IEEE.

Chien, S. I.-J., & Kuchipudi, C. M. (2003). Dynamic travel time prediction with real-time and historic data. *Journal of Transportation Engineering*, *129*(6), 608–616.

Chien, S., Liu, X., & Ozbay, K. (2003). Predicting travel times for the South Jersey real-time motorist information system. *Transportation Research Record: Journal of the Transportation Research Board*, (1855), 32–40.

Clark, S. (2003). Traffic prediction using multivariate nonparametric regression. *Journal of Transportation Engineering*, *129*(2), 161–168.

Coufal, D., & Turunen, E. (2003). Short term prediction of highway travel time using data mining and neuro-fuzzy methods. In *Proc. IFSA* (pp. 175–182).

Crittin, F., & Bierlaire, M. (2002). An efficient algorithm for real-time estimation and prediction of dynamic OD table. In *Swiss Transport Research Conference*.

Cui, Z., Ke, R., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. *ArXiv Preprint ArXiv:1801.02143*.

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, *28*(4), 269–287.

Dia, H. (2001). An object-oriented neural network approach to short-term traffic forecasting. *European Journal of Operational Research*, *131*(2), 253–261. https://doi.org/10.1016/S0377-2217(00)00125-9

Duan, Y., Lv, Y., & Wang, F.-Y. (2016). Travel time prediction with LSTM neural network. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC),* pp. 1053–1058. IEEE.

Fabritiis, C. de, Ragona, R., & Valenti, G. (2008). Traffic estimation and prediction based on real time floating car data. In *2008 11th International IEEE Conference on Intelligent Transportation Systems* (pp. 197–203). https://doi.org/10.1109/ITSC.2008.4732534

Fu, R., Zhang, Z., & Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)* (pp. 324–328). https://doi.org/10.1109/YAC.2016.7804912

Ghosh, B., Basu, B., & O'Mahony, M. (2009). Multivariate short-term traffic flow forecasting using time-series analysis. *IEEE Transactions on Intelligent Transportation Systems*, *10*(2), 246–254. https://doi.org/10.1109/TITS.2009.2021448

Gopi, G., Dauwels, J., Asif, M. T., Ashwin, S., Mitrovic, N., Rasheed, U., & Jaillet, P. (2013). Bayesian support vector regression for traffic speed prediction with error bars. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)* (pp. 136–141). https://doi.org/10.1109/ITSC.2013.6728223

Guo, J., Huang, W., & 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. https://doi.org/10.1016/j.trc.2014.02.006

Habtie, A. B., Abraham, A., & Midekso, D. (2017). Artificial neural network based real-time urban road traffic state estimation framework. In *Computational Intelligence in Wireless Sensor Networks* (pp. 73–97). Springer, Cham. https://doi.org/10.1007/978-3-319-47715-2_4

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

Handley, S., Langley, P., & Rauscher, F. A. (1998). Learning to predict the duration of an automobile trip. In *KDD*. pp. 219–223.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science (New York, N.Y.)*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hong, W. C. (2011). Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm. *Neurocomputing*, *74*(12), 2096–2107. https://doi.org/10.1016/j.neucom.2010.12.032

Huang, S. H., & Ran, B. (2003). An application of neural network on traffic speed prediction under adverse weather condition. Presented at the *Transportation Research Board 82nd Annual Meeting.* Transportation Research Board. https://trid.trb.org/view/646541

Huang, W., Song, G., Hong, H., & Xie, K. (2014). Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, *15*(5), 2191–2201. https://doi.org/10.1109/TITS.2014.2311123

Ishak, S., Kotha, P., & 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, Y., Wu, J., Ben-Akiva, M., Seshadri, R., & Du, Y. (2017). Rainfall-integrated traffic speed prediction using deep learning method. *IET Intelligent Transport Systems*, *11*(9), 531–536. https://doi.org/10.1049/iet-its.2016.0257

Jia, Y., Wu, J., & Du, Y. (2016). Traffic speed prediction using deep learning method. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC),* pp. 1217–1222. IEEE.

Jiang, X., & Adeli, H. (2005). Dynamic wavelet neural network model for traffic flow forecasting. *Journal of Transportation Engineering*, *131*(10), 771–779.

Jiang, H., Zou, Y., Zhang, S., Tang, J., & Wang, Y. (2016). Short-term speed prediction using remote microwave sensor data: Machine learning versus statistical model. *Mathematical Problems in Engineering*, vol. 2016. https://doi:10.1155/2016/9236156

Kamarianakis, Y., & Prastacos, P. (2003). Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transportation Research Record: Journal of the Transportation Research Board*, (1857), 74–84.

Kamarianakis, Y., Kanas, A., & Prastacos, P. (2005). Modeling traffic volatility dynamics in an urban network. *Transportation Research Record: Journal of the Transportation Research Board*, (1923), 18–27.

Kindzerske, M., & Ni, D. (2007). Composite nearest neighbor nonparametric regression to improve traffic prediction. *Transportation Research Record: Journal of the Transportation Research Board*, *1993*, 30–35. https://doi.org/10.3141/1993-05

Kuchipudi, C., & Chien, S. (2003). Development of a hybrid model for dynamic travel-time prediction. *Transportation Research Record: Journal of the Transportation Research Board*, (1855), 22–31.

Kwon, E., & Stephanedes, Y. J. (1994). Comparative evaluation of adaptive and neural-network exit demand prediction for freeway control. *Transportation Research Record*, 66–66.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., & Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.

Lee, E.-M., Kim, J.-H., & Yoon, W.-S. (2007). Traffic speed prediction under weekday, time, and neighboring links' speed: Back propagation neural network approach. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues* (pp. 626–635). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74171-8_62

Lee, S., & Fambro, D. (1999). Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, (1678), 179–188.

Lee, S., Lee, Y.-I., & Cho, B. (2006). Short-term travel speed prediction models in car navigation systems. *Journal of Advanced Transportation*, *40*(2), 122–139. https://doi.org/10.1002/atr.5670400203

Lee, Y. (2009). Freeway travel time forecast using artificial neural networks with cluster method. In *2009 12th International Conference on Information Fusion* (pp. 1331–1338).

Li, K. L., Zhai, C. J., & Xu, J. M. (2017). Short-term traffic flow prediction using a methodology based on ARIMA and RBF-ANN. In *2017 Chinese Automation Congress (CAC)* (pp. 2804–2807). https://doi.org/10.1109/CAC.2017.8243253

Li, L., He, S., Zhang, J., & Ran, B. (2016). Short-term highway traffic flow prediction based on a hybrid strategy considering temporal–spatial information. *Journal of Advanced Transportation*, *50*(8), 2029–2040. https://doi.org/10.1002/atr.1443

Lingras, P., & Mountford, P. (2001). Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting. In *Engineering of Intelligent Systems* (pp. 290–299). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45517-5_33

Lint, J. W. C. van. (2008). Online learning solutions for freeway travel time prediction. *IEEE Transactions on Intelligent Transportation Systems*, *9*(1), 38–47. https://doi.org/10.1109/TITS.2008.915649

Lint, J. van, Hoogendoorn, S., & 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.

Liu, Y., Yang, X., Ma, W., & Wang, Y. (2018). A deep ensemble learning method for short-term travel time prediction. Presented at the Transportation Research Board 97th Annual Meeting Transportation Research Board. https://trid.trb.org/view/1497195

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y. (2015). Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 865–873. https://doi.org/10.1109/TITS.2014.2345663

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., & Wang, Y. (2017). Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, *17*(4), 818. https://doi.org/10.3390/s17040818

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015b). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, *54*, 187–197. https://doi.org/10.1016/j.trc.2015.03.014

Ma, X., Yu, H., Wang, Y., & Wang, Y. (2015a). Large-scale transportation network congestion evolution prediction using deep learning theory. *PloS One*, *10*(3), e0119044. https://doi.org/10.1371/journal.pone.0119044

Moretti, F., Pizzuti, S., Panzieri, S., & Annunziato, M. (2015). Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling. *Neurocomputing*, *167*, 3–7. https://doi.org/10.1016/j.neucom.2014.08.100

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

Oswald, R. K., Scherer, W. T., & Smith, B. L. (2000). Traffic flow forecasting using approximate nearest neighbor nonparametric regression. Retrieved from https://trid.trb.org/view/661764

Park, B., Messer, C., & Urbanik II, T. (1998). Short-term freeway traffic volume forecasting using radial basis function neural network. *Transportation Research Record: Journal of the Transportation Research Board*, (1651), 39–47.

Park, D., & Rilett, L. (1998). Forecasting multiple-period freeway link travel times using modular neural networks. *Transportation Research Record: Journal of the Transportation Research Board*, (1617), 163–170.

Park, J., Li, D., Murphey, Y. L., Kristinsson, J., McGee, R., Kuang, M., & Phillips, T. (2011). Real time vehicle speed prediction using a neural network traffic model. In *The 2011 International Joint Conference on Neural Networks* (pp. 2991–2996). https://doi.org/10.1109/IJCNN.2011.6033614

Pavlyuk, D. (2017). Short-term traffic forecasting using multivariate autoregressive models. *Procedia Engineering*, *178*, 57–66. https://doi.org/10.1016/j.proeng.2017.01.062

Qi, Y., & Ishak, S. (2013). Stochastic approach for short-term freeway traffic prediction during peak periods. *IEEE Transactions on Intelligent Transportation Systems*, *14*(2), 660–672. https://doi.org/10.1109/TITS.2012.2227475

Smith, B. L., & Demetsky, M. J. (1994). Short-term traffic flow prediction: Neural network approach. *Transportation Research Record*, (1453). https://trid.trb.org/view/424677

Smith, B., & Demetsky, M. (1996). Multiple-interval freeway traffic flow forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, (1554), 136–141.

Smith, B. L., & Oswald, R. K. (2003). Meeting real-time traffic flow forecasting requirements with imprecise computations. *Computer-Aided Civil and Infrastructure Engineering*, *18*(3), 201–213. https://doi.org/10.1111/1467-8667.00310

Smith, B. L., Williams, B. M., & Oswald, R. K. (2002). Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, *10*(4), 303–321.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. *ArXiv:1409.4842 [Cs]*. http://arxiv.org/abs/1409.4842

Szeto, W. Y., Ghosh, B., Basu, B., & O'Mahony, M. (2009). Multivariate traffic forecasting technique using cell transmission model and SARIMA model. *Journal of Transportation Engineering*, *135*(9), 658–667.

Tan, H., Xuan, X., Wu, Y., Zhong, Z., & Ran, B. (2016). A comparison of traffic flow prediction methods based on DBN. In *CICTP 2016* (pp. 273–283).

Tang, J., Liu, F., Zou, Y., Zhang, W., & 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. https://doi.org/10.1109/TITS.2016.2643005

Texas Transportation Institute. (2007). *2007 Urban Mobility Report*. College Station, TX: Texas A&M University.

Vanajakshi, L., & Rilett, L. R. (2007). Support vector machine technique for the short term prediction of travel time. In *2007 IEEE Intelligent Vehicles Symposium* (pp. 600–605). https://doi.org/10.1109/IVS.2007.4290181

Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C. (2005). Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. *Transportation Research Part C: Emerging Technologies*, *13*(3), 211–234. https://doi.org/10.1016/j.trc.2005.04.007

Wang, H., Liu, L., Qian, Z., Wei, H., & Dong, S. (2014). Empirical mode decomposition-autoregressive integrated moving average: Hybrid short-term traffic speed prediction model. *Transportation Research Record: Journal of the Transportation Research Board*, *2460*, 66–76. https://doi.org/10.3141/2460-08

Wang, J., & Shi, Q. (2013). Short-term traffic speed forecasting hybrid model based on Chaos–Wavelet Analysis-Support Vector Machine theory. *Transportation Research Part C: Emerging Technologies*, *27*, 219–232. https://doi.org/10.1016/j.trc.2012.08.004

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

Wang, Y., Papageorgiou, M., & Messmer, A. (2006). RENAISSANCE—A unified macroscopic model-based approach to real-time freeway network traffic surveillance. *Transportation Research Part C: Emerging Technologies*, *14*(3), 190–212. https://doi.org/10.1016/j.trc.2006.06.001

Wang, Y., Wang, H., & Xia, L. (2005). Highway traffic prediction with neural network and genetic algorithms. In *IEEE International Conference on Vehicular Electronics and Safety,* pp. 211–216. IEEE.

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

Williams, B. (2001). Multivariate vehicular traffic flow prediction: Evaluation of ARIMAX modeling. *Transportation Research Record: Journal of the Transportation Research Board*, (1776), 194–200.

Williams, B. M., & 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.

Williams, B., Durvasula, P., & Brown, D. (1998). Urban freeway traffic flow prediction: Application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transportation Research Record: Journal of the Transportation Research Board*, (1644), 132–141.

Wu, C.-H., Ho, J.-M., & Lee, D. T. (2004). Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, *5*(4), 276–281. https://doi.org/10.1109/TITS.2004.837813

Wu, Y., & Tan, H. (2016). Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *ArXiv:1612.01022 [Cs]*. http://arxiv.org/abs/1612.01022

Xia, J. (2006). Dynamic freeway travel time prediction using single loop detector and incident data. *University of Kentucky Doctoral Dissertations*. 315. https://uknowledge.uky.edu/gradschool_diss/315

Xie, Y., & Zhang, Y. (2006). A wavelet network model for short-term traffic volume forecasting. *Journal of Intelligent Transportation Systems*, *10*(3), 141–150. https://doi.org/10.1080/15472450600798551

Xie, Y., Zhang, Y., & Ye, Z. (2007). Short-term traffic volume forecasting using Kalman filter with discrete wavelet decomposition. *Computer-Aided Civil and Infrastructure Engineering*, *22*(5), 326–334. https://doi.org/10.1111/j.1467-8667.2007.00489.x

Yang, F., Yin, Z., Liu, H., & Ran, B. (2004). Online recursive algorithm for short-term traffic prediction. *Transportation Research Record: Journal of the Transportation Research Board*, (1879), 1–8.

Yang, J.-S. (2005). Travel time prediction using the GPS test vehicle and Kalman filtering techniques. In *Proceedings of the 2005, American Control Conference, 2005.* (pp. 2128–2133 vol. 3). https://doi.org/10.1109/ACC.2005.1470285

Yasdi, R. (1999). Prediction of road traffic using a neural network approach. *Neural Computing & Applications*, *8*(2), 135–142. https://doi.org/10.1007/s005210050015

Ye, Q., Szeto, W. Y., & Wong, S. C. (2012). Short-term traffic speed forecasting based on data recorded at irregular intervals. *IEEE Transactions on Intelligent Transportation Systems*, *13*(4), 1727–1737. https://doi.org/10.1109/TITS.2012.2203122

Yi, H., Jung, H., & Bae, S. (2017). Deep neural networks for traffic flow prediction. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp),* (pp. 328–331). IEEE.

Yildirim, U., & Cataltepe, Z. (2008). Short time traffic speed prediction using data from a number of different sensor locations. In *2008 23rd International Symposium on Computer and Information Sciences* (pp. 1–6). https://doi.org/10.1109/ISCIS.2008.4717955

Yin, H., Wong, S. C., Xu, J., & Wong, C. K. (2002). Urban traffic flow prediction using a fuzzy-neural approach. *Transportation Research Part C: Emerging Technologies*, *10*(2), 85–98. https://doi.org/10.1016/S0968-090X(01)00004-3

You, J., & Kim, T. J. (2000). Development and evaluation of a hybrid travel time forecasting model. *Transportation Research Part C: Emerging Technologies*, *8*(1–6), 231–256.

Yu, H., Wu, Z., Wang, S., Wang, Y., & Ma, X. (2017). Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *ArXiv:1705.02699 [Cs]*. http://arxiv.org/abs/1705.02699

Yu, B., Yang, Z., & Yao, B. (2006). Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems*, *10*(4), 151–158. https://doi.org/10.1080/15472450600981009

Yun, S.-Y., Namkoong, S., Rho, J.-H., Shin, S.-W., & Choi, J.-U. (1998). A performance evaluation of neural network models in traffic volume forecasting. *Mathematical and Computer Modelling*, *27*(9–11), 293–310.

Zeng, D., Xu, J., Gu, J., Liu, L., & Xu, G. (2008). Short term traffic flow prediction using hybrid ARIMA and ANN models. In *2008 Workshop on Power Electronics and Intelligent Transportation System* (pp. 621–625). https://doi.org/10.1109/PEITS.2008.135

Zeng, X., & Zhang, Y. (2013). Development of recurrent neural network considering temporal-spatial input dynamics for freeway travel time modeling. *Computer-Aided Civil and Infrastructure Engineering*, *28*(5), 359–371. https://doi.org/10.1111/mice.12000

Zhang, L., Liu, Q., Yang, W., Wei, N., & Dong, D. (2013). An improved k-nearest neighbor model for short-term traffic flow prediction. *Procedia—Social and Behavioral Sciences*, *96*, 653–662. https://doi.org/10.1016/j.sbspro.2013.08.076

Zhang, Y., & Liu, Y. (2009). Traffic forecasting using least squares support vector machines. *Transportmetrica*, *5*(3), 193–213. https://doi.org/10.1080/18128600902823216

Zhang, Y., & Xie, Y. (2007). Forecasting of short-term freeway volume with v-support vector machines. *Transportation Research Record*, (2024), 92–99. https://doi.org/10.3141/2024-11

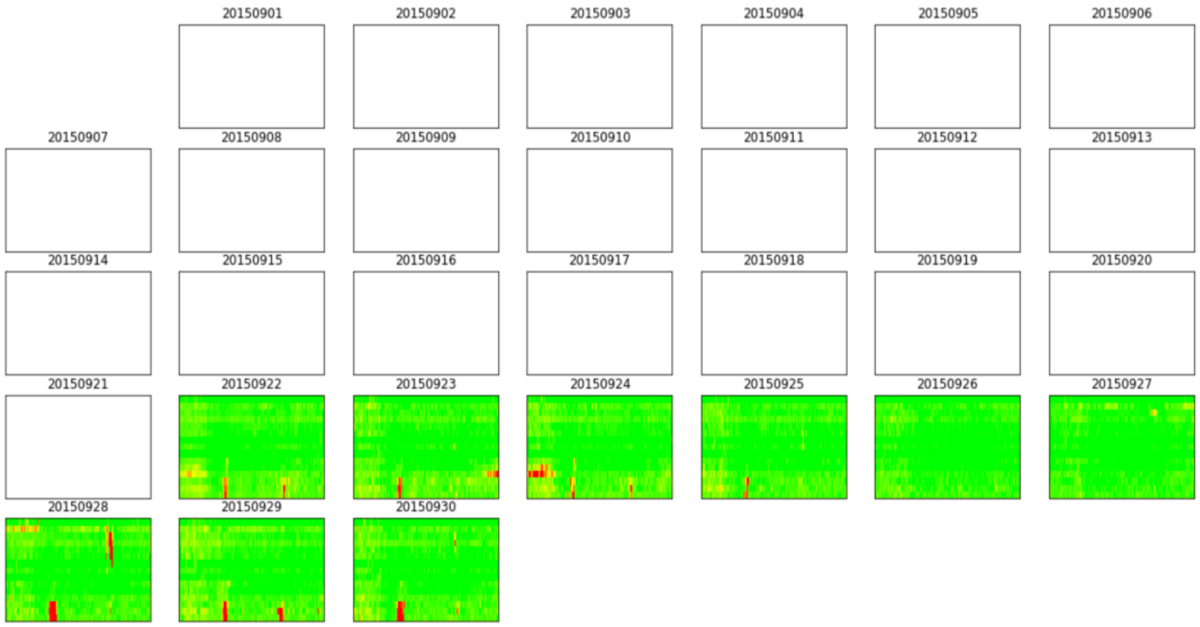
Zhao, Z., Chen, W., Wu, X., Chen, P. C. Y., & Liu, J. (2017). LSTM network: A deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, *11*(2), 68–75. https://doi.org/10.1049/iet-its.2016.0208

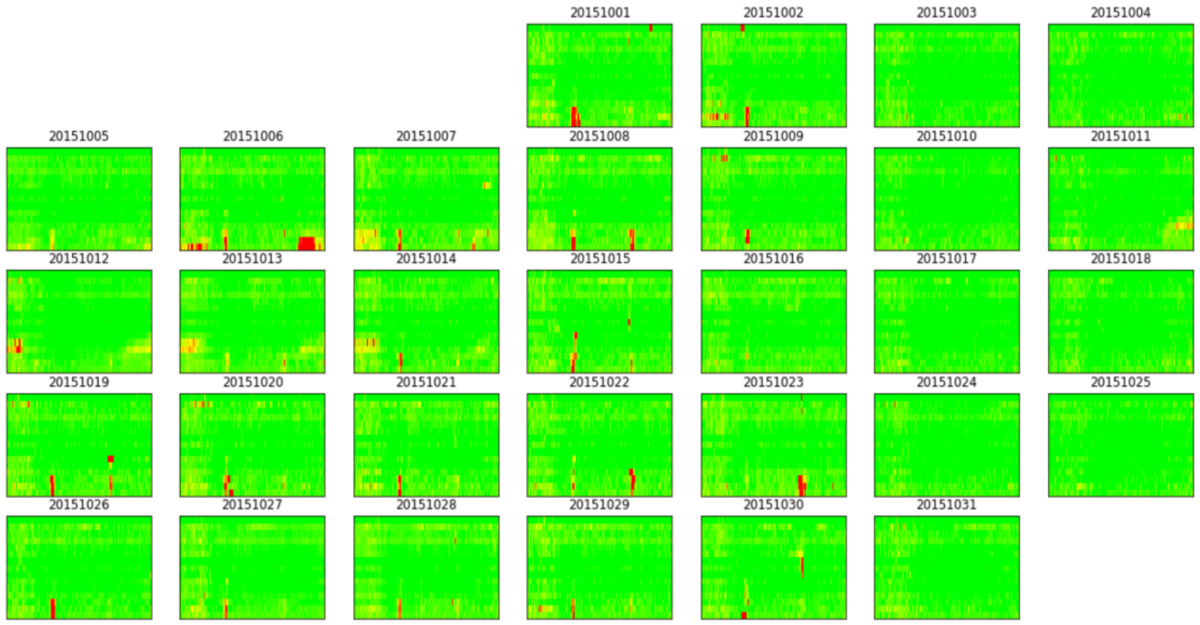
Zheng, W., Lee, D.-H., & Shi, Q. (2006). Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of Transportation Engineering*, *132*(2), 114–121.

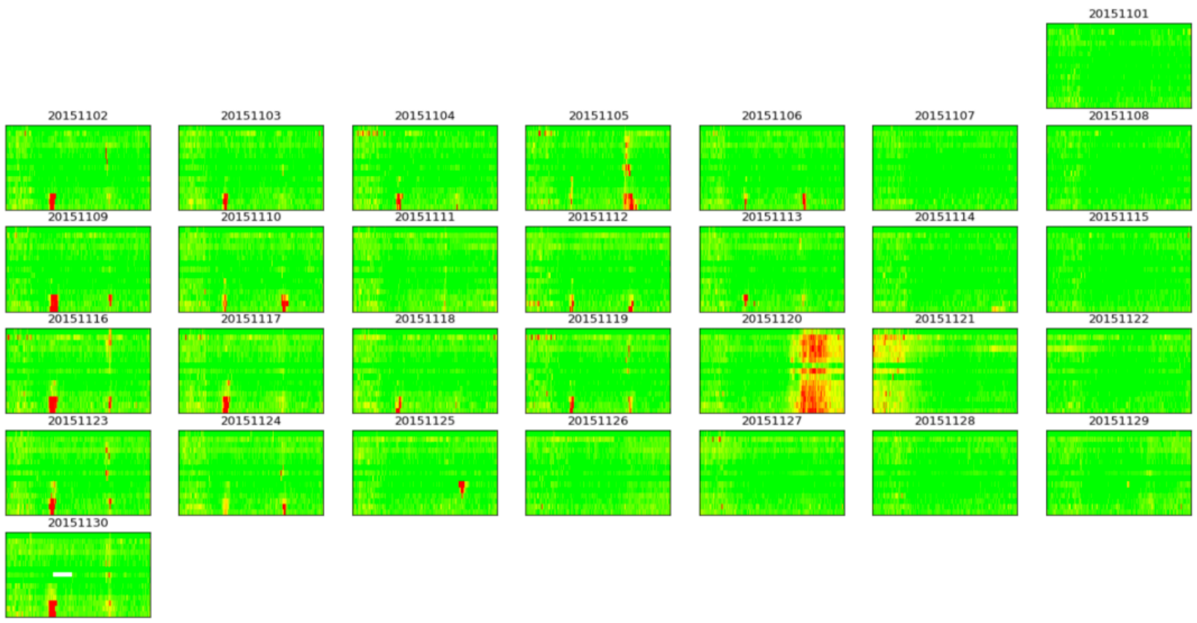
Zhong, J., & Ling, S. (2015). Key factors of k-nearest neighbours nonparametric regression in short-time traffic flow forecasting. In *Proceedings of the 21st International Conference on Industrial Engineering and Engineering Management 2014* (pp. 9–12). Paris: Atlantis Press. https://doi.org/10.2991/978-94-6239-102-4_2

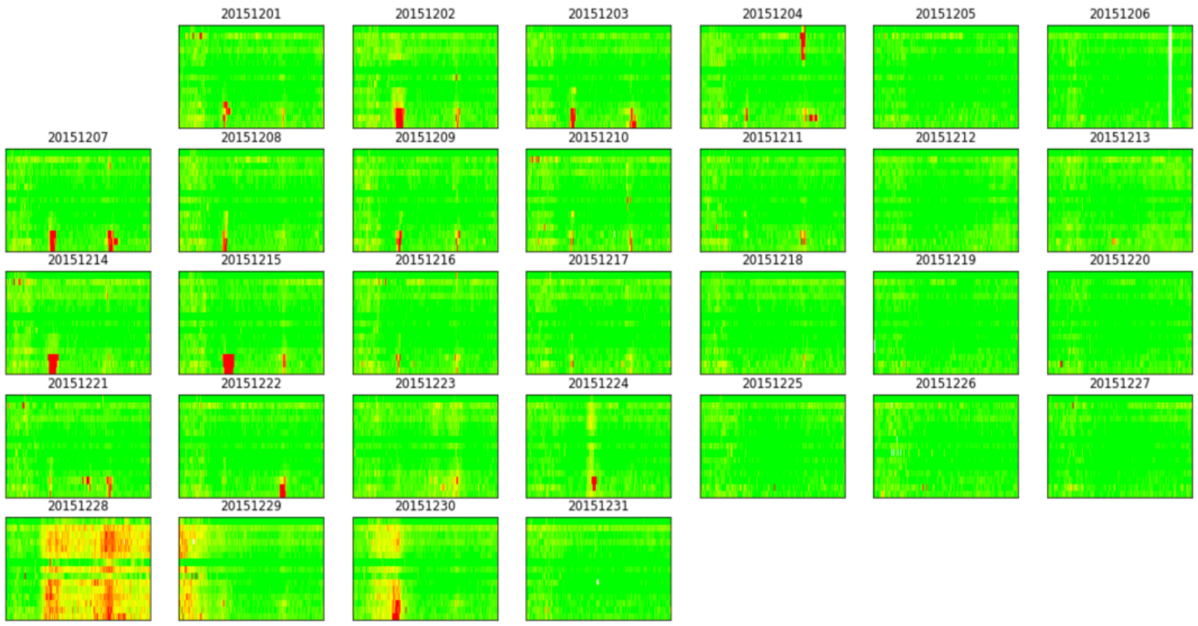
Zou, Y., Hua, X., Zhang, Y., & Wang, Y. (2015). Hybrid short-term freeway speed prediction methods based on periodic analysis. *Canadian Journal of Civil Engineering*, *42*(8), 570–582. https://doi.org/10.1139/cjce-2014-0447

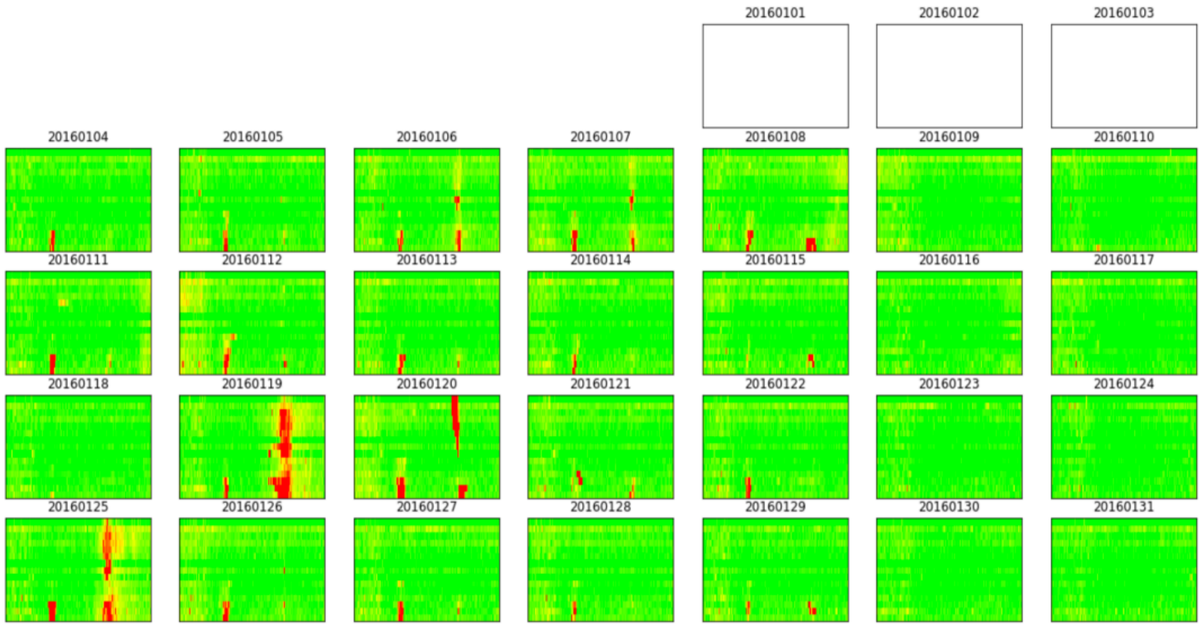
APPENDIX. CALENDAR SPEED PLOTS OF THE STUDIED DATASET

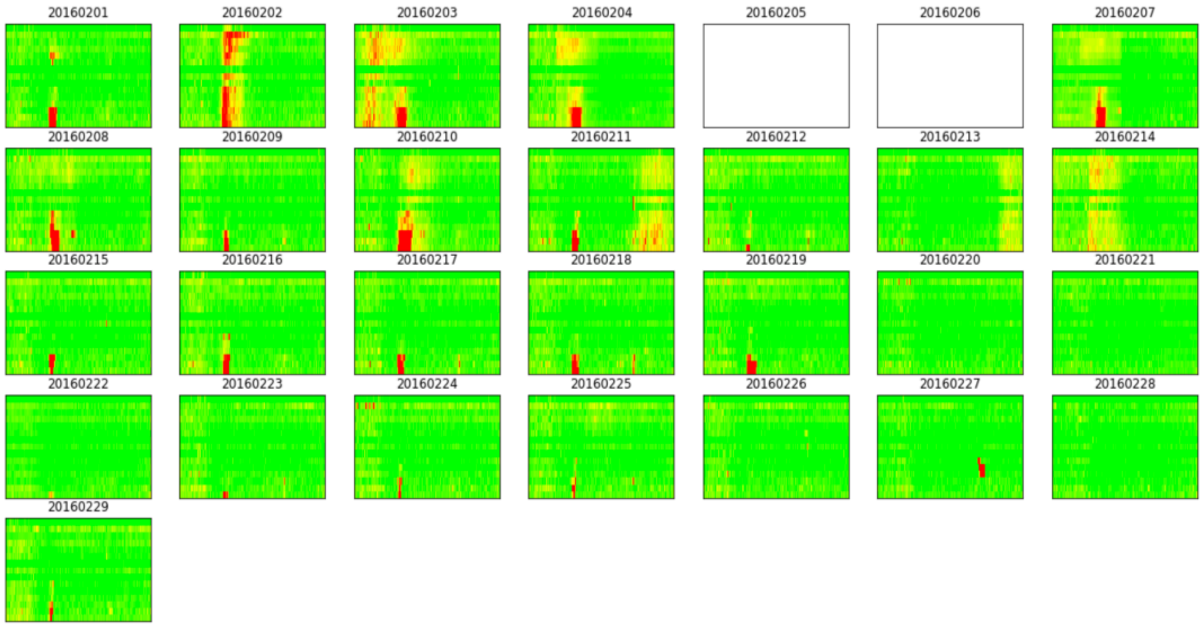


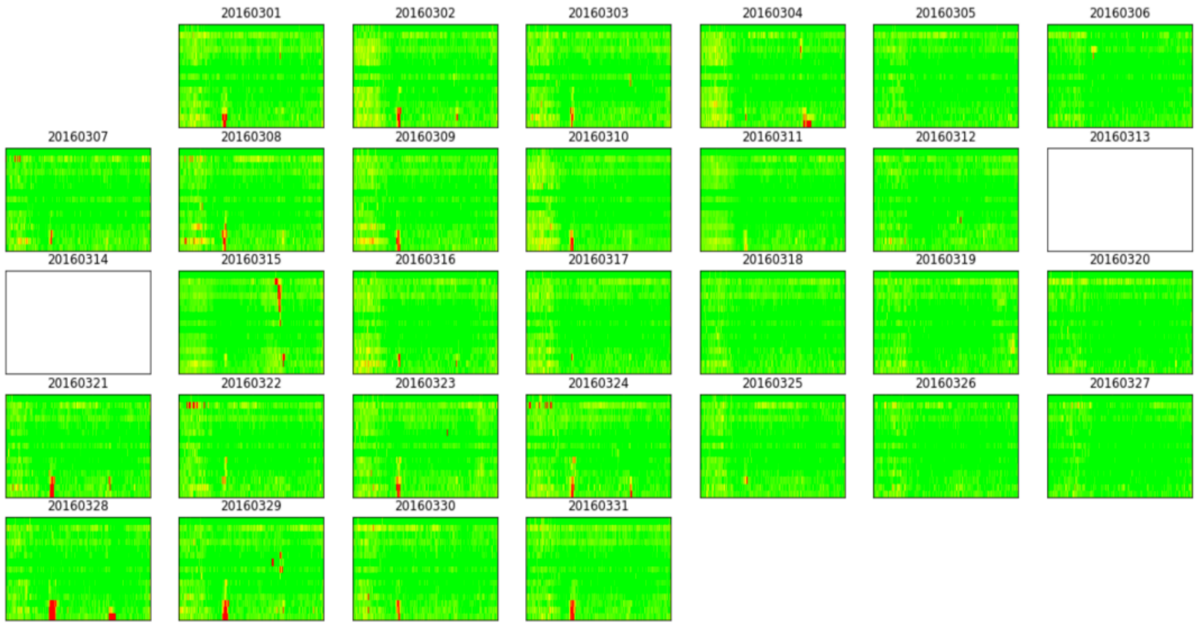


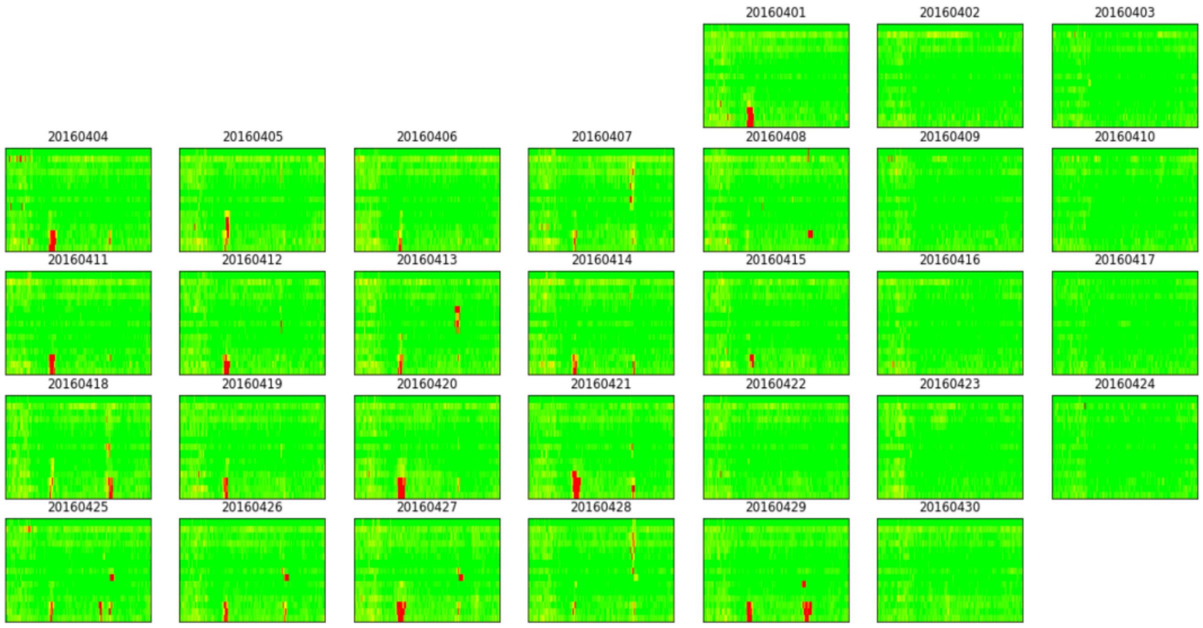

20151101
20151102
20151103
20151104
20151105
20151106
20151107
20151108
20151109
20151110
20151111
20151112
20151113
20151114
20151115
20151116
20151117
20151118
20151119
20151120
20151121
20151122
20151123
20151124
20151125
20151126
20151127
20151128
20151129
20151130

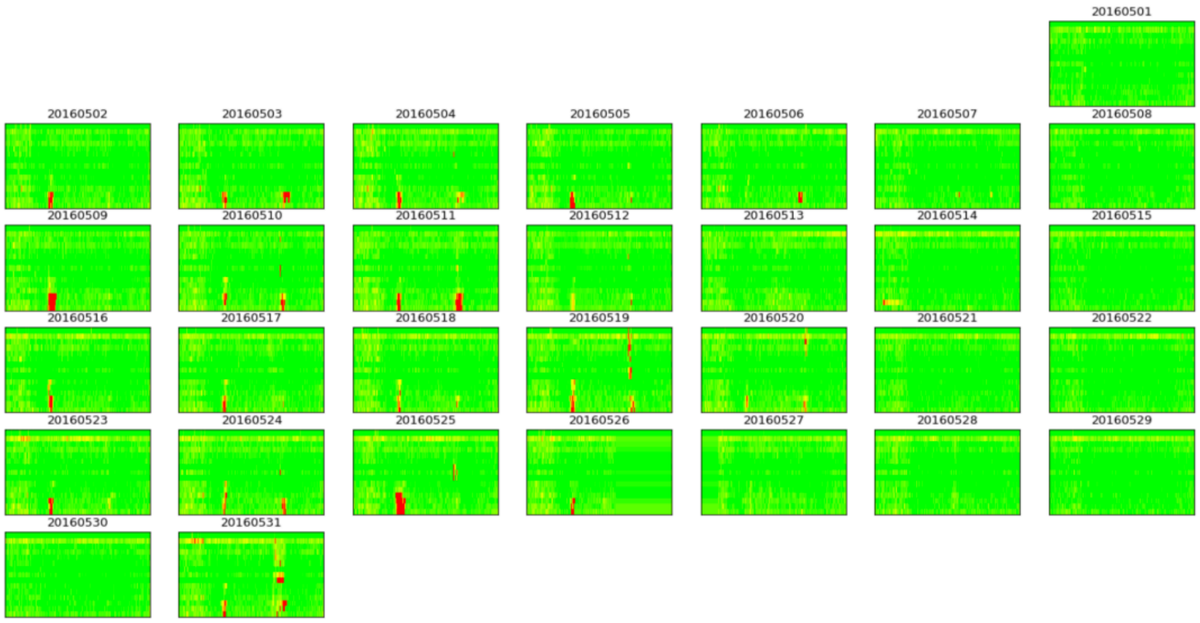

20151201
20151202
20151203
20151204
20151205
20151206
20151207
20151208
20151209
20151210
20151211
20151212
20151213
20151214
20151215
20151216
20151217
20151218
20151219
20151220
20151221
20151222
20151223
20151224
20151225
20151226
20151227
20151228
20151229
20151230
20151231

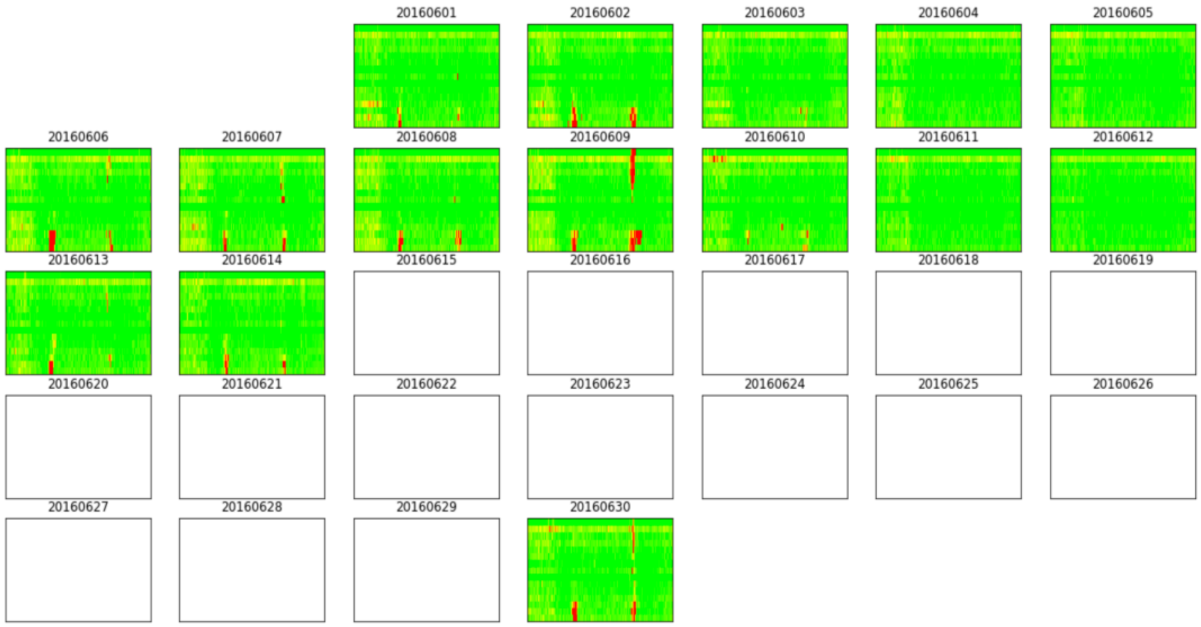

20160101
20160102
20160103
20160104
20160105
20160106
20160107
20160108
20160109
20160110
20160111
20160112
20160113
20160114
20160115
20160116
20160117
20160118
20160119
20160120
20160121
20160122
20160123
20160124
20160125
20160126
20160127
20160128
20160129
20160130
20160131

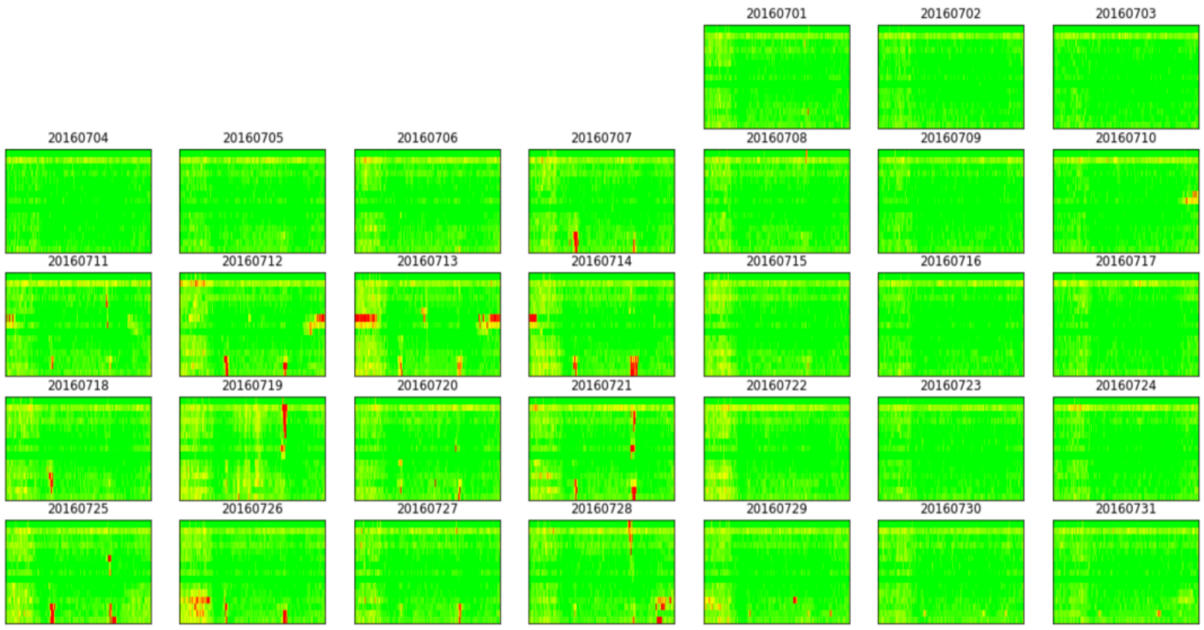

20160201
20160202
20160203
20160204
20160205
20160206
20160207
20160208
20160209
20160210
20160211
20160212
20160213
20160214
20160215
20160216
20160217
20160218
20160219
20160220
20160221
20160222
20160223
20160224
20160225
20160226
20160227
20160228
20160229

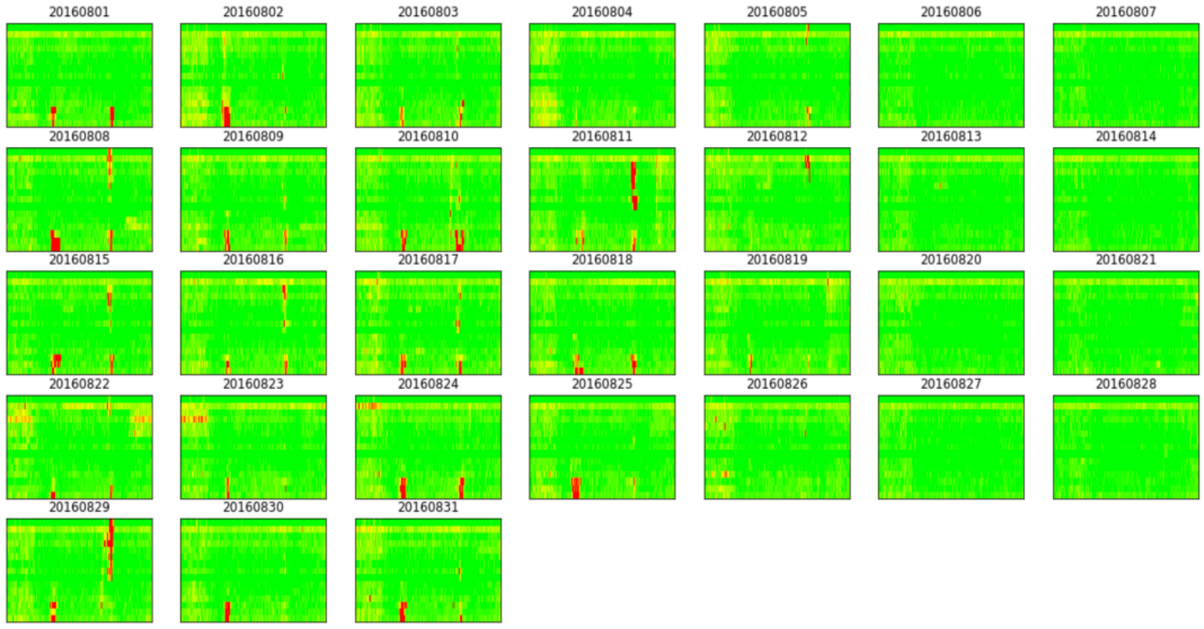

20160301
20160302
20160303
20160304
20160305
20160306
20160307
20160308
20160309
20160310
20160311
20160312
20160313
20160314
20160315
20160316
20160317
20160318
20160319
20160320
20160321
20160322
20160323
20160324
20160325
20160326
20160327
20160328
20160329
20160330
20160331

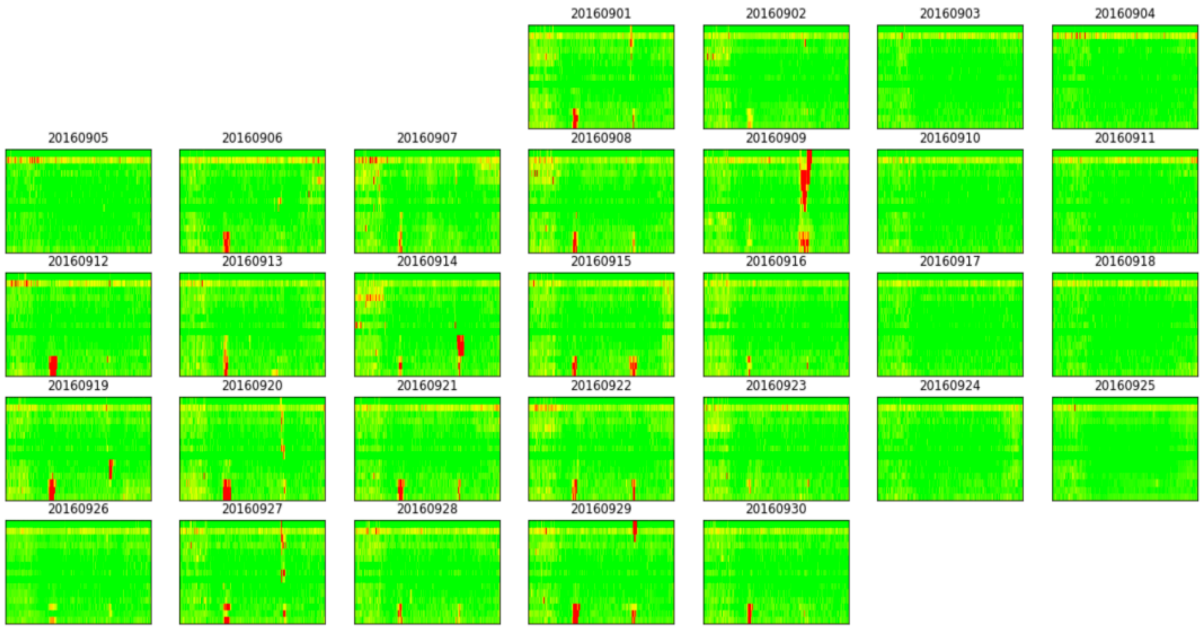

20160401
20160402
20160403
20160404
20160405
20160406
20160407
20160408
20160409
20160410
20160411
20160412
20160413
20160414
20160415
20160416
20160417
20160418
20160419
20160420
20160421
20160422
20160423
20160424
20160425
20160426
20160427
20160428
20160429
20160430

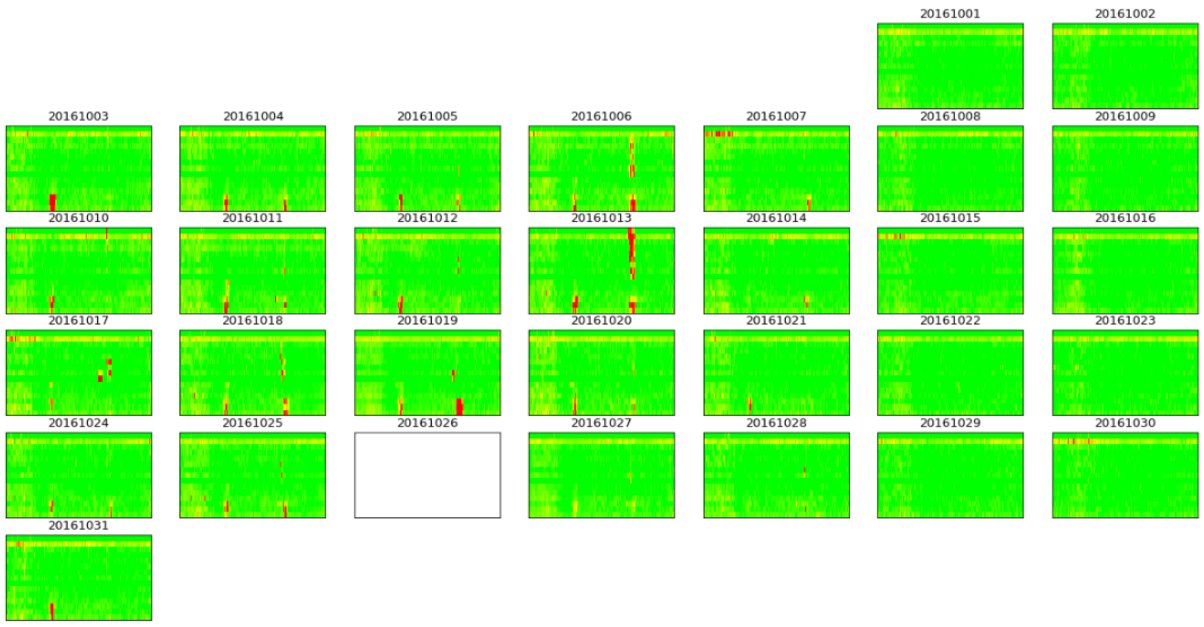

20160501
20160502
20160503
20160504
20160505
20160506
20160507
20160508
20160509
20160510
20160511
20160512
20160513
20160514
20160515
20160516
20160517
20160518
20160519
20160520
20160521
20160522
20160523
20160524
20160525
20160526
20160527
20160528
20160529
20160530
20160531

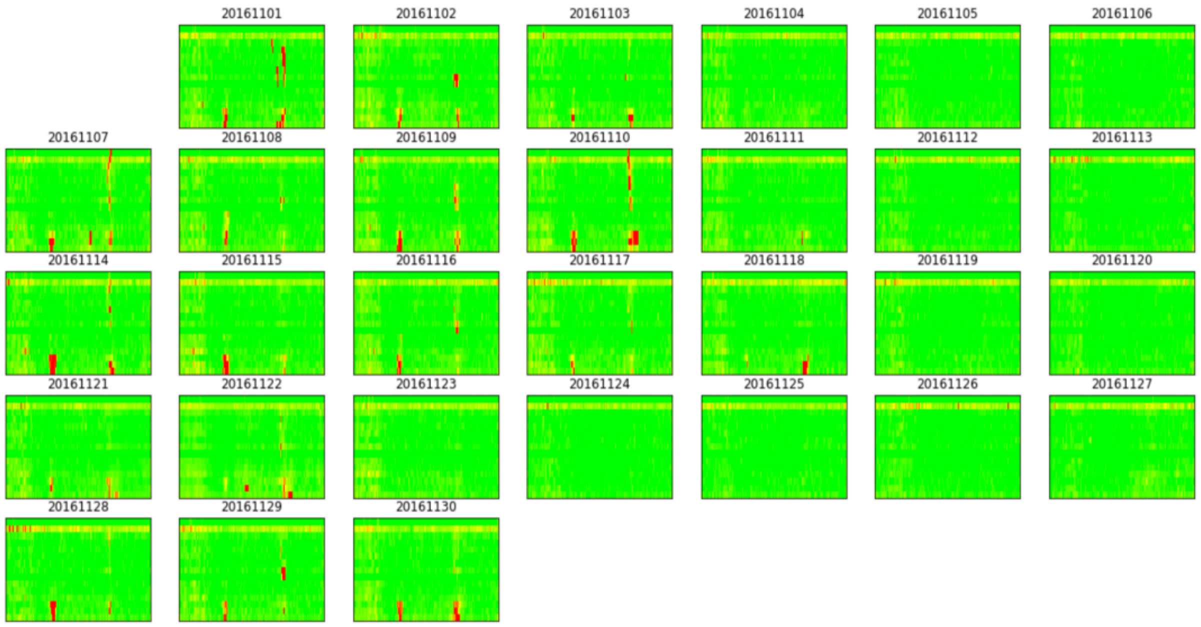

20160601
20160602
20160603
20160604
20160605
20160606
20160607
20160608
20160609
20160610
20160611
20160612
20160613
20160614
20160615
20160616
20160617
20160618
20160619
20160620
20160621
20160622
20160623
20160624
20160625
20160626
20160627
20160628
20160629
20160630

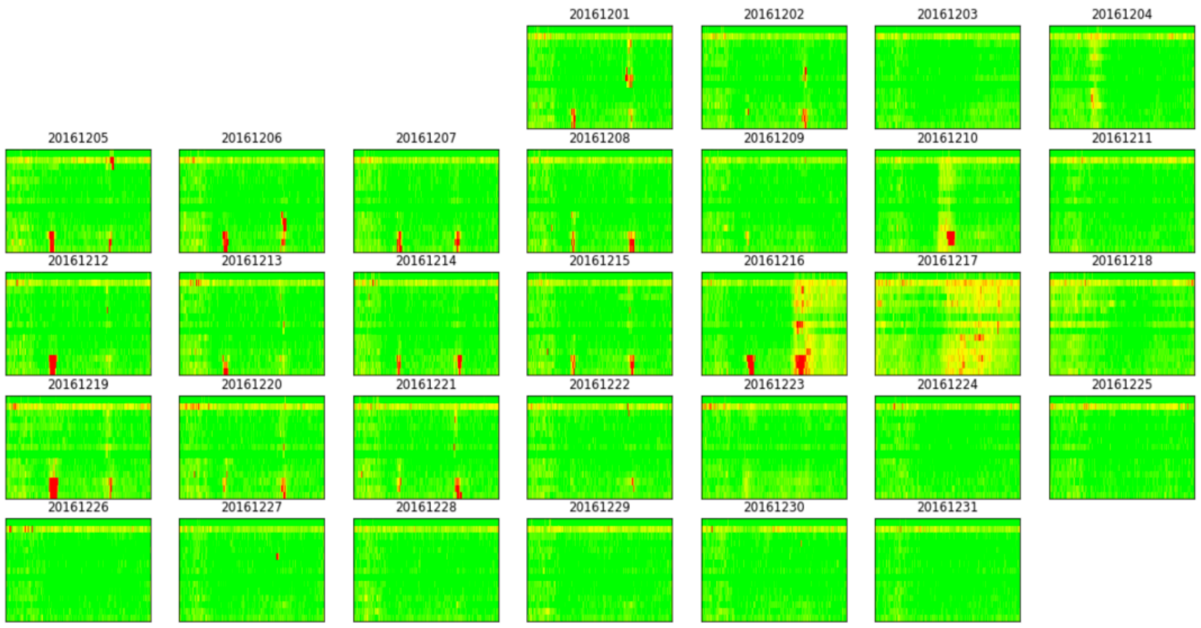

20160701
20160702
20160703
20160704
20160705
20160706
20160707
20160708
20160709
20160710
20160711
20160712
20160713
20160714
20160715
20160716
20160717
20160718
20160719
20160720
20160721
20160722
20160723
20160724
20160725
20160726
20160727
20160728
20160729
20160730
20160731


20160801
20160802
20160803
20160804
20160805
20160806
20160807
20160808
20160809
20160810
20160811
20160812
20160813
20160814
20160815
20160816
20160817
20160818
20160819
20160820
20160821
20160822
20160823
20160824
20160825
20160826
20160827
20160828
20160829
20160830
20160831


20160901
20160902
20160903
20160904
20160905
20160906
20160907
20160908
20160909
20160910
20160911
20160912
20160913
20160914
20160915
20160916
20160917
20160918
20160919
20160920
20160921
20160922
20160923
20160924
20160925
20160926
20160927
20160928
20160929
20160930


20161001
20161002
20161003
20161004
20161005
20161006
20161007
20161008
20161009
20161010
20161011
20161012
20161013
20161014
20161015
20161016
20161017
20161018
20161019
20161020
20161021
20161022
20161023
20161024
20161025
20161026
20161027
20161028
20161029
20161030
20161031


20161101
20161102
20161103
20161104
20161105
20161106
20161107
20161108
20161109
20161110
20161111
20161112
20161113
20161114
20161115
20161116
20161117
20161118
20161119
20161120
20161121
20161122
20161123
20161124
20161125
20161126
20161127
20161128
20161129
20161130


20161201
20161202
20161203
20161204
20161205
20161206
20161207
20161208
20161209
20161210
20161211
20161212
20161213
20161214
20161215
20161216
20161217
20161218
20161219
20161220
20161221
20161222
20161223
20161224
20161225
20161226
20161227
20161228
20161229
20161230
20161231