Short-Term Traffic Forecasting: Modeling and Learning Spatio-Temporal Relations in Transportation Networks Using Graph Neural Networks

Behrooz Shahsavari
Pieter Abbeel

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-243
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-243.html

December 17, 2015

Short–Term Traffic Forecasting: Modeling and Learning Spatio–Temporal Relations in Transportation Networks Using Graph Neural Networks

by Behrooz Shahsavari

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor P. Abbeel
Research Advisor

(Date)

* * * * * * *

Professor R. Horowitz
Second Reader

(Date)

Short–Term Traffic Forecasting: Modeling and Learning Spatio–Temporal Relations in Transportation Networks Using Graph Neural Networks

Abstract

Short–Term Traffic Forecasting: Modeling and Learning Spatio–Temporal Relations in Transportation Networks Using Graph Neural Networks

by

Behrooz Shahsavari

Master of Science in Computer Sciences

University of California, Berkeley

Professor Pieter Abbeel, Chair

This report addresses the problem of traffic conditions forecasting based on *big data* with a novel *artificial intelligence* approach. We propose an empirical data–driven graph–oriented model that can process spatio–temporal measurements. The proposed intelligent model not only incorporates observations from multiple locations, but also, explicitly takes into account the spatial interrelations (between the sensor locations) that are forced by the traffic network topology. We abstract the data collected in a transportation network by a graph that has a set of "nodes" corresponding to the sensor locations and a set of "edges" representing the spatial interrelations governed by the network topology. Both entities are associated with real valued feature vectors. For instance, a history of traffic flow, occupancy and speed measured by a sensor form the feature vector associated with the corresponding node. On the other hand, the road characteristics such as length, capacity and direction constitute the edge feature vectors. A *Graph Neural Network* is trained in a supervised fashion to predict future traffic conditions based on the stated graph–structured data.

This model combines the advantages of methods like *Cell Transmission Model* that benefits from knowing causalities enforced by traffic network topology, and advantages of neural network models that can extract very complex and nonlinear relations after being trained on *big data*. Moreover, the proposed model is "robust" to sample missing.

A comprehensive empirical study is conducted on traffic data obtained from *PeMS* database. A method is proposed to constitute the spatial interrelations (i.e. graph edges) between the sensor locations by deploying Google Maps (Directions) API. We evaluate the effectiveness of the proposed prediction method in locations with simple and complex dynamics (e.g. the intersection of several highways with multitude on– and off–ramps) individually.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Intelligent Transportation Systems (ITSs) aim to enhance the operational efficiency of current transportation infrastructures by benefiting from the well established and/or emerging advanced algorithms, computers with ever increasing computing power, and distributed sensors that collect an abundant amount of information from traffic networks. Among these three crucial components of an ITS, the last two have been growing rapidly in recent years. Many countries around the world have started embedding sensors, such as inductive–loop detectors, magnetometers, video image processors, microwave and laser radar sensors in their traffic networks Tewolde (2012). Beside the data from these sensory domains, web–based and mobile technologies provide accurate data about the road states and even details regarding individual cars. The overwhelming data collected by these emerging sensory technologies has introduced the notion of *Big Data* in the field of traffic management and control by making large and rich datasets available to the researchers and traffic engineers. It is expected that in the future not only the data volume will increase rapidly, but also data from variety of new domains will be available as the *Internet of Things (IoT)* manifests in transportation systems Vlahogianni (2015). In addition to aboundness of data, researchers now have access to more computing power than ever before thanks to hardware developments and emergence of *cloud–computing* that provides easy and cheap access to powerful computers.

This availability of data and computation power has created unique opportunities for researchers to develop *data–driven* models and advanced algorithms to transform raw data to some informative knowledge that can be utilized for management and control of a transportation system. The success of many ITSs strongly depends on a *traffic estimation and prediction system (TrEPS)* that uses advanced traffic models to obtain timely and accurate estimates of emerging traffic states such as traffic *flow, occupancy and speed.* These predictions can be utilized to select proactive strategies for meeting various traffic control, management, and operation objectives Lieu (2000). For instance, TrEPS plays a crucial role in *advanced traveler information systems* and *advanced traffic management systems* that attempt to optimize traffic signals timing and driving routing to relieve traffic congestion

and travel time problems facing commuters. Since the late 1970's (Ahmed & Cook (1979)) a great deal of research has been focused on this matter, especially on *Short–term Traffic Forecasting*, by incorporating an extensive variety of mathematical models and apparatuses from different fields such as statistics, signal processing, and control theory. However, in recent years, the inimitable data availability and rapid processing facilities has directed many researchers toward novel data driven empirical algorithms, that have been systematically growing in parallel to the well–founded mathematical models that are based on macroscopic and microscopic theories Yuan *et al.* (2012); Wang & Papageorgiou (2005). This migration from analytical to data driven modeling has been marked by a significant increase of *computational intelligence (CI)* and *data mining* approaches to analyzing the data. In this work, we revisit the problem of short–term traffic forecasting with a novel CI approach and propose an empirical data–driven graph–oriented model that can process measurements from multitude sources to accurately predict traffic states in "relatively" long horizons.

In a general framework, the problem of traffic forecasting can be formalized as follows. Let $X_i(t)$ be a real valued vector that contains traffic measurements at time step t obtained from a point of traffic network that is indexed by i. For instance, the state X can have components representing the traffic flow, occupancy and speed measured by a particular inductive–loop detector that is indexed by i. It may also include other types of information that are not necessarily measured by a traffic sensor. For instance, weather conditions or the existence of special events can be mapped to the set of real numbers and then be included in X. In the simplest case, *univariate* traffic forecasting attempts to infer temporal correlation between scalar measurements obtained at one location to predict the future values at the same location. That is, given n_t past values of the states $\{X_i(t), X_i(t-1), ..., X_i(t-n_t+1)\}$, where $X_{\cdot}(.) \in \Re$, one wants to find an accurate map $\mathscr{M}^*$ from these observations to the traffic state at δ steps ahead $X_i(t+\delta)$

$$\mathscr{M}^* := \arg\min_{\mathscr{M}} \|X_i(t+\delta) - \mathscr{M}\left(t, i, X_i(t), X_i(t-1), ..., X_i(t-n_t+1)\right)\|_d$$

where $\|.\|_d$ denotes a distance metric that is desired to be as small as possible. The prediction horizon δ, the number of past observed values n_t and the error metric $\|.\|$ are important components of this formalism and will be discussed in detail throughout this work. Another approach is *multivariate* prediction which is, in general, more accurate and more complex since it deploys multi–dimensional data that has information about the target location as well as a set of other spatially correlated locations. In this case, the problem is inferring an estimate of $X_{i_1}(t+\delta) \in \Re^m$ given the history of observations at location i_1 and other *correlated* locations, say $\{X_{i_2}(t), X_{i_2}(t-1), ...\}$,..., $\{X_{i_n}(t), X_{i_n}(t-1), ...\}$

$$\begin{aligned}\mathscr{M}^* : \quad = \arg\min_{\mathscr{M}} \|C^T [X_i\left(t+\delta\right) - \mathscr{M}\left(t, i_1, X_{i_1}\left(t\right), ..., X_{i_1}\left(t-n_t+1\right),\right. \\ \left. ..., X_{i_n}\left(t\right), ..., X_{i_n}\left(t-n_t+1\right)\right)] \|_d.\end{aligned} \tag{1.1}$$

The vector C represents the weights on each element of prediction error. For instance, when the states contain weather conditions information, the corresponding component(s) in C should be zero since we are not interested in predicting these quantities in the future.

As we will see in our literature review in the next two sections, it has been found that prediction accuracy can be increased, especially in longer horizons, when in addition to the temporal correlations between the observations, spatial interrelations between different sensors are taken into account. Statistical models that consider spatio–temporal correlations of data are limited and most of the work has been outlined in a CI framework, especially by deploying neural networks. These data–driven methods in the literature do not explicitly consider the transportation network dynamics; rather, they rely on a learning process in order to extract the underlying spatial relationships between the unstructured input data and the future traffic conditions. For instance, in related work that deploys neural network models the spatial relationships between observations are removed from the input data and it is left to the neural network to extract the causality and hierarchical relationships in the learning process. These methods have the drawback that the input vector dimension increases by taking into account the observations from more locations. Furthermore, these models operate as static pattern regressors and cannot manipulate dynamic information appropriately.

Our objective is to overcome these shortcomings by developing an empirical data–driven traffic model that can analyze graph structured data that not only presents temporal observations, but also preserves full information about the spatial interrelations between the observations collected from different locations. It has been shown that models with learning capabilities can extract general rules from the available examples more effectively when data representation faithfully preserves the environment properties Bianchini & Maggini (2013). For a traffic network, a graph can describe precisely the network dynamics by modeling it through a set of nodes that correspond to road segments equipped with sensors (e.g. inductive–loop detectors) and edges that correspond to the roads between the nodes. Both entities are associated with observation vectors which in our application correspond to the measurements obtained by the sensors (e.g. flow, occupancy and speed) and road characteristics (e.g. direction and length). An example of these graph structures is shown in Fig. 1.1. Each node in this graph corresponds to a vehicle detector station (VDS) (in California, U.S.A.), and the edges are created based on the road network topology. âŹł Our work will be based on the Graph Neural Network (GNN) model Scarselli *et al.* (2009b) which is a multi–layer perceptron (MLP) that can process input data encoded as general undirected/directed labeled graphs. GNN exploits an information diffusion mechanism in which a set of entities representing the graph nodes are linked to each other based on the graph connectivity. The notion of information associated with nodes is represented by adaptable "state" vectors. In this framework, the prediction obtained for each node is determined based on the observations associated with that particular node in conjunction with the states that encode information diffused to the target node from all other nodes via network topological dependencies. For instance, a GNN predicts the traffic condition at VDS 715944 in Fig. 1.1, by processing the entire graph. Note that for this example, a univariate model only looks at VDS 715944 observations and ignores the other nodes. On the other hand, existing multivariate approaches can consider all nodes observations, but they ignore the edges. Moreover, these models cannot handle the case when the underlying topology varies due to missing

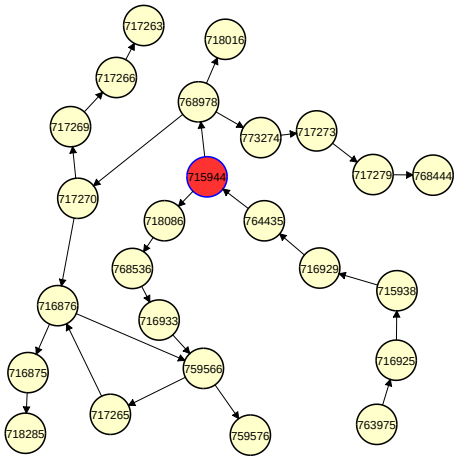


Figure 1.1: A graph representation of spatially interconnected vehicle detector stations in a transport network.

measurements.

The remainder of this report is organized as follows. In sections 1.2 and 1.3 in the current chapter, we will briefly review the literature on short–term traffic prediction and especially the methodologies that deploy spatio–temporal correlations systematically for this purpose. In chapter 2 we will present fundamentals of the graph neural network including the model architecture and a learning paradigm. The process of modeling a traffic network as a GNN is also presented in that chapter. Chapter 3 is devoted to a comprehensive empirical study on traffic data obtained from Caltrans Performance Measurement System (PeMS) database. We have considered data of "district 7" that includes Los Angeles and Ventura counties in the U.S. and proposed a systematic way of sampling VDSs such that the results can be generalized to the whole network. Most of the traffic databases, including PeMS, only provide sensor measurements and do not furnish spatial relations between the sensors. For instance, we can only obtain node observations in Fig. 1.1 via PeMS and constructing the edges is not a trivial task. We will propose a mechanism that can retrieve these relations by deploying Google Maps (Directions) APIs efficiently. In section 3.3 different GNN models are trained for various case studies and the results are compared to other statistical and CI methods that are known as the baselines and state of the art for the problem of short–term traffic forecasting. Lastly, conclusions are drawn and possible research directions for future work are outlined in chapter 4.

1.2 Literature on Short–term Traffic Forecasting

Short–term traffic forecasting is most commonly analyzed by two approaches, namely classical statistical perspectives and computational intelligence. In general, the prediction

models considered by these two approaches can be categorized to *parametric* and *non–parametric* models. However, as we will see throughout this section most of the parametric models deploy statistical methodologies, whereas the non–parametric models are mostly treated in a CI framework. Comprehensive studies of the existing literature on these approaches from 1980's to 2014 can be found in four survey articles. Adeli Adeli (2001) reviewed neural network articles related to civil engineering, including traffic forecasting, that were published in archival research journals from 1989 to 2000. Van Lint and Van Hinsbergen Van Lint & Van Hinsbergen (2012) reviewed *Artificial Intelligence (AI)* techniques for short–term traffic forecasting and provided a taxonomy of the many different approaches reported in the literature. Vlahogianni provided a comprehensive critical review of the literature up to 2003 Vlahogianni *et al.* (2004), and recently published another survey paper that focuses on emerging challenges and opportunities arose by recent developments in technological and analytical aspects of intelligent transpiration systems Vlahogianni *et al.* (2014). We briefly review the literature on both approaches in this section. The next section is devoted to discussing, in more detail, the existing multivariate methods that consider spatio–temporal dependencies rather than relying only on temporal relations.

Most, if not all, of the early work on traffic forecasting employed parametric models and statistical analysis on time–series to infer the traffic characteristics, such as flow, in the future based on statistics extracted from historical data. Ahmed's work in 1979 (Ahmed & Cook (1979)) can be noted as one of the earliest literature that introduces this approach. He investigated the application of Box–Jenkins method Box & Jenkins (1976) to predict freeway traffic volume and occupancy time series. Ahmed reported that the *Auto–Regressive Integrated Moving–Average (ARIMA)* methods were found to be more accurate in representing freeway time–series data compared to *moving–average, double–exponential smoothing, and Trigg and Leach adaptive models* Trigg & Leach (1967). ARIMA models are one of the most general class of models for forecasting a time series which can be made to be "stationary". An ARIMA model is defined by three non–negative integer numbers, commonly specified by the notation ARIMA(p, d, n) where

- p is the number of autoregressive terms,
- d is the number of nonseasonal differences needed for stationarity,
- n is the number of lagged forecast errors in the prediction equation.

Let q^{-1} be the *one step lag operator*. The difference equation for an AIRMA model is

$$\left(1 - \sum_{i=1}^{p} b_i q^{-i}\right) \left(1 - q^{-1}\right)^d Y(t) = \mu + \left(1 - \sum_{i=1}^{n} a_i q^{-i}\right) X(t), \tag{1.2}$$

where $Y(t)$ is the output of system. The coefficients a_i, b_i and μ should be determined based on historical data or in an adaptive and online fashion. Many well known time–series models are special cases of ARIMA model. For instance, *Random Walk* model that employs

$Y(t) = Y(t-1)$ is ARIMA$(0,1,0)$ with $\mu = 0$. Since Ahmed's work, the time–series models, especially ARIMA, have been the basis of numerous other methods for traffic prediction Moorthy & Ratcliffe (1988). The *KARIMA* method Van Der Voort *et al.* (1996) uses a hybrid architecture that deploys a Kohonen self–organizing map Kohonen (1998) as an initial classifier and an individually tuned ARIMA model associated with each determined class. It was illustrated that the explicit separation of the tasks of classification and functional approximation could improve the forecasting performance compared to a single ARIMA model.

The *Seasonal ARIMA (SARIMA)* model is another variant that explicitly considers the *seasonality* of traffic time–series in the analysis Williams *et al.* (1998); Williams & Hoel (1999). The SARIMA method is motivated by the fact that a general traffic time–series does not meet stationarity conditions due to the existence of periodic patterns in the series. For instance, a weekly pattern tied to working days, and a daily pattern tied to working hours can be observed in traffic flow. Given these patterns, it follows that creating a series composed of the differences between traffic observations and the observations one "season" prior should remove the predominant time dependencies in the statistics and provide a transformed process that is more "stationary" Williams & Hoel (2003). More formally, a SARIMA model denoted by SARIMA$(p,d,n)(P,D,N)(s)$ has a difference equation in form of

$$\left(1-\sum_{i=1}^{p} b_i q^{-i}\right)\left(1-\sum_{i=1}^{P} B_i q^{-i\times s}\right)\left(1-q^{-1}\right)^d\left(1-q^{-s}\right)^D Y(t) = \mu + \left(1-\sum_{i=1}^{n} a_i q^{-i}\right)\left(1-\sum_{i=1}^{N} A_i q^{-i\times s}\right) X(t), \tag{1.3}$$

where s is denotes the "season" interval. For instance, when the notion of season refers to weekly patterns, the value of s is equal to $7 \times M$ where M is the number of samples per day. The parameters P, N and D, respectively, represent the seasonal autoregressive polynomial order, the seasonal moving average polynomial order, and the order of seasonal differences. More recently, an *Auto–Regressive Moving Average with an eXogenous input (ARMAX)* model with an optimal multiple–step–ahead predictor of traffic demand was proposed by Wu et al. Wu *et al.* (2014b,a). The technique is based on combining the most recent measured flow data with a nominal historical flow profile where the predictor parameters are estimated with an ARMAX model.

The vast majority of time–series techniques are univariate, meaning that the observations are scalar–valued and each model can only estimate the future of one quantity at one target point in the network. On the opposite side, there are multivariate models that attempt to fit a single model from vector-valued observations to the underlying traffic dynamics at multiple locations (e.g. at multiple loop detectors). These methods are mostly inherited from univariate models and have been generalized to cope with multiple time-series. The *Vector Auto–Regressive Moving Average (VARMA)* model and *Space–Time AIRMA (STARIMA)* model are such generalizations Kamarianakis & Prastacos (2003). The VARMA model

considers N time–series and describes a set of $N \times N$ autoregressive and moving average parameter matrices to represent all autocorrelations and cross–correlations among them. The STARIMA model can be viewed as a special case of VARMA model where a set of constraints are posed to mimic the topology of a spatial network and result in a drastic reduction in the number of estimated parameters. This model was first introduced by Pfeifer and Deutsch in a series of papers in 1980's Pfeifer & Deutrch (1980); Pfeifer & Deutsch (1980), and later was applied to traffic flow time–series by Kamarianakis Kamarianakis & Prastacos (2005). The *Structural Time–series Model (STM)* is a parsimonious and computationally simple multivariate algorithm that is not based on a generalization of a univariate model Ghosh *et al.* (2009). Non–parametric models that have been applied to the problem of short-term traffic forecasting deploy variety of statistical methods such as Gaussian maximum likelihood Lin (2001), Markov chain Yu *et al.* (2003); Sun *et al.* (2004), Bayesian networks Sun *et al.* (2006), and Kalman Filtering Xie *et al.* (2007); Wang & Papageorgiou (2005).

Statistical methods are based on solid mathematical foundations. As Karlaftis and Vlahogianni suggest Karlaftis & Vlahogianni (2011), these methods are most effective when a priori knowledge about the functional relationship of traffic variables is available. Furthermore, these methodologies provide interpretable results and causalities, and verify the statistical properties of the underlying mechanism. However, the statistical methods rely on a set of constraining assumptions that may fail when dealing with complex and highly nonlinear data. For instance, it was realized in 1990's that traffic flow exhibits discontinuities in its temporal evolution Newell (1993); Hall *et al.* (1992) which makes formalizing the prediction as a simple time series problem questionable. Moreover, varying statistical behavior, strongly nonlinear features near boundary conditions, and highly transitional behavior in traffic flow that was indicated by Kerner (Kerner (2004)) and Vlahogianni (Vlahogianni *et al.* (2006)) are difficult to be modeled by a single statistical prediction approach.

These difficulties with the statistical (mostly parametric) models, and on the other side, the recent technological advances in rapid processing and overwhelming traffic data gathered from different domains have brought unprecedented opportunities for data–driven non–parametric methods. A great deal of research has been focused on deploying computational intelligence and non–parametric models to predict traffic conditions Karlaftis & Vlahogianni (2011). From an analytical point of view, the *k–Nearest Neighbors (kNN)* algorithms is one of the simplest non–parametric models. The exact kNN algorithm and an approximation to that, which is computationally more efficient, have been applied to traffic conditions prediction Kim *et al.* (2005); Oswald *et al.* (2000). However, it has been shown that kNN–like methods can outperform only naïve algorithms such as random walk Smith *et al.* (2002). Chang et al. proposed a multi–interval traffic volume prediction based on the kNN algorithm and illustrated a higher performance than ARIMA and SARIMA methods by empirical study Chang *et al.* (2010). Nevertheless, the algorithm suffers from *the curse of dimensionality* and has high time complexity of query when the number of locations (sensors) and samples are large. Non parametric regression Davis & Nihan (1991) and local linear regression models Sun *et al.* (2003) are other machine learning methods that have been applied to traffic time–series prediction.

The *Support Vector Regression (SVR)* algorithm is a non–parametric method that is based on statistical learning theory Vapnik & Vapnik (1998); Smola & Schölkopf (2004). The original form of SVR and numerous variants of it have been deployed for the purpose of traffic prediction by the researchers. Ding et al. used the SVR for traffic flow prediction Ding *et al.* (2002) and Wu et al. applied it for travel time estimation and compared the results to naïve algorithms Wu *et al.* (2004). Attempts on developing an online version of SVR can also be found in the literature Su *et al.* (2007); Castro-Neto *et al.* (2009); Jeong *et al.* (2013). Lippi et al. proposed two SVR models based on an idea to benefit from prediction accuracy of SARIMA model and computational efficiency of SVR algorithm Lippi *et al.* (2013). The models used RBF and linear seasonal kernels and were compared to the original form of SVR (with RBF kernel) and many other forecasting methods. The comparison illustrated that these new methods outperform the simple SVR model thanks to explicitly incorporating the seasonality of data into the kernels. The successfulness of these methods raises the question if there are other useful features, especially in a multivariate framework, that can be considered in the kernel. Even for considering the seasonal effects, it is still not clear that what seasonal period, e.g. one week or one day, should be considered; or, whether it is better to rely on most recent seasons or it is more beneficial to use all previous seasons. These types of ambiguities in creating hand–engineered features motivates the use of *Neural Networks (NN)* as the algorithms that can extract informative features from raw data. NNs do not require determining specific structures in the data and, indeed, are "intelligent" in the sense that during the learning process the structure emerges from an unstructured beginning.

A great deal of successful work on traffic modeling and conditions prediction, especially in multivariate frameworks, is based on neural networks. The multivariate models are of high importance since they are not subjected to two shortcomings that are common among many univariate models: (1) the traffic evolution is inherently a temporal and spatial phenomenon, and accordingly, the univariate methods based solely on previous observations at the location of interest miss potential information embedded in observations attained at other locations Williams (2001). (2) in a univariate paradigm, a model should be incorporated for each individual location which requires significant memory, computation and time to develop. In other words, the first matter is in regard to the dimension of input domain of the prediction model, whereas the latter concerns the output domain dimensionality. As was mentioned earlier, there are article on statistical based methods (e.g. Stathopoulos & Karlaftis (2003); Kamarianakis & Prastacos (2003, 2005); Williams (2001)) for traffic conditions prediction in a multivariate framework. However, this topic of research is mostly dominated by NN based methods due to: (1) the underlying dynamics governed by both temporal and spatial dependencies between observations in time and space is complex and can be strongly non-linear. Neural networks are powerful models that can capture very complex relations. (2) crafting hand–engineered features that extract all information from data spread in time and space is laborious. Neural networks are data–driven and adaptive models that can extract the features without the need of a priori assumptions. (3) from a methodological point of view, these models are very flexible in that generalization from univariate to multivariate, or more generally, changing the number of locations incorporated in the input or output of

the model, requires insignificant effort.

In early 1990's (Clark *et al.* (1993)) these advantages motivated researchers to deploy neural networks for traffic condition prediction. NN models can be categorized into two groups, namely the static and dynamic models, based on their architectures. The majority of early work including Clark *et al.* (1993); Dougherty & Cobbett (1997); Dougherty *et al.* (1994) deployed static networks such as simple feedforward Multi–Layer Perceptrons (MLP). The temporal and spatial relationships between data samples in this approach should be augmented in the input data during a preprocessing phase. For instance, the input vector when the current sample $X_{i_1}(t)$ and the previous one $X_{i_1}(t-1)$ at location i_1 are considered in accordance with the corresponding values at location i_2, the input vector can be in the form of $\begin{bmatrix} X_{i_1}^T(t) & X_{i_1}^T(t-1) & X_{i_2}^T(t) & X_{i_2}^T(t-1) \end{bmatrix}^T$ or any other permutation of these components. In addition to the aforementioned papers, static models are deployed for exit demand prediction by feedforward neural networks (FNN) Kwon & Stephanedes (1994), forecasting of traffic conditions in urban road networks Vythoulkas (1993), modeling complex nature of intersection queue dynamics Chang & Su (1995), and urban traffic congestion prediction Gilmore & Abe (1995). A comprehensive study on the effect of various factors on the results of the short–time traffic flow and speed prediction on motorways can be found in Innamaa (2000). Besides this classical approach to determining the proper structure and parameters of a NN based on a sensitivity analysis, more involved methodologies based on optimization techniques have been deployed by the researchers for this purpose Vlahogianni *et al.* (2005).

The second approach uses dynamic neural network models to capture temporal patterns in time–series of traffic data. Traffic conditions predictors have been developed based on the Time Delay NNs (TDNN) and in some work they were optimized by genetic algorithms Lingras & Mountford (2001); Abdulhai *et al.* (2002). Other variants of TDNN such as a recursive model Zhang (2000) and a comparison among them can be found in Yun *et al.* (1998). The use of Recurrent NNs (RNN) have also been investigated for traffic prediction. Dia and his collaborators used a Time–Lag RNN for speed prediction Dia (2001) and Van Lint attempted to model state space dynamics for travel time prediction by RNNs Van Lint *et al.* (2002). Numerous other neural network models applied to similar problems in the domain of traffic prediction can be traced in the literature. The radial basis MLPs Park *et al.* (1998), Fuzzy NNs Yin *et al.* (2002); Ishak & Alecsandru (2004), Bayesian combined NN Zheng *et al.* (2006), echo state NNs Yisheng *et al.* (2011), wavelet–based NN Jiang & Adeli (2005); Xie & Zhang (2006) are such models. Researchers have also considered hybrid models that combine static and dynamic neural networks. For instance, Alecsandru et al. deployed a hybrid model that integrates a model–based approach (to improve prediction performance under non–recurrent conditions) with a memory–based approach (to cope with recurrent phenomena) Alecsandru & Ishak (2004).

The most, if not all, of the literature prior to 2013 on the application of neural networks in traffic forecasting deploys "shallow" structures, meaning that the network has at most one hidden layer. This is probably due to the prevailing interpretation of universal approximation theorem which claims that, "in principle", one large enough hidden layer is adequate for approximating any function. However, in recent years, the researchers have proved that in

many practical applications, coping with real–world data, a "deep architecture" is much more effective than a shallow one when an appropriate training algorithm is utilized Bengio *et al.* (2015). This has emerged a new area of machine learning research known as *Deep Learning*. To the best of our knowledge, deep learning was first applied to traffic (flow) forecasting by Huang and collaborators in 2013 Huang *et al.* (2013). They used a hierarchical structure that had a Deep Belief Network (DBN) in the bottom and a (multi–task) regression layer on the top. The DBN was for unsupervised feature extraction and the regression layer was trained to predict the traffic flow based on these extracted features. More recently, Lv et al. used a deep stacked autoencoder model for traffic flow prediction Lv *et al.* (2015). Both papers used California Performance Measurement System (PeMS) (PeMS (n.d.)) database for a comparison study.

As mentioned earlier, our proposed method is multivariate and based on a special dynamic neural network model. Hence, this method can potentially benefit from advantages that were mentioned for *data–driven non–parametric multivariate* models. Moreover, our method considers spatio–temporal relations presented by graph structured data since this type of data structure can preserve topological dependencies. This is the main distinction with other NN models in the literature since they all use unstructured data and rely on the learning process to extract topological and hierarchical relationships. We will review literature that systematically takes into account the spatial dependencies and compare them to our proposed model in the following section.

1.3 Traffic Prediction Methods that Consider Spatial Dependencies

Traffic conditions at any time and location are correlated with the past observations at the same location as well as neighboring places. This spatio–temporal interrelation is the basis of some methods such as the well known Cell Transmission Model (CTM) Daganzo (1994) that explicitly takes spatial and temporal relationships into account to mimic the network dynamics. Chandra and Al–Deek studied the effect of upstream as well as downstream locations on the traffic at a specific location Chandra & Al-Deek (2008). The speeds from a station at the center of this location were checked for cross–correlations with stations at upstream and downstream. It was formalized in this study that the past values of upstream as well as downstream stations influence the future values at a station and therefore can be used for prediction. Head (1995) indicated that incorporating far away observations from the location of interest can increase the accuracy in longer prediction horizons. Cheng at al. examined the spatio–temporal autocorrelation structure of road networks and assessed that the autocorrelation is heterogeneous in space and dynamic in time Cheng *et al.* (2012). Kamarianakis compared ARMA–based multivariate approaches, such as STARIMA and VARMA, with univariate ARIMA model and concluded that the multivariate time–series methods can outperform the classical univariate models in certain cases Kamarianakis & Prastacos (2003). Tebaldi et al. used downstream flows in a hierarchical statistical modeling frame-

work and reported that the improvements in very short time (1 minute) prediction can be radically high in cases of very marked breakdown of freeway flows on some links Tebaldi *et al.* (2002). Stathopoulos and Karlaftis developed time–series state space models that were fed with upstream detectors data to improve on the predictions of downstream locations Stathopoulos & Karlaftis (2003). Kamarianakis et al. in another work split traffic regimes based on the data nonlinearity and for each regime and measurement location, implemented a LASSO estimator to perform model selection and coefficient estimation simultaneously Kamarianakis *et al.* (2012). Du and collaborators proposed an adaptive information fusion model to predict the link travel time distribution by iteratively combining past information with the real–time information available at discrete time points Du *et al.* (2012). Bayen and collaborators in a series of articles studied arterial traffic estimation from GPS probe vehicle data by modeling the evolution of traffic states by probabilistic graphical models Herring *et al.* (2010a,b); Hunter *et al.* (2009); Hofleitner *et al.* (2012). However, the majority of voluminous literature that uses statistical approaches, especially the time–series–based methods, ignores the spatial relationships, probably because it requires multivariate analysis which can make inference very complex in statistical based approaches.

Non–parametric models have been used for this purpose too. Haworth and Cheng developed a non–parametric spatio–temporal kernel regression model that used the current traffic patterns of the upstream and downstream neighboring links to forecast travel time values of road links under the assumption of sensor malfunction Haworth & Cheng (2012). Sun et al. investigated the combination of multi–link models with Multi Task Learning (MTL) Sun *et al.* (2012). They also used graphical LASSO to extract the most informative historical flows from all of the links in the whole transportation system, and then constructed a neural network with the extracted data to predict traffic flows.

All the aforementioned literature emphasizes that spatial characteristics of traffic network play a crucial role in accurate prediction of traffic conditions, especially in longer horizons. The existing literature can be divided into two categories: (1) methods like CTM that uses an exact spatial representation in modeling the traffic dynamics. The advantage of these methods is that the system evolution obeys the exact network dynamics governed by the network topology. However, these models have two deficiencies. First, they are not robust to sample missing and unmodeled dynamics (e.g. on– and off–ramps with no sensors). Second, they are constrained by a set of assumptions such as piecewise affine relation of flow and density as in CTM. (2) data–driven methods that do not explicitly consider the network dynamics; rather, they rely on the learning process to extract the underlying relationships between the unstructured input data and the future traffic conditions. For instance, in neural network models for multivariate prediction, the input data is constructed by concatenating the observations from different sensors into one single large vector Clark *et al.* (1993); Huang *et al.* (2013, 2014); Lv *et al.* (2015). In this case, the spatial relationships between observations is removed from the input data and the data is fed in an unstructured form. Note that this is analogous to the static nerual networks for time–series where the causality sequence of measurements is ignored. In both cases, it is left to the neural network to learn the causality and hierarchical relationships in the learning process. These have the

drawback that the input vector dimension increases by taking into account observations from more locations. Furthermore, these models operate as static pattern regressors and cannot manipulate dynamic information appropriately. As Vlahogianni (2015) remarked in a recent paper, research on incorporating network dynamics in traffic forecasting is not mature yet. Methodologies aimed to capture spatial traffic features accurately have not received enough attention, and especially, at a network level, limited research has been conducted Cheng *et al.* (2012).

One of our objectives is to overcome these shortcomings by developing a traffic model that can analyze a graph structured data that not only presents temporal observations, but also preserves full information about the spatial interrelations between the observations collected from different locations. We believe that the ability of graph structured data to represent traffic network topology is of high importance since input data encoding is a crucial step in the design of a data mining model. It has been shown that models with learning capabilities can extract general rules from the available examples more effectively when data representation faithfully preserves the environment properties Bianchini & Maggini (2013). Determining the structure for input data to gain this characteristic strongly depends on the problem domain. In the simplest form, the decision taken by the agent relies only on a single event that is described by a certain number of measurements that can be encoded as a vector of real numbers. Indeed, this is the approach taken by all work on traffic prediction that uses CI. However, for more complex tasks or a better decision making, a successful approach may require to exploit structured data, that provide a more balanced split of the intrinsic complexity between the task of data representation and that of data processing Bianchini & Maggini (2013).

For a traffic network, a graph can describe the network dynamics by modeling it through a set of nodes that correspond to the sensors (e.g. loop detectors) and edges that correspond to the roads between the nodes. Both entities are associated with observation vectors which in our application correspond to the measurements obtained by the sensors and road characteristics. Even if it is possible to obtain simpler representations from complex data structures, this process always implies the loss of information, an inaccurate modeling of the problem at hand or a higher complexity in the input data distribution. For instance, when the input data to a multivariate traffic model is a real valued vector, the size of input vector increases by taking more locations into account. Moreover, this representation is not capable of dealing with missing data since the vector size will be changing, unless a maximum number of sensors is assumed and a padding technique is used to encode the missing sensor measurements. It is worth noting that in such a data representation, the relationships among the sensors in the network would be encoded by their position in the vector, making it more difficult to exploit the structural regularities by an algorithm designed to process generic vectors.

In the next chapter we will focus on graph–structured data and present the graph neural network model which is able to process this type of data structure and cope with time–varying topologies. The fundamental of GNN model including its architecture and a learning paradigm for optimizing weights will be discussed and the process of modeling a traffic

network as a GNN will be presented.

Chapter 2

Proposed Methodology

2.1 Processing Structured Data

It is crucial to utilize an appropriate method of encoding when the input data to a machine learning technique is obtained from a structured domain. An encoding method should ideally preserve data and any information embedded in its structure properties to facilitate the extraction of useful rules (e.g. features) from training dataset. For instance, when data is attained from scattered sensors, the topological dependency of information on each sensor may be lost if an appropriate encoding stage is not exploited.

Different encoding methods can be adopted based on the input domain structure. In general, for simple tasks and data structures, the input data can be represented by a vector and be effectively used as long as the resulting data is not subjected to *the curse of dimensionality* Bellman *et al.* (1961). For instance, time series methods commonly rely on a finite horizon of past values of measurements that can be represented by linked lists – in which the nodes and links abstract the measurement values and timing sequences. Whether the time series algorithm is univariate or multivariate, the input data can be naturally represented by a vector containing the measurement values in a fixed (but arbitrary) order. However, a natural mapping from structured data to Euclidean space (vector form) is not necessarily possible for all domains/tasks and encoding phase can cause loss of information (e.g. when structure is ignored), modeling inaccuracy or high complexity in the input data distribution.

For complex tasks, the intensive data processing required for digesting data in vector form can be reduced by exploiting more complex data structures. In this research, we do not discuss a general framework for addressing this issue. Rather, we focus on graph structured data since a road network can be modeled by a graph precisely. Traffic state measurements, such as flow and occupancy, are obtained from a set of distributed sensors. This data can be represented by a graph structure in which the nodes and links (edges) represent the "sensors" and "relationships" between them, respectively. The measurements at each sensor form the observation vector associated with the corresponding node. Different notions can be adopted to define the "relationships" between the nodes. The spatial relationships that delineate the network topology are such concepts. We will return to this topic and provide a detailed

discussion later.

A traditional approach to apply machine learning techniques to graph structured data is to map the input data to Euclidean space in a preprocessing phase Haykin & Network (2004). For instance, the mapped data can be represented by a long vector with two sets of elements. The first set should contain the node observations and the second set should encode the graph connectivity. As an example, for a directed graph with m nodes, a natural way of encoding the connectivity matrix is to stack all elements in a vector with m^2 binary values. However, such an encoding method would not be appropriate when m is large, or when it is time–varying. Moreover, an algorithm designed to process generic vectors might not be successful in extracting structural regularities that are now encoded in the vector by exploiting the vector element indices in order to model the positional relationships among the graph nodes. Similar phenomenon might arise when the aforementioned binary string of length m^2 is replaced by a shorter string that encodes the graph connectivity in a numeral system with a larger base.

2.1.1 Machine Learning Techniques for Graph Structured Input Data

A great deal of research effort has been focused on developing machine learning techniques that can preserve topological dependencies among graph nodes and incorporate full information in the data processing phase. Early work include Elman's recurrent neural networks Elman (1990), Pollack's recurrent auto–associative memory Pollack (1990) and Sperduti's neural networks Sperduti & Starita (1997) for classification. Recursive neural networks Frasconi *et al.* (1998); Sperduti & Starita (1997); Hagenbuchner *et al.* (2003); Bianchini *et al.* (2005), and Markov chains Brin & Page (2012); Kleinberg (1999) are the main methodologies applied to this data domain.

Recursive neural networks can cope with directed acyclic graphs in graph–focused tasks by estimating the parameters of a mapping entity that transforms graph structured input data to Euclidean space. This type of neural networks has attracted researchers in the field of drug design Schmitt & Goller (1998), inductive learning in symbolic domains Küchler & Goller (1996), face spotting Bianchini *et al.* (2003) and logo recognition Francesconi *et al.* (1998). On the other hand, Markov chain models can emulate processes where the causal connections among events are represented by graphs.

Recently, a more general model denoted as "the Graph Neural Network" was proposed Scarselli *et al.* (2009b) which can be considered as an extension of both recursive neural networks and Markov chain models. The GNN extends recursive neural networks since it can process a more general class of graphs including cyclic, directed, and undirected graphs, and it can deal with node–focused applications without any preprocessing steps. The approach extends Markov models by the introduction of a learning algorithm and by enlarging the class of processes that can be modeled. GNNs are also related to support vector machines that exploit special kernels – e.g. Kondor & Lafferty (2002) – to process graphs Hammer & Jain (2004); Gärtner *et al.* (2004) since they both encode the input graph into an inter-

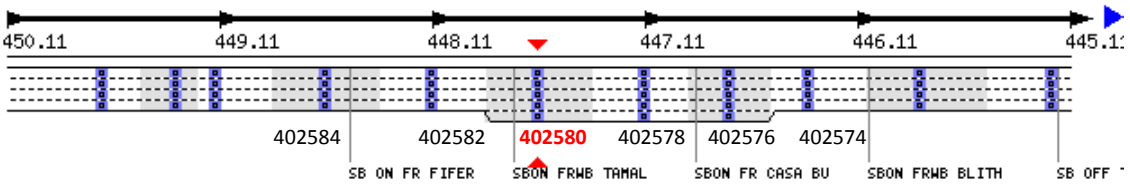


Figure 2.1: A section of `U.S. Highway 101 (South)` that contains eleven VDSs. The inductive–loops are shown by the blue squares. The numbers on the bottom and top of the road show the station IDs and the *absolute postmiles*[1]respectively. *(Reprinted from PeMS website.)*

nal representation. However, the GNN is a more general model since the internal encoding is learned, while in support vector machine, it is designed by the user Kondor & Lafferty (2002); Kashima *et al.* (2003); Collins & Duffy (2001). The computational capability of GNNs is theoretically proved in Scarselli *et al.* (2009a) by showing a type of universal approximation property, and it is successfully applied to different fields including large–scale recommender systems Pucci *et al.* (2006), web spam detection Di Noi *et al.* (2010); Scarselli *et al.* (2013); Belahcen *et al.* (2015) relational learning Uwents *et al.* (2011), text structure mining Chau *et al.* (2009); Muratore *et al.* (2010), image classification Quek *et al.* (2011); Jothi & Rani (2013), long term dependencies learning Bandinelli *et al.* (2010), sub–graph matching Baskararaja & Manickavasagam (2012) and chemical compound classification Barcz & Jankowski (2014).

GNNs exploit an information diffusion mechanism in which a set of entities representing the graph nodes are linked to each other based on the graph connectivity. The notion of information associated with nodes is represented by adaptable "state" vectors. The output of each node is then determined based on the observations of that particular node in conjunction with the stationary states that encode information gathered from all other nodes and topological dependencies. This scheme is similar to *cellular neural networks* Chua & Yang (1988) and *Hopfield's neural networks* Hopfield (1982). However, it differs from both models in that it can cope with more general classes of graphs, and it adopts a more general diffusion mechanism. The technical aspects of GNN models are discussed after the following section that provides required notations.

2.2 The GNN Model

A graph G is defined by a pair (N, E), where N and E denote the set of nodes and the edges among them respectively. We model a traffic network by a graph in which the nodes correspond to the locations that are equipped with vehicle detection stations (VDSs)

[1]The postmile is the means by which California tracks highway mileage. The postmile starts at zero at the western or southern end of the route or at the western or southern boundary of the county through which the route is traveling.

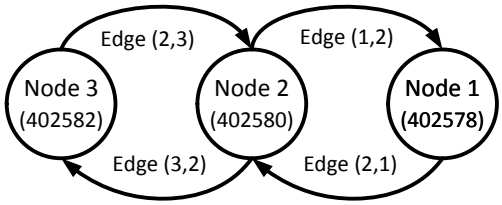


Figure 2.2: A graph that represents three VDSs in `U.S. Highway 101 (South)` and their spatial dependencies.

and the edges abstract the roads between these locations. For instance, consider a section of `U.S. Highway 101 (South)` that is shown in Fig. 2.1, and assume that the traffic flow at VDS 402580 is supposed to be predicted based on the past measurements obtained from the same station and VDS 402582 (upstream), as well as VDS 402578 (downstream). These three VDSs in accordance with the spatial dependencies between them can be represented by the 3–node graph shown in Fig. 2.2. The edges show the dependencies between the traffic conditions at node locations. For instance, the edge from Node 1 to Node 2 implies that the traffic condition at Node 2 is affected by the traffic condition at Node 1 (which is associated with its downstream VDS). On the other hand, the edge $(1, 2)$ implies that traffic conditions at Node 1 depend on the upstream traffic. We use this graph as a running example throughout this chapter to illustrate the GNN model.

Observations: Each node and edge in the graph is associated with a real vector–valued *observation*. We denote the observation vectors assigned to node n and edge (n, m) by $l_n \in \Re^{d_N}$ and $l_{(n,m)} \in \Re^{d_E}$ respectively, where d_N and d_E refer to the dimension of these vectors. The observations are **known** and form a part of the input data to the predictor which will be discussed later. We refer to the observation instances at (discrete) time step t by $l_n(t)$ and $l_{(n,m)}(t)$. In the traffic prediction application, $l_n(t)$ can include any traffic related quantity – such as flow, occupancy, speed, etc. – that is associated with VDS n at time step t and a finite horizon of past measurements. Throughout this work we denote the number of past measurements included in such a horizon as n_t. Let $flw_n(t)$, $occ_n(t)$ and $spd_n(t)$ be the flow, occupancy and speed measured by VDS n at time step t respectively. In our running example, we assume that these measurements at time t and the last 3 steps form each node's observation vector

$$l_1(t) := \begin{bmatrix} flw_1(t) \\ occ_1(t) \\ spd_1(t) \\ \vdots \\ flw_1(t-3) \\ occ_1(t-3) \\ spd_1(t-3) \end{bmatrix}, \qquad l_2(t) := \begin{bmatrix} flw_2(t) \\ occ_2(t) \\ spd_2(t) \\ \vdots \\ flw_2(t-3) \\ occ_2(t-3) \\ spd_2(t-3) \end{bmatrix}, \qquad l_3(t) := \begin{bmatrix} flw_3(t) \\ occ_3(t) \\ spd_3(t) \\ \vdots \\ flw_3(t-3) \\ occ_3(t-3) \\ spd_3(t-3) \end{bmatrix}.$$

Different measures that (partially) characterize the network dynamics and are related to the roads can be adopted for defining the (observation) vectors associated with the edges. For instance, one can use the distance between two connected locations for the corresponding edge if the edge direction is aligned with the road direction; otherwise, the negative of this value can be used. In our running example, since the distance between subsequent VDSs is 0.5 miles (see the *postmiles* in Fig. 2.1) this definition yields to

$$l_{(1,2)} = 0.5 \qquad l_{(2,1)} = -0.5 \qquad l_{(2,3)} = 0.5 \qquad l_{(3,2)} = -0.5 \ \cdot$$

Note that this definition yields to a set of time–invariant values that are not actual "observations". However, the GNN model is not constrained to this case and can parse graphs with edges that have time–varying values.

Targets: Each node of a graph is associated with a **known** *target* vector. Let $t_n(t)$ be the target vector of node n at time t and suppose that our objective is to predict the traffic flow in 2 steps ahead. The target values associated with the nodes of our graph example are

$$t_1(t) = flw_1(t+2) \qquad t_2(t) = flw_2(t+2) \qquad t_3(t) = flw_3(t+2).$$

These targets are related to the aforementioned observations and the graph topology through an unknown function g^*. The intuitive idea underlying the GNN model is that this information for a node n is split into two parts: (1) the node observation vector l_n, and (2) the information gathered from other nodes, edges and the graph topology. It is assumed that the latter form of information is encoded to an **unobservable** *adaptable state* denoted by $x_n^a \in \Re^s$

$$t_n = g^*(x_n^a, l_n). \tag{2.1}$$

A graphical model that illustrates this type of causalities for our running example is shown in Fig. 2.3. The colored nodes in the figure correspond to the known values. The "states" and causalities depicted by solid arrows are explained in the following part. The dotted arrows show that each target depends on the observation and the "state" associated with the same node (analogous to (2.1)). Note that both g^* and the states are **unknown** in (2.1).

States: The state of each node is aimed to collect related data (about the target) from the neighboring nodes' observations, arriving edges and the states of other nodes. Returning to Fig. 2.3, we note that the solid arrows are outlined based on this structure which is, indeed, governed by the network topology. For instance, in our running example, the state of node 1 presents the contribution of the traffic conditions at other nodes in changing the flow at node 1 in the future. This contribution is related to node 2 and edge $(1, 2)$ directly (note the arrows from l_2 and $l_{(1,2)}$ to x_2^a) and to the other nodes and edges indirectly (through x_2^a).

The GNN model considers a mapping called the *transition function*, which is denoted by f^*, to model the dependencies that are illustrated by the solid arrows in Fig. 2.3. More

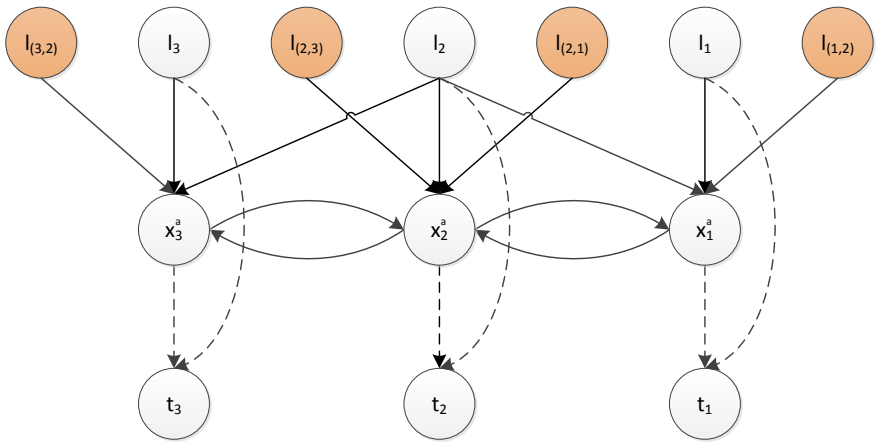


Figure 2.3: A graphical model that illustrates the causalities considered by the GNN model for our running example.

explicitly, for each node n, the transition function encodes the observations of adjacent nodes $l_{ne[n]}$, arriving edges $l_{co[n]}$ and neighbors' states $x^a_{ne[n]}$ into the current state x^a_n

$$x^a_n = f^*(l_{\text{ne}[n]}, l_{\text{co}[n]}, x^a_{\text{ne}[n]}). \tag{2.2}$$

For instance, in our running example we have $x^a_1 = f^*(\{l_1, l_2\}, \{l_{(1,2)}\}, \{x^a_2\})$. Similar to g^*, the transition function f^* is unknown. Based on (2.2), learning the states is equivalent to learning the transition function f^* and solving (2.2).

Objective: The GNN model receives a graph augmented with the aforementioned observations and generates an output vector for each node. These outputs are desired to be as close as possible to the targets. We denote the output node n at time t by $o_n(t)$. The outputs for the example under our study are of the form

$$o_1(t) = \widehat{flw}_1(t+2) \qquad o_2(t) = \widehat{flw}_2(t+2) \qquad o_3(t) = \widehat{flw}_3(t+2).$$

where $\widehat{flw}_n(t+2)$ is the flow of node n at $t+2$ predicted at time step t.

The GNN model exploits a parametric function f_w to mimic the dynamics of the transition function f^* (2.2) and estimate the sates

$$x_n = f_w(l_{\text{ne}[n]}, l_{\text{co}[n]}, x_{\text{ne}[n]}). \tag{2.3}$$

where x_n is the estimate of the actual state x^a_n. Furthermore, it deploys another parametric function, g_w, that aims to mimic the mapping from the states and observations to the outputs (c.f. the dotted arrows in Fig. 2.3 and (2.1))

$$o_n = g_w(x_n, l_n). \tag{2.4}$$

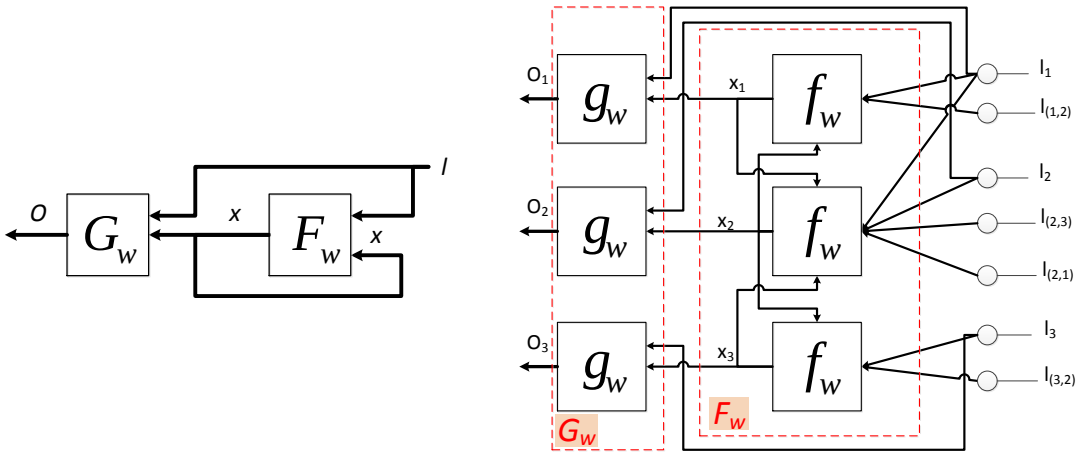


Figure 2.4: Succinct representation of the transition (F_w) and output (G_w) functions in the GNN model.

Different models can be adopted to structure the two functions f_w and g_w. The GNN model uses a special type of recursive neural networks as f_w and a feedforward fully connected NN as g_w. We refer the reader to Scarselli *et al.* (2009b) for the details of f_w. A supervised learning algorithm for training the weights of these two NNs is given in the sequel.

Let x, o and l denote the sets of all states, outputs and observations respectively. The GNN model defined by (2.3) and (2.4) in a succinct form can be presented by

$$x = F_w(x, l) \tag{2.5a}$$

$$o = G_w(x, l) \tag{2.5b}$$

which is illustrated by two block diagrams in Fig. 2.4.

The GNN dynamics presented in (2.3)–(2.4) implies that a successful implementation requires

1. Parametric functions for f_w and g_w. For instance, g_w can be a multi layer neural network and f_w can be either a linear or nonlinear function as long as (2.3) has solution.

2. A method to solve (2.3). Note that for a given f_w, the states appear in both the input and the output.

3. An algorithm that can learn the parameters of f_w and g_w based on an example set that includes pairs of the form (*targets*; *observations and graph topology*).

The next three subsections briefly discuss these requirements.

2.2.1 Method for State Computation

The uniqueness and existence of a solution to (2.5a) is guaranteed by Banach's theorem Khamsi & Kirk (2011) when F_w is a contraction map with respect to (w.r.t.) the state.

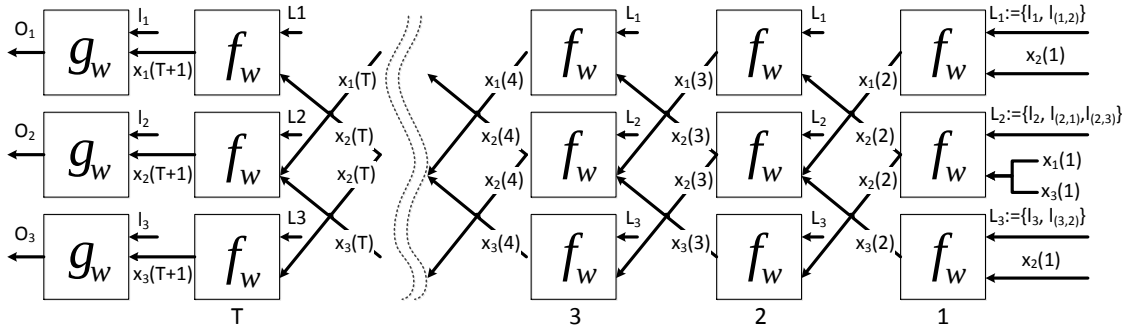


Figure 2.5: Schematic of an unfolded network that approximates the GNN for the 3–node graph shown in Fig. 2.2. All units of unfolded transition function use an identical f_w. Connections between layers follow the graph topology (c.f. (2.3)).

Moreover, the solution is equal to the convergence point of the following iteration

$$x_n(i+1) = f_w\left(l_{\mathrm{ne}[n]}, l_{\mathrm{co}[n]}, x_{\mathrm{ne}[n]}\left(i\right)\right) \tag{2.6}$$

initialized from an arbitrary state $x_n(1)$. Suppose that this update rule after T iterations yields to a set of states that satisfy some convergence criteria. For our running example, this iteration is equivalent to unfolding the transition function F_w as is illustrated in Fig. 2.5. The next subsection discusses parametric functions for f_w that yield to F_w's that are contraction maps.

2.2.2 Parametric Models for Transition and Output Functions

The output function g_w can be any general parametric function as long as it can be trained in a supervise paradigm, and the gradient of its output w.r.t. its input can be calculated. The original model proposed by Scarselli *et al.* (2009b) that is also deployed in this work is a multilayer feedforward neural network.

As for the transition function f_w, the choices are limited to the parametric functions that define a contraction map with respect to the states. Linear and nonlinear models that satisfy this constraint are discussed by Scarselli *et al.* (2009b). The nonlinear form that is adopted here is realized by a multilayer feedforward neural network that is guaranteed to produce a contraction map F_w when a penalized quadratic cost function in the from of

$$e_w := \sum_{j=1}^{p} \sum_{i \in N} \left(t_{i,j} - o_{i,j}\right)^2 + \beta L_\mu \left(\left\| \frac{\partial F_w}{\partial x} \right\|\right) \tag{2.7}$$

is used. Here, p denotes the number of example graphs in the train set, and $t_{i,j}$ stands for the "target" value at node i of graph j. The penalty function L_μ is equal to $(y-\mu)^2$ if $y \geq \mu$; otherwise, it equals 0. The two parameters β and $\mu \in (0,1)$ determine the penalty weight and the desired contraction intensity of F_w. The linear model for F_w and the proof for the contraction property can be found in Scarselli *et al.* (2009b).

2.2.3 Multiple Encoders

The transition network F_w implements an information diffusion mechanism that generates the states. The original form of GNN exploits one block of this encoding mechanism. The states are in general multidimensional, i.e. in $\Re^s$ where $s \geq 1$, and it is shown by Scarselli *et al.* (2009b) that the computational complexity of the most expensive instructions of the GNN training algorithm is related to s^2. Therefore, learning complex patterns through encoding layers that output high dimensional states is computationally intensive. Here, we propose an alternative that is inspired by the use of multiple filters in convolutional neural networks (CNN) LeCun & Bengio (1995). The architecture of a CNN is designed to benefit from the grid structure of the input domain, which is indeed a special form of graph structure. A convolutional layer in a CNN exploits several filters to provide different feature maps. The input to the filters is common among them and each filter attempts to extract a special feature from the input domain. Here, we adopt the same idea and propose using multiple *transition functions* that are aimed to extract different features of the spatio–temporal dependencies between the graph nodes.

Let n_f be the number of transition functions. These functions all receive the same graph as the input, but, attempt to reach individual states (equilibrium points). Analogous to (2.3), the states are computed by

$$x_n^i = f_{w_i}(l_{\text{ne}[n]}, l_{\text{co}[n]}, x^i_{\text{ne}[n]}) \quad i \in \{1, 2, \cdots, n_f\} \tag{2.8}$$

$$o_n = g_w(\{x_n^1, x_n^2, \cdots, x_n^{n_f}\}, l_n) \tag{2.9}$$

where x_n^i denotes the state outputted by encoding block f_{w_i}. Note that in (2.8) the transition functions perform independently and no information is interchanged between f_{w_i} and f_{w_j} when $i \neq j$. The states that contribute in shaping x_n^i are generated by the same transition function. A succinct form of this mapping can be written as

$$x^i = F_{w_i}(x^i, l), \quad i \in \{1, 2, \cdots, n_f\} \tag{2.10a}$$

$$o = G_w(\{x^1, x^2, \cdots, x^{n_f}\}, l) \tag{2.10b}$$

where $x^i := \{x_1^i, x_2^i, \cdots, x_{|N|}^i\}$. An abstract form of this interconnection for a GNN with $n_f = 3$ is shown in Fig. 2.6. We will show by experimental results that such an architecture with multiple transition functions and smaller states – at least in the application under our study – outperforms in both accuracy and training speed the original GNN model with one larger transition network that generates higher dimensional states.

2.2.4 Learning Algorithm

The learning objective is to find a set of parameters that form the functions F_{w_i}'s and G_w in (2.10) such that the cost function (2.7) is minimized. For simplicity, we only consider one transition function F_w. The algorithm can be extended to the case of multiple transition functions easily. A gradient–descent based algorithm is proposed by Scarselli *et al.* (2009b) that consists of the following three steps in each iteration t:

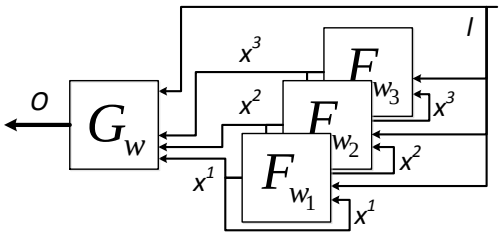


Figure 2.6: The porposed GNN model with three individual transition functions.

1. Let w_f and w_g denote the weights of f_w and g_w respectively. These weights are initialized randomly at iteration 0. For given values of w_f and w_g, the states are iteratively calculated by (2.6). Then, the outputs are computed based on the states and the observations using (2.4).

2. The gradients of cost function with respect to the parameters, $\frac{\partial e_w}{\partial w_f}$ and $\frac{\partial e_w}{\partial w_g}$, are computed by the method explained in the sequel.

3. Let w be a vector that contains both w_f and w_g. This vector is updated by

$$w(t+1) = w(t) + \frac{\partial e_w}{\partial w}|_{w=w(t),e_w=e_w(t)}$$

   where $\alpha(t)$ is the learning rate.

The first step was discussed in section 2.2.1 and the last step is straightforward. As for step 2, the gradient of cost function with respect to the weights in g_w, i.e. $\frac{\partial e_w}{\partial w_g}$, is calculated by backpropagation method for multilayer feedforward neural networks. The gradient computation with respect to the transition function (f_w) weights, i.e. $\frac{\partial e_w}{\partial w_f}$, is adopted from backpropagation–through–time algorithm that is used for recurrent neural networks Frasconi *et al.* (1998); Miller *et al.* (1995). The key idea is that the transition network is unfolded from time T back to an initial time t_0 similar to the schematic that was earlier shown in Fig. 2.5 for the forward path. This unfolding process creates $T - t_0 + 1$ layers that contains copies of f_w that are connected to each other based on the topological relations between the graph nodes (see Fig. 2.5).

Carrying out the unfolding procedure provides a network that consists of a set of (identical) feedforward neural networks. Therefore, the backpropagation through time is the same as traditional backpropagation on the unfolded network. The only difference is that all unfolded layers use the same weight vector w_f. This implies that the gradient of cost function w.r.t. w_f should be obtained by summing the gradient of the cost function w.r.t. the weights of all layers. In fact, backward unfolding is continued till this sum converges.

Backpropagation through time requires all values of states (c.f. 2.5) generated by the layers of unfolded network during the forward path. A more memory efficient approach based on the algorithms for backpropagation in recurrent neural networks Almeida (1990); Pineda (1987) is proposed in Scarselli *et al.* (2009b). The key idea is that, for large T, the

states in the deep layers converge to a stable equilibrium point and the states generated by the shallow layers have negligible effects in the gradient calculation. Accordingly, the states in all layers are approximated by the final state $x(T)$. Based on this unfolding process, the gradient of cost w.r.t. w can be calculated by

$$\frac{\partial e_w}{\partial w} = \frac{\partial e_w}{\partial o}\frac{\partial G_w}{\partial w}(x, l_N) + z\frac{\partial F_w}{\partial w}(x, l)$$

where z stands for the partial derivative of the cost function w.r.t. the states and can be iteratively calculated by

$$z(t) = z(t+1)\frac{\partial F_w}{\partial x}(x, l) + \frac{\partial e_w}{\partial o}\frac{\partial G_w}{\partial x}(x, l_N).$$

The sequence $z(t)$ converges to the fixed point z as $t \to -\infty$ (refer to Theorem 2 in Scarselli *et al.* (2009b) for the proof of this claim).

2.2.5 Putting It All Together

The update rule in (2.6) for calculating the equilibrium point of (2.5a) along with the traditional backpropagation through G_w and the backpropagation through time for F_w provides the machinery to define an explicit supervised learning algorithm for the GNN model. Algorithm 1 illustrates the details of the training procedure. Note that the states are passed as an input to `Forward(.,.)` procedure even though the states converge to the equilibrium point from any arbitrary initial value. The reason behind this is that the new equilibrium point is close to the last one when the learning factor is small. Therefore, the state calculated in the last forward path is a good point for initializing the iterations of the current forward path.

Algorithm 1 Supervised Learning Algorithm for the GNN Model

procedure TRAIN
  Initialize w, x
  $t \leftarrow 0$
  while (stopping criterion not satisfied) **do**
    $[\, x,\ o\,] \leftarrow \text{Forward}(w, x)$
    $\frac{\partial e_w}{\partial w} \leftarrow \text{Backward}(w, x, o)$
    $w \leftarrow w - \alpha(t)\frac{\partial e_w}{\partial w}$
    $t \leftarrow t + 1$
    update learning factor $\alpha(t)$
  end while
end procedure

procedure FORWARD(w, x)
  $x_{new} \leftarrow F_w(x, l)$
  while (convergence criterion not satisfied) **do**
    $x \leftarrow x_{new}$
    $x_{new} \leftarrow F_w(x, l)$
  end while
  $x \leftarrow x_{new}$
  $o \leftarrow G_w(x, l_N)$
  return $[\, x,\ o\,]$
end procedure

procedure BACKWARD(w, x, o)
  $A \leftarrow \frac{\partial F_w}{\partial x}(x, l)$
  $B \leftarrow \frac{\partial e_w}{\partial o}\frac{\partial G_w}{\partial x}(x, l_N)$
  initialize z (arbitrarily)
  $z_{new} \leftarrow zA + B$
  while (convergence criterion not satisfied) **do**
    $z \leftarrow z_{new}$
    $z_{new} \leftarrow zA + B$
  end while
  $z \leftarrow z_{new}$
  return $z\frac{\partial F_w}{\partial w}(x, l) + \frac{\partial e_w}{\partial o}\frac{\partial G_w}{\partial w}(x, l_N)$
end procedure

Chapter 3

Empirical Study

3.1 Dataset

The traffic dataset we have used in this research is obtained from California Department of Transportation (Caltrans) Performance Measurement System (PeMS) that measures traffic states by over 41'000 detectors (as of Aug, 2015). PeMS project started in 1999 as a university research project, was introduced in 2001 by Chen *et al.* (2001) and rapidly attracted many researchers in the following years Choe *et al.* (2002); Varaiya (2002); Chen (2003); Shekhar (2004); Lippi *et al.* (2010); Guo & Williams (2010); Lippi *et al.* (2013); Pan *et al.* (2013); Huang *et al.* (2014); Lv *et al.* (2015). This measurement system provides access to real–time and historical performance data in many formats to help engineers and researchers understand transportation performance, identify problems, and formulate solutions. PeMS utilizes a web–based interface PeMS (n.d.) to provide free and public access to the traffic dataset. The system covers more than 30'000 miles directional distance and incorporates more than 6'800 controllers, 41'000 detectors and 16'000 traffic census stations – all reported in Aug, 2015.

PeMS collects data from various types of vehicle detector stations, including inductive loops, side–fire radar, and magnetometers. Inductive–loop, the most common type of detection device currently used by Caltrans, records data from loops installed at specific locations on the freeway. The machine senses when vehicles travel over the loops by reading the number of times (flow) and for how long the inductance of the loops changes (occupancy). The controller sends this information to the District TMC via modem every 30 seconds. A schematic for PeMS data collection, analysis and publication work flow is shown in Fig. 3.1.

The state of California traffic network is divided into 11 districts in PeMS. We selected data of district 7 that includes Los Angeles and Ventura counties – which incorporate approximately 100 cities. This district has attracted many researchers Chen *et al.* (2003); Rice & Van Zwet (2004); Chen *et al.* (2005); Lippi *et al.* (2013); Pan *et al.* (2013) since it has the largest number of detectors ($\sim 10'000$) and most complex freeway network (consisted of over 40 highways). The smallest data samples, in terms of temporal aggregation, from VDSs are individual 5–minute lane points. We have aggregated data to 15–min disjoint intervals

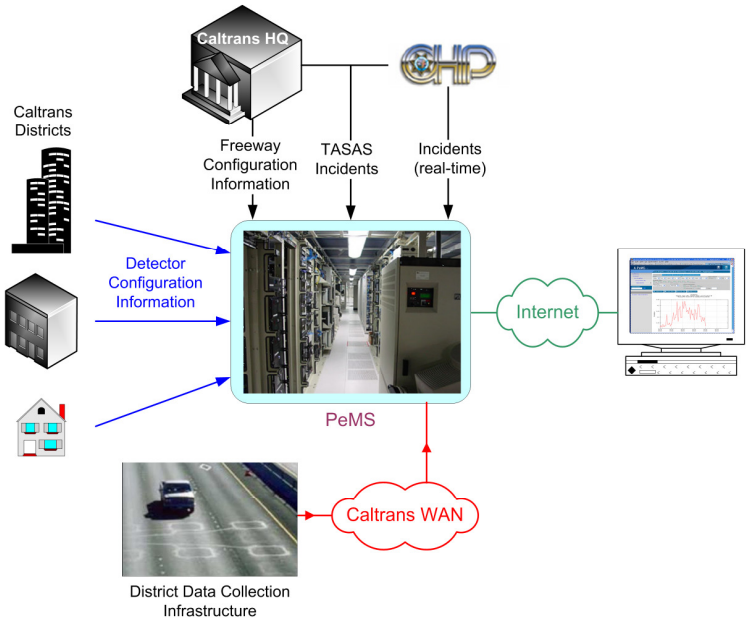


Figure 3.1: PeMS data collection and analysis infrastructure. (Reprinted from PeMS manual available on Caltrans (2014))

in order to increase the signal to noise ratio as many researchers have suggested Smith & Demetsky (1997); Smith *et al.* (2002); Shekhar & Williams (2008). Holidays in 2015 listed in Table 3.1 are removed from the dataset since the traffic trend in these days is considerably different than the regular days.

Performing experiments on this overwhelming amount of data and obtaining experimental results that can represent the prediction performance over the whole district either relies on time consuming computations, or requires careful considerations in sampling the data and monitoring its quality. Unfortunately, these two factors are ignored in many short–term traffic forecasting studies on this dataset. Many researchers have chosen a number of stations without any justification for their particular selection. Since the number of considered stations in research experiments, e.g. 20, is considerably smaller than the total number of available stations, i.e. over 3'600, the results may not generalize to the whole network. A traffic network may have a relatively small number of locations that are of utmost importance and have complex dynamics. In this case, uniform sampling of locations and focusing on the average behavior which is governed by the majority of locations with simple dynamics may overestimate the prediction algorithm effectiveness. Similar to spatial aggregation, high level of aggregation in temporal domain (e.g. in Lv *et al.* (2015) that aggregates the data of all detectors along a freeway to represent the whole freeway by a single aggregated value) has direct implications on the temporal structure of a time series because it eliminates variation in the data and alters most properties Chen *et al.* (2012); Dunne & Ghosh (2011).

The other important aspect in sampling the stations is "data quality", which has not re-

Table 3.1: 2014 Holidays in the United States that are removed from the dataset.

Date	Description	Day of Week
01/01/2014	New Year's Day	Wednesday
01/20/2014	Martin Luther King, Jr. Day	Monday
02/17/2014	Washington's Birthday	Monday
05/26/2014	Memorial Day	Monday
07/04/2014	Independence Day	Friday
09/01/2014	Labor Day	Monday
10/13/2014	Columbus Day	Monday
11/11/2014	Veterans Day	Tuesday
11/27/2014	Thanksgiving Day	Thursday
12/25/2014	Christmas Day	Thursday

ceived enough attention in experiments relying on PeMS datasets. Vehicle detectors provide the best source of real–time traffic data available to Caltrans. However, the data stream can contain missing or incorrect values that require careful analysis to produce reliable results. Therefore, the data used in an experiment could be "original" (obtained from a detector) or be "imputed". In the latter case, spatial correlation between detectors and temporal correlation with the past measurements are used by numerous imputation algorithms Chen *et al.* (2003) to correct the data. When imputed values are used in data driven prediction methods, the results are questionable since the input and target data are not necessarily original. Moreover, the input data may contain information about the future if the imputation method was not following time causality. PeMS reports a reliable measure of data quality, called *percent observed*, that represents the ratio of actual measurements to the total reported data.

The station candidates in our experiments are chosen based on the data *variance* and *quality* in order to achieve generality and accuracy. The notion of *variance* is quantified based on the deviation of flow from its seasonal (periodic) mean obtained by averaging on the same days of the past weeks. Suppose that the train set includes N_T seasons of length T. For instance, when seasonality corresponds to weekly periodicity and data is aggregated to 15–min intervals, the period T is 96×7. The deviation of the state $X(t)$ from seasonal mean is then

$$X_d(t) := X(t) - \bar{X}(\text{mod}(t, T))$$

where the seasonal mean $\bar{X}(k)$ is

$$\bar{X}(k) := \frac{1}{N_T} \sum_{m=0}^{N_T-1} X(mT + k) \quad 0 \leq k \leq T - 1.$$

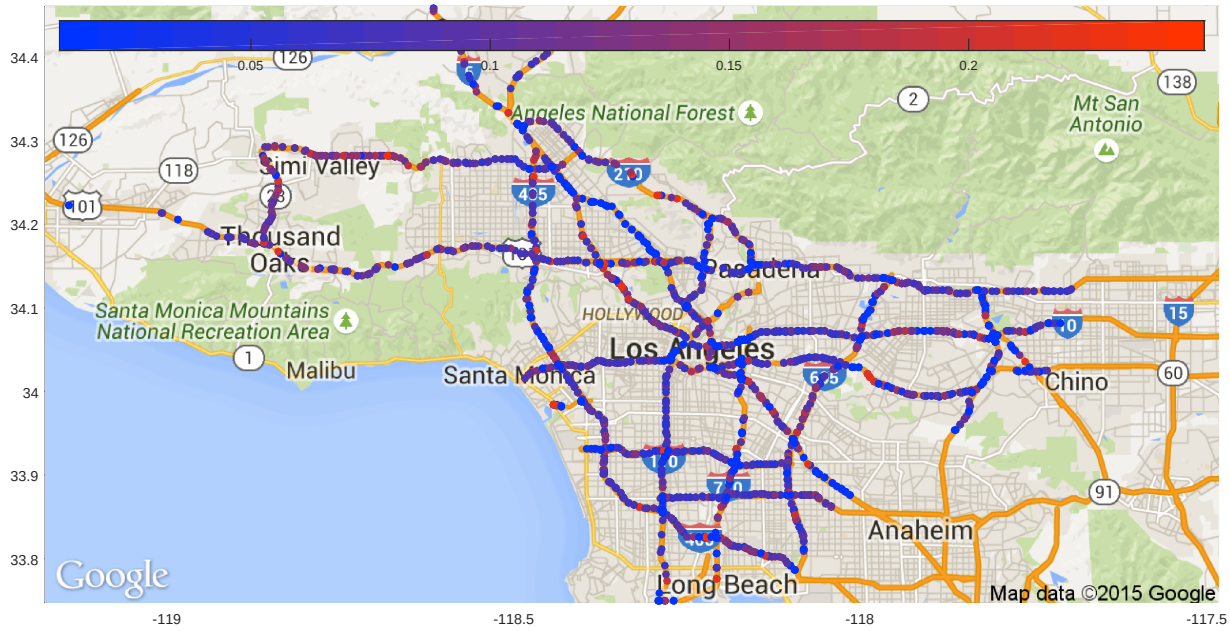


Figure 3.2: RMS of flow deviation from seasonal mean for all VDSs in District 7 (LA/Ventura counties). The axes represent longitude and latitude in a geographic coordinate system.

Then, the data variance can be related to the root mean square of deviation by

$$V := \frac{1}{N_T T} \sum_{t=0}^{N_T T-1} |X_d(t)|^2. \tag{3.1}$$

Similarly, the relative deviation can be quantified by

$$V_r := \frac{1}{N_T T} \sum_{t=0}^{N_T T-1} |\frac{X_d(t)}{X(t)}|^2. \tag{3.2}$$

In general, larger V or V_r means larger variation from intra–day trends. Note that (3.1) is similar to the mean variance of a cyclo–stationary process. We can also think of V or V_r as the prediction performance when the seasonal mean is used as the predictor. Therefore, small V or V_r implies that prediction methods as simple as historical means are competent to obtain high levels of accuracy. The V_r value for flow of all *mainline* (ML) and *freeway–to–freeway* (FF) stations in district 7 are color coded and shown in Fig. 3.2. The light blue nodes correspond to the stations with simple behavior and the light red ones depict the highly non–stationary nodes. Figure 3.3 shows a closer view of the stations deviation in the LA area.

We have divided the stations into three categories based on relative variance V_r. These groups are called low, mid and high variance and are defined by three intervals on V_r, namely

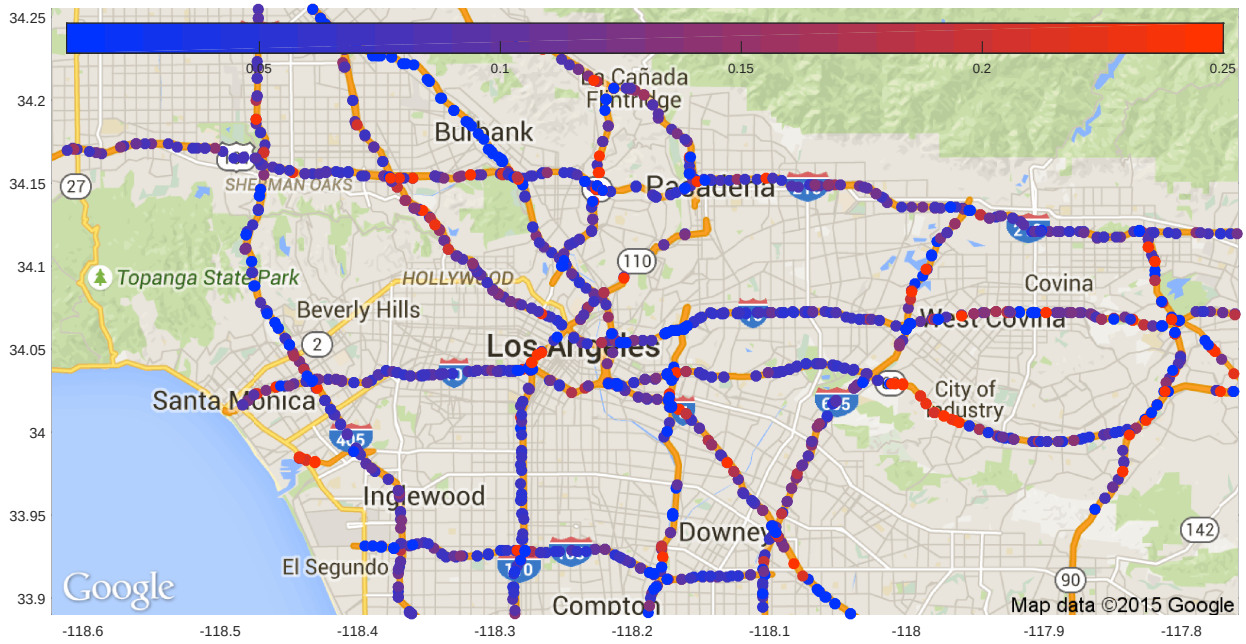


Figure 3.3: RMS of relative flow deviation after removing seasonal mean in LA and its neighborhood. The axes represent longitude and latitude in a geographic coordinate system.

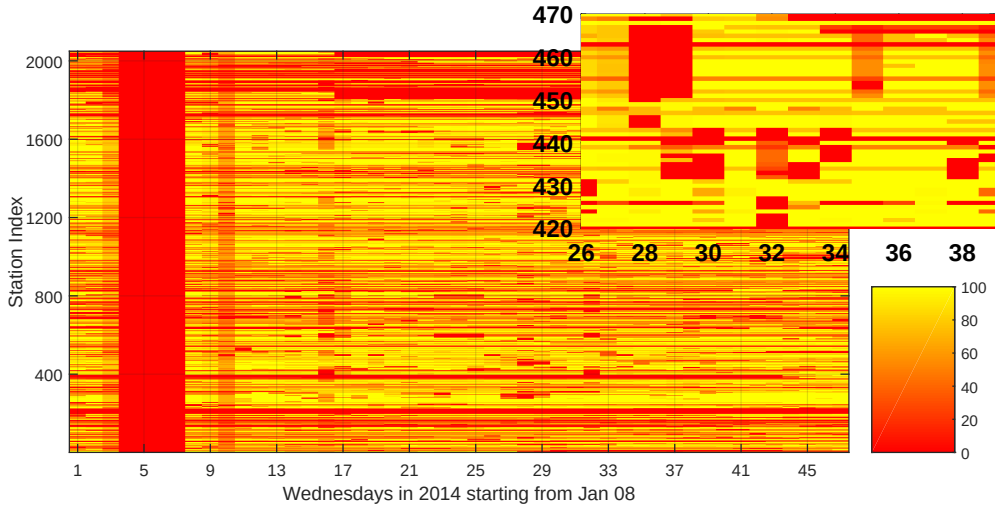


Figure 3.4: Historical values of "%Observed" for "FF" and "ML" stations in district 7. The stations are indexed from 1 to 2049 and the Wednesdays (Jan 08–Nov 30, 2014) are indexed from 1 to 47. The sub–figure on top shows a closer view of a portion of the main plot.

$V_r \leq 9\%$ for low variance (light blue circles), $9\% < V_r \leq 15\%$ for mid variance (purple circles) and $15\% \leq V_r$ for high variance stations (light red circles).

As mentioned earlier, we incorporate the data quality in addition to the stationarity of

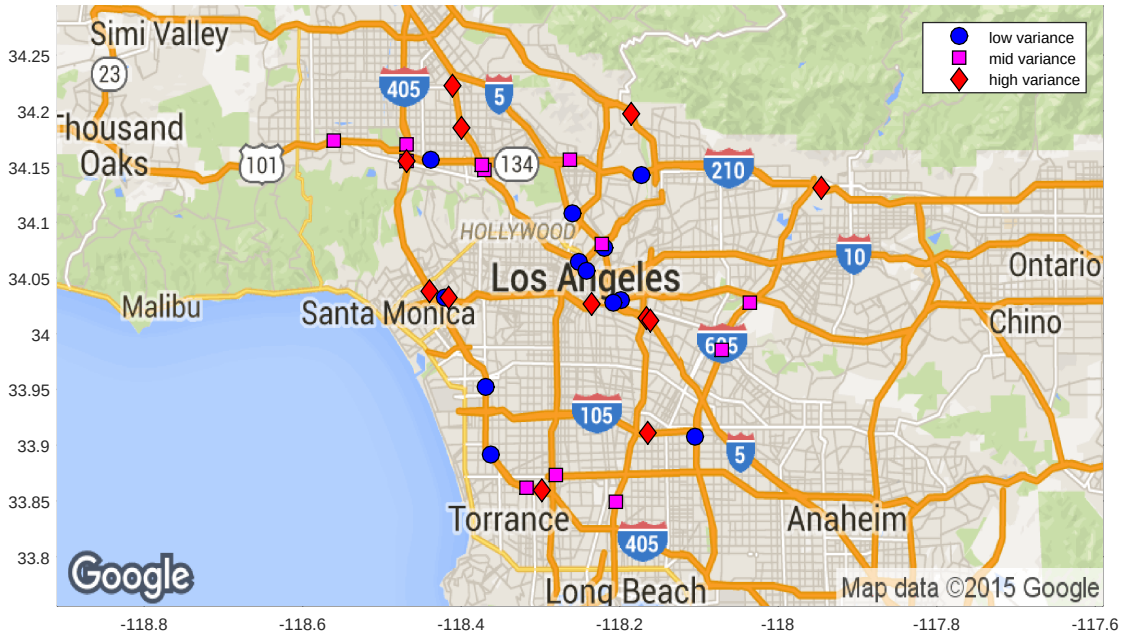


Figure 3.5: Location of stations selected based on seasonality variance and "%Observer" value for the empirical study.

data in sampling the stations for experiments. The data quality, in term of %Observed is shown in Fig. 3.4 for the same stations. The figure only shows the Wednesdays in 2014, starting from Jan 08 (Jan 01 was a holiday) until Nov 30. As shown in the figure, especially in the subplot on top, numerous measurements are missed (red cells). Among the stations that are categorized based on their variances, we only consider the ones that have at least 50% observation and select 12 stations from each category. These stations are shown in Fig. 3.5 and are listed in Table. 3.2.

Table 3.2: The set of stations used in our experiments. These stations have high quality data and they are categorized based on their flow deviation from seasonal mean.

Deviation	ID	Name	Freeway	Type	Lanes	Longitude	Latitude
LOW	716956	FLETCHER	5–N	ML	5	–118.259	34.108
	717264	LORENA	60–E	ML	5	–118.199	34.03
	717599	SAN RAFAEL	134–W	ML	4	–118.172	34.143
	717848	FM 105 EB	605–S	ML	4	–118.105	33.907
	718277	INGLEWOOD 2	405–S	ML	4	–118.362	33.891
	737292	WESTWOOD	10–W	ML	5	–118.421	34.032
	761492	CENTURY 2	405–N	ML	6	–118.369	33.952
	763237	PASADENA	5–N	ML	4	–118.219	34.077
	765182	N OF 110	101–N	ML	5	–118.251	34.065
	769444	HAZELTINE	101–S	ML	5	–118.438	34.156
	773023	BROADWAY	101–N	ML	4	–118.242	34.057
	773281	EUCLID	60–E	ML	5	–118.208	34.028
MID	716670	VAN NESS	405–N	ML	4	–118.318	33.862
	716810	ROSEHEDGE	605–S	ML	4	–118.072	33.986
	716946	AVE 26	5–S	ML	3	–118.222	34.081
	717490	VINELAND	101–S	ML	5	–118.371	34.147
	717582	CENTRAL	134–W	ML	4	–118.262	34.156
	717816	VENTURA	405–S	ML	5	–118.469	34.155
	717942	PECK 2	605–S	ML	4	–118.036	34.028
	717962	DEL AMO 2	710–N	ML	4	–118.205	33.849
	718141	MOORPARK	101–N	ML	3	–118.375	34.151
	764425	FIGUEROA	91–W	ML	3	–118.281	33.873
	767053	BURBANK 2	405–S	ML	4	–118.468	34.169
	772610	CORBIN	101–S	ML	5	–118.561	34.173
HIGH	715944	FERRIS	5–N	ML	4	–118.166	34.014
	716456	FACADE	105–E	ML	5	–118.164	33.911
	716573	ROSCOE 2	170–S	ML	3	–118.411	34.223
	717742	NORMANDIE1	405–N	ML	4	–118.298	33.86
	717795	PICO	405–S	ML	6	–118.439	34.038
	717816	VENTURA	405–S	ML	5	–118.469	34.155
	737158	OLYMPIC 2	10–E	ML	5	–118.235	34.026
	763325	OVERLAND	10–W	ML	4	–118.416	34.032
	764115	VICTORY 1	170–N	ML	4	–118.401	34.185
	764435	TRIGGS	5–N	ML	4	–118.161	34.011
	769806	FOOTHILL	210–E	ML	4	–118.185	34.197
	772858	S.G. RIVER	210–W	ML	4	–117.946	34.131

3.2 Retrieving Spatial Interconnections

The graph structures that abstract the spatial correlation between traffic stations can be constructed based on the freeway network anatomy that governs the traffic stream. Let n be the station under study and N_n be the set of stations that are supposed to collaborate in predicting traffic state at n. Note N_n includes n since the n's measurements are usually very informative about the future of its traffic state Chandra & Al-Deek (2008). A natural way of defining the traffic network in a graph structured framework is to associate nodes to n and N_n. We will abuse notation and use the same symbols for referring to the actual traffic monitoring stations as well as their node representatives in a graph domain. Therefore, n refers to the actual station as well as the node considered in the corresponding graph(s). For each processed station n at time t, we define a graph $G_n(t)$ by the pair $(N_n(t), E_n(t))$ where $N_n(t)$ is a discrete set of size $s_n(t)$ and $E_n(t)$ is an $s_n(t) \times s_n(t)$ square matrix representing the connections between the nodes (stations). Construction of these two crucial graph elements is discussed in the following two sections.

3.2.1 Graph Edges

The edges are aimed to sketch how information diffuses throughout the graph. This can be abstracted by considering a set of connections between the nodes that are realized by the graph edges. The notion of connection between two nodes of the graph G_n has a counterpart in the actual traffic network. That is, one node is connected to another node only if the traffic stream can diffuse from the first node to the second one – a dependency that is inherently governed by the traffic network anatomy. These connections are easy to retrieve from the physical location of nodes when they are in a road section with no injection or leakage – e.g. no on–ramp and off–ramp in a highway section. Indeed, it is obvious that the cars at the current point are coming from an upstream region and will pass the downstream station shortly. Therefore, each node should be simply connected to its upstream and downstream neighbors. This type of simple dependency has been studied by Chandra & Al-Deek (2008, 2009); Vlahogianni *et al.* (2005). The mainline VDS with ID 402580 considered in Li *et al.* (n.d.) is such an example. Figure. 2.1 shows this station and its neighbors that are all located in U.S. 101 (south) highway in district 4 of PeMS. The "upstream" and "downstream" stations with respect to this station can be easily determined based on their geographic location and highway direction. For instance, the Euclidean distance between station 402578 and other stations in U.S. 101–south highway as in Fig. 2.1 determines the two neighbors of this station, and by comparing the latitude of these stations the upstream (norther) and downstream (souther) stations will be distinguished.

However, when it comes to a complicated network such as an intersection of multiple freeways, freeway sections with on–ramps/off–ramps and more complex structures as in arterial road networks, inferring connections is a more involved process. In this case, defining the upstream and downstream neighbors is not an easy task and the dependencies cannot be determined only based on sensor locations. The Euclidean distance will not be a good

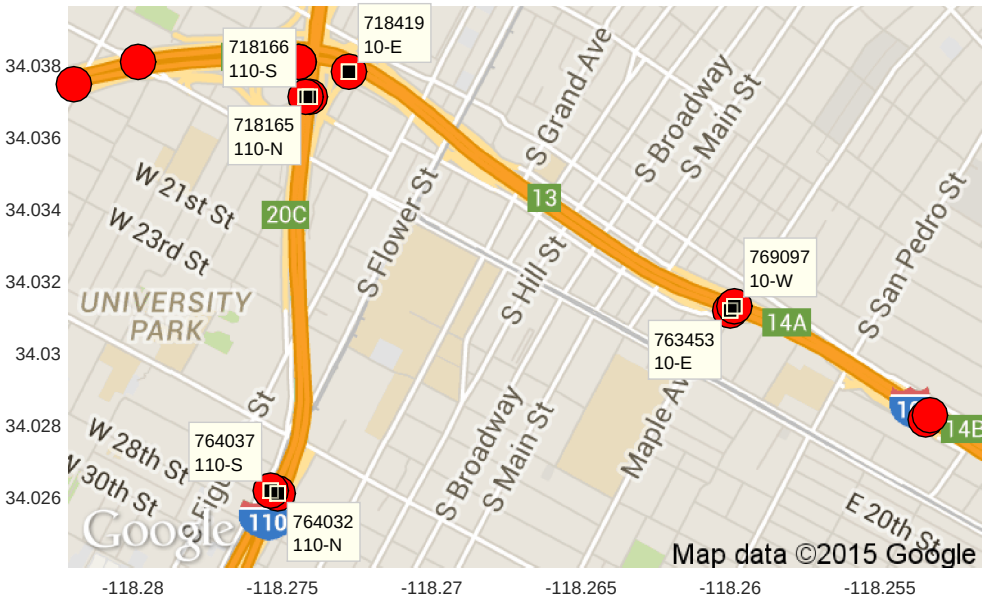


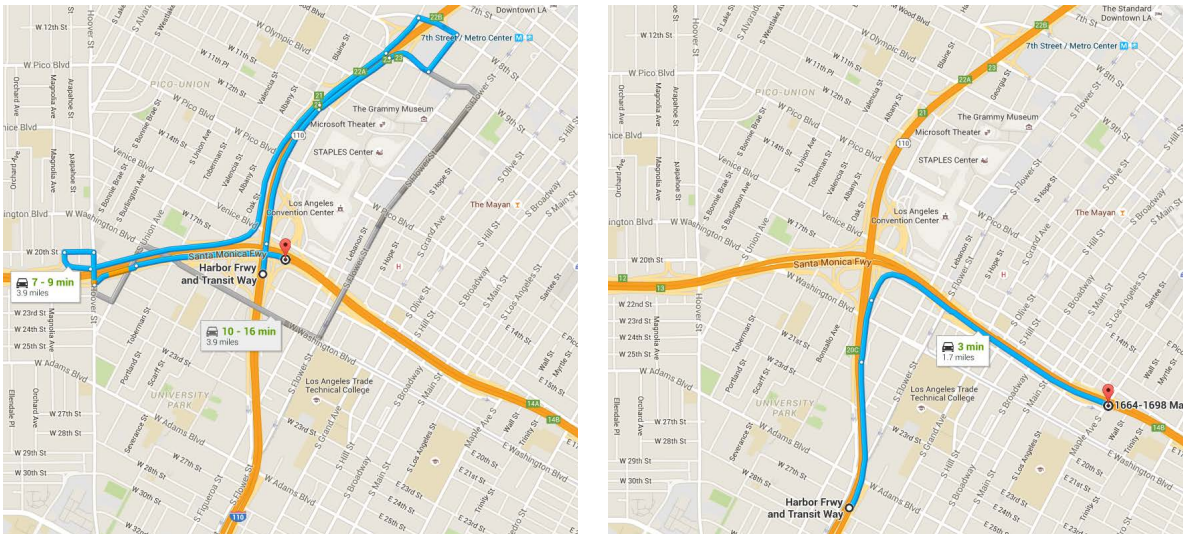
Figure 3.6: `U.S. Route 110` and `10` intersection. The Euclidean distance between the VDS locations cannot retrieve the correct spatial interrelations in this case.

measure of dependency in that case because sensors from different roads might be close to each other in Euclidean space while no short road exists between them. Figure 3.6 shows a typical intersection (highways 10 and 110). The two stations 718165 and 718419 are very close in Euclidean space, but in fact the traffic stream does not flow from 718165 to 718419 directly since the first station is located after the exit from 110–N and 718419 is located before the entrance to 10–E. Indeed, in this particular case the two stations 764032 and 763453 are the ones that characterize the traffic stream from 110–N to 10–E even though the direct distance between them is not the smallest one.

An alternative measure for determining dependencies is the "road distance" between stations. It lacks the aforementioned shortcoming since it is an implicit function of traffic network topology. For instance, suppose n_1 is a station in 110–N and n_2 denotes any station in 10–E. It can be shown that among all pairs of (n_1, n_2) the smallest road distance, which is 1.7 miles, is achieved when n_1 and n_2 correspond to stations 764032 and 763453 respectively. The road paths between these two stations and the other pair of stations mentioned above are illustrated in Fig. 3.7.

The main obstacle in using road distances to retrieve spatial dependencies is that obtaining such a measure requires a profound database about the road networks. We use "The Google Directions API"[1] – a service that calculates directions between locations using an HTTP request – in order to obtain road distances between two locations – recall that PeMS provide latitude/longitude geographic coordinates for stations. Calculating directions is a

[1] https://developers.google.com/maps/documentation/directions/intro



(a) (b)

Figure 3.7: `U.S. Route 110` and `10` intersection. (a) Shows the two stations in 110–N and 10–E that have the smallest Euclidean distance. The road distance between these two stations, reported by Google Maps, is 3.9 miles. (b) Illustrates the two stations that have the smallest road distance (1.7 miles). These stations characterize the traffic stream from 110–N to 10–E. (images from Google map)

time and resource intensive task and Google limits the number of requests per day (2500 free requests per day as of Aug 2015). Since there are numerous stations installed in the freeway network, calculating the direction for each possible pair of stations is a time consuming process. We categorize the graph edges to two groups, namely the edges that connect two stations on the same freeway (including the direction) and the ones that connect stations that belong to two freeways (i.e. at intersections). Constructing the first group of edges is straight forward as discussed above. As for the second group, we can consider any possible pair of stations that do not share the same freeway and ignore the ones that have the Euclidean distance greater than a threshold. This is motivated by the fact that the road distance between two locations is bounded below by the Euclidean distance. Therefore, for a suitably chosen threshold it can be guaranteed that the two stations with a larger Euclidean distance do not characterize the intersection of two roads. This heuristic considerably decreases the need of acquiring driving directions from the API. The pair of stations that are not filtered out by this criterion will be evaluated by calling the Google Directions API and the pair providing the minimum distance is chosen as the one modeling interconnection of two roads. When no pair passes the fist filtering phase, we conclude that the two roads do not intersect. Note that there is other information reported by the API such as the driving duration (in minutes) and also the number of steps (e.g. turning, merging, etc.) required to reach the destination. This information can also be utilized to define some sort of distance metrics between two locations, especially in this particular case that deals only with rela-

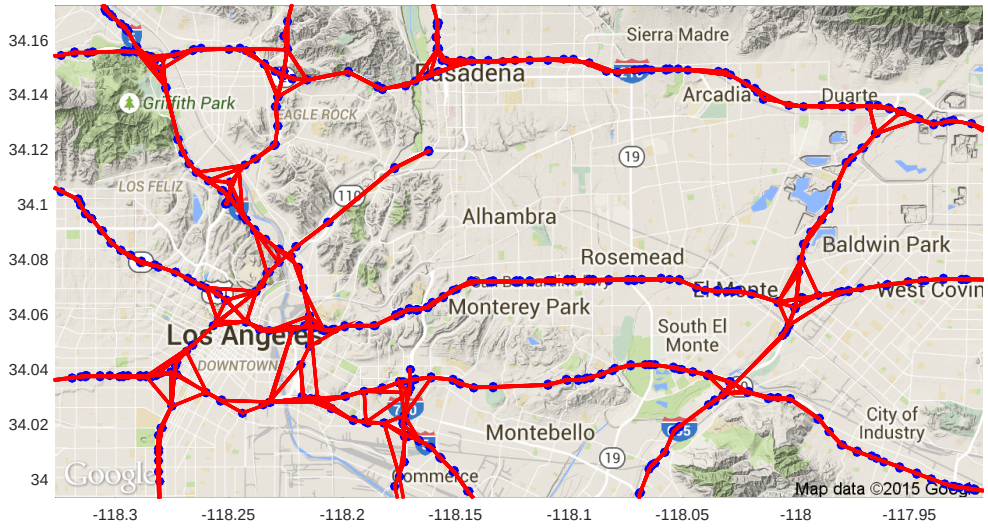


Figure 3.8: Stations spatial interrelations represented by a graph structure. The nodes, shown by blue circles, represent the stations and the graph edges, shown by red lines, denote station dependencies enforced by the network topology. Edge directions are not shown.

tive distances. For instance, we can determine the intersection by looking for the minimum driving duration from one station to another one and define the threshold on the number of steps required to take the path. Our experiments show that both approaches will result in the same connectivity set when the thresholds are set correctly. Figure 3.8 shows the graph edges created by following the aforementioned procedure. The edges are indeed directed from each node (station) toward its downstream neighbors. However, the directions are not shown in the figure due the space limitation.

We refer to this graph, that consists of all nodes and edges in district 7, as the global graph associated with the whole district 7. In the following parts, subgraph structures that present local spatial correlations around a target station will be constructed based on the global graph.

3.2.2 Local Receptive Fields

Prior work on utilizing spatial relations in traffic forecasting has illustrated that a longer prediction horizon can be achieved by broadening the set of stations considered in the forecasting analysis Head (1995); Stathopoulos & Karlaftis (2003); Vlahogianni *et al.* (2004). On the other hand, it was also being remarked by the same line of research that the information attained from far locations is subjected to distortion and non stationarity, especially in complex road networks. Moreover, processing information from numerous far sensors is not plausible from a computation point of view since it requires intensive calculation and time. These two factors motivate us to consider local receptive fields around the stations under study, rather than processing the full global graph that contains data from over 2000

stations spread in an approximately 40 miles × 40 miles region.

Different notions of locality and neighborhood can be adopted for defining local receptive fields. Here, we define a neighborhood of radius r around a target station n, say $N_{n,r}$, as the set of all nodes that can be reached from the target after passing at most r edges without respecting the edge directions. For instance, neighborhoods of radius 0 contains only the target, i.e. $N_{n,0} = \{n\}$, and as of radius 1 includes the target in addition to the nodes sharing an edge with the target.

Suppose C is the connectivity matrix of the global graph in a logical format. More formally, $C_{i,j} = 1$, if there exists an edge from node (station) i to node j and it is zero otherwise. It is obvious that for any target n, the neighborhood of radius 1 contains only the nodes with non–zero values in the n^{th} row and n^{th} column of C. The neighborhoods with larger radius, $r > 1$, can be determined similarly based on the n^{th} row of C^r and $(C^T)^r$. This can be shown by induction. Assume that the claim is true for some $r \geq 1$ and $|N|$ denotes the cardinality of the node set. We have

$$(C^{r+1})_{i,j} = \sum_{m=1}^{|N|} (C^r)_{i,m} C_{m,j} \tag{3.3}$$

which implies that $(C^{r+1})_{i,j} > 0$ if there exists at least one m such that $(C^r)_{i,m} > 0$ and $C_{m,j} > 0$. This is equivalent to reaching m in at most r steps and then going to j by passing another link. Therefore, there exists a path from i to j with at most $r + 1$ links if and only if $(C^{r+1})_{i,j} > 0$. Since this is true for $r = 1$, it also holds for any integer $r > 1$. The same argument can be made for finding upstream nodes that are at most r edges far from n by looking at $(C^T)^r$. A set of receptive fields with different radius r around station 715944 (a high variance node) is shown in Fig. 3.9. For instance, Fig. 3.9a depicts the stations that are located 1 link away from station 715944 on upstream (one node) and downstream (two nodes). Figure 3.10 depicts the histogram of graph sizes when radius of receptive field is 10. The plot illustrates that the notions of "upstream" and "downstream" stations become complex when the receptive field radius increases. When the node is located in a freeway where no other road is close to it, it will have 10 stations in upstream and 10 stations in downstream, which in total makes a graph of 21 nodes (including itself). This is the straight forward case that was mentioned above. However, the histogram shows that only 12% (244 from 2049 stations) fall in this category. The remaining 88% of the graphs have a more complex topology and can have a very wide range of node numbers.

Measurement at any node in $N_{n,r}$ might be missed at any time instance t. Hence, it is required to monitor the $\%Observed$ quantity of nodes to assure that the data is high quality. We consider a threshold on the $\%Observed$ value for each node in $N_{n,r}$ and include a node in the dataset if its $\%Observed$ value is higher than the threshold. This set, denoted by $N_n(t)$, determines the nodes of the local graph $G_n(t)$. The set of edges for $G_n(t)$ can be extracted from C by only considering the rows and column corresponding to the elements of $N_n(t)$. Note that the graph might become disconnected once the "unhealthy" nodes are removed. In that case, we can remove the parts that do not contain n since the information from these

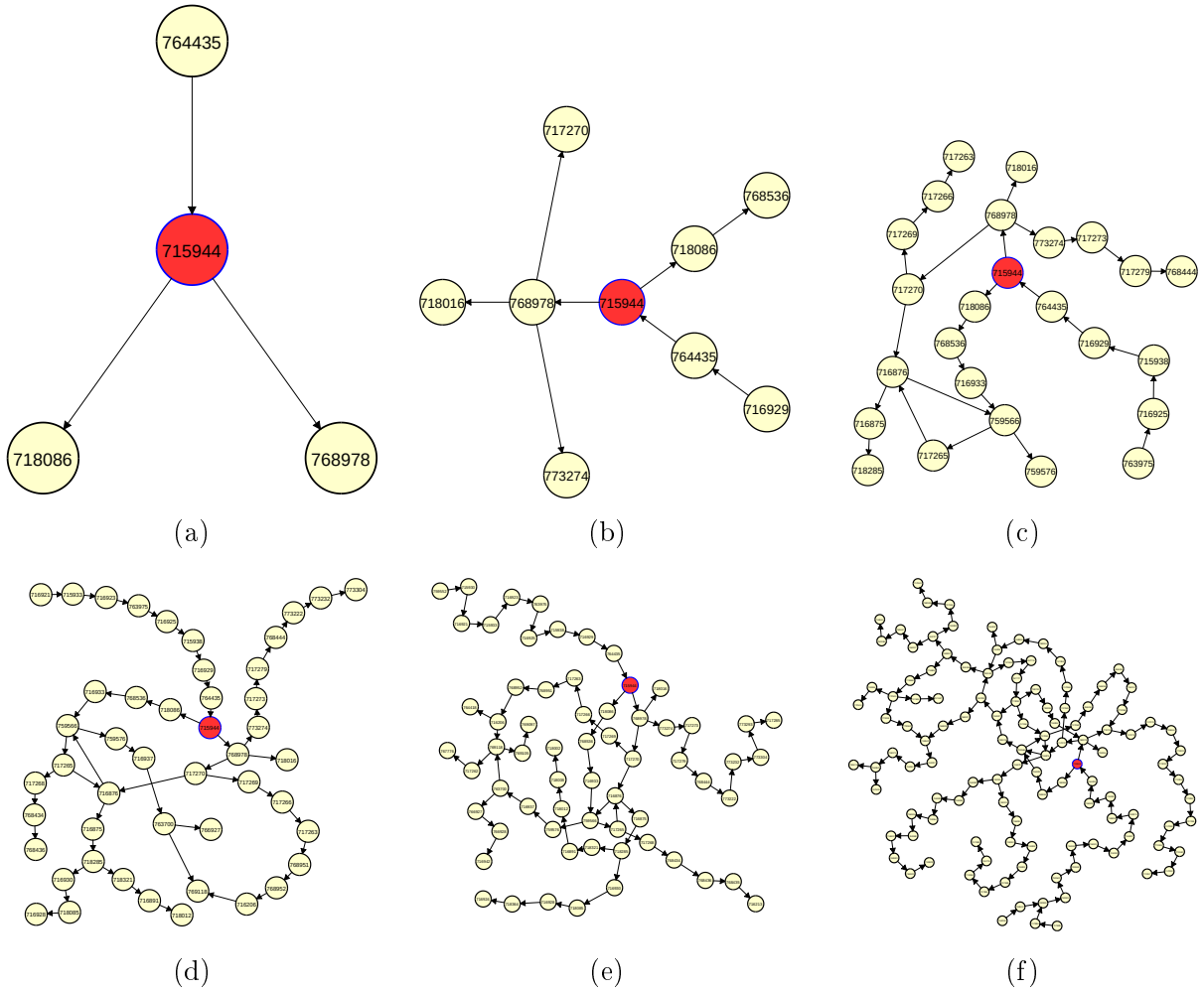


Figure 3.9: Receptive fields with different radii around station 715944. (a) $r = 1$, (b) $r = 2$, (c) $r = 5$, (d) $r = 8$, (e) $r = 10$, (f) $r = 15$.

nodes will not diffuse to n. Another possibility is to connect the children of "unhealthy" nodes to their grand parents and repeat this procedure till separated parts get attached to the subgraph that contains n.

3.3 Results of Case Studies

The proposed prediction methodology is applied to the aforementioned dataset and the results for flow forecasting for different GNN configurations are compared with various algorithms from trivial predictors to methods known as the state of the art by the ITS community. These methods include

1. Random Walk (RW): This is the simplest method which assumes that the state will

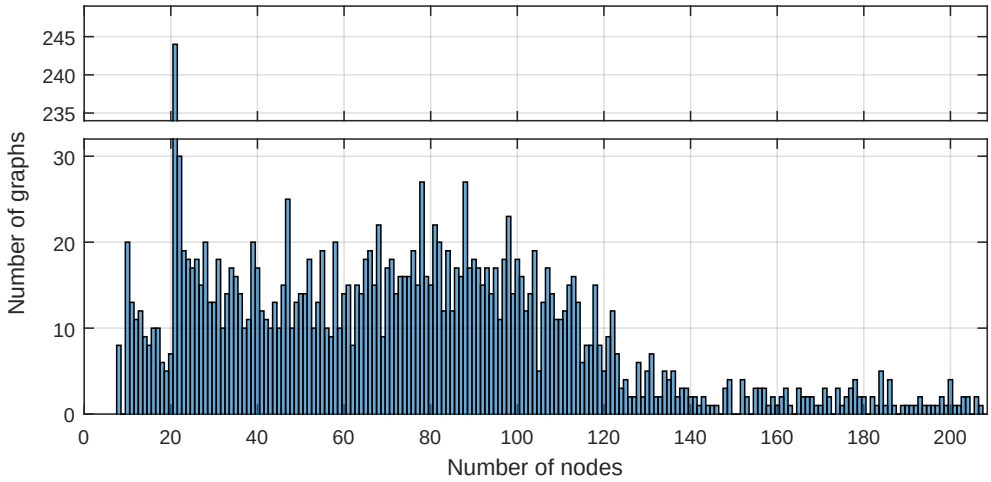


Figure 3.10: Graph size histogram for receptive field of radius 10 around station 715944. 244 for graphs out of 2049 fall in a straightforward category which has 21 nodes (large bin at 21). The remaining graphs correspond to more complex cases.

not change in the future. Therefore, predicts the state at any future time as equal to the current state.

2. Seasonal Mean (SM): The traffic state at future time is predicted as equal to the average of corresponding states at past "seasons". As was noted by Williams & Hoel (2003) and quantified more recently by Chandra & Al-Deek (2009), the seasonality effect is more apparent when measured over 1–day or 1–week time intervals. Here, we consider weekly trends as the seasonal means. For instance, the SM used on a Wednesday is obtained only from past Wednesdays.

3. Autoregressive integrated moving average (ARIMA) Moorthy & Ratcliffe (1988); Lee & Fambro (1999): A grid search is carried over all admissible values of p, d and q which are, respectively, less than $p_{\max} = 5$, $d_{\max} = 2$ and $q_{\max} = 5$. See (1.2) for the details.

4. Seasonal autoregressive integrated moving average (SARIMA) Williams *et al.* (1998); Williams & Hoel (2003): The parameters of SARIMA model in (1.3) are $(1, 0, 1)(0, 1, 1)(7\times 96)$. This model is similar to Lippi *et al.* (2013) and Williams & Hoel (2003).

5. Support vector regression models (SVR) with RBF kernel: We use the same model parameters as SVR_{RBF} in Table II of Lippi *et al.* (2013).

6. Vector auto regressive (VAR) model Chandra & Al-Deek (2009): The model used here is VAR(6) with 2 upstream and downstream sensors. Details of this model can be found in Chandra & Al-Deek (2009).

7. Feedforward Neural Network (FNN) Vlahogianni *et al.* (2005); Lv *et al.* (2015): "Shallow" (2–layer) MLP and deep architectures are both considered and the system configuration parameters are optimized by a grid search.

One of the biggest challenges in effective utilization of neural networks – despite what type of MLP is being used – is finding an optimal network structure. We use a grid search method, due to the ease of parallelization, to find the (sub)optimal hyperparameters. The grid search is carried out by a pool of 12 workers on a `3.40GHz Intel(R) Core(TM) i7-3770` processor. The design hyperparameters that should be determined for the graph neural network model are as follows:

1. Number of layers, number of neurons in each layer and type of activation functions in the output network g_w in (2.4).
2. Number of encoding blocks f_{w_i}, n_f, defined in (2.8).
3. States dimension s (recall that $x^i \in \Re^s$).
4. Number of layers, neurons in each layer and activation functions for $f_{w,i}$.
5. Parameters of termination criterion.

The second type of hyperparameters correspond to the modeling phase

6. Input domain: any combination of *flow*, *occupancy* and *reported speed* (i.e. speed that is measured directly and not inferred).
7. Spatial extent: the neighborhood radius for defining receptive fields (c.f. 3.2.2).
8. Temporal extent: number of past observations used in the input – i.e. n_t in (1.1).
9. Edge labels: an edge (n_1, n_2) can be either unlabeled or labeled with a value that encodes the distance between n_1 and n_2 as well as the direction of edge. In the latter case, the labels are defined with respect to each node. For instance, when edge (n_1, n_2) is used for x_{n_1} in (2.3), it can have a label of $+0.5$ which means that the direction of edge is toward n_1 and the distance from n_2 is 0.5 miles. The same edge, when it is used in the processing of x_{n_2} will have a label of -0.5 encoding that n_1 is located 0.5 miles downstream from n_2.

Performance Metrics: The prediction accuracy is commonly reported by the *Mean Relative Percent Error* (MRPE) that is defined as

$$\text{MRPE} := \frac{1}{m}\sum_{i=1}^{m} \left|\frac{X^i - \hat{X}^i}{X^i}\right| \times 100 \tag{3.4}$$

where X^i and $\hat{X}^i$ denote the actual and predicted values (e.g. for flow) of data instance i. However, this criterion is sensitive to the small values of X^i. In practice, even a single instance i can increase the MRPE of a "good" predictor. On the other hand, a model that fits to all data instances is not necessarily beneficial since it might be overfitted. Here, we evaluate the performance by a modified MRPE metric that is called $\text{MRPE}_{6\sigma}$. We consider an interval of $\pm 6\sigma$ around the MRPE and consider the values that fall outside the interval as *outliers*. Then, MRPE is recalculate for the values that are located inside this interval

$$\sigma^2 := \frac{1}{m}\sum_{i=1}^{m}\left(|\frac{X^i - \hat{X}^i}{X^i}| \times 100 - MRPE\right)^2$$

$$I := \{i \mid \text{MRPE} - 6\sigma \leq 100|\frac{X^i - \hat{X}^i}{X^i}| \leq MRPE + 6\sigma\}$$

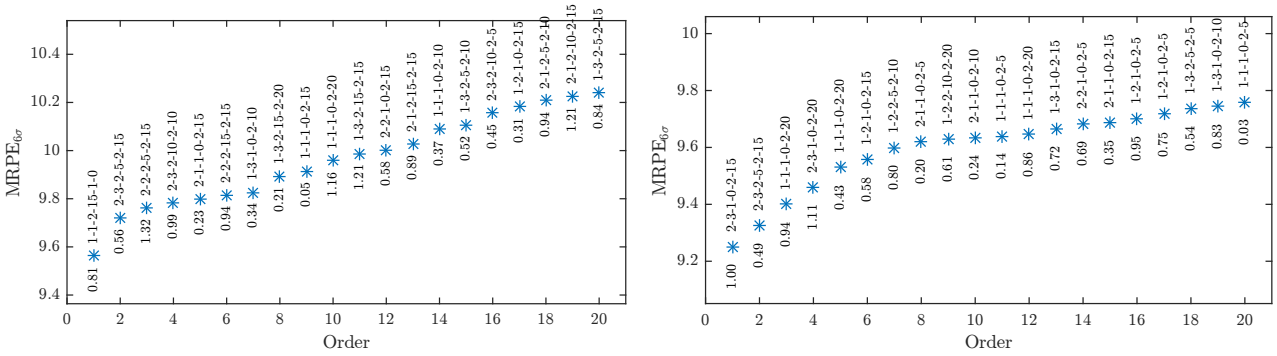
$$\text{MRPE}_{6\sigma} := \frac{1}{|I|}\sum_{i\in I}|\frac{X^i - \hat{X}^i}{X^i}| \times 100.$$

We observed in our experiments that at most 0.2% of data (1 out of 500) corresponds to the outliers. However, this small portion could increase the MRPE by 1% which is relatively a large value.

Training and Termination Criterion: The MRPE can be used directly as the cost function in training. However, expression (3.4) shows that the MRPE (with squared terms) is related to a weighted MSE criterion when the error terms are scaled by $1/|X^i|$. Therefore, minimizing MRPE may result in a predictor that is tailored for low profile regimes. We use the mean squared error (MSE) as the training objective. Twenty percent of the data is separated for validation and the same amount is specified to the test set. The training procedure stops when the $\text{MRPE}_{6\sigma}$ on validation set increases for 10 consecutive updates. Moreover, the error on validation set is tracked to determine the best weights achieved during training. After termination, this set of weights define the predictor.

Selecting GNN Hyperparameters: We use a linear activation function in the output layers of f_{w_i}'s and g_w, and introduce nonlinearities by using *tanh* function in their hidden layers. In our search grid, the number of hidden layers for both networks is varied from 0 to 3 with stride 1 and the number of neurons in each layer is chosen between 5 and 25 with a stride of 5. The number of states, s, and encoding functions, n_f, are both spanned from 1 to 5.

Effects of the two types of hyperparameters (HPs), namely GNN parameters and modeling parameters on flow prediction are studied in the following two sections. In all experiments in these sections we only use flow data to predict flow. Other possibilities which incorporates occupancy and speed are considered thereafter.



(a) Prediction based on the 2 most recent observations, $n_t = 2$.

(b) Prediction based on the 10 most recent observations, $n_t = 10$.

Figure 3.11: 30–minute flow prediction $\text{MRPE}_{6\sigma}$ for the 20 best sets of GNN hyperparameters. The hyperparameters are coded on top of each point in (3.5) format. The standard deviation over 5 tests is shown below each point.

Effects of GNN HPs: The first group of results corresponds to 30–minute flow prediction (i.e. 2 steps ahead) for different GNN hyperparameters and fixed modeling parameters. The graph depth, r, in this case was set to 2 and the number of past (flow) observations, n_t, was chosen equal to 2 and 10. Edge labels were considered according to the paradigm mentioned above. The best 20 sets of GNN hyperparameters – ordered by a grid search – when the last two observations are considered in the input, $n_t = 2$, are shown in Fig. 3.11a. Similar results for $n_t = 10$, i.e. considering the observations from the past 2.5 hours, are shown in Fig. 3.11b. These hyperparameters achieve the minimum average of $\text{MRPE}_{6\sigma}$ over the stations listed in Table. 3.2. Each point on the figures corresponds to the set of hyperparameters that is coded on top of it. The vertical axis shows the $\text{MRPE}_{6\sigma}$ associated with those parameters and the horizontal axis shows its ranking among all sets of parameters evaluated in the grid search. The number under each point shows the standard deviation of MRPE calculated on 5 tests. The hyperparameters on top of the points are coded in the following format:

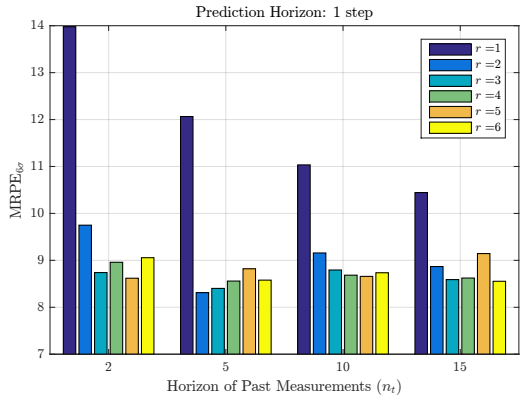
$$
\begin{array}{ll}
H_g & \text{number of neurons in the hidden layers of } g_w \\
| & \\
L_g & \text{number of layers (hidden + output) in each } g_w \\
| & \\
H_f & \text{number of neurons in each transition network } f_w \\
| & \\
L_f & \text{number of layers (hidden + output) in each } f_w \\
| & \\
n_s & \text{number of states outputed by each } f_w \\
| & \\
n_f & \text{number of transition networks } f_w\text{'s}
\end{array}
\tag{3.5}
$$

For instance, $2-3-2-5-2-15$ represents a GNN with two transition blocks that each outputs 3 states. Each transition layer has 1 hidden layer with 5 neurons. The output network of this GNN is a neural network with 1 hidden layer and 15 neurons.

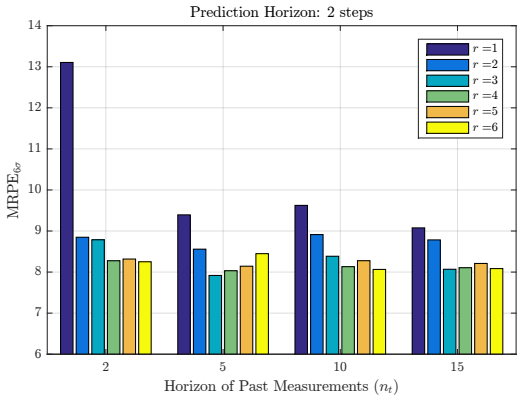
Note that the GNN model becomes a simple feedforward neural network when the transition blocks are removed ($n_f = 0$). Although, we have considered $n_f = 0$ in the grid, no hyperparameter set with $n_f = 0$ was among the best sets that are shown in figures 3.11a and 3.11b. This implies that all the GNN models listed in the figure outperform all FNN models in the grid. It can be drawn from the figures that among the first 5 sets in the two plots, 7 models out of 10 have 2 transition networks. Moreover, the first "low–variance" model in both cases is a GNN with 2 transition blocks. We use GNN model denoted by $2-3-2-5-2-15$ for the next set of results since it is a high accuracy–low variance predictor in both cases.

Effects of Modeling HPs: The second set of results illustrate the performance of a given GNN model for different flow prediction horizons and modeling hyperparameters. More explicitly, we vary the prediction horizon, number of past (flow) observations available to the predictor (n_t) and the radius of receptive field (r) when the GNN model is $2-3-2-5-2-15$. Figures 3.12a and 3.12b illustrate the $\text{MRPE}_{6\sigma}$ in 15-min and 30-min prediction for 6 different receptive field radii and 4 past observation horizons. The figures show that the flow in 1 and 2 steps ahead has the highest temporal correlation with the past 5 observations. Less number of past observations is less informative. On the other hand, more number of past observations increases the number of model weights without providing a rich enough input which can lead to finding a worse local optimum. The figures also show the importance of considering spatial interrelations in the prediction process. Prediction solely based on the target sensor results in an inferior performance compared to the all cases that information from neighbors is also taken into account ($r > 1$).

The same set of parameters are evaluated in 45, 60, 75 and 90 minutes predictions and the results are shown in Fig. 3.13a, 3.13b, 3.14a and 3.14b respectively. The effectiveness of considering multiple spatially correlated sensors in the prediction performance is more clear in these figures. Comparing them demonstrates that neighboring sensors play a crucial role when the prediction horizon is long, and indeed, the states of far sensors are more informative than the observations obtained at the target.

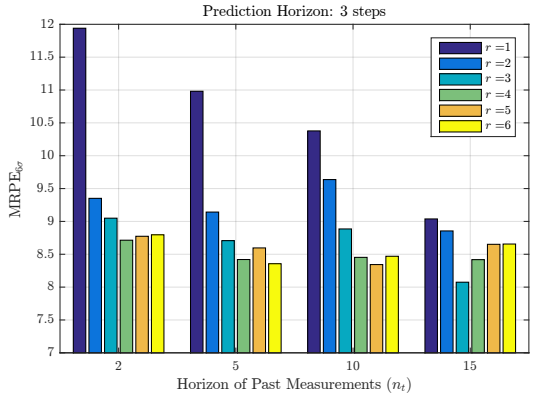


(a) GNN performance in 15–minute prediction.

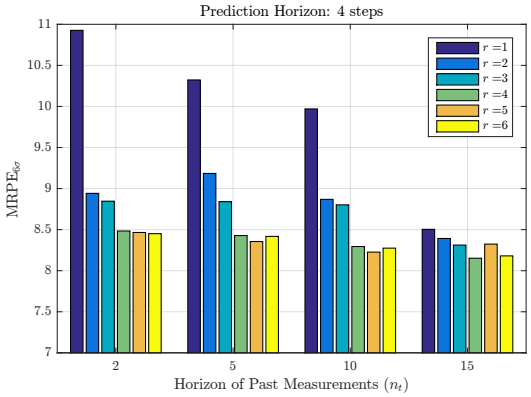


(b) GNN performance in 30–minute prediction.

Figure 3.12: Each group of bars corresponds to one value of n_t which is the number of past observations considered in the input of GNN. Each bar color correspons to one spacial extent radius (r) that is indicated in the legend.

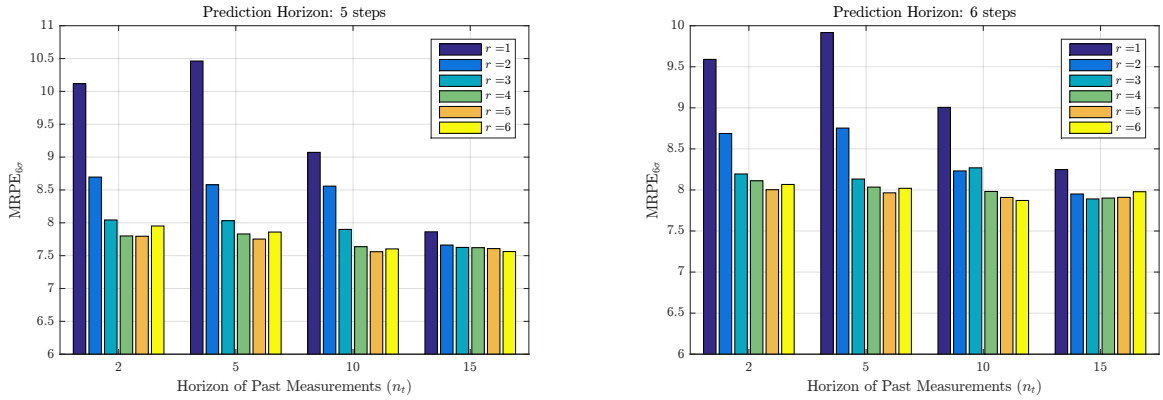


(a) GNN performance in 45–minute prediction.



(b) GNN performance in 60–minute prediction.

Figure 3.13: Each group of bars corresponds to one value of n_t which is the number of past observations considered in the input of GNN. Each bar color correspons to one spacial extent radius (r) that is indicated in the legend.



(a) GNN performance in 75–minute prediction. (b) GNN performance in 90–minute prediction.

Figure 3.14: Each group of bars corresponds to one value of n_t which is the number of past observations considered in the input of GNN. Each bar color correspons to one spacial extent radius (r) that is indicated in the legend.

The optimal temporal and spatial extents for these 6 prediction horizons are listed in Table 3.3. The results show that the optimal spatial extent increases by the prediction horizon whereas the temporal extent does not follow this trend. This is probably because the states at multiple steps ahead are weakly correlated with the states at multitude steps before. As a result, increasing the number of model weights can lead to finding a poor local optimum when some elements of the input data are not informative. The error contribution from each location for this optimal setting is depicted in Fig. 3.15

Table 3.3: Optimal length of past observations horizon and radius of receptive field in flow prediction.

Prediction Horizon	Past Measurements Horizon	Radius of Receptive Field	MRPE$_{6\sigma}$
15	5	2	8.31%
30	5	3	7.92%
45	15	3	8.07%
60	15	4	8.15%
75	10	5	7.56%
90Ãň	10	6	7.87%

The time–series of actual and predicted flow for one of the high–variance VDSs are shown in Fig. 3.16 and 3.17. As can be seen from the figures, the GNN is able to keep the forecasting error small even when the flow profile deviates largely from the seasonal mean and the prediction horizon is large. The histogram of of the absolute error is depicted in Fig. 3.18.

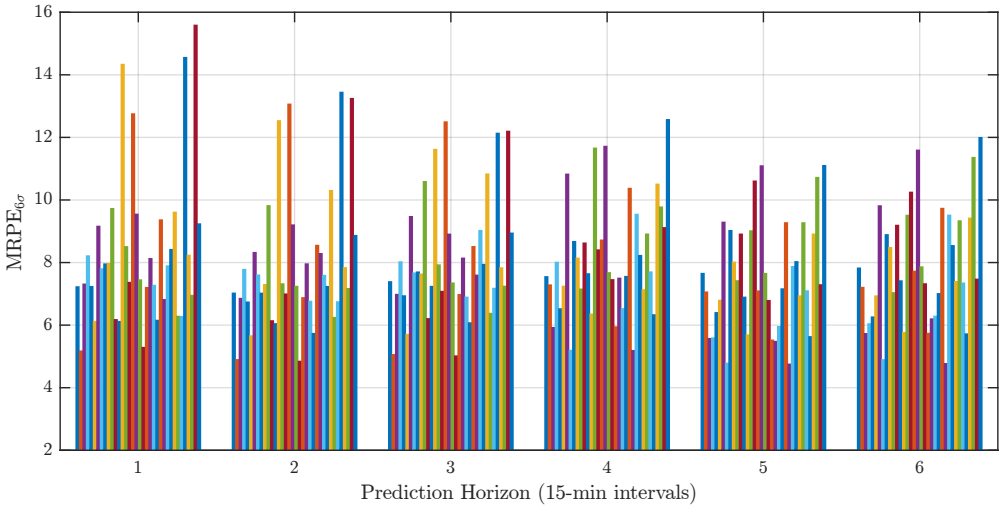


Figure 3.15: MRPE$_{6\sigma}$ for all 36 stations when optimal settings listed in Table 3.3 are applied. (The input data only contains flow.)

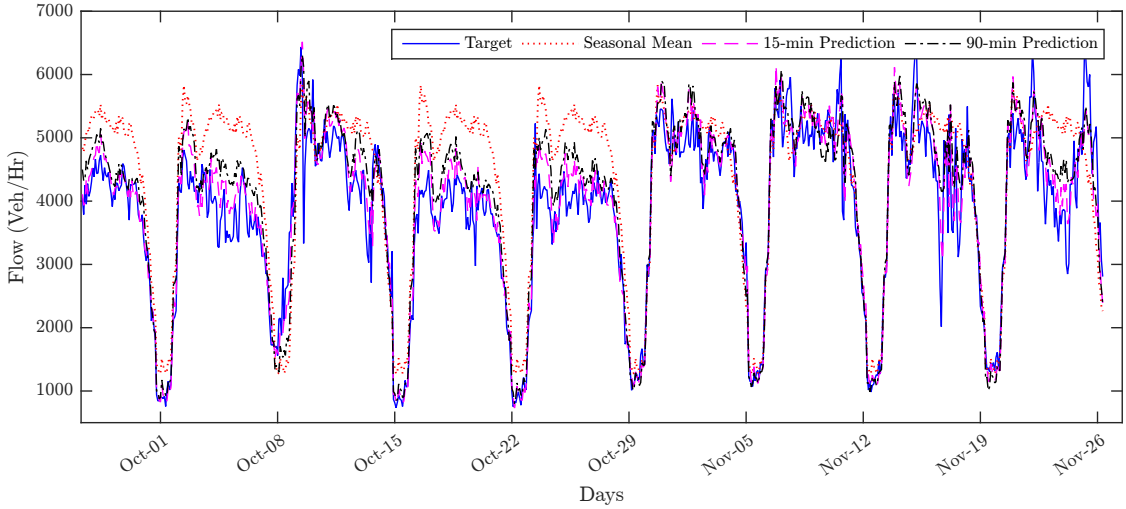


Figure 3.16: Actual and predicted traffic flow at VDS 717742 in all Wednesdays of the test set. The end of each day is indicated on the horizontal axes. The input data only contains traffic flow.

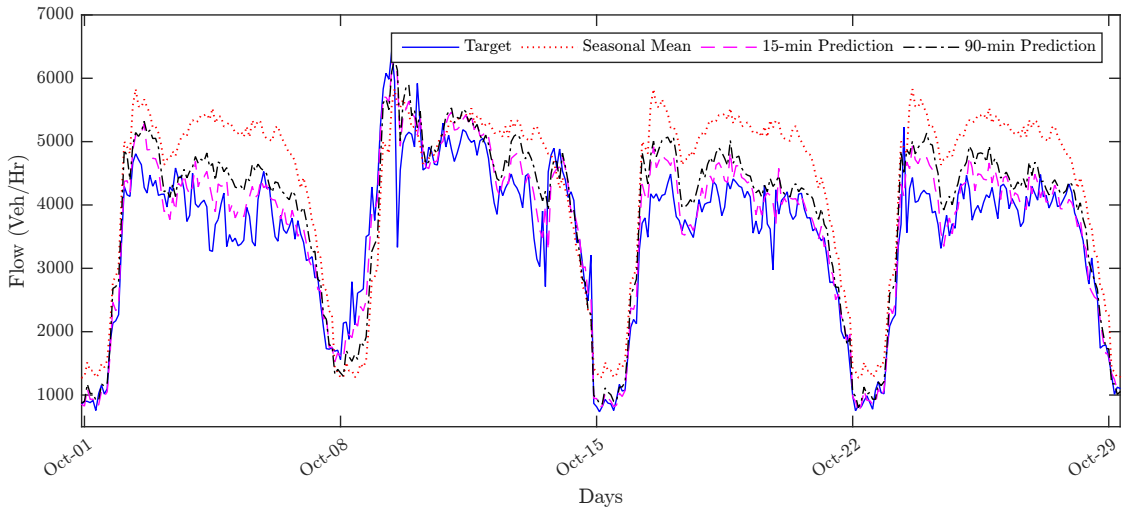


Figure 3.17: Actual and predicted traffic flow at VDS 717742 in four Wednesdays of the test set. The end of each day is indicated on the horizontal axes. The input data only contains traffic flow.

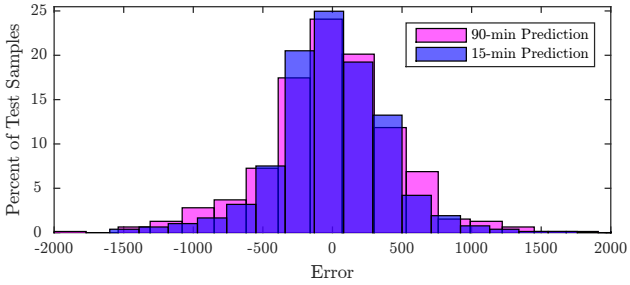


Figure 3.18: Flow prediction error (Veh/Hr) histogram for 15 and 90 minutes ahead prediction at VDS 717742. The input data only contains traffic flow.

Input Data: Flow was the only input quantity in the previous case studies. Occupancy and speed are two other measurements that are obtained by VDSs. Accordingly, there are three other possibilities of input data that can be used for prediction, namely "flow and occupancy", "flow and speed", and "flow and occupancy and speed". The same type of analysis as the last two sections is performed on these three alternatives. The optimal results for "low", "mid" and "high" variance sets (c.f. Table 3.2) and different forecasting horizons are illustrated in Fig. 3.19. The figure shows that incorporating both flow and occupancy in the input results in a slightly better performance than solely relying on flow. The average over of $MRPE_{6\sigma}$ over all stations is shown in Fig. 3.20.

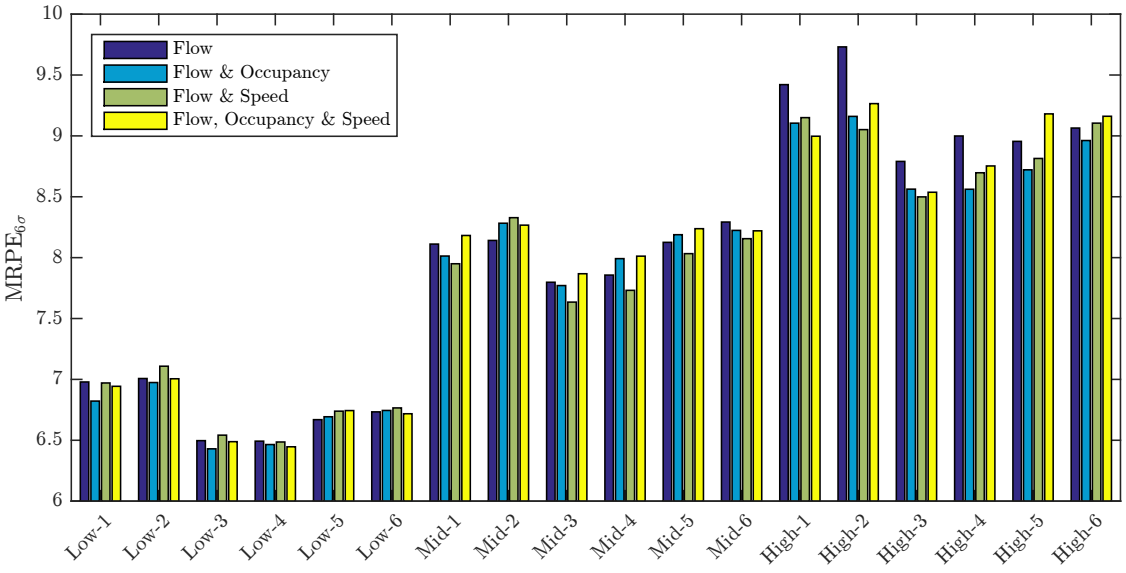


Figure 3.19: Average MRPE$_{6\sigma}$ for "Low", "Mid" and "High" variance station sets and different prediction horizons (indicated next to the variance type).

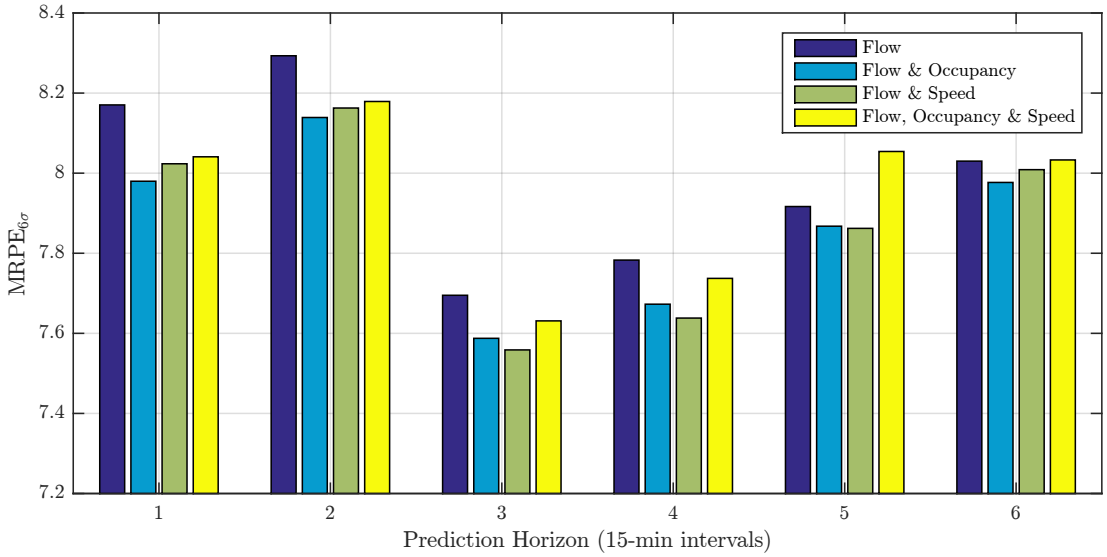


Figure 3.20: Average MRPE$_{6\sigma}$ over all stations for different input data types.

Comparison with Other Methods: The performance of GNN model $2-3-2-5-2-15$ with the optimal settings given in Table 3.3 and the input data consisted of flow and occupancy is compared to the other methods that were listed in the beginning of this section. Figures 3.21, 3.22 and 3.23 respectively illustrate the average MRPE$_{6\sigma}$ at the locations that have profiles with "low", "mid" and "high" deviation from the seasonal mean.

The first figure shows that the GNN error is larger than most of other methods in 15, 30 and 45 minutes ahead prediction at locations that the flow profile follows the seasonal mean closely. The GNN performance stays roughly constant at higher forecasting horizons while most of the other methods, especially the time–series based models, degrade. Although the GNN error is slightly smaller than other methods, it may not be a suitable model for this type of locations since this model is significantly more complex than other methods such as SARIMA which achieves approximately the same performance. Indeed, based on the figure, even the performance of naive SM predictor may be satisfactory in this case.

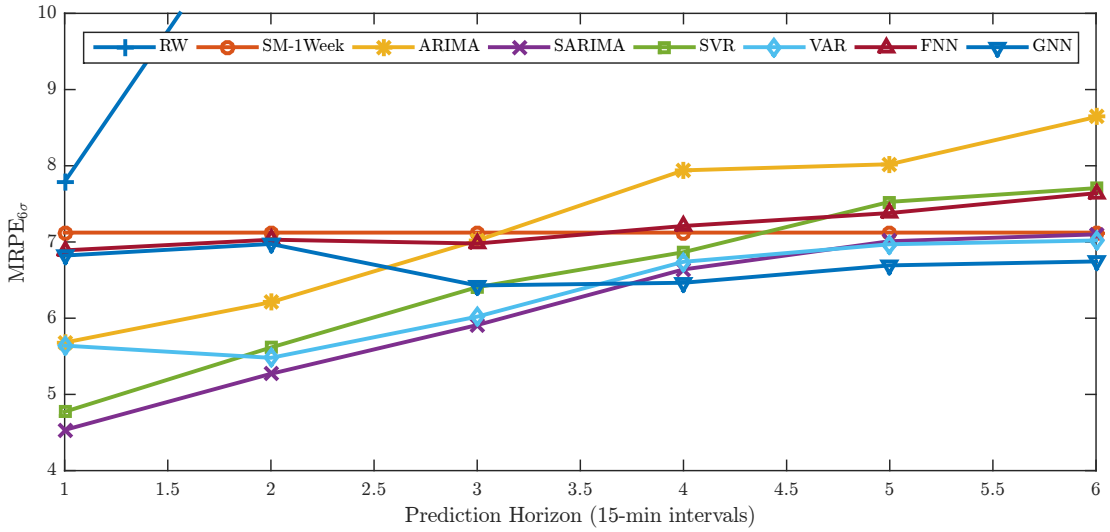


Figure 3.21: Comparison of prediction performance for the locations that have "low" deviation from seasonal mean. (For clearness, the error of RW method is not shown completely since it quickly grows to 28%.)

As shown in Fig. 3.22, at locations with mid deviation from seasonal mean, the ARIMA–like models (SARIMA, VAR and ARIMA) outperform the GNN in 15 and 30-min forecasting. However, the error of these methods grow with the prediction horizon while the GNN performance is more or less the same at longer forecasting. Accordingly, starting from 45-minute prediction the GNN outperforms other models and in 90-minute prediction its performance is significantly higher than other methods. This behavior is more clear among locations that have profiles with high deviation from seasonal mean. Figure 3.23 shows that the GNN model outperforms all other models for 30-minute prediction and larger horizons. It is interesting that in 15-minute forecasting the RW method is the second best predictor in this case. This shows that the high frequency flow variation is large and weakly correlated with the past observations. As a result, the improvement gained by time–series methods such as ARIMA and VAR is insignificant, and the methods that are based on the seasonality of data fail. The exact value of error for each of these methods can be found in Table 3.4.

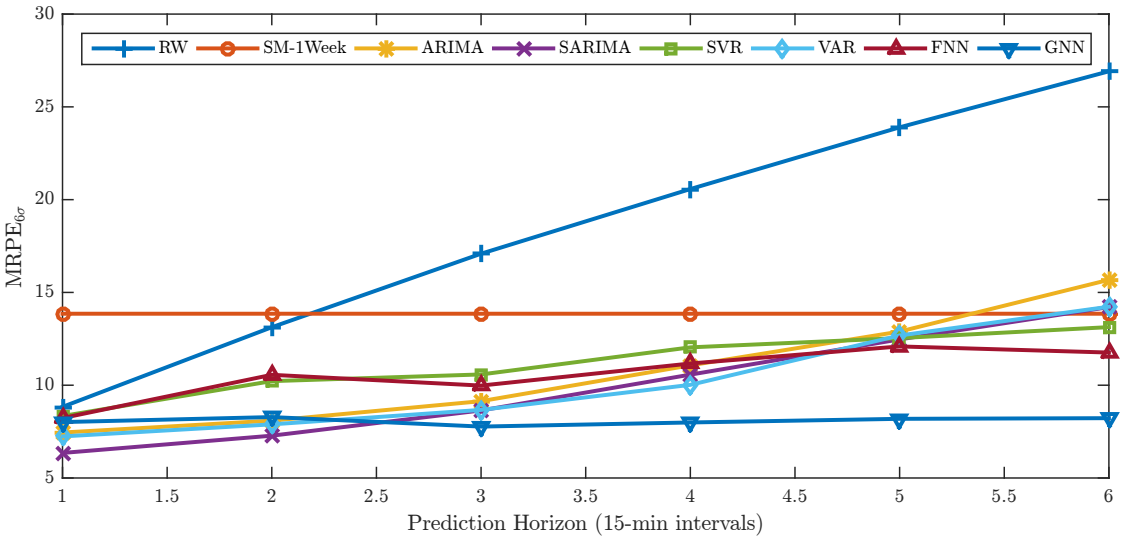


Figure 3.22: Comparison of prediction performance in the locations that have "mid" deviation from seasonal mean.

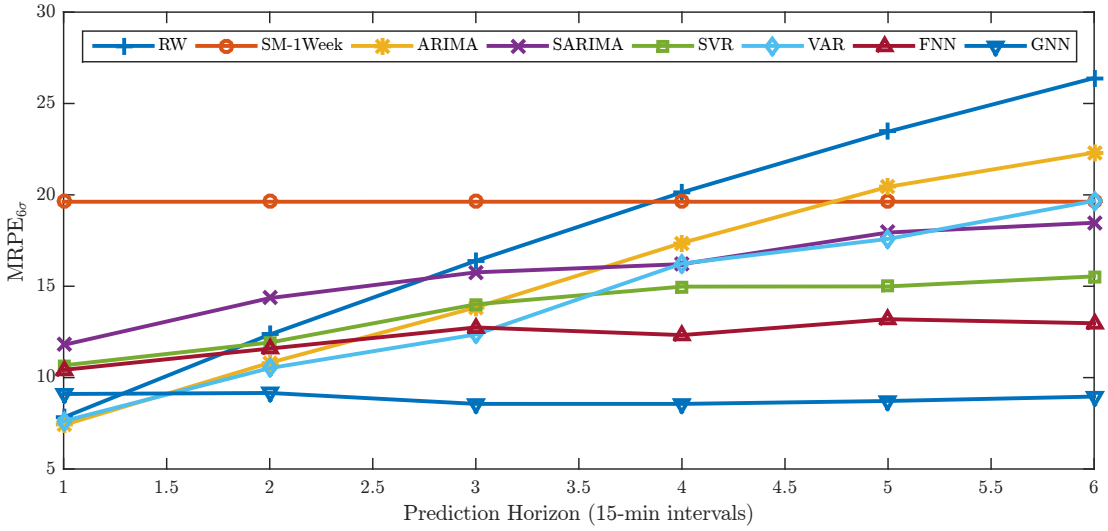


Figure 3.23: Comparison of prediction performance in the locations that have "high" deviation from seasonal mean.

In summary, the simple univariate models ARIMA or SARIMA can outperform the GNN model in 1 or at most 2-step ahead prediction problems. However, the difference is slight. On the other hand, when it comes to a longer forecasting horizons, the GNN performance is larger than other methods and the improvement becomes very significant at long horizons. This is because the simple time–series models are more capable of learning local and short

term temporal relations, while the GNN is more efficient in extracting long term dependencies and learning the network dynamics.

Table 3.4: Average $MRPE_{6\sigma}$ for different prediction methods, deviations from seasonal mean and prediction horizons.

Deviation from SM	Method	Delay 1	2	3	4	5	6
High	RW	7.82	12.36	16.39	20.14	23.46	26.39
	SM-1Week	19.62	19.62	19.62	19.62	19.62	19.62
	ARIMA	**7.41**	10.81	13.84	17.37	20.44	22.31
	SARIMA	11.79	14.35	15.76	16.21	17.94	18.47
	SVR	10.67	11.92	14.01	14.98	14.99	15.54
	VAR	7.62	10.52	12.36	16.24	17.59	19.67
	FNN	10.42	11.59	12.75	12.33	13.20	12.97
	GNN	9.10	**9.16**	**8.56**	**8.56**	**8.72**	**8.96**
Mid	RW	8.83	13.12	17.09	20.57	23.90	26.92
	SM-1Week	13.85	13.85	13.85	13.85	13.85	13.85
	ARIMA	7.45	8.09	9.14	11.07	12.89	15.68
	SARIMA	**6.35**	7.28	8.64	10.57	12.47	14.21
	SVR	8.34	10.22	10.58	12.04	12.54	13.13
	VAR	7.24	**7.89**	8.67	10.02	12.68	14.23
	FNN	8.23	10.56	9.98	11.17	12.09	11.76
	GNN	8.01	8.28	**7.77**	**7.99**	**8.19**	**8.22**
Low	RW	7.79	11.74	15.41	18.73	21.65	24.36
	SM-1Week	7.12	7.12	7.12	7.12	7.12	7.12
	ARIMA	5.68	6.21	7.02	7.94	8.02	8.64
	SARIMA	**4.53**	**5.27**	**5.91**	6.64	7.01	7.10
	SVR	4.77	5.62	6.41	6.87	7.53	7.71
	VAR	5.64	5.48	6.02	6.74	6.97	7.02
	FNN	6.89	7.03	6.98	7.21	7.38	7.64
	GNN	6.82	6.97	6.43	**6.46**	**6.69**	**6.74**

Chapter 4

Conclusion

The availability of abundant traffic data and computation power emerged in recent years motivated us to revisit the problem of short–term traffic forecasting by a computational intelligence approach and propose a *data–driven* traffic model and prediction algorithm.

The data–driven methods in the literature do not explicitly consider the transportation network dynamics; rather, they rely on a learning process in order to extract the underlying spatial relationships between the unstructured input data and the future traffic conditions. These methods have a few drawbacks: (1) they attempt to retrieve relational and hierarchical dependencies that was lost during converting structured data to an unstructured form that can be processed by them. Indeed, this information that corresponds to the spatial interrelations between different sources of data is known and can be explicitly presented by a structured data. (2) the input vector dimension increases by taking into account the observations from more locations (3) they cannot manipulate dynamic information appropriately.

Our objective was to overcome these shortcomings by developing an empirical data–driven traffic model that can analyze a graph structured data that not only presents temporal observations, but also preserves full information about the spatial interrelations between the observations collected from different locations. The key idea was that in a traffic network, a graph can describe precisely the network dynamics by modeling it through a set of nodes that correspond to road segments equipped with sensors (e.g. inductive–loop detectors) and edges that correspond to the roads between the nodes. Both entities possess labels which in our application correspond to the measurements obtained by the sensors (e.g. flow, occupancy and speed) and road characteristics (e.g. direction and length).

We showed that the Graph Neural Network (GNN) models which is can process such graph–oriented models of traffic network for traffic state forecasting. Comprehensive case studies on real–world data from PeMS database and comparison with state of the art methods illustrated that: (1) in very short prediction horizons (e.g. 15 or at most 30 minutes) and regular traffic regimes, the accuracy of proposed model is negligibly lower than the state of art, (2) in longer prediction horizons (above 30 minutes) and complex regimes the GNN performance is significantly higher than other methods. This is because the GNN takes advantage of data from multiple sources and also information about their spatial dependencies.

The empirical study in this work was focused on flow prediction. We investigated the effect of incorporating occupancy and speed in the input data. As a future work, one can investigate whether predicting multiple quantities (e.g. flow and occupancy) at the same time, using a shared representation benefits from *multi-task learning*. Another future work could be investigating deep architectures of GNNs that are consisted of multiple layers of transition and output layers. Different types of pretraining methods used in deep–learning can be applied to these deep models.

Bibliography

Abdulhai, Baher, Porwal, Himanshu, & Recker, Will. 2002. Short-term traffic flow prediction using neuro-genetic algorithms. *ITS Journal-Intelligent Transportation Systems Journal*, **7**(1), 3–41.

Adeli, Hojjat. 2001. Neural networks in civil engineering: 1989–2000. *Computer-Aided Civil and Infrastructure Engineering*, **16**(2), 126–142.

Ahmed, Mohammed S, & Cook, Allen R. 1979. *Analysis of freeway traffic time-series data by using Box-Jenkins techniques.*

Alecsandru, Ciprian, & Ishak, Sherif. 2004. Hybrid model-based and memory-based traffic prediction system. *Transportation Research Record: Journal of the Transportation Research Board*, 59–70.

Almeida, Luis B. 1990. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. *Pages 102–111 of: Artificial neural networks.* IEEE Press.

Bandinelli, Niccolo, Bianchini, Monica, & Scarselli, Franco. 2010. Learning long-term dependencies using layered graph neural networks. *Pages 1–8 of: Neural Networks (IJCNN), The 2010 International Joint Conference on.* IEEE.

Barcz, Aleksy, & Jankowski, Stanisław. 2014. Graph Neural Networks for 3D Bravais Lattices Classification. *Pages 76–86 of: Neural Networks and Artificial Intelligence.* Springer.

Baskararaja, GnanaJothi Raja, & Manickavasagam, MeenaRani Sundaramoorthy. 2012. Subgraph Matching Using Graph Neural Network. *Journal of Intelligent Learning Systems and Applications*, **4**(04), 274.

Belahcen, Anas, Bianchini, Monica, & Scarselli, Franco. 2015. Web Spam Detection Using Transductive (Inductive Graph Neural Networks. *Pages 83–91 of: Advances in Neural Networks: Computational and Theoretical Issues.* Springer.

Bellman, Richard, Bellman, Richard Ernest, Bellman, Richard Ernest, & Bellman, Richard Ernest. 1961. *Adaptive control processes: a guided tour.* Vol. 4. Princeton university press Princeton.

Bengio, Yoshua, Goodfellow, Ian J., & Courville, Aaron. 2015. *Deep Learning.* Book in preparation for MIT Press.

Bianchini, Monica, & Maggini, Marco. 2013. Supervised Neural Network Models for Processing Graphs. *Pages 67–96 of: Handbook on Neural Information Processing.* Springer.

Bianchini, Monica, Mazzoni, Paolo, Sarti, Lorenzo, & Scarselli, Franco. 2003. Face spotting in color images using recursive neural networks. *Pages 76–81 of: IAPR–TC3 International*

Workshop on Artificial Neural Networks in Pattern Recognition, Florence (Italy). Citeseer.

Bianchini, Monica, Maggini, Marco, Sarti, Lorenzo, & Scarselli, Franco. 2005. Recursive neural networks for processing graphs with labelled edges: Theory and applications. *Neural Networks*, **18**(8), 1040–1050.

Box, George EP, & Jenkins, Gwilym M. 1976. Time series analysis, control, and forecasting. *San Francisco, CA: Holden Day*, **3226**(3228), 10.

Brin, Sergey, & Page, Lawrence. 2012. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, **56**(18), 3825–3833.

Caltrans. 2014. An Introduction to the California Department of Transportation Performance Measurement System (PeMS).

Castro-Neto, Manoel, Jeong, Young-Seon, Jeong, Myong-Kee, & Han, Lee D. 2009. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, **36**(3), 6164–6173.

Chandra, Srinivasa, & Al-Deek, Haitham. 2008. Cross-correlation analysis and multivariate prediction of spatial time series of freeway traffic speeds. *Transportation Research Record: Journal of the Transportation Research Board*, 64–76.

Chandra, Srinivasa Ravi, & Al-Deek, Haitham. 2009. Predictions of freeway traffic speeds and volumes using vector autoregressive models. *Journal of Intelligent Transportation Systems*, **13**(2), 53–72.

Chang, Gang-Len, & Su, Chih-Chiang. 1995. Predicting intersection queue with neural network models. *Transportation Research Part C: Emerging Technologies*, **3**(3), 175–191.

Chang, Hyunho, Park, Dongjoo, Lee, Seungjae, Lee, Hosang, & Baek, Seungkirl. 2010. Dynamic multi-interval bus travel time prediction using bus transit data. *Transportmetrica*, **6**(1), 19–38.

Chau, Rowena, Tsoi, Ah Chung, Hagenbuchner, Markus, & Lee, Vincent. 2009. A conceptlink graph for text structure mining. *Pages 141–150 of: Proceedings of the Thirty-Second Australasian Conference on Computer Science-Volume 91*. Australian Computer Society, Inc.

Chen, Chao. 2003. Freeway performance measurement system (PeMS). *California Partners for Advanced Transit and Highways (PATH)*.

Chen, Chao, Petty, Karl, Skabardonis, Alexander, Varaiya, Pravin, & Jia, Zhanfeng. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record: Journal of the Transportation Research Board*, 96–102.

Chen, Chao, Kwon, Jaimyoung, Rice, John, Skabardonis, Alexander, & Varaiya, Pravin. 2003. Detecting errors and imputing missing data for single-loop surveillance systems. *Transportation Research Record: Journal of the Transportation Research Board*, 160–167.

Chen, Chao, Varaiya, Pravin, & Kwon, Jaimyoung. 2005. An empirical assessment of traffic operations. *Pages 105–124 of: Proceedings of the 16th International Symposium on Transportation and Traffic Theory*.

Chen, Chenyi, Wang, Yin, Li, Li, Hu, Jianming, & Zhang, Zuo. 2012. The retrieval of intra-day trend and its influence on traffic prediction. *Transportation research part C: emerging technologies*, **22**, 103–118.

Cheng, Tao, Haworth, James, & Wang, Jiaqiu. 2012. Spatio-temporal autocorrelation of road network data. *Journal of Geographical Systems*, **14**(4), 389–413.

Choe, Tom, Skabardonis, Alexander, & Varaiya, Pravin. 2002. Freeway performance measurement system: operational analysis tool. *Transportation Research Record: Journal of the Transportation Research Board*, 67–75.

Chua, Leon O, & Yang, Lin. 1988. Cellular neural networks: Applications. *Circuits and Systems, IEEE Transactions on*, **35**(10), 1273–1290.

Clark, Stepen D, Dougherty, Mark S, & Kirby, Howard R. 1993. The use of neural networks and time series models for short term traffic forecasting: a comparative study. *In: Transaction Planning Methods. Proceedings of Seminar D Held at the PTRC European Transport, Highways, and Planning 21st Summer Annual Meeting (September 13-17, 1993).*

Collins, Michael, & Duffy, Nigel. 2001. Convolution kernels for natural language. *Pages 625–632 of: Advances in neural information processing systems.*

Daganzo, Carlos 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.

Davis, Gary A, & Nihan, Nancy L. 1991. Nonparametric Regression and Short-Term Freeway Traffic Forecasting. *Journal of Transportation Engineering.*

Di Noi, Lucia, Hagenbuchner, Markus, Scarselli, Franco, & Tsoi, Ah Chung. 2010. Web spam detection by probability mapping graphsoms and graph neural networks. *Pages 372–381 of: Artificial Neural Networks–ICANN 2010.* Springer.

Dia, Hussein. 2001. An object-oriented neural network approach to short-term traffic forecasting. *European Journal of Operational Research*, **131**(2), 253–261.

Ding, Ailine, Zhao, Xangmo, & Jiao, LiCheng. 2002. Traffic flow time series prediction based on statistics learning theory. *Pages 727–730 of: Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on.* IEEE.

Dougherty, Mark S, & Cobbett, Mark R. 1997. Short-term inter-urban traffic forecasts using neural networks. *International journal of forecasting*, **13**(1), 21–31.

Dougherty, MS, Kirby, HR, & Boyle, RD. 1994. Using neural networks to recognise, predict and model traffic. *Artificial intelligence applications to traffic engineering*, 235–250.

Du, Lili, Peeta, Srinivas, & Kim, Yong Hoon. 2012. An adaptive information fusion model to predict the short-term link travel time distribution in dynamic traffic networks. *Transportation Research Part B: Methodological*, **46**(1), 235–252.

Dunne, Stephen, & Ghosh, Bidisha. 2011. Regime-based short-term multivariate traffic condition forecasting algorithm. *Journal of Transportation Engineering*, **138**(4), 455–466.

Elman, Jeffrey L. 1990. Finding structure in time. *Cognitive science*, **14**(2), 179–211.

Francesconi, Enrico, Frasconi, Paolo, Gori, Marco, Marinai, Simone, Sheng, JQ, Soda, Giovanni, & Sperduti, Alessandro. 1998. Logo recognition by recursive neural networks. *Pages 104–117 of: Graphics Recognition Algorithms and Systems.* Springer.

Frasconi, Paolo, Gori, Marco, & Sperduti, Alessandro. 1998. A general framework for adaptive processing of data structures. *Neural Networks, IEEE Transactions on*, **9**(5), 768–786.

Gärtner, Thomas, Lloyd, John W, & Flach, Peter A. 2004. Kernels and distances for struc-

tured data. *Machine Learning*, **57**(3), 205–232.

Ghosh, Bidisha, Basu, Biswajit, & O'Mahony, Margaret. 2009. Multivariate short-term traffic flow forecasting using time-series analysis. *Intelligent Transportation Systems, IEEE Transactions on*, **10**(2), 246–254.

Gilmore, John F, & Abe, Naohiko. 1995. Neural network models for traffic control and congestion prediction. *Journal of Intelligent Transportation Systems*, **2**(3), 231–252.

Guo, Jianhua, & Williams, Billy. 2010. Real-time short-term traffic speed level forecasting and uncertainty quantification using layered Kalman filters. *Transportation Research Record: Journal of the Transportation Research Board*, 28–37.

Hagenbuchner, Markus, Sperduti, Alessandro, & Tsoi, Ah Chung. 2003. A self-organizing map for adaptive processing of structured data. *Neural Networks, IEEE Transactions on*, **14**(3), 491–505.

Hall, Fred L, Hurdle, VF, & Banks, James H. 1992. Synthesis of recent work on the nature of speed-flow and flow-occupancy (or density) relationships on freeways. *Transportation Research Record*.

Hammer, Barbara, & Jain, Brijnesh J. 2004. Neural methods for non-standard data. *Pages 281–292 of: ESANN*. Citeseer.

Haworth, James, & Cheng, Tao. 2012. Non-parametric regression for space–time forecasting under missing data. *Computers, Environment and Urban Systems*, **36**(6), 538–550.

Haykin, Simon, & Network, Neural. 2004. A comprehensive foundation. *Neural Networks*, **2**(2004).

Head, K Larry. 1995. Event-based short-term traffic flow prediction model. *Transportation Research Record*, 45–52.

Herring, Ryan, Hofleitner, Aude, Abbeel, Pieter, & Bayen, Alexandre. 2010a. Estimating arterial traffic conditions using sparse probe data. *Pages 929–936 of: Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE.

Herring, Ryan, Hofleitner, Aude, Amin, Saurabh, Nasr, T, Khalek, A, Abbeel, Pieter, & Bayen, Alexandre. 2010b. Using mobile phones to forecast arterial traffic through statistical learning. *In: 89th Transportation Research Board Annual Meeting, Washington DC*. Citeseer.

Hofleitner, Aude, Herring, Ryan, Abbeel, Pieter, & Bayen, Alexandre. 2012. Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network. *Intelligent Transportation Systems, IEEE Transactions on*, **13**(4), 1679–1693.

Hopfield, John J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, **79**(8), 2554–2558.

Huang, Wenhao, Hong, Haikun, Li, Man, Hu, Weisong, Song, Guojie, & Xie, Kunqing. 2013. Deep architecture for traffic flow prediction. *Pages 165–176 of: Advanced Data Mining and Applications*. Springer.

Huang, Wenhao, Song, Guojie, Hong, Haikun, & Xie, Kunqing. 2014. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *Intelligent Transportation Systems, IEEE Transactions on*, **15**(5), 2191–2201.

Hunter, Timothy, Herring, Ryan, Abbeel, Pieter, & Bayen, Alexandre. 2009. Path and travel time inference from GPS probe vehicle data. *NIPS Analyzing Networks and Learning with Graphs*, **12**, 1.

Innamaa, Satu. 2000. Short-term prediction of traffic situation using MLP-neural networks. *Pages 6–9 of: Proceedings of the 7th world congress on intelligent transport systems, Turin, Italy.*

Ishak, Sherif, & Alecsandru, Ciprian. 2004. Optimizing traffic prediction performance of neural networks under various topological, input, and traffic condition settings. *Journal of Transportation Engineering*, **130**(4), 452–465.

Jeong, Young-Seon, Byon, Young-Ji, Mendonca Castro-Neto, Manoel, & Easa, Said M. 2013. Supervised weighting-online learning algorithm for short-term traffic flow prediction. *Intelligent Transportation Systems, IEEE Transactions on*, **14**(4), 1700–1707.

Jiang, Xiaomo, & Adeli, Hojjat. 2005. Dynamic wavelet neural network model for traffic flow forecasting. *Journal of transportation engineering*, **131**(10), 771–779.

Jothi, RB Gnana, & Rani, SM Meena. 2013. Edge Based Graph Neural Network to Recognize Semigraph Representation of English Alphabets. *Pages 402–412 of: Mining Intelligence and Knowledge Exploration*. Springer.

Kamarianakis, Yiannis, & Prastacos, Poulicos. 2003. Forecasting traffic flow conditions in an urban network: comparison of multivariate and univariate approaches. *Transportation Research Record: Journal of the Transportation Research Board*, 74–84.

Kamarianakis, Yiannis, & Prastacos, Poulicos. 2005. Space–time modeling of traffic flow. *Computers & Geosciences*, **31**(2), 119–133.

Kamarianakis, Yiannis, Shen, Wei, & Wynter, Laura. 2012. Real-time road traffic forecasting using regime-switching space-time models and adaptive LASSO. *Applied Stochastic Models in Business and Industry*, **28**(4), 297–315.

Karlaftis, MG, & Vlahogianni, EI. 2011. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, **19**(3), 387–399.

Kashima, Hisashi, Tsuda, Koji, & Inokuchi, Akihiro. 2003. Marginalized kernels between labeled graphs. *Pages 321–328 of: ICML*, vol. 3.

Kerner, Boris S. 2004. Three-phase traffic theory and highway capacity. *Physica A: Statistical Mechanics and its Applications*, **333**, 379–440.

Khamsi, Mohamed A, & Kirk, William A. 2011. *An introduction to metric spaces and fixed point theory*. Vol. 53. John Wiley & Sons.

Kim, Taehyung, Kim, Hyoungsoo, & Lovell, David J. 2005. Traffic flow forecasting: overcoming memoryless property in nearest neighbor non-parametric regression. *Pages 965–969 of: Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*. IEEE.

Kleinberg, Jon M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, **46**(5), 604–632.

Kohonen, Teuvo. 1998. The self-organizing map. *Neurocomputing*, **21**(1), 1–6.

Kondor, Risi Imre, & Lafferty, John. 2002. Diffusion kernels on graphs and other discrete structures. *Pages 315–322 of: Proceedings of the 19th international conference on machine*

learning.

Küchler, Andreas, & Goller, Christoph. 1996. Inductive learning in symbolic domains using structure-driven recurrent neural networks. *Pages 183–197 of: KI-96: Advances in Artificial Intelligence*. Springer.

Kwon, Eli, & Stephanedes, Yorgos J. 1994. Comparative evaluation of adaptive and neural-network exit demand prediction for freeway control. *Transportation Research Record*.

LeCun, Yann, & Bengio, Yoshua. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, **3361**(10).

Lee, Sangsoo, & Fambro, Daniel. 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*, 179–188.

Li, Li, Su, Xiaonan, Zhang, Yi, Lin, Yuetong, & Li, Zhiheng. Trend Modeling for Traffic Time Series Analysis: An Integrated Study.

Lieu, Henry C. 2000. *Traffic estimation and prediction system*. Tech. rept.

Lin, Wei-Hua. 2001. A Gaussian maximum likelihood formulation for short-term forecasting of traffic flow. *Pages 150–155 of: Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*. IEEE.

Lingras, Pawan, & Mountford, Paul. 2001. Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting. *Pages 290–299 of: Engineering of Intelligent Systems*. Springer.

Lippi, Marco, Bertini, Matteo, & Frasconi, Paolo. 2010. Collective traffic forecasting. *Pages 259–273 of: Machine Learning and Knowledge Discovery in Databases*. Springer.

Lippi, Marco, Bertini, Matteo, & Frasconi, Paolo. 2013. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *Intelligent Transportation Systems, IEEE Transactions on*, **14**(2), 871–882.

Lv, Yisheng, Duan, Yanjie, Kang, Wenwen, Li, Zhengxi, & Wang, Fei-Yue. 2015. Traffic flow prediction with big data: a deep learning approach. *Intelligent Transportation Systems, IEEE Transactions on*, **16**(2), 865–873.

Miller, W Thomas, Werbos, Paul J, & Sutton, Richard S. 1995. *Neural networks for control*. MIT press.

Moorthy, CK, & Ratcliffe, BG. 1988. Short term traffic forecasting using time series methods. *Transportation planning and technology*, **12**(1), 45–56.

Muratore, Donatella, Hagenbuchner, Markus, Scarselli, Franco, & Tsoi, Ah Chung. 2010. *Sentence extraction by graph neural networks*. Springer.

Newell, Gordon F. 1993. A simplified theory of kinematic waves in highway traffic, part I: General theory. *Transportation Research Part B: Methodological*, **27**(4), 281–287.

Oswald, R Keith, Scherer, William T, & Smith, Brian L. 2000. Traffic flow forecasting using approximate nearest neighbor nonparametric regression. *Final project of ITS Center project: Traffic forecasting: non-parametric regressions*.

Pan, TL, Sumalee, Agachai, Zhong, Ren-Xin, & Indra-Payoong, Nakorn. 2013. Short-term traffic state prediction based on temporal–spatial correlation. *Intelligent Transportation Systems, IEEE Transactions on*, **14**(3), 1242–1254.

Park, Byungkyu, Messer, Carroll, & Urbanik II, Thomas. 1998. Short-term freeway traffic volume forecasting using radial basis function neural network. *Transportation Research Record: Journal of the Transportation Research Board*, 39–47.

PeMS. *Caltrans Performance Measurement System (PeMS). Available on http://pems.dot.ca.gov/.*

Pfeifer, Phillip E, & Deutrch, Stuart Jay. 1980. A three-stage iterative procedure for space-time modeling phillip. *Technometrics*, **22**(1), 35–47.

Pfeifer, Phillip E, & Deutsch, Stuart Jay. 1980. Identification and interpretation of first order space-time ARMA models. *Technometrics*, **22**(3), 397–408.

Pineda, Fernando J. 1987. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, **59**(19), 2229.

Pollack, Jordan B. 1990. Recursive distributed representations. *Artificial Intelligence*, **46**(1), 77–105.

Pucci, A, Gori, M, Hagenbuchner, M, Scarselli, F, & Tsoi, AC. 2006. Investigation into the application of graph neural networks to large-scale recommender systems. *Systems Science*, **32**(4), 17–26.

Quek, Alyssa, Wang, Zhiyong, Zhang, Jian, & Feng, Dagan. 2011. Structural image classification with graph neural networks. *Pages 416–421 of: Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on.* IEEE.

Rice, John, & Van Zwet, Erik. 2004. A simple and effective method for predicting travel times on freeways. *Intelligent Transportation Systems, IEEE Transactions on*, **5**(3), 200–207.

Scarselli, Franco, Gori, Marco, Tsoi, Ah Chung, Hagenbuchner, Markus, & Monfardini, Gabriele. 2009a. Computational capabilities of graph neural networks. *Neural Networks, IEEE Transactions on*, **20**(1), 81–102.

Scarselli, Franco, Gori, Marco, Tsoi, Ah Chung, Hagenbuchner, Markus, & Monfardini, Gabriele. 2009b. The graph neural network model. *Neural Networks, IEEE Transactions on*, **20**(1), 61–80.

Scarselli, Franco, Tsoi, Ah Chung, Hagenbuchner, Markus, & Di Noi, Lucia. 2013. Solving graph data issues using a layered architecture approach with applications to web spam detection. *Neural Networks*, **48**, 78–90.

Schmitt, Thorsten, & Goller, Christoph. 1998. Relating chemical structure to activity: An application of the neural folding architecture. *Pages 170–177 of: Proc. Workshop Fuzzy-Neuro Syst./Conf. Eng. Appl. Neural Netw.*

Shekhar, Shashank. 2004. Recursive Methods for Forecasting Short-term Traffic Flow Using Seasonal ARIMA Time Series Model.

Shekhar, Shashank, & Williams, Billy. 2008. Adaptive seasonal time series models for forecasting short-term traffic flow. *Transportation Research Record: Journal of the Transportation Research Board*, 116–125.

Smith, Brian L, & Demetsky, Michael J. 1997. Traffic flow forecasting: comparison of modeling approaches. *Journal of transportation engineering.*

Smith, Brian L, Williams, Billy M, & Oswald, R Keith. 2002. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C:*

Emerging Technologies, **10**(4), 303–321.

Smola, Alex J, & Schölkopf, Bernhard. 2004. A tutorial on support vector regression. *Statistics and computing*, **14**(3), 199–222.

Sperduti, Alessandro, & Starita, Antonina. 1997. Supervised neural networks for the classification of structures. *Neural Networks, IEEE Transactions on*, **8**(3), 714–735.

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

Su, Haowei, Zhang, Ling, & Yu, Shu. 2007. Short-term traffic flow prediction based on incremental support vector regression. *Pages 640–645 of: Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 1. IEEE.

Sun, Hongyu, Liu, Henry X, Xiao, Heng, He, Rachel R, & Ran, Bin. 2003. Short term traffic forecasting using the local linear regression model. *In: 82nd Annual Meeting of the Transportation Research Board, Washington, DC*.

Sun, Shiliang, Yu, Guoqiang, & Zhang, Changshui. 2004. Short-term traffic flow forecasting using sampling Markov Chain method with incomplete data. *Pages 437–441 of: Intelligent Vehicles Symposium, 2004 IEEE*. IEEE.

Sun, Shiliang, Zhang, Changshui, & Yu, Guoqiang. 2006. A Bayesian network approach to traffic flow forecasting. *Intelligent Transportation Systems, IEEE Transactions on*, **7**(1), 124–132.

Sun, Shiliang, Huang, Rongqing, & Gao, Ya. 2012. Network-scale traffic modeling and forecasting with graphical lasso and neural networks. *Journal of Transportation Engineering*.

Tebaldi, Claudia, West, Mike, Karr, Alan F, *et al.* 2002. Statistical analyses of freeway traffic flows. *Journal of Forecasting*, **21**(1), 39–68.

Tewolde, Girma S. 2012. Sensor and network technology for intelligent transportation systems. *Pages 1–7 of: Electro/Information Technology (EIT), 2012 IEEE International Conference on*. IEEE.

Trigg, DW, & Leach, AG. 1967. Exponential smoothing with an adaptive response rate. *OR*, 53–59.

Uwents, Werner, Monfardini, Gabriele, Blockeel, Hendrik, Gori, Marco, & Scarselli, Franco. 2011. Neural networks for relational learning: an experimental comparison. *Machine Learning*, **82**(3), 315–349.

Van Der Voort, Mascha, Dougherty, Mark, & Watson, Susan. 1996. Combining Kohonen maps with ARIMA time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies*, **4**(5), 307–318.

Van Lint, J, 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*, 30–39.

Van Lint, JWC, & Van Hinsbergen, CPIJ. 2012. Short-term traffic and travel time prediction models. *Artificial Intelligence Applications to Critical Transportation Issues*, **22**.

Vapnik, Vladimir Naumovich, & Vapnik, Vlamimir. 1998. *Statistical learning theory*. Vol. 1.

Wiley New York.

Varaiya, Pravin. 2002. California's performance measurement system: Improving freeway efficiency through transportation intelligence. *TR News*.

Vlahogianni, Eleni I. 2015. Computational Intelligence and Optimization for Transportation Big Data: Challenges and Opportunities. *Pages 107–128 of: Engineering and Applied Sciences Optimization*. Springer.

Vlahogianni, Eleni I, Golias, John C, & Karlaftis, Matthew G. 2004. Short-term traffic forecasting: Overview of objectives and methods. *Transport reviews*, **24**(5), 533–557.

Vlahogianni, Eleni I, Karlaftis, Matthew G, & Golias, John 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.

Vlahogianni, Eleni I, Karlaftis, Matthew G, & Golias, John C. 2006. Statistical methods for detecting nonlinearity and non-stationarity in univariate short-term time-series of traffic volume. *Transportation Research Part C: Emerging Technologies*, **14**(5), 351–367.

Vlahogianni, Eleni I, Karlaftis, Matthew G, & Golias, John C. 2014. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, **43**, 3–19.

Vythoulkas, P. 1993. Alternative approaches to short term traffic forecasting for use in driver information systems. *Transportation and traffic theory*, **12**, 485–506.

Wang, Yibing, & Papageorgiou, Markos. 2005. Real-time freeway traffic state estimation based on extended Kalman filter: a general approach. *Transportation Research Part B: Methodological*, **39**(2), 141–167.

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

Williams, Billy, Durvasula, Priya, & Brown, Donald. 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*, 132–141.

Williams, Billy M, & Hoel, Lester A. 1999. *Modeling and forecasting vehicular traffic flow as a seasonal stochastic time series process*. Tech. rept.

Williams, Billy M, & Hoel, Lester A. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of transportation engineering*, **129**(6), 664–672.

Wu, Cheng-Ju, Schreiter, Thomas, & Horowitz, Roberto. 2014a. Multiple-clustering ARMAX-based predictor and its application to freeway traffic flow prediction. *Pages 4397–4403 of: American Control Conference (ACC), 2014*. IEEE.

Wu, Cheng-Ju, Schreiter, Thomas, Horowitz, Roberto, & Gomes, Gabriel. 2014b. Traffic Flow Prediction Using Optimal Autoregressive Moving Average with Exogenous Input-Based Predictors. *Transportation Research Record: Journal of the Transportation Research Board*, 125–132.

Wu, Chun-Hsin, Ho, Jan-Ming, & Lee, Der-Tsai. 2004. Travel-time prediction with support

vector regression. *Intelligent Transportation Systems, IEEE Transactions on*, **5**(4), 276–281.

Xie, Yuanchang, & Zhang, Yunlong. 2006. A wavelet network model for short-term traffic volume forecasting. *Journal of Intelligent Transportation Systems*, **10**(3), 141–150.

Xie, Yuanchang, Zhang, Yunlong, & Ye, Zhirui. 2007. Short-Term Traffic Volume Forecasting Using Kalman Filter with Discrete Wavelet Decomposition. *Computer-Aided Civil and Infrastructure Engineering*, **22**(5), 326–334.

Yin, Hongbin, Wong, S_C, Xu, Jianmin, & Wong, CK. 2002. Urban traffic flow prediction using a fuzzy-neural approach. *Transportation Research Part C: Emerging Technologies*, **10**(2), 85–98.

Yisheng, An, Qingsong, Song, & Xiangmo, Zhao. 2011. Short-term traffic flow forecasting via echo state neural networks. *Pages 844–847 of: Natural Computation (ICNC), 2011 Seventh International Conference on*, vol. 2. IEEE.

Yu, Guoqiang, Hu, Jianming, Zhang, Changshui, Zhuang, Like, & Song, Jingyan. 2003. Short-term traffic flow forecasting based on Markov chain model. *Pages 208–212 of: Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*. IEEE.

Yuan, Yufei, Van Lint, JWC, Wilson, R Eddie, Wageningen-Kessels, Van, Hoogendoorn, Serge P, *et al.* 2012. Real-time Lagrangian traffic state estimator for freeways. *Intelligent Transportation Systems, IEEE Transactions on*, **13**(1), 59–70.

Yun, S-Y, Namkoong, Seong, 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), 293–310.

Zhang, HM. 2000. Recursive prediction of traffic conditions with neural network models. *Journal of Transportation Engineering*, **126**(6), 472–481.

Zheng, Weizhong, Lee, Der-Horng, & Shi, Qixin. 2006. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of transportation engineering*, **132**(2), 114–121.