SPATIOTEMPORAL ANOMALY DETECTION AND PREDICTION OF ANOMALY PROPAGATION PATH USING LSTM NETWORKS

By

Sanchita Basak

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

December 12, 2020

Nashville, Tennessee

Approved:

Dr. Abhishek Dubey

Dr. Aniruddha Gokhale

Dedicated to my family in Kolkata, India

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor Dr. Abhishek Dubey for his guidance in conducting the research. I would also like to thank Dr. Aniruddha Gokhale for being a member of my thesis committee. I would like to express a special note of thanks to my friends Scott Eisele, Chinmaya Samal, Fangzhou Sun, Geoffrey Pettet, Michael Wilbur, Shreyas Ramakrishna, and Nithin Guruswamy for their constant support during my time at Vanderbilt.

I would like to acknowledge the research funding from National Science Foundation (NSF) and Siemens CT for supporting various research projects in which I have participated.

I want to convey my heartfelt gratitude to Dr. Shyamali Mukherjee and Dr. Saptarshi Sengupta for always staying by my side when the going got tough.

None of this would have come to fruition without the constant love and support of my family based in Kolkata, India. This one is for you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Introduction

I.1 Problem Overview

A Cyber Physical System (CPS) is an integration of cyber and physical parts. Examples include transportation networks (Deka et al., 2018), water networks (Wei and Li, 2015), smart grid (Yu and Xue, 2016) etc. We can retrieve the information about the physical systems through various forms of sensors and build models that ensure safe and correct operation of such systems. Faults that affect this efficient and accurate operation of these systems need to be identified.

Modern Cyber Physical System is a collection of connected nodes and edges where information diffuses among various nodes of the network. Each node is associated with a sensor which measures the system state. In a large scale geographically distributed Cyber Physical System some nodes are in close proximity of each other whereas some are far away. The nodes are mutually influential and are connected in a way that the state of a particular node is affected by the state of the nodes in its neighborhood. The hypothesis is that if there is a fault or anomaly in a sensor associated with a node then there must be anomalies in the neighboring sensors as well. The objective is to try to identify anomalies in each node and identify the anomalies that are in neighboring nodes that appear within a specified time interval and analyze if those anomalies are connected or not. Connected anomaly indicates multiple anomalies that occur in a close spatial neighborhood within a short interval of time and appear due to a common reason and progresses in a cascaded way.

A networked Cyber Physical System can be represented as a connected graph with multiple nodes and edges that share information and influence each other. The research questions that this thesis aims to solve are as follows: (a) how to capture the neighborhood information in a networked Cyber Physical System, (b) how to capture temporal progression of system variables in networked CPS, (c) how to select past temporal windows that affect future system states, (d) how to diagnose connected anomalies and differentiate between cyber and physical anomalies and (e) how to prognosticate anomaly propagation path.

Thus the objective of this research is to identify and correlate anomalies in networked Cyber Physical Systems. Once an anomaly is identified it needs to be categorized whether it is due to a cyber attack or due to some physical incidents. If multiple anomalies are observed, the correlation among them should be investigated and the cascaded progression of anomalies should be analyzed. From the sequence of this chained anomaly progression, the root cause i.e., the source of anomaly needs to be isolated. Also the focus is to prognosticate future progression of anomaly in networked Cyber Physical Systems.

An efficient way to identify anomalous behavior in non-stationary Cyber Physical Systems is to look at the context specific data and study the changing distributions of data over time and learn the temporal variations of this data to take into account the variances and uncertainties introduced by temporal scales. However, along with temporal variations it is important to characterize the spatial influences that govern the system dynamics. Existing research in this area ignores the micro-scale, subsystem level spatial influence in modeling system properties (Ma et al., 2015b). Thus there remains a need to develop a robust, system-wide framework that works at a granularity of capturing the specificity of micro-level sub-systemic spatial interactions. In this context we looked at the neighborhood specific information as in a small neighborhood, that is at a very small systemic level the data distributions should have influence over each other and should cluster together. So it is important to note how the observation in a single neighborhood gets impacted by the observations nearby. In this context it is important to chose optimal neighborhood as that helps avoid lack of specificity as well as breach of causality. Thus it takes into account the exact information flow among the actual neighbors. This is proposed to solve using graph based approaches. One of the primary reasons why the graph data is useful is because of its structural correlations that help discover underlying data patterns. It is important to note that multi-dimensional data objects are not always completely independent of each other when they are part of the same subsystem. To capture their interdependence in multi-dimensional space, incorporation of graphical structure can help capture the correlations among various data objects. On the basis of that, an integrated architecture to model the system as a directed connected graph encapsulating the micro-scale spatial interactions is created for large scale Cyber Physical systems.

This thesis aims to develop a common framework for diagnosis and prognosis of anomaly for a networked Cyber Physical System. The solution approach to mitigate these research challenges are explained with traffic networks which is an example of a large scale networked Cyber Physical System.

In the context of traffic networks, anomaly is defined as the sudden and huge difference between the predicted speed value of a road segment and the observed speed value of that road segment. The goal is to identify whether an anomaly is due to a cyber incident or a physical incident. As a first step, we continuously predict the traffic speed of each road segment based on the speed of its neighbors. If a sudden and huge difference between the predicted and observed speed is noticed from statistical measurements, an anomalous prediction is identified. Next, a diagnostic framework is built which relies on connecting the dots between anomalous predictions of multiple roads in a neighborhood that occur within a short time window. Consequently a graphical representation to capture the flow of anomaly propagation is created. Once the anomalies are identified it is investigated if they are connected and occur in a close neighborhood within a short time. If this condition holds true then the anomalous occurrences are identified to have evolved from a physical incident, otherwise they are attributed to cyber attacks on the sensors. Also the thesis addresses prognosticating the future states of anomaly propagation. In context of traffic networks the thesis develops a citywide connected recurrent neural architecture to prognosticate traffic congestion propagation patterns and discusses the likelihood of congestion propagation from a source of congestion to its neighboring segments.

I.2 Scope of the Thesis

This thesis focuses on developing application-specific spatiotemporal anomaly detection and prognosis frameworks for networked Cyber Physical Systems. In this thesis two research questions are explored:

- The first question is how to capture the neighborhood information in a networked cyber-physical system as well as capture temporal correlations of system variables to identify system anomalies. Once an anomaly is identified how can one traverse the anomaly propagation path to identify the source of anomaly. This question is addressed in chapter II of the thesis. In this case, we use real traffic data from Nashville, TN to demonstrate a novel anomaly detector and a Timed Failure Propagation Graph (TFPG) (Abdelwahed et al., 2009) based diagnostics mechanism. The novelty lies in the ability to capture the the spatial information and the interconnections of the traffic network as well as the use of recurrent neural network architectures to learn and predict the operation of a graph edge as a function of its immediate peers, including both incoming and outgoing branches.

- The second question that the thesis looks at is how to predict anomaly propagation path in advance. For example, when a congestion is detected at a particular road segment of the traffic network can we predict how the congestion will propagate to the neighboring road segments and when. We answer this question in chapter III of the thesis. In the past, this problem has mostly been addressed by modelling the traffic congestion over some standard physical phenomenon through which it is difficult to capture all the modalities of such a dynamic and complex system. While other recent works have focused on applying a generalized data-driven technique on the whole network at once, they often ignore intersection characteristics. On the contrary, we propose a city-wide ensemble of intersection level connected Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network models and propose mechanisms for identifying congestion events using the predictions from the networks. To reduce the search space of likely congestion sinks we use the likelihood of congestion propagation in neighboring road segments of a congestion source that we learn from the past historical data. We validated our congestion forecasting framework on the real world traffic data of Nashville, TN, USA.

I.3 Outline of the Thesis

A descriptive exposition of data-driven detection of anomalies and cascading failures in traffic networks is laid out in Chapter II. In this Chapter the problem statement is first described, the background is elaborated on, the traffic speed prediction model is proposed and how that is used in detection of anomalies is detailed. A discussion follows about the cascading effect of traffic congestion and finding the root cause. Next, in Chapter III an analysis of the cascading effect of traffic congestion using LSTM networks is introduced. In order to do that, a citywide connected LSTM fabric and a congestion propagation framework are introduced. Chapter IV concludes the thesis with possible directions of future research.

CHAPTER II

Data-Driven Detection of Anomalies and Cascading Failures in Traffic Networks

II.1 Problem scope

Since the emergence of smart cities, a major focus has been in the area of Intelligent Transportation System. These systems provide researchers with unique opportunities to study the dynamics of road traffic. In this chapter, we address the first research question, that is how to capture the neighborhood information in a connected cyber-physical system as well as capture temporal correlations of system variables to identify system anomalies. Once an anomaly is identified we also discuss how can one traverse the anomaly propagation path to identify the source of anomaly. The first step of the proposed anomaly detection model relies on an effective traffic speed prediction architecture on top of which an effective anomaly detection and anomaly source identification approaches are built. The work discussed in this chapter is published in (Basak et al., 2019).

Traffic predictions can be performed based on two different approaches: model-driven and data-driven (Barros et al., 2015). In model-driven approaches, we have a physical model that represents the network topology, incorporating information about intersections, road segments, signals, geographical coordinates of Traffic Message Channel (TMC), etc. In data-driven approaches, information regarding various forms of traffic measurements, such as speed and congestion factor, are needed for training, which can be obtained from sensors, such as induction-loop detectors placed in the road network.

Our aim here is to combine model-driven and data-driven approaches to build an effective traffic prediction architecture. We use the physical model of the network to generate a directed graph that captures the spatial interconnections within the network. The temporal dependencies of the flow patterns are captured by training recurrent neural network architectures using significant amounts of sensor data. Thus, combining the model-driven and data-driven approaches, we can assess the evolution of the traffic state of the entire road network.

We demonstrate our approach using real traffic data from Nashville, TN, USA obtained via the HERE API (her, 2020). In particular, we study the efficacy of building traffic-speed predictors using two different approaches, Long-Short Term Memory Networks (LSTMs) and Gaussian Process Regression (GPR). For both approaches, we model the speed of each road segment in the network as a function of its neighboring road segments, and build specialized traffic predictors for each edge of the entire network.

We develop the traffic speed prediction model keeping two objectives in mind: 1) detection of anomalous sensor readings and 2) a model to capture the dynamics of congestion propagation in a cascaded way. The disruptive events in the traffic network causing anomalous sensor readings can be due to malicious sensor attacks involving data manipulation as well as real physical incidents creating congestion. For sensor anomaly and attack detection, we introduced additive and deductive anomalies in the real-time traffic data and showed the ability of the trained traffic predictors to identify the attacks using statistical control charts. We also analyzed the precision and recall of this anomaly detection scheme.

Next, the cascading effect of congestion in a traffic network is analyzed where congestions/perturbations created at a local level at a targeted road segment can propagate backwards like a wave to affect a larger part of the network leading to chained congestions. To analize such effects in a large-scale traffic network, we use the SUMO (Simulation of Urban Mobility) (Behrisch et al., 2011) traffic simulator to access real-time traffic simulation and monitor as well as analyze traffic patterns under the influence of congestion. We trained traffic predictors with data collected from SUMO under normal operating conditions and showed that the pre-trained models effectively predicted the real-time cascading effect of congestions spreading out to the neighboring road segments. Once a persisting congestion is noted in a road segment, we identified the root-cause of the cascaded congestion by finding the target road where the congestion started using Timed Failure Propagation Graphs

(TFPG) (Abdelwahed et al., 2009).

Contributions Our contributions in this chapter are:

- Building efficient LSTM based traffic predictors in an unique way of modelling each road segment in a large scale traffic network as a function of its neighboring roads and comparing its performance with that of Recurrent Neural Network and Gaussian Process Regression. We achieved an accurate prediction model with an average loss of 6.55×10^{-4} on normalized speed values.

- These traffic predictors combined with statistical control chart CUSUM are able to detect anomalies in sensor reading with high precision and recall indicating an AUC of 0.8507 of the precision-recall curve.

- We formulated the traffic congestion propagation as a Timed Failure Propagation Graph to identify the root cause of failure in the network.

Thus we are interested in developing data-driven detectors to identify the following disruptive events: (a) sensor attacks, that is, cyber-attacks against smart sensors by a networked adversary which may change the measurements values arbitrarily, and (b) physical incidents, such as motor vehicle accidents, that occur randomly and may cause a cascade of traffic disruptions throughout the road network by creating chained traffic congestion. In such cases, identification of the root cause of an event can help eliminate the cascaded propagation of congestion.

II.1.1 Definitions

Transportation Network Graph: A graph representing our system model is defined as $G = (V,E)$ where V is a set of nodes. E is the set of road segments connecting the nodes. In the graph, let $v_i \in V$ denote a node and $e_{ij} = (v_i, v_j) \in E$ represent an edge.

in, out: The *in* operator $in : V \rightarrow 2^E$ gives all the edges for which this node v is the destination. When the *out* operator is applied to a node $out : V \rightarrow 2^E$ it gives all the edge for which this node v is the source.

in degree, out degree: The *in degree* of a node v is the number of road segments incoming to the node and can be calculated as $|in(v)|$, whereas, the *out degree* of a node v is the number of road segments outgoing from the node and is calculated as $|out(v)|$.

Traffic Message Channels (TMC): We call an edge a traffic message channel (TMC) if it has timestamped traffic speed data associated with it. We denote the set of TMC as $TMC \subseteq E$. Each sensor $(s \in S)(s : TMC \times T \rightarrow \Re^{+})$ represents the speed readings of each traffic message channel at times T.

Jam Factor: We also have a function J that provides the jam factor, a value between 0 to 1 that describes the congestion on road. 0 means no congestion and 1 means the observed speed is zero.

***k*-hop incoming neighbors**: These are the k-nearest hops of the incoming edges feeding traffic into an edge. The set of k hop incoming neighbors $N_{in}^{k}(e)$ can be defined recursively as $\bigcup_{x \in N_{in}^{k-1}(e)}(in(src(x)))$ using the definition of 1 hop neighbors, $N_{in}^{1}(e) = \bigcup_{x \in in(src(e))}(in(src(x)))$

***k*-hop outgoing neighbors**: These are the k-nearest hops of edges taking traffic away from an edge via its out node. We define this function recursively as well. $N_{out}^{1}(e) = \bigcup_{x \in out(dst(e))}(out(dst(x)))$. Given the set N_{out}^{k-1}, the set N_{out}^{k} can be defined as $N_{out}^{k}(e) = \bigcup_{x \in N_{out}^{k-1}(e)}(out(dst(x)))$.

Physical incident: By *physical incidents*, we mean the failure circumstances such that the real speed of the edge is significantly below maximum speed. This can typically be explained by motor-vehicle accidents that occur randomly and may cause a cascade of traffic disruptions throughout the road network. An incident can cause disruptions (i) directly through traffic congestion, which may propagate to connected roads, and/or (ii) indirectly by forcing vehicles to take alternative routes, even before reaching the areas that are not affected by direct congestion.

Logical incident: A *logical incident* is the hypothesis that the observed speed of the edge is significantly different from its speed under normal operating condition. A logical incident can be caused by a fault in the sensor or by an adversary. The disruptive events alter the traffic speed attribute where physical incidents have effect on real traffic speed, but the sensor failures and attacks change the observed or sensed traffic speed but not the real speed.

AUC: AUC is the area under the precision recall (PR) curve. It is used as an indicator of efficiency of the anomaly detection approaches discussed in the chapter. The greater the AUC of the PR curve, the better is the detection model.

II.1.2 Problem Definition and Dataset

We are concerned with the following problem. Given a transportation graph and a sequence of real-time speed readings,detect the occurrence of the anomalous events. Performance in detecting anomalies is provided by quantifiable measures such as false positives, true positives, false negatives, true negatives as well as precision and recall. Second, we want to identify the root cause of an event (e.g., if a congestion event on a road causes a large disruption through cascades of reroutes, we need to identify the original congestion event as the root cause). In this chapter, we specifically study this problem for the transportation network of Nashville, TN. In particular, we use the data collected by our team from HERE to setup the problem. The data contains timestamped representation of information regarding speed, jam factor, free flow speed, etc. for each Traffic Message Channel (TMC) ID. Each TMC ID identifies a specific road segment and represents the sensor information for that particular segment.

To inject physical incidents and study their effect, we use the microscopic traffic simulator SUMO, which we have configured for Nashville.

II.2 Related Work

The approach that we will describe later in this chapter is combining three active areas of research (a) building a predictor to forecast the normal congestion events and the expected speeds on the road network; (b) using these predictors to build anomaly detectors; and (c) developing a cascade model to study the progression of congestion and effectively isolate the root causes. We make the assumption of single root failure (physical incident). The model of computation we use and describe later in this chapter can support multi-failure hypothesis. But we leave that for our future work.

II.2.1 Existing Work on Traffic Forecast with Machine Learning

Ma et al. (Ma et al., 2015a) presented an LSTM neural network to predict travel speed using microwave detector data. They collected 1-month traffic speed data from two sites in Beijing expressway. They compared three different typologies of recurrent neural network (i.e. Elman NN, TDNN and NARX NN) as well as other non-parametric and parametric methods (i.e. SVM, Time Series and Kalman Filter) with the LSTM NN based on the same dataset. The numerical experiments proved that the LSTM NN performes better than other algorithm in terms of accuracy and stability. Tian et al. (Tian and Pan, 2015) introduced a model called Long Short-Term Memory Recurrent Neural Network (LSTM RNN) which represents long-term dependencies and determines the optimal time lags for time series problems. The study used data from the Caltrans Performance Measurement System (PeMS) and included a comparison of the LSTM RNN model with other four established prediction models, i.e., RW (Random Walk), SVM (Support Vector Machine), FFNN (Feed Forward NN) and SAE (Sum of Absolute Errors). This study mainly analyzed the temporal influence on traffic flow but does not consider other factors, such as spatial impact from neighbour observation stations, weather conditions, etc.

Polson et al. (Polson and Sokolov, 2017a) developed a deep learning architecture which combined a linear model that was fitted using l_1 regularization and a sequence of $tanh$ layers. The first layer identified spatiotemporal relations among predictors and other segments modelled nonlinear relationships. The study provided a twofold analysis of short-term traffic forecasts from deep learning. It demonstrated that deep learning provides a significant advancement over linear models. A good review of Deep Learning technologies used in forecasting analysis can be found in (Sengupta et al., 2019b). Prior work on traffic forecasting has also been carried out with multi agent based approaches. Hu aet al. (Hu et al., 2014) used Particle Swarm Optimization for traffic flow prediction. Some recent swarm-based algorithms listed in [(Sengupta et al., 2019a), (Sengupta et al., 2018)] can also be used in this purpose.

For short term traffic volume forecasting, Zhao et al. (Zhao et al., 2017) proposed a cascaded LSTM network by combining the interaction among the road network in both the time and spatial domain. They showed that the proposed LSTM network approach for traffic volume prediction is sturdy and had a minimum MRE (Mean Relative Error) compared to other models such as ARIMA (Autoregressive Integrated Moving Average) model, SVM (Support Vector Machine) and SAE (Stacked Auto-encoder). LSTM and RNN architectures also outperformed other techniques in numerous applications, such as language learning [(Gers and Schmidhuber, 2001)], connected handwriting recognition [(Graves and Schmidhuber, 2009)], Remaining Useful Life Prediction of hard disks [(Basak et al., 2019b)].

In comparison our approach we model each road segment in the network as a function of its neighboring roads and use that relationship for prediction. When we compared our performance with that of RNN and Gaussian Process Regression we saw that we achieved a better prediction model with an average loss of 6.55×10^{-4} calculated on Normalized Speed.

II.2.2 Existing Work on Traffic Anomaly Prediction

Zygouras et al. (Zygouras et al., 2015) presented an approach to identify anomalous sensors and resolve whether irregular measurements are due to faulty sensors or unusual traffic. The proposed method was implemented by using the Lambda Architecture which combined a batch processing framework (i.e. Hadoop3) and a distributed stream processing system (i.e. Storm4) for efficiently processing both historical and real-time data. The authors also developed a Crowdsourcing system to extract answers from the human crowd based on the MapReduce paradigm. The study recognised anomalous SCATS (Sydney Coordinated AdaptiveTraffic System) sensors from Dublin city with three methods; Pearson's correlation, cross-correlation and multivariate ARIMA model. The three different outlier detection techniques identified a complementary set of faulty sensors. The study gave a detailed experimental evaluation to prove that their proposed approach effectively resolved the source of irregular measurements in real-time.

Lu et al. (Lu et al., 2008) provided a systematic study of previous loop fault detection and data correction methods, and also systematic classification of possible faults and the reasons behind them at different levels. According to the study, existing work on loop fault detection and data correction/imputation may be divided into three levels which lead to different viewpoints for loop fault detection and data correction: macroscopic such as: (a) TMC/PeMS level; (b) mesoscopic – a stretch of freeway; and (c) microscopic – control cabinet level. These three levels of approaches are complementary to each other although they study the problem from different aspects using a different level of data.

In this work we used statistical control chart CUSUM to identify malicious sensor attacks with high precision and recall indicating an AUC of 0.8507 of the precision-recall curve.

II.2.3 Existing Work on Cascading Failures

Daqing et al. (Daqing et al., 2014) studied the long-range spatial correlation of cascading failures and their evolution with time to predict system collapse in case of power grid failures and traffic congestion. Zhang et al. (Zhang et al., 2015) employed an improved form of Coupled Map Lattice(CML) model to analyze the cascading failures on Beijing Traffic network. They considered the traffic network topology and tested on various attack strategies and how the scale of failure varies with external perturbations, coupling strengths and attack strategies. Liang et al. (Liang et al., 2017) proposed a data-driven approach $CasInf$ to study the cascading patterns of traffic propagation through maximizing the likelihood function from the available data. They treated it as a submodular function maximization problem providing near-optimal performance guarantees.

In this chapter, the spatiotemporal correlations of anomaly have been formulated as a separate directed graph. The concept of Timed Failure Propagation Graph (TFPG) in analyzing and correlating traffic congestion propagation pattern has been introduced in this research. Apart from analyzing the cascading effects of traffic congestion on the neighboring road segments of the network, the innovation of this work lies in showing that the source of congestion can be isolated by formulating the congestion propagation problem as a Timed Failure Propagation Graph.

II.3 Traffic Speed Prediction Model

For the Nashville dataset, we have 3,724 unique TMCs. For each TMC we have collected speed values for a total of 6000 timesteps. Each timestep specifies a small time interval of 10 minutes.

First, a matrix of dimension *(total number of timesteps $\times$ traffic speed for all unique TMC IDs)* (6000×3724) is formed. Some of the TMCs do not have speed value recorded. To interpolate the missing speed value of a particular TMC, we are considering the speed values of all the neighbouring TMCs for the preceding and succeeding timestep using data imputation.

Since we consider the speed of the neighbors for predicting the speed of a TMC we must ensure that we normalize the speeds (see definition 1). The normalized speeds are defined to be in between 0 and 1 and help ensure data ranges are balanced between the road segments. This is required for building a good predictive model.

Definition 1 (Normalized speed) *The normalized speed of a TMC (definition II.1.1) is a ratio of its current speed with the average of speeds for times when the jam factor (definition*

II.1.1) is zero.

For each TMC, $N^1_{in}(TMC)$ and $N^1_{out}(TMC)$ give the set of its immediate incoming and outgoing neighbors respectively. For each TMC, the normalized speed values for each of its neighbors (including incoming and outgoing) are treated as input features whereas the normalized speed of the target TMC is treated as the label. We applied both Recurrent Neural Networks and Long Short Term Memory Networks to build the traffic predictors for each TMC in the traffic network.

The number of timesteps to look back in order to predict the result for current timestep has been chosen in a way that produces the least loss. The timesteps are varied from 5 to 20. From the experimental results, we have seen that for RNN, ten timesteps provide a stable outcome whereas LSTM gives better result with 15 timesteps. Table II.1 shows the average loss on test data calculated over normalized speeds for different timesteps produced by RNN and LSTM.

Table II.1: Average Loss with Different Number of Timesteps

Number of Timesteps to Look Back	**Average Loss from RNN**	**Average Loss from LSTM**
5	0.0007797	0.0007032
10	**0.0006966**	0.0007063
15	0.0006976	**0.0006805**
20	0.0006966	0.0006853

RNN and LSTM take the input as a three dimensional matrix of dimension specified as $(Samples \times timesteps \times features)$ where number of features is equal to the total number of Neighbouring TMCs. As the sample labels for a particular TMC is the normalized traffic speed value of that TMC, the network learns to predict the speed at any timestep for the target TMC given past 10 timesteps of data inputs form its neighbors. The sample matrices are split randomly into *Training Set* and *Test Set* (70% Training and 30% Testing).

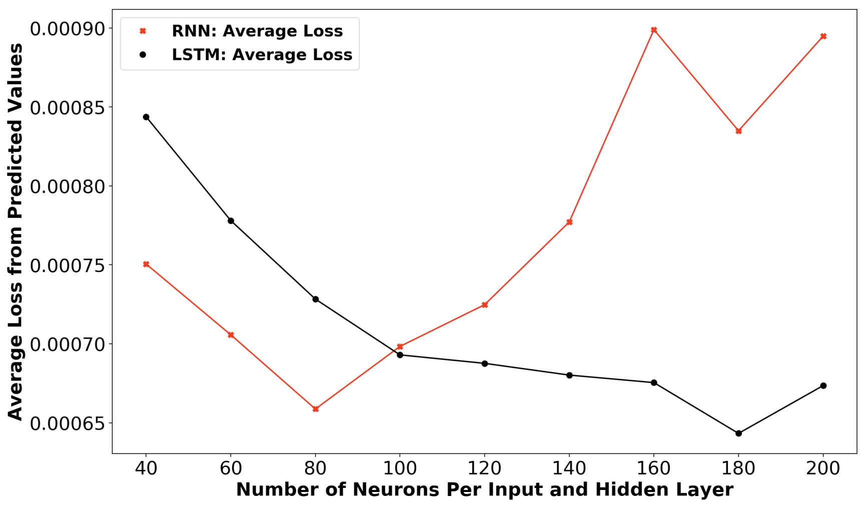

Figure II.1: Average loss from predicted speed values by RNN and LSTM for different number of neurons

II.3.1 Prediction Using Recurrent Neural Network

For Recurrent Neural Network (RNN) prediction model, we have tried a different number of neurons (from 40 to 200) in the input and hidden layers. We ran the models with a different number of neurons for the first 100 TMC. From the average losses, we have found out that RNN works better with 80 neurons. Figure II.1 shows the average losses produced by RNN and LSTM for the different number of neurons. The average losses provided by RNN show a downward trend for 40 to 80 neurons. Afterwards, as the number of neurons increases, the average loss also increases.

We have used *Mean Squared Error* as the loss function for RNN. For training the deep neural model, we have used *Adam* optimizer. Figure II.2 shows the predicted speed value and actual speed value of the first TMC for the first 400 timesteps. The loss of this prediction is 3.388×10^{-5}.

II.3.2 Prediction Using Long Short-Term Memory

For the LSTM model, we have predicted normalized speed values for the different number of neurons (40 to 200). The average losses show a downward trend with the increasing

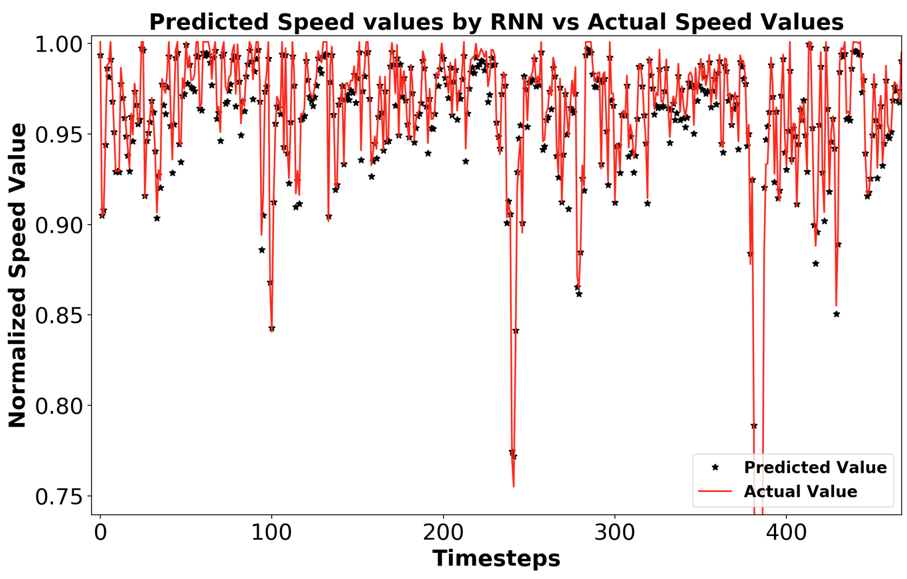

Figure II.2: Predicted speed values by RNN vs actual speed values for the test data of first TMC

number of neurons. According to our experiment, 180 neurons in both input and hidden layer produces the least average loss. The loss function is defined in terms of mean squared error. Figure II.1 shows the average loss produced by RNN and LSTM with varying number of neurons. It is visible from the figure that RNN converges with 80 neurons while LSTM needs 180 neurons. So, in our LSTM model, we have used 180 neurons.

Figure II.3 shows the predicted speed value and actual speed value of the first TMC for the first 400 timesteps. The loss of this prediction is 2.704×10^{-5}.

II.3.3 Comparison Between RNN and LSTM

To compare which model is producing a better result, we have run the model with their optimal number of neurons and timesteps. Based on our experiments, the optimal number of neurons for RNN and LSTM is 80 and 180 respectively. RNN works the best with 10 timesteps and LSTM with 15 timesteps. So, we ran both the models for the first 100 TMC to see which delivers the best result. Figure II.4 shows the losses for the first 100 TMCs. It is visible in the figure that LSTM produces the least loss in most cases. The average loss from RNN is 7.04×10^{-4}, and average loss from LSTM is 6.55×10^{-4}. So, LSTM works

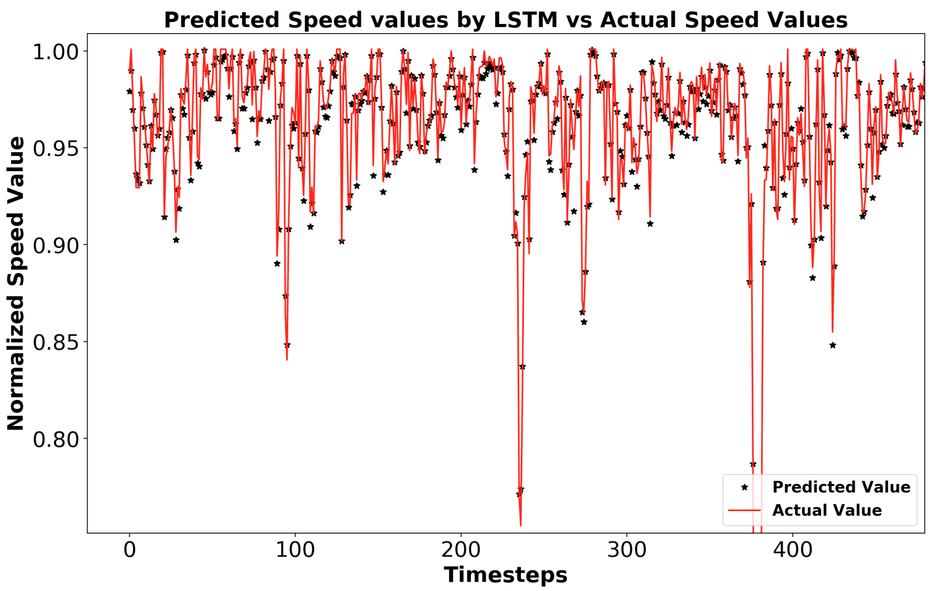

Figure II.3: Predicted speed values by LSTM vs actual speed values for the test data of the first TMC

best for this dataset.

II.3.4 Prediction Using Gaussian Process Regression

Other than neural networks, we have also used Gaussian Process Regression (Rasmussen and Williams, 2005) which is a Bayesian approach for modelling functional relationships to build traffic predictors. The underlying assumption in this process is that the prior distribution of the regression function is considered to be a multivariate Gaussian distribution. By calculating the covariance matrix for the labeled data and covariance vector between labeled and new test data points and taking the measurement noise into account, the prediction result for the test data points can be obtained (Ghafouri et al., 2017). In this work, we have used Radial Basis Function (RBF) as the kernel. Figure II.5 compares the root mean square losses of the prediction results produced by LSTM and Gaussian Process Regression for the first 100 TMCs. The average loss from Gaussian Process Regression is 0.0178 whereas LSTM produces an average loss of 6.55×10^{-4} showing that LSTM works best for this traffic speed prediction problem.

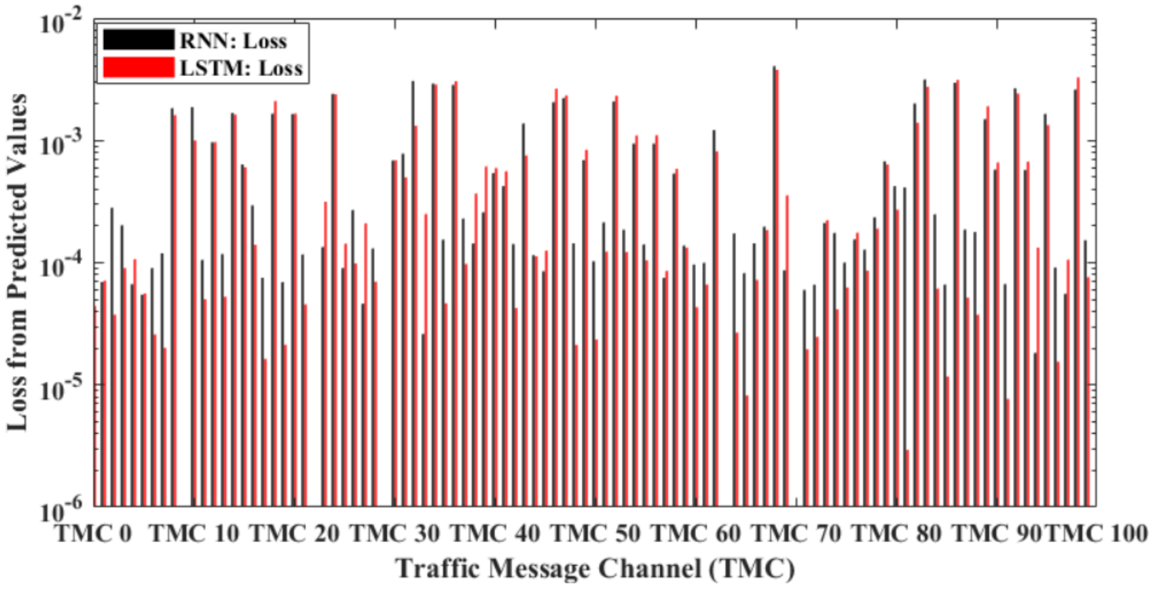

Figure II.4: Loss from predicted speed values by LSTM and RNN for the first 100 TMC (difference is shown in Log scale)

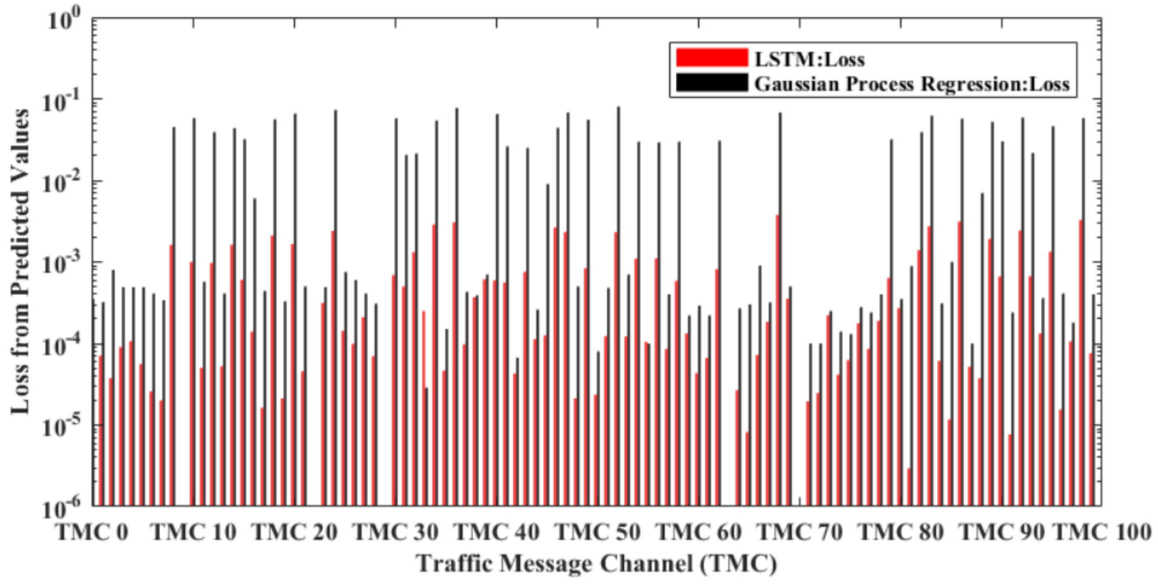

Figure II.5: Comparison of LSTM and Gaussian Process Regression based on predicting the speed of a chosen TMC

II.4 Detection of Anomalies

The goal here is to identify anomalous sensor readings in the traffic networks. Anomalous sensor readings can arise due to sensor attacks as faults can be artificially injected in the data stream associated with a sensor by a networked adversary. So it is important to build effective anomaly detectors so that we can mitigate the effects by replacing the erroneous or missing data with predictions based on correct values from other sensors through data imputation.

Each TMC ID is associated with a sensor s_i whose value is predicted through the set of sensors $(s_j \in S, i \neq j)$ placed in the neighboring (incoming and outgoing) road segments. The anomalous sensor readings can be detected by calculating the difference between the prediction and the real-time sensor measurement. The time series data representing this difference can be used for identifying anomaly. The anomalies in the sensor data can be introduced in two ways: additive or deductive. In case of additive anomalies, the sensor readings are increased arbitrarily compared to normal operating conditions. Conversely, for deductive anomalies, the sensor readings are decreased compared to the normal conditions. We must inject anomalies artificially into the real data since we need ground-truth labels for anomalies in order to validate the detection approach, but we do not have any labels corresponding to anomalous readings of real data. Figure II.6 shows and example of differences between the predicted and the actual real-time sensor measurements during an additive sensor attack.

In this work, we use Cumulative Sum Control chart (CUSUM) (Page, 1954), which is a statistical control chart to track the variation of timeseries data. This algorithm is used to identify the timestamp when the anomaly started and ended, the amplitude of change, and an alarm (timestamp of when the anomaly was detected).

By choosing a threshold, we can control the number of false positives and negatives, i.e., we can modulate the sensitivity of the algorithm for anomaly detection. The upper $(usum_s^t)$ and lower $(lsum_s^t)$ cumulative sums are defined as:

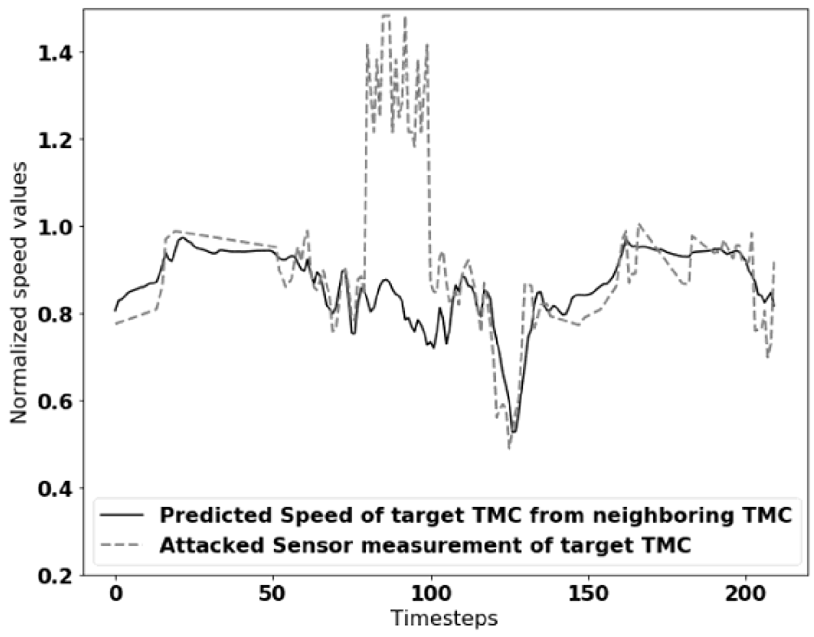

Figure II.6: Introducing additive anomaly into sensor readings of a TMC

$$usum_s^t = \max\{0, usum_s^{t-1} + x_s^t - \mu - k\} \tag{II.1}$$

$$lsum_s^t = \min\{0, lsum_s^{t-1} + x_s^t - \mu + k\} \tag{II.2}$$

The CUSUM criterion detects a sample x_s^t of sensor s to be anomalous at timestamp t, if $(usum_s^t > \eta_s)$ or $(lsum_s^t < \eta_s)$, where η_s is the detection threshold for sensor s.

Figure II.7 shows the detection of anomaly for the case described in Figure II.6. We introduced an additive sensor attack between the time window of (80,100) and the difference between the predicted speed through LSTM and the sensor data subjected to the attack has been fed to CUSUM, which triggered the alarm at 80th and 100th instant, identifying the actual time of attack. This anomaly identification can be carried out online as we continuously feed the difference between the prediction and sensor measurements. This validates the fact that using traffic predictors combined with change detection algorithm CUSUM, online identification of anomalies is possible.

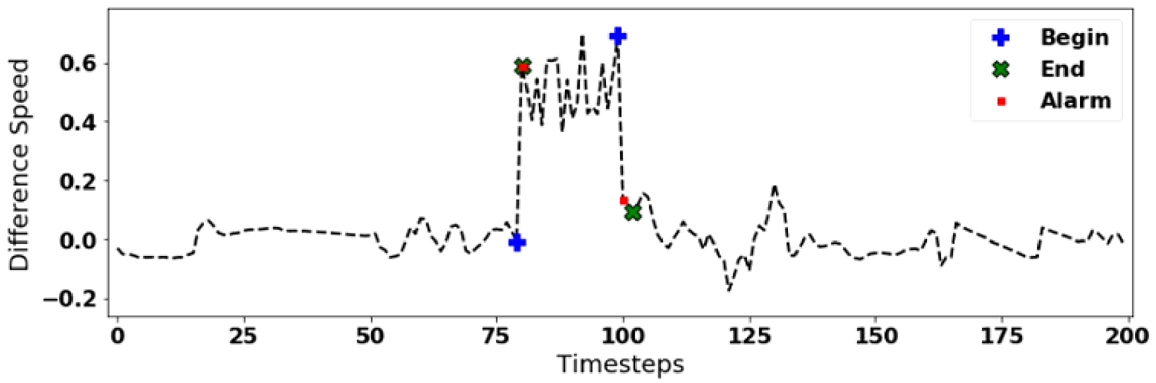

Figure II.7: Detection of anomaly through CUSUM algorithm

To compare the efficiency of the anomaly detection scheme between the approach combining LSTM based traffic predictors and CUSUM and on the other hand, Gaussian Process Regression based traffic predictors and CUSUM, we show the Precision-Recall curve for both the approaches by varying the anomaly detection thresholds similarly. Series of randomly generated additive and deductive anomalies have been introduced in the sensor data and the above mentioned approaches have been applied on the same altered data to identify the anomalies. Figure II.8 shows the Precision Recall curves of LSTM and Gaussian Process Regression showing their comparative efficiency in identifying anomalies. The Area Under Curve (AUC) for Gaussian Process Regression is 0.4070 whereas the AUC for LSTM based approach is 0.8507 showing its superiority in identifying anomalies all other conditions remaining equal. We expected LSTM to perform better in anomaly detection because we had already seen in Figure II.5 that it predicts traffic speed more accurately

It is to be noted that anomalies in sensor data can also be due to physical incidents. However, the presence of any physical incidents can be deduced by Timed failure Propagation Graphs indicating a sequence of anomalies. This is described in detail in Section II.5.

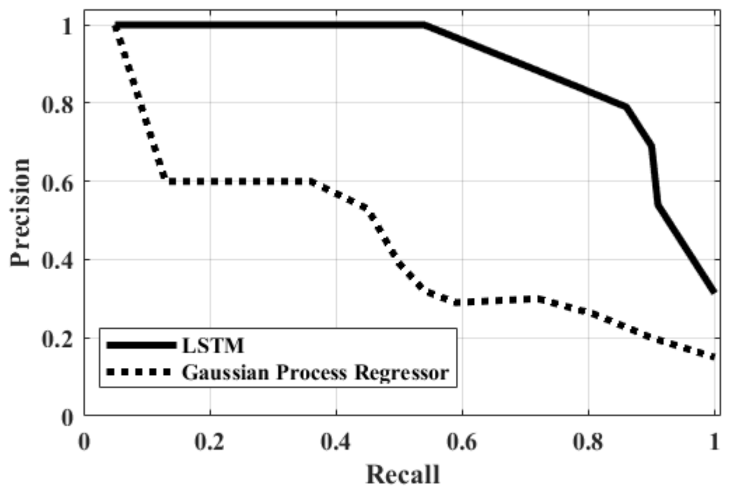

Figure II.8: Precision Recall curves of LSTM and Gaussian Process Regression showing their comparative efficiency in identifying anomaly

II.5 Cascading Effect of Traffic Congestion

In a large-scale interconnected system such as a traffic network, congestion in one (or some) parts can lead to congestion in other, connected parts as well. In this chapter, our goal is to identify the pattern of how congestion originating from one road segment propagates backwards to the incoming branches of the road segment, creating a cascading effect of traffic congestion.

To study the spread of road congestion, we used SUMO, which is a microscopic traffic simulator. SUMO allows us to introduce congestion by manipulating a running simulation and to measure road traffic using simulated traffic sensors. All of the experiments in this section are based on SUMO simulations. We simulated congestion scenarios on a part of Nashville's road network, which we downloaded from OpenStreetMap (ope, 2020). Figure II.9 shows the part of the road network that we used in our simulations. For our experiments, we introduced congestion at road segment R1.

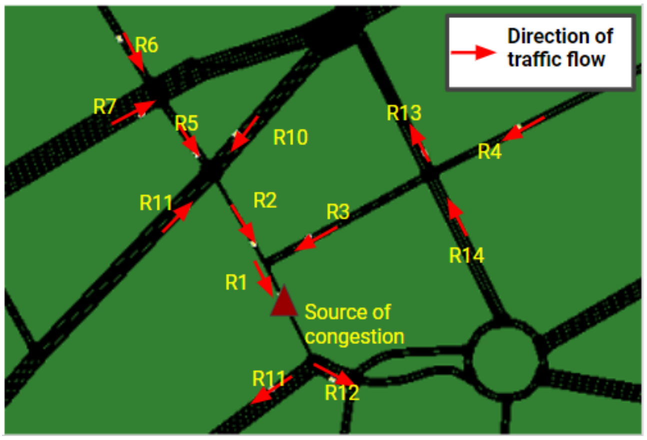

Figure II.9: Part of the Nashville traffic network showing the source of congestion and the direction of traffic flow

II.5.1 Congestion Simulation

Figure II.10 depicts an instance of congestion simulation, where the vehicles at the target road R1 completely stop due to some incident. The graph shows how the effect of the congestion propagates backwards to affect all the incoming road segments of R1. Following the congestion at R1, the observed speed at its first hop neighbors R2 and R3 drops immediately; whereas the speed at its second hop neighbors R4 and R5 drops one minute later. Vehicle speed at the third hop neighbor R7 drops following the speed drop at R5.

We trained traffic speed predictors for each road segment using the data collected from SUMO. For training the predictors, we modeled the speed of each road segment as a function of its neighboring road segments, all working under normal operating conditions, so that each predictor learns how speed at the target road segment depends on its neighbors. Then, we tested whether they can predict the speed at a road segment based on the speed at its neighboring road segments under the influence of congestion.

II.5.2 Effect of Physical Incidents

In section II.4 we discussed anomaly detection when anomaly was introduced at a particular road segment whereas the neighboring road segments were working under normal operating conditions. So, the traffic predictors predicted the speed of the target road based

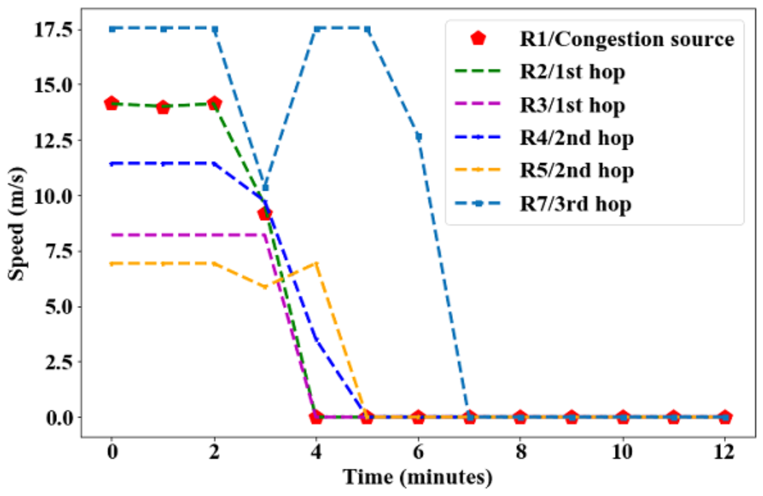

Figure II.10: Congestion instance: vehicles at target road R1 completely stop due to some incident.

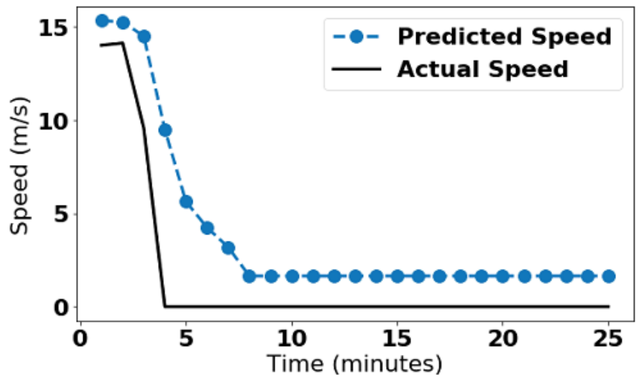

Figure II.11: Prediction result for 1st hop neighbor R2

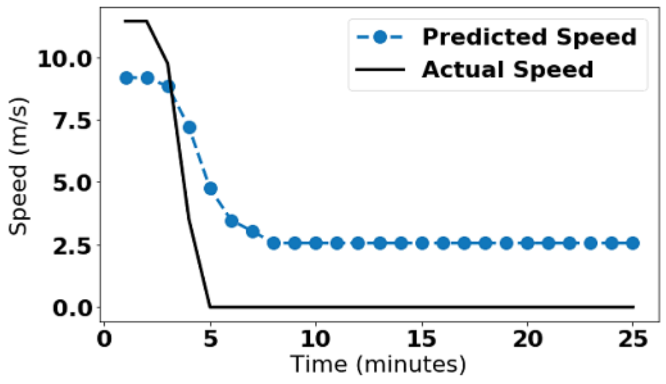

Figure II.12: Prediction result for 2nd hop neighbor R4

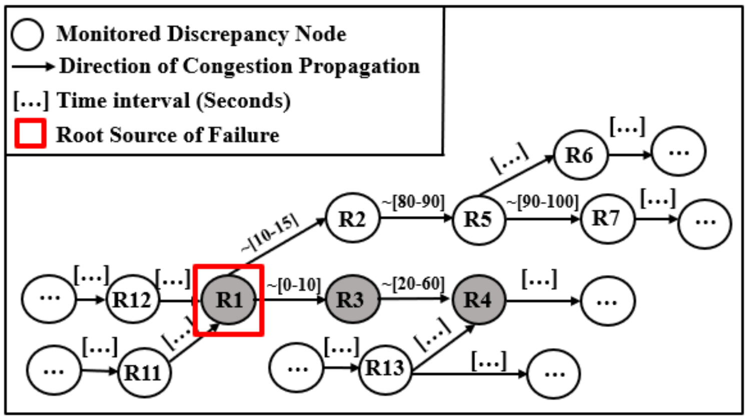
Monitored Discrepancy Node
Direction of Congestion Propagation
[...] Time interval (Seconds)
Root Source of Failure
R1
R2
R3
R4
R5
R6
R7
R11
R12
R13
~[10-15]
~[0-10]
~[20-60]
~[80-90]
~[90-100]
[...]

Figure II.13: TFPG model of congestion propagation

on the speed of the normally operating neighbors. As a result, the prediction result for the target road produced normal speed values as the output which deviated from the anomalous sensor readings showing large difference in the actual and predicted speed.

In case of a physical congestion in a road segment the traffic speed of the target road segment experiences a sudden decrease in speed while its neighbors are still operating under congestion free condition. So the prediction of speed for that road segment is off by some margin from the actual speed at that current time as the prediction is based on speed of the neighbors who are still working normally. Under this condition our LSTM based traffic predictors should raise an alarm due to the large deviation between actual and predicted speed. However, as time progresses and congestion propagates to the neighboring roads, the traffic predictor for the target road starts giving predicted result close to the actual decreased speed as the neighbors are also getting congested. Once the difference between the actual and predicted result goes down the alarm turns off[1]. Figures II.11 and II.12 show that the time at which the congestion started there is a large difference between actual and predicted speed and then the difference decreases with progression of time. We observe this sequence of alarms (as they turn on) for each road as a time series to hypothesize the source of the physical incident.

[1]Note that this is because the LSTM is predicting based on recent history

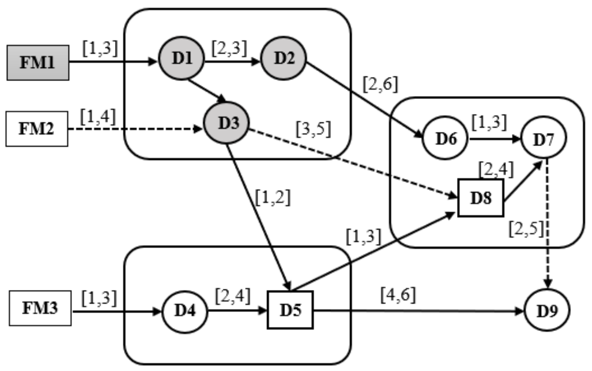

Figure II.14: An illustration of a TFPG Model with Failure Modes (FM), Discrepancies (D), and fault propagation links. Labels on edges indicate delay in (min,max) values.

II.5.3 Timed Failure Propagation Graph of Traffic Network

We can identify the source of congestion efficiently using a Timed Failure Propagation Graph (TFPG) (Abdelwahed et al., 2009). TFPGs capture the causality and temporal pattern of failure propagation in complex systems. A timed failure propagation graph (TFPG) is a labeled directed graph where nodes are either failure modes or discrepancies. Discrepancies are the failure effects, some of which may be observable. Edges in TFPG represent the causality of the fault propagation and edge labels capture operating modes in which the failure effect can propagate over the edge, as well as a time-interval by which the failure effect could be delayed (see figure II.14).

Figure II.13 shows a TFPG model capturing the propagation of congestion among the edges of the network described in Figure II.9. To create a TFPG model for the traffic network, we start with a directed graph of the traffic network, where each road segment in the network corresponds to a discrepancy node in the TFPG. The direction of the edges between the TFPG nodes is opposite to the direction of traffic flows in the traffic network since congestion propagates in the opposite direction of traffic flow. The TFPG is comprised of a non-empty set of discrepancy nodes (DN). Each edge e_{TFPG} in the TFPG model represents the direction of congestion propagation between two road segments with an approximate minimum $e_{TFPG}[tmin]$ to maximum $e_{TFPG}[tmax]$ time bound. The time for congestion propagation are subject to some fluctuations depending on specific time of days and other external factors. These time bounds are obtained from the simulation, which we set up by creating congestion scenarios in each edge of the network and calculating the time bounds within which the congestion propagates from one DN to other. All the discrepancy nodes in the TFPG are OR type as they are activated when the congestion propagates from any of their parent nodes within the specified time bound. Certain discrepancy nodes are consistently monitored, i.e., we have traffic predictors for this discrepancy nodes;

Note that monitoring all the discrepancy nodes in a large-scale traffic network is computationally expensive. There are various ways for selecting monitored nodes of a graph under the constraint of maximum number of allowed nodes that can be monitored and can be treated as an optimization problem. (Davis et al., 2016) discussed hill-climbing algorithm which starts with an initial seed node for placing the first monitor and goes on placing the next monitors on the highest degree neighbors. (Wijegunawardana et al., 2017) discussed strategies of monitor placement based on graph topology and colors of nodes. Other than some well-known monitor placement strategies such as smart random sampling, red score, most red neighbors, the authors proposed a learning based monitor assignment strategy. As there are numerous well-established methodologies for this problem, we do not discuss it any further.

II.5.4 Diagnosis

In a traffic network, congestion created at a source road segment propagates to its incoming neighbors. So if the root cause of an observed congestion at a certain road segment is to be found, then the root must lie within its k-hop outgoing neighbors in the traffic network. Note that the direction of traffic flow in the network is opposite to the direction of the congestion propagation shown in TFPG. Hence, once an alarm is observed from one of the monitored discrepancy node, a hypothesis is made such that the root failure node must lie within a subset of k-hop incoming discrepancy nodes in the TFPG. So, starting from a monitored alarm at a monitored discrepancy node, traverse through the TFPG, in a backward manner, and check if their corresponding alarms have been activated within the time range specified and go up to k-hop incoming discrepancy nodes, until the alarm at k-th hop discrepancy node is not activated but alarms till (k-1) th hop discrepancy nodes have been activated, so that we know that the source of congestion was at (k-1)th hop discrepancy node. At each hop, the subset of DNs whose alarms are not observed from the set of DNs at that hop are eliminated from the hypothesis set, so that the hypothesis set for finding the root of failure shrinks continuously and ultimately boils down to a single discrepancy node which is the source of congestion.

II.5.5 Case Study

Here we present a case study, where we try to find the source of congestion for road segment $R4$ (see Figure II.9) where an alarm has been observed in the corresponding DN after 5 minutes from start of simulation. For the root cause diagnosis we first check for its first hop incoming neighbors R3 and R13, out of which the alarm of R3 has been activated almost 60 seconds ago and the time bound for congestion propagation from $R3$ to $R4$ is (20-60) seconds as shown in the TFPG model in Figure II.13. However DN R13 is inactive following by the same logic. Next we check when the alarms of the immediate incoming neighbors of only R3 triggered, and find the alarm of R1 to be activated within specified time bound. Then we stop checking further as the alarms associated with none of the immediate incoming neighbors of R1 is activated, returning R1 as the source of congestion correctly.

II.6 Outcome

In this chapter methodologies for capturing neighborhood specific information for a large-scale connected cyber-physical system has been introduced. Architecture of Long Short Term Memory networks have been put up to capture both the spatial and temporal correlations of system variables to identify system anomalies. The cascading effect of traffic congestion has been simulated using a traffic simulator SUMO and its impact on the traffic speeds in the neighboring region of the source of congestion have been predicted. The most interesting contribution of this work lies in formulating the cascading effect of congestion propagation problem as a Timed Failure Propagation Graph. The source of congestion has been shown to be identified by traversing through the TFPG.

CHAPTER III

Analyzing the Cascading Effect of Traffic Congestion Using LSTM Networks

III.1 Problem Scope

In a large-scale interconnected system such as a traffic network, it is important to study the effect of cascading failures, where failure in one part of the system eventually triggers failure in other parts of the system. In this chapter we are trying to answer the second research question, that is how to predict anomaly propagation path in advance. For example, when a congestion is detected at a particular road segment of the traffic network can we predict how the congestion will propagate to the neighboring road segments and when. The work discussed in this chapter is published in (Basak et al., 2019a).

In past, traffic congestion prediction has been carried out in both model-driven and data-driven approaches. Model-driven approaches are based upon mathematical modelling to capture traffic congestion dynamics. For example, Fei et al. (Fei et al., 2017) modeled the traffic congestion inspired by shockwave theory, Arnott (Arnott, 2013) represented the network dynamics as a bathtub model. However, accurate modeling of the dynamic behavior of a complex system such as traffic networks using standard mathematical or statistical methods is a challenging task because the speed distributions in a large scale dynamic system like traffic network cannot be always modeled by predetermined distributions and all the modalities of such a dynamic and complex system cannot be captured (Ma et al., 2015b).

On the contrary, in case of data-driven approaches the complex functional relationships among several influencing factors can be learned by studying large amounts of data without relying on any standard and fixed statistical relation. Recent works on data driven approaches (Polson and Sokolov, 2017b), Ma et al. (2015b) in traffic prediction considered the traffic network as a homogeneous system and were mostly focused on applying

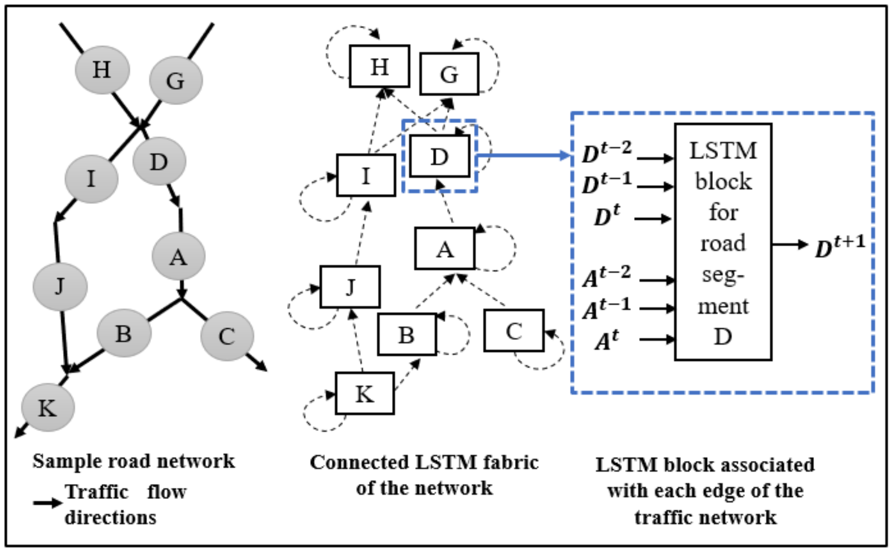
H
G
I
D
A
J
B
C
K
Sample road network
Traffic flow directions
H
G
D
I
A
J
B
C
K
Connected LSTM fabric of the network
D^{t-2}
D^{t-1}
D^{t}
A^{t-2}
A^{t-1}
A^{t}
LSTM block for road seg-ment D
D^{t+1}
LSTM block associated with each edge of the traffic network

Figure III.1: A sample road network and corresponding connected LSTM networks.

a generalized single architecture for the entire network at once ignoring road intersection-specific information. Thus it is difficult to capture the dynamically changing influence of each neighbor on a certain target road segment.

To address this gap, we developed a citywide congestion forecasting framework that works at a much higher granularity tailored towards capturing the specificity of each traffic intersections of the network. To develop such an integrated architecture we modeled the traffic network to a directed connected graph encapsulating the spatial interconnections where each neighbor of a road segment is a function of spatial distance as well as traffic flow directions. Along with modelling spatial dependencies, the temporal aspect of the traffic flow has been captured by multiple recurrent neural network architectures. Our approach has been validated with the real world traffic data from Nashville, USA collected from the HERE API.

Figure III.1 illustrates the representation of a sample road network with directions of traffic flow and its corresponding framework of the connected fabric of neural architectures. These neural modules associated with each and every edge of the network takes into account the information from itself and its outgoing neighbors for certain past sequences

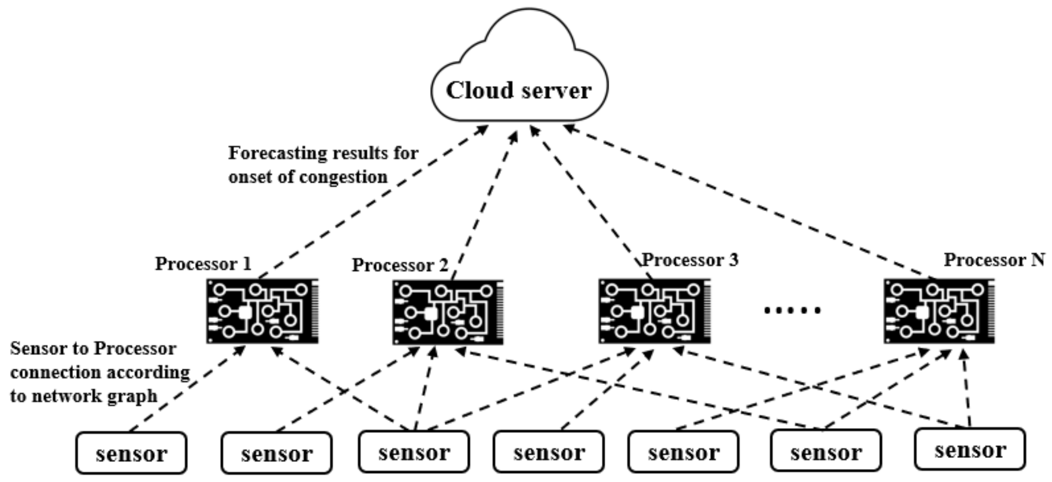

Figure III.2: Each computing processor associated with each road segment in the network collects the speed of the neighboring segments according to the graphical model of the network, process them to forecast the speed and send the results to a central cloud server which can be used for taking traffic routing decisions

upto current time to determine the future traffic state of the target edge. Figure III.2 shows a deployment diagram where each computing processor associated with each road segment in the network collects the traffic speed of the neighboring segments from the associated sensors according to the graphical model of the network, process them to forecast the speed of the target road segment and send the results to a central cloud server, which can be used for taking traffic routing decisions.

Prior research work has been based on training the network with several congestion specific incidents and learning from them (Pan et al., 2015) and applying the learnt models on similar traffic incidents in future. Here, we do not train our model on specific congestion incidents as being trained on specific incident data may limit the model's performance on similar situations only. Rather, we trained our architecture on all possible traffic conditions observed over the entire city for almost one and a half month and tailored our algorithms to identify the congestion propagation phenomenon from them.

Contributions Our contributions in this chapter are:

- We developed a city-wide connected congestion forecasting framework by incorporating intersection-specific information. We performed spatiotemporal modelling of the transportation network by expressing the network as a directed connected graph and used Long Short Term Memory (LSTM) networks to learn the distribution of the traffic speed of a target road in future as a function of the past sequences of observed speed of the target road and its immediate outgoing neighbors.

- We describe algorithms for identifying congestion events at any part of the network based on spatial and temporal correlations of the traffic speed at any road segment and its associated neighborhood. We also reduce the search space of the real time congestion forecasting algorithm by making it focus on intersections with a higher likelihood of congestion progression as learned from the historical data.
- The congestion forecasting framework has been validated by applying it on ten congestion events identified from the real traffic data of Nashville. We effectively identified the onset of congestion in each of the neighboring segments of the congestion source with an average precision of 0.9269 and an average recall of 0.9118 tested over those ten events.

This chapter solves the following problem: if a congestion event is observed at a certain road segment at any point of time in the transportation network, when does its effect propagate to its $k-hop$ incoming neighbors. We define the congestion event and the congestion cascade below.

III.1.1 Definitions

Definition 2 (Congestion Event (CE)) *A Congestion Event (CE) at an edge e is a tuple* $CE(e) = (t, s(e,t))$ *where* $s(e,t) \leq 0.6 * FF(e)$.

Xiong et al. (Xiong et al., 2018) used reduction of 50% speed compared to free flow speed as an indicator of congestion. We use the congestion criteria described as above.

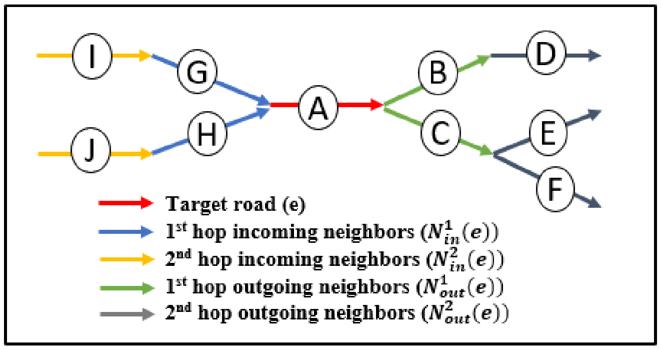
I
J
G
H
A
B
C
D
E
F
Target road (e)
1^{st} hop incoming neighbors ($N_{in}^1(e)$)
2^{nd} hop incoming neighbors ($N_{in}^2(e)$)
1^{st} hop outgoing neighbors ($N_{out}^1(e)$)
2^{nd} hop outgoing neighbors ($N_{out}^2(e)$)

Figure III.3: An illustrative representation of k-hop incoming and outgoing neighbors. In this figure, 'A' is the target road, 'G' and 'H' are its 1st hop incoming neighbors, 'I' and 'J' are its 2nd hop incoming neighbors, 'B' and 'C' are its 1st hop outgoing neighbors and 'D','E' and 'F' are its 2nd hop outgoing neighbors.

Definition 3 ($\Delta-$ Cascade Event) *The Delta Cascade Event is defined as a congestion event where more than 50% of first hop neighbors ($N_{in}^1(e)$) show 60% speed reduction with Δ time steps. We say e is the source of the cascade event.*

Figure III.3 provides an example of k-hop incoming and outgoing neighbors. Given a city network and the data collected from TMC segments our goal is to find the $\Delta-$ Cascade Events across the city and show that without training specifically on the cascade or congestion events we can identify the time of propagation of congestion up to the $k-hop$ incoming neighboring segments where k varies from one to three.

III.2 Related Work

Previous work on traffic congestion analysis have adopted several model-driven approaches. Fei et al. (Fei et al., 2017) proposed a time-variant model of congestion boundary founded on shockwave theory which is based on the analogy of traffic with the fluid flow. The parameters used in the model are specific to the design and features of a particular type of road segment and thus is difficult to be generalized to new traffic scenarios. Arnott (Arnott, 2013) modeled the traffic congestion as a bathtub model to analyze the traffic conditions in downtown areas during rush hour. The model compares the traffic inflow and outflow to the water flowing into and out of the tub with the height of water being proportional to the traffic density. The author proposed time-varying congestion pricing in situations where the demand is higher than the capacity. Long et al. (Long et al., 2008) proposed a congestion propagation framework inspired by the cell transmission phenomenon to identify network congestion bottleneck under various traffic demand scenarios.

Xiong et al. (Xiong et al., 2018) predicted congestion propagation patterns by constructing propagation graphs as a sequence of the traffic conditions of the road segments to identify in which of the roads the congestion will propagate from the source. Several other Deep Learning approaches (Sengupta et al., 2019c) are suitable to be applied to congestion forecasting. Zhang et al. (Zhang et al., 2019) carried out traffic congestion prediction by taking the snapshots of the traffic network as images and trained deep autoencoder architectures to predict the congestion levels. Polson et al. (Polson and Sokolov, 2017b) developed a Deep Learning architecture to predict traffic flows where the first layer identifies the spatiotemporal relations among predictors and the second layer models the non-linearities. They commented that the recent observations are stronger predictors than the historical values in predicting future traffic conditions.

Ma et al. (Ma et al., 2015b) used data driven techniques to analyze the congestion evolution in transportation network. They used conditional Restricted Boltzmann Machine (RBM) and Recurrent Neural Networks (RNN) to predict traffic congestion applied to Global Positioning System (GPS) data collected from taxi rides. The authors commented on the fact that although their prediction model performed well, in future they would like to explore the possibility incorporating spatial interactions among adjacent road segments in order to improve prediction accuracy. The use of recurrent neural networks specially Long Short Term Memory (LSTM) Networks for short term traffic volume prediction has also been evidenced in Zhao et al. (Zhao et al., 2017). Due to the exceptional capability of learning temporal sequence, LSTMs are used in various other domains including language learning (Gers and Schmidhuber, 2001), prognostics (Basak et al., 2019b) as well as traffic prediction. Tian et al. (Tian and Pan, 2015) also compared the traffic speed prediction performance by LSTM-RNN, with that of Support Vector Machine, Random Walk and Feed-forward neural networks and showed the supremacy of the LSTM-RNN model.

Past research works in traffic congestion forecasting using data driven approaches were contingent upon a single network approach where the entire information of the network state at any point of time is inputted and flattened as a vector. As a result, we lose the specific neighborhood information obtained from the network graph because the flattened vector does not incorporate the spatial closeness information along with the traffic data. Instead, in this work, we use multiple recurrent architectures with specific attention to each of the traffic channels in the network. Thus, our models are tailored towards capturing the specific dynamic relationships of any traffic channel and its neighbors which is not possible for a single neural network architecture for the entire city to provide the same level of resolution of encoding such inter-relationships.

III.3 Citywide Connected LSTM Fabric

LSTM is a form of recurrent neural network with the capability of processing sequences of data. It was proposed by Hochreiter and Schmidhuber (Hochreiter and Schmidhuber, 1997). LSTM prevents the vanishing and exploding gradient problem encountered in recurrent neural networks so that they are capable of capturing long temporal dependencies using backpropagation through time. They are used in this work to model the temporal dependencies of the traffic speed that will affect the speed in future. We build a connected LSTM based architecture that is intersection specific. To model the future speed of a particular TMC we use the information from its relevant neighboring segments. Now, in a transportation network traffic flows to a road from its incoming neighbor but congestion flows in a reverse direction of traffic flow, i.e., from an outgoing neighbor to a target road. As the congestion moves in a sequence, the speed forecasting detector for a target road is trained on the traffic data of the target road segment and its immediate outgoing neighbor, since congestion flows from an outgoing neighbor to a target road. In our previous work

(Basak et al., 2019), we used information from both the incoming and outgoing neighbors to model real-time traffic speed but as we are now concerned with predicting future traffic speed under the influence of congestion we use the information from the outgoing neighbors only.

The function of the traffic predictors for speed forecasting $\forall e \in TMC$ can be expressed as:

$$s(e)^{c_t+p} = f(\langle s(e) \rangle_{c_t-j}^{c_t}, \langle N_{out}^1(e) \rangle_{c_t-j}^{c_t}) \tag{III.1}$$

Where $s(e)$ denotes the speed of any TMC e, c_t denotes the current timestep, p is the number of timesteps we are going to predict ahead in future and j is the number of past timesteps to look back. So, future traffic states of the TMC $s(e)$, evaluated at current timestep c_t, has been modeled as a function (f) of traffic states of its own and its immediate outgoing neighbors' speed $(N_{out}^1(e))$ from timestep $(c_t - j)$ to c_t. The traffic predictors take into account the normalized speed data of each TMC, normalized w.r.t. the free flow speed. Each TMC in the network has such LSTM based traffic predictor associated with it. Figure III.1 shows an example of a sample road network and its corresponding connected fabric of LSTM.

Our approach is unique in the sense that it takes into account the information from neighbors to forecast the traffic speed of a target road. To solely analyze the importance and influence of neighboring road segments in determining the future traffic speed of a target road segment, we trained two simple feed-forward networks with same architecture, optimizer and loss functions. The first network is trained to forecast traffic speed using the information from the neighboring road segments and the second network is trained to forecast the traffic speed without using any information from neighbors. Figure III.4 shows the comparison of the mean squared errors (MSE) in forecasting the traffic speed over five randomly chosen TMC IDs. We observe that, given same architectural constraints the forecasts using the neighbors' information have far less MSE than the forecasts without using

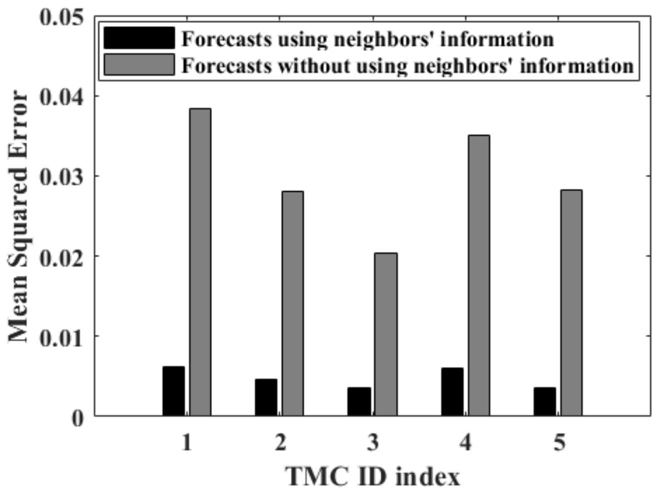

Figure III.4: Comparison of mean Squared error in traffic forecasting with and without using neighborhood information to solely evaluate the importance of using neighborhood information in the traffic prediction architecture.

the neighbor's information clearly indicating the need for using neighborhood information in traffic forecasting.

III.3.1 Selecting number of past observations

Selecting the number of past observations is an important hyperparameter to tune the LSTM models. We look back two past sequences of the traffic speed i.e., we look back into the past 20 minutes of the data for predicting the future traffic speed. Choosing longer time sequence doesn't improve performance in this case, because the future speed can be more closely approximated with speeds in recent history. Figure III.5 compares the mean squared errors (MSE) associated with different number of past observations taken into account while predicting the future traffic speed. It shows that MSE is not decreasing as we take more number of past data samples into account and is least when looking back for two timesteps. Hence, we choose the hyper-parameter representing the number of past observations as two.

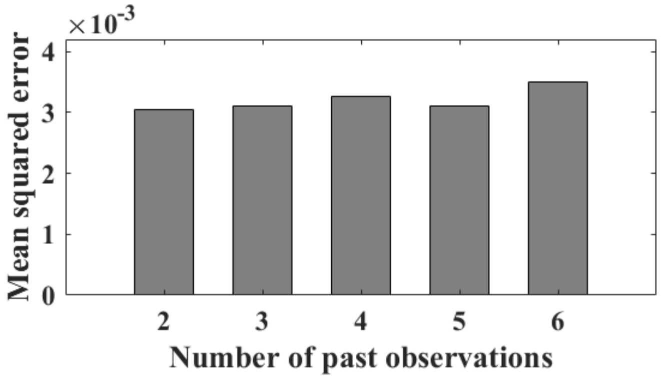

Figure III.5: Comparison of the mean squared errors of taking different number of past observations in predicting the future speed

III.3.2 Selecting the time resolution for the LSTM fabric

The timestep i.e., the interval at which the traffic data is discretely sampled is a critical hyperparameter. Figure III.6 ‘a’, ‘b’ and ‘c’ show a total of 500 minutes of data collected at an interval of 1 minute, 5 minutes and 10 minutes respectively. When we predict multiple timesteps ahead, the error in prediction increase gradually. If we choose data collected at one-minute time interval, then we need to predict 10 times to get a prediction after 10 minutes, which includes the error accumulated at each level of prediction. Instead if we choose data sampled at 10 minute time interval, then we just need to predict once to get a 10-minute ahead prediction, given we do not lose much information by sampling the data at 10 minute interval.

When plot ‘a’ is regenerated from plot ‘c’ by making each datapoint of plot ‘c’ represent same values for 10 corresponding samples of plot ‘a’, the mean squared error between the actual signal in plot‘a’ and the regenerated signal of plot ‘a’ from the downsampled version in plot ‘c’ is only 0.00138. Generally, for a normal data distribution, 95% of the data remain within two standard deviations from the mean. Also, there are no two consecutive datapoints in plot ‘c’, where the change in signal values is more than two standard deviations of the data samples in plot ‘a’. Hence we chose the timestep as 10 minutes for this work. Our predicted results using LSTMs with $timestep = 10$ are in multiples of 10 minute

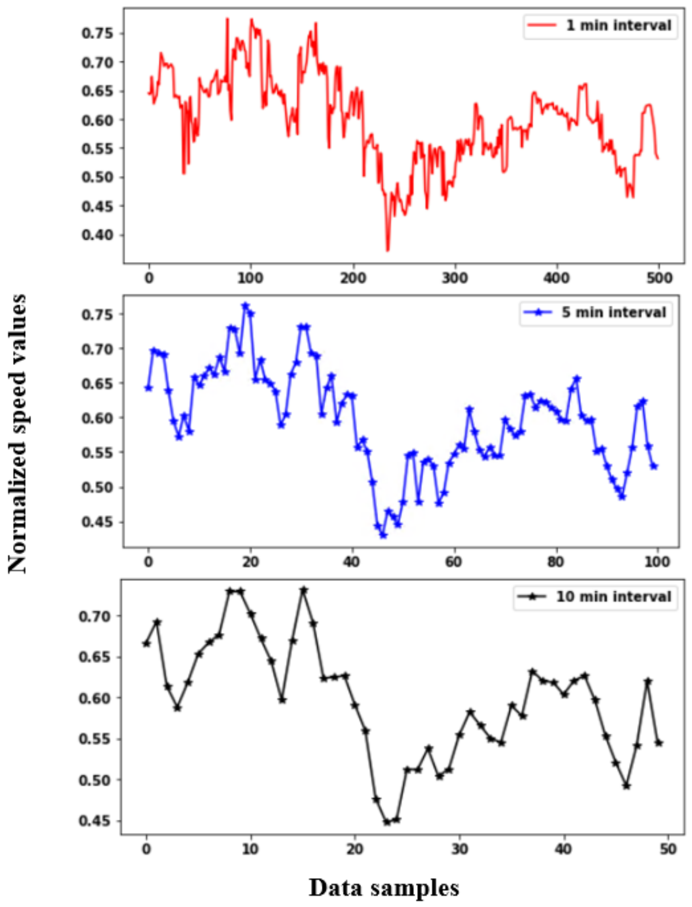

Figure III.6: Selecting hyper-parameter: time constant at which the data should be sampled.

time intervals. Later in this chapter we show how we fine-tune our solution to predict congestion times in multiple of 5 minute time intervals using LSTMs with $timestep = 5$.

III.3.3 LSTM architecture

We used a two-layered deep LSTM network for each traffic predictor with 100 units in each layer and a dense output layer. We used the mean squared error (MSE) between the predicted and actual speed as the loss function and the 'Adam' optimizer for optimizing the loss function.

III.3.4 Predicting multiple timesteps ahead

Using the connected LSTM fabric we can predict multiple timesteps ahead in future. As we want to predict ahead from current time, we require the information upto k-hop neighbors of a target road to predict the traffic speed for 'k' number of timesteps in advance. For example, a one-step ahead prediction requires the past and current traffic speed of the 1st hop neighbors, whereas, a two-step ahead prediction requires the one-step ahead predictions of the target road segment as well as that of the 1st hop neighbors to be treated as input. Now, the one-step ahead predictions of the 1st hop neighbors require the traffic information from their neighbors, i.e., the 2nd hop neighbors of the target road. So, for a two-step ahead prediction we need information upto 2nd hop neighbors.

Figure III.7 shows predictions upto three timesteps ahead in the future incorporating information upto 3rd hop neighbors following similar approach. The 0-th timestamp is the current time and we predict one, two and three timesteps ahead from the current time. This is how the connected fabric of LSTM architectures inter-dependently can produce multi-timestep ahead predictions. But the difference between actual and predicted speed while predicting three timesteps ahead is 1.3414 times more than that of two timesteps ahead and 2.6857 times more than that of one timestep ahead. So as we move further away in the future, the difference between the actual and predicted speed will increase as shown in Figure III.7.

III.4 Congestion Prognostics

In this section we describe our approach of building a congestion forecasting framework with an overall connected fabric of LSTM architectures.

III.4.1 Training Phase

We choose the data from 01.01.2018 to 01.27.2018 to train the traffic predictors for each TMC. We employ the LSTM fabric discussed in Section III.3 to train the network. The trained model for each TMC is saved which is used again in the congestion forecasting

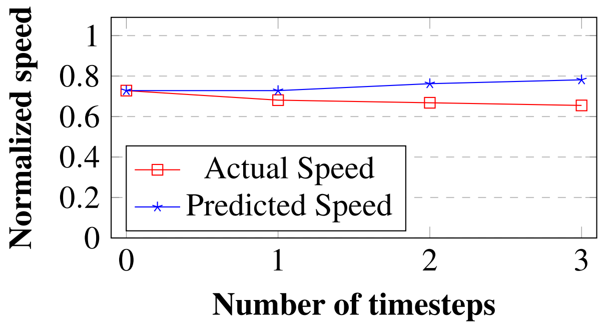

Figure III.7: Predicting normalized traffic speed of TMC '4424-0.12847' upto three timesteps, i.e., 30 minutes ahead from current time using the citywide connected LSTM fabric.

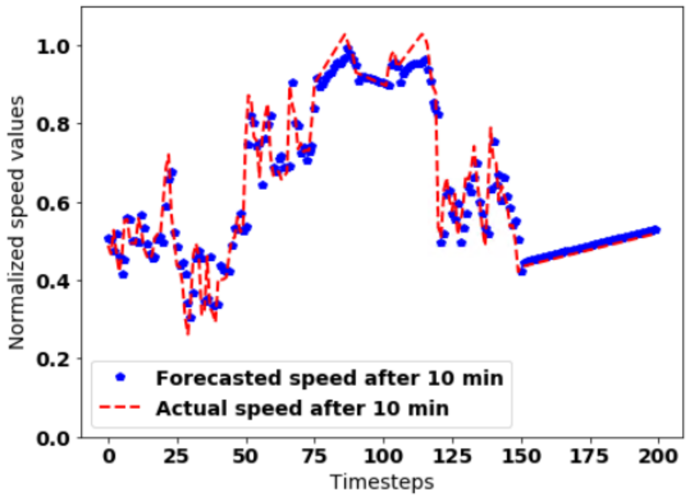

Figure III.8: Comparing the forecasted speed after 10 minutes and the actual observed speed after 10 minutes on a TMC ID having five neighbors.

phase. Figure III.8 shows an example of the traffic speed forecasting performance on a road segment having five neighbors. It shows that the forecasted speed after ten minutes and the actual speed after ten minutes overlap each other with a mean squared error of 0.0046.

In this work, we also compare the mean squared error (MSE) in predicting one vs. multiple timesteps ahead in future for multiple TMCs. Figure III.9 compares the mean squared errors in forecasting one, two, three and four timesteps ahead for 45 TMCs out of 3724 TMCs in Nashville and shows that the error increases vastly as we predict for times much ahead in the future.

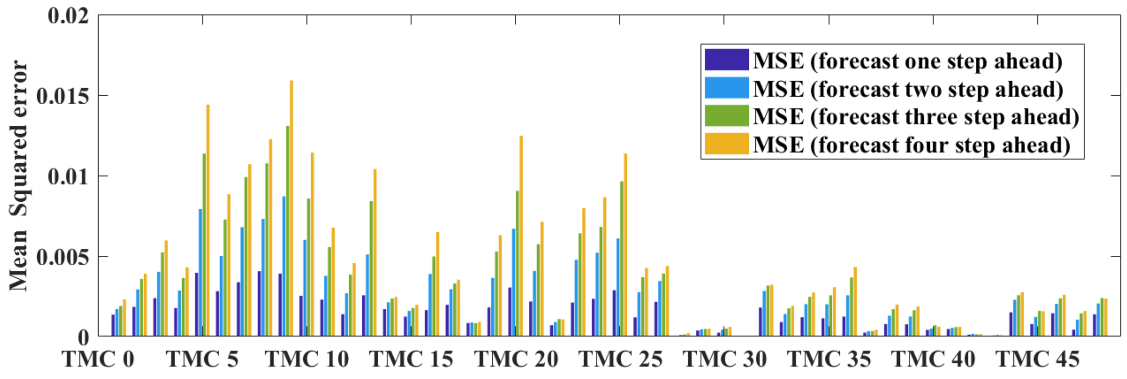

Figure III.9: Comparison of the mean squared errors among forecasting one, two, three and four timesteps ahead respectively for 45 TMC out of 3724 TMC in Nashville. The plot shows first 45 TMC only for brevity.

III.4.2 Congestion Forecasting Algorithm

Algorithm 1 illustrates the overall congestion forecasting architecture. Once congestion is identified at a target road segment the algorithm starts with gathering the 1st hop incoming neighbors $N_{in}^{1}(e)$, For each of those 1st hop neighbors, it finds the 2nd hop incoming neighbors denoted as $N_{in}^{2}(e)$. It repeats the process for 3rd hop incoming neighbors to find a set of it denoted as $N_{in}^{3}(e)$. These subsets of 1st, 2nd and 3rd hop neighbors constitutes the set of total neighbors denoted as $\tilde{N}$.

The function $predict\_next(e, timestep)$ calls the pre-trained LSTM forecasting module to predict the speed for a certain TMC edge e ($e \in TMC$) based on the values of its neighboring segments as discussed in equation III.1. It predicts the speed of edge e one time-step ahead in the future which is 10 minutes in this case. When the decrease in speed between two consecutive forecasts for a given tmc e is more than a detection threshold (δ) indicating a sudden and sharp drop in forecasted speed for the specified tmc, and the forecasted speed is less than or equal to 60% of the free-flow speed, the algorithm turns on the corresponding flag for the tmc e and forecasts a congestion to start at that tmc from the next timestep. So, the accuracy of this algorithm depends on the detection threshold δ indicating how much percentage of dip in forecasted speed from that of the previous timestep would trigger the initiation of congestion. Empirically we found that the algorithm works best

Algorithm 1: Algorithm to forecast congestion from a source up to its 3rd hop incoming neighbors

```
1: Input: Congestion event $CE$ at TMC $e$ at timestep $n$
2: $\tilde{N} = []$
3: $\tilde{N}.append(N_{in}^{1}(e))$
4: for each $i$ in $N_{in}^{1}(e)$ do
5:     $\tilde{N}.append(N_{in}^{2}(e))$
6:     for each $j$ in $N_{in}^{2}(e)$ do
7:         $\tilde{N}.append(N_{in}^{3}(e))$
8:     end for
9: end for
10: for timestep $n : n+10$ do
11:     $flag = zeros(length(\tilde{N}))$
12:     for each $i$ in $\tilde{N}$ do
13:         if $predict\_next(i,n-1) - predict\_next(i,n) \geq \delta * predict\_next(i,n-1)$ and
            $predict\_next(i,n) \leq 0.6 * FF(e)$ then
14:             $flag[i] = 1$
15:             Output $\tilde{N}$ will have onset of congestion at timestep $(n+1)$
16:         end if
17:     end for
18:     for each $j$ in flag do
19:         if $flag[j] = 1$ then
20:             $\tilde{N}.delete(\tilde{N}[i])$
21:         end if
22:     end for
23: end for
```

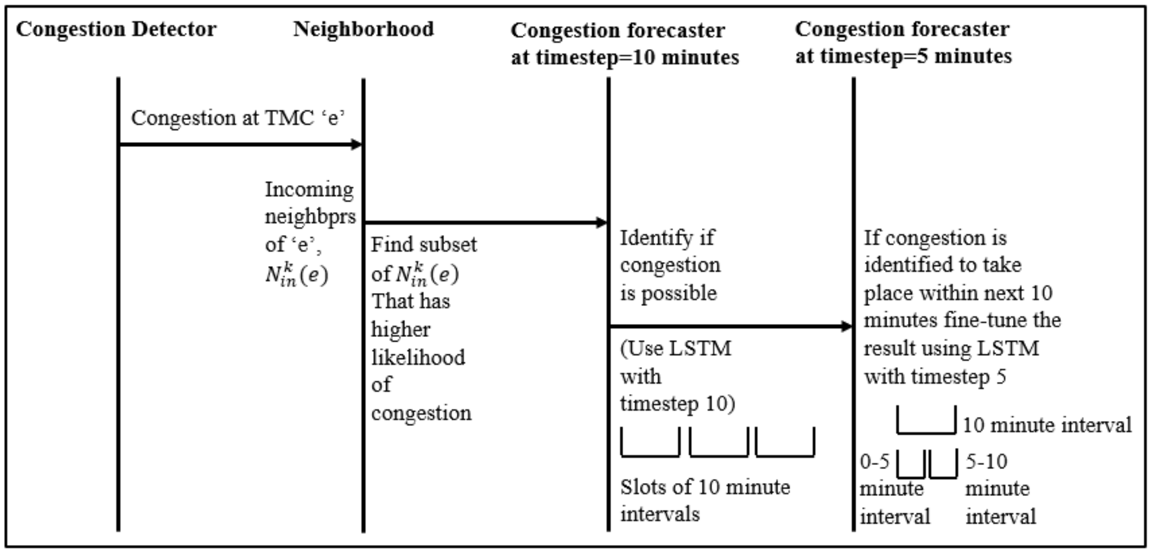
Congestion Detector
Neighborhood
Congestion forecaster at timestep=10 minutes
Congestion forecaster at timestep=5 minutes
Congestion at TMC 'e'
Incoming neighbprs of 'e', $N_{in}^{k}(e)$
Find subset of $N_{in}^{k}(e)$ That has higher likelihood of congestion
Identify if congestion is possible
(Use LSTM with timestep 10)
Slots of 10 minute intervals
If congestion is identified to take place within next 10 minutes fine-tune the result using LSTM with timestep 5
10 minute interval
0-5 minute interval
5-10 minute interval

Figure III.10: An illustration of the overall congestion forecasting framework

with a detection threshold between 0.1 to 0.15.

At each timestep the algorithm checks for the TMCs whose flags have been turned on and eliminates those from the list of $\tilde{N}$. In essence the algorithm starts with checking if a congestion is forecasted to start within the next 10 minutes for all the relevant 1st, 2nd and 3rd hop neighbors and then goes on eliminating the neighbors where congestion gets started. As in the 1st hop neighbors the congestion start earlier, they get eliminated from the list first, so that in the next timestep, the computation is carried out only for their corresponding 2nd and 3rd hop neighbors to output the corresponding time for onset of congestion for them.

This algorithm uses two sets of LSTMs we have. The LSTM with $timestep = 10$ uses the data collected at 10 minute intervals and predict time of onset of congestion at multiples of 10 minutes. Once congestion is forecasted within next ten minutes, the solution can be fine tuned by predicting whether the congestion will start within next 0 to 5 minutes or within next 5 to 10 minutes by using LSTMs with $timestep = 5$. This kind of prediction applies the same algorithm using the data sampled at 5 minute intervals.

Figure III.10 shows a diagram explaining the overall congestion forecasting framework.

III.4.3 Identifying likelihood of congestion propagation

Algorithm 2: Algorithm to identify which of the incoming neighbors of a congestion source have higher likelihood of congestion propagation. This algorithm is shown for the first hop neighbors but can also be applied to the second and third hop neighbors.

```
1: $\hat{cn}^1(e) = []$
2: for $i$ in $N_{in}^1(e)$ do
3:     $ev1 = 0$
4:     $ev2 = 0$
5:     if $s[e][n] < 0.6 * FF$ then
6:         $ev1 = ev1 + 1$
7:         $temp = 0$
8:         $speed\_array = s[i][n : n+4]$
9:         if any item in $speed\_array < 0.6 * FF$ then
10:            $temp = 1$
11:        end if
12:        if $temp = 1$ then
13:            $ev2 = ev2 + 1$
14:        end if
15:    end if
16:    $likelihood[i] = ev2/ev1$
17:    if $likelihood[i] > 0.5$ then
18:        $\hat{cn}^1(e).append(i)$
19:    end if
20: end for
```

Algorithm 2 aims to find out the likelihood of congestion propagation from a source road to a destination road. It identifies which of the incoming neighbors of a target road segment have higher likelihood of congestion propagation. By doing so, we can reduce the execution time of algorithm 1 by testing for onset of congestion for only those neighbors at each hop where the likelihood of congestion propagation are higher given historical records, instead of testing for congestion for all the incoming neighbors at each hop.

The algorithm keeps track of two kinds of events. Event *ev*1 corresponds to the phenomenon where a significant speed decrease is observed at any target road. Event *ev*2 corresponds to the phenomenon where a significant speed decrease is observed at any of its incoming neighbors within the time range of start time of congestion in target road, upto

Δ timesteps from that time. Δ is a heuristic and is chosen as 4 in this case with the assumption that a congestion if progresses from source to neighbor, should take place within 4 timesteps. The choice of Δ will vary according to the problem. For each neighbor the algorithm checks the number of times the event $ev1$ and $ev2$ occurred and saves the ratio of $ev2/ev1$ as $likelihood$ which signifies the proportion of times the congestion created at the source propagated to the corresponding neighbors.

From the historical observations this $likelihood$ of congestion propagation for each source destination pair can be found out and can be updated in real time, as more and more such cases are encountered. If the $likelihood$ is more than 50%, i.e., more than half of the times the congestion from source propagated to a particular neighbor given historical records, then this particular neighbor is appended to the set of most likely neighbors to be affected by congestion at source road e and this set is denoted as $\hat{cn}(e)$ of a road segment 'e'. The k hop $\hat{cn}(e)$ is denoted as $\hat{cn}^k(e)$, which indicates the subset of the neighbors of 'e' at $k-th$ hop that have higher likelihood of getting affected by the congestion at 'e', where k=1,2,3.

So, $\hat{cn}^1(e) \subset N^1_{in}(e)$, such that when we run our overall congestion forecasting algorithm described in Algorithm 1, we run the congestion forecasts for $\hat{cn}^1(e)$ only, instead of the whole set of $N^1_{in}(e)$. Thus we are reducing the execution time of the overall congestion forecasting algorithm by an order of $\hat{cn}^1(e)/N^1_{in}(e)$ for each of the road segments. Xiong et al. (Xiong et al., 2018) also used congestion propagation probabilities to construct propagation graphs from congestion matrices. But, in this work we use the likelihood of congestion propagation to save the time complexity of the overall congestion forecasting algorithm. However, our algorithm can still be applied to edges that are left out.

III.5 Validation

We validate our algorithm on the Nashville dataset. The data from January 28, 2018 to February 12, 2018 was used for validation purposes. The outline of this section is as fol-

Algorithm 3: Algorithm to identify $\Delta - Cascade - Event$ from Nashville traffic data

```
for each $e$ in $TMC$ list do
  for each timestep $n$ do
    if $(s[e][n]$ and $s[e][n+1]) < 0.6 * FF$ then
      for $i$ in $N^1_{in}(e)$ do
        $count = 0$
        $temp = 0$
        $s\_array = s[i][n:n+4]$
        if any item in $s\_array < 0.6 * FF$ then
          $temp = 1$
        end if
        if $temp = 1$ then
          $count = count + 1$
        end if
      end for
      if $count \geq 0.5 * |N^1_{in}(e)|)$ then
        Output: TMC $e$ has congestion at timestep $n$
      end if
    end if
  end for
end for
```

lows. We first identify the set of congestion events (definition 2). Then we discuss the results. We specifically look at one of the congestion events and show how we can further resolve the time to propagation to a 5 minute resolution.

III.5.1 Cascade Event Dataset

The procedure for finding the cascade events from validation dataset (see Algorithm 3) starts with checking for TMC IDs whose current speeds are less than 60% of the free flow speed (FF) for two consecutive timesteps n and $n+1$. Then for each of the incoming neighbors $N^1_{in}(e)$ for TMC e it checks their corresponding normalized speed from timestep n to $n+\Delta$. We select $\Delta = 4$ for this purpose as the hypothesis is if there is a congestion event that affects a neighborhood, then the congestion propagation between any two consecutive hops are within this Δ number of timesteps. The parameter Δ is just a heuristic here and will vary depending on the problem. If it detects congestion in any of the incoming neighbors

Table III.1: Summary of the congestion forecasting result for ten congestion events whose precision and recall values are shown in Figure III.11. The congestion sources, the date and time of onset of congestion at source, the actual and predicted times of onset of congestion at each of the neighbors. Note that there are multiple neighbor rows for the same congestion source, one for each incoming neighbor at that hop distance. Dashes indicate that there were no congestion events on the neighbors.

Index	Congestion source (ID)	Congestion source (Road name)	Date	Time	1-hop neighbors		2-hop neighbors		3-hop neighbors	
					Actual	Predicted	Actual	Predicted	Actual	Predicted
1	7413+3.57391	Hillsboro Pike	02.01.2018	16:30	16:40	16:40	-	-	-	-
2	4564+0.68565	I-24	01.30.2018	18:00	18:20	18:20	-	-	-	-
3	4418-0.94469	Charlotte Avenue	01.29.2018	16:20	16:30	16:30	16:40	16:40	-	-
4	4470+1.91003	I-24	02.02.2018	14:40	14:50	14:50	-	-	-	-
5	6847-1.51788	Memorial Boulevard	01.31.2018	15:00	15:00	15:10	15:30	15:20	-	-
							15:30	15:20		
6	6841+0.23911	South Church Street	02.09.2018	14:10	14:20	14:20	15:00	15:10	-	-
					14:50	15:00				
7	5041+1.16158	Dickerson Pike	01.30.2018	15:20	15:20	15:20	16:00	16:00	-	-
					15:20	15:20				
					15:50	16:00				
8	6017+0.46437	US 231	02.05.2018	06:30	06:50	06:50	07:40	07:30	-	-
					-	07:10	07:50	07:50		
					07:20	07:10				
9	8649-0.30317	West End Avenue	02.09.2018	10:40	11:00	11:00	10:40	10:50	-	-
					11:10	11:10	11:10	11:20		
							11:20	11:20		
10	13710-0.32285	21st Ave North	02.02.2018	06:50	06:50	06:50	07:00	07:00	07:00	07:00
									07:10	07:00
									07:30	07:30
							07:40	07:20	07:30	07:30
									07:20	07:20
									07:30	07:30

within this specified time range, it turns the flag *temp* on for that road as specified in Algorithm 3. After that the algorithm counts the number of times the flag *temp* turned on and sum them up. This count indicates how many incoming neighbors showed the sign of congestion within that time range. If more than or equal to 50% of the incoming neighbors showed the effect of congestion, then the algorithm classifies it as a congestion event and outputs the traffic network edge '*e*' has congestion at timestep *n*. The assumption here is, that not all of the neighbors necessarily need to be congested in a dynamic real-world traffic scenario. By identifying these cascaded congestion events, we are creating a validation set to verify the proposed congestion forecasting algorithm. We have identified ten such events from the Nashville dataset.

III.5.2 Congestion Progression Using 10 minute resolution LSTM

We validate our algorithms on a total of ten congestion events identified across Nashville. To give a more precise idea of the efficacy of the algorithm we calculated the corresponding precision and recall values in identifying the onset of congestion in each of the neighboring road segments. For each road segment we carried out an experiment for three consecutive timesteps including the actual time of onset of congestion and one timestep before and after that and classified whether the proposed algorithm outputted the presence of congestion or not for those timesteps and compared them with true conditions. When the onset of congestion is correctly identified, we consider it to be true positive. When the algorithm forecasts the onset of congestion before the actual onset, it is considered as false positive for those number of timesteps during when congestion was forecasted but was not actually present. When the algorithm forecasts the onset of congestion after the actual onset, it is considered to be false negative for those number of timesteps during when congestion was not forecasted but was actually present.

We test our algorithms only on neighbors that had higher likelihood congestion propagation as outlined in Algorithm 2. We present various scenarios where the congestion is confined within the 1st hop neighbors itself or affects a larger number of neighbors ranging upto the 3rd hop. Table III.1 summarizes the congestion forecasting results by comparing the actual and predicted time for onset of congestion w.r.t. the time of onset of congestion at source for each specific congestion event. It also reports the event index, TMC ID of the congestion source and the time of actual onset of congestion for the congestion source outputted by Algorithm 3 on which our approach was tested. It only shows the results for the neighbors where likelihood of congestion propagation was higher according to Algorithm 2. Figure III.11 shows the corresponding results of the precision and recall values in identifying the onset of congestion 10 minutes in advance in the neighboring segments and the corresponding number of neighbors that got affected by the congestion for ten different congestion events. The average precision and recall are obtained as 0.9269 and 0.9118 re-

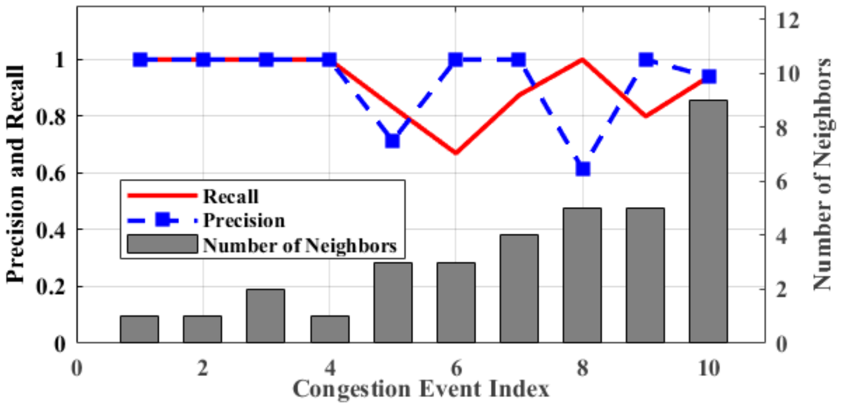

Figure III.11: Precision and recall values identifying the onset of congestion in all 1st, 2nd and 3rd hop neighboring segments of a congestion source tested over ten congestion events.

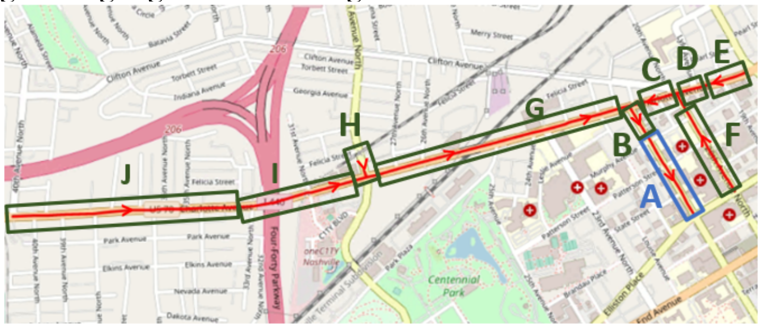

Figure III.12: Road segment for congestion event 10 in Table III.1. The source road of congestion is road segment ‘A’. Following the congestion at the source road segment, the congestion propagates to the 1st (‘B’), 2nd (‘C’, ‘G’) and 3rd hop (‘D’, ‘E’, ‘F’, ‘H’, ‘I’, ‘J’) incoming neighbors respectively.

spectively tested on these ten events. The variance of these precision and recall values are recorded as 0.02 and 0.0131 respectively.

When a congestion is predicted for a neighboring segment within next ten minutes, we fine tune our solution to identify whether the congestion will take place within next 0 to 5 minutes or next 5 to 10 minutes. Table III.2 summarizes the actual and predicted time of onset of congestion for all the neighbors using LSTM with $timestep = 5$. It refers back to the event index 10 in table III.1 and identifies whether the congestion is going to take place in the 0-5 minute time-slot or 5-10 minute time-slot. The average precision and recall values for identifying the onset of congestion in one of the two possible higher resolution time-slots are calculated as 0.75 and 0.92 respectively.

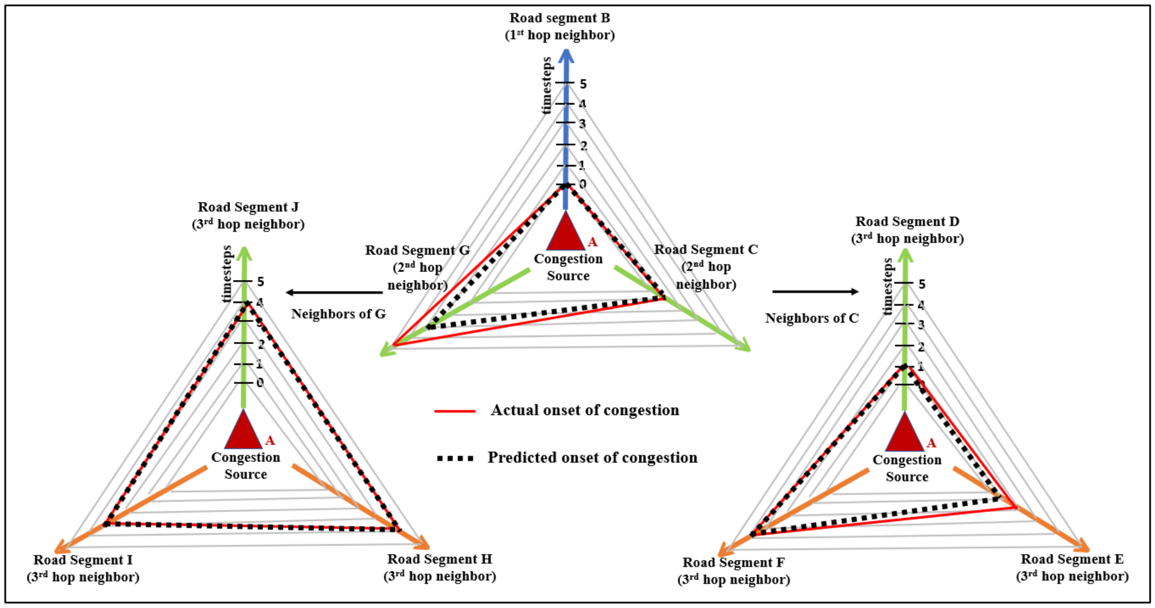

Figure III.13: Radar chart showing the accuracy of forecasting results applied to the for road section and congestion event shown in Figure III.12

Table III.2: The table shows the actual and predicted time for onset of congestion w.r.t. the time of onset of congestion at source at 5 minute resolution.

Neighbors of TMC ID '13710-0.32285'	Actual	Predicted
B	06:40-06:45	06:40-06:45
C	06:50-06:55	6:55-07:00
G	07:35-07:40	07:10-07:15
D	06:55-07:00	06:50-06:55
E	07:05-07:10	06:55-07:00
F	07:20-07:25	07:20-07:25
J	07:20-07:25	07:20-07:25
H	07:10-07:15	07:10-07:15
I	07:20-07:25	07:20-07:25

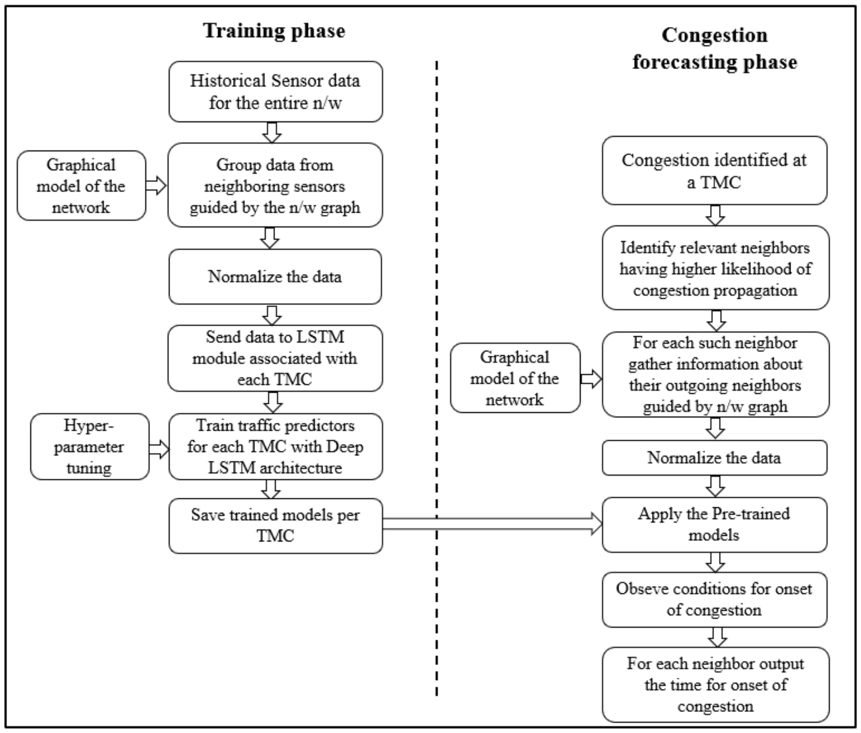
Training phase
Historical Sensor data for the entire n/w
Graphical model of the network
Group data from neighboring sensors guided by the n/w graph
Normalize the data
Send data to LSTM module associated with each TMC
Hyper-parameter tuning
Train traffic predictors for each TMC with Deep LSTM architecture
Save trained models per TMC
Congestion forecasting phase
Congestion identified at a TMC
Identify relevant neighbors having higher likelihood of congestion propagation
Graphical model of the network
For each such neighbor gather information about their outgoing neighbors guided by n/w graph
Normalize the data
Apply the Pre-trained models
Obseve conditions for onset of congestion
For each neighbor output the time for onset of congestion

Figure III.14: Workflow of the congestion forecasting framework.

III.5.3 Fine tuning progression results Using LSTM with timestep=5

Figure III.12 shows the transportation network for congestion event index 10 described in table III.1. For better understanding of the result on the cascaded congestion prediction using LSTM with $timestep = 10$ we present Figure III.13 which shows the effectiveness of the algorithm in identifying the onset of congestion in each of the neighbors through three different radar charts. The chart in the middle shows the results for one 1st and two 2nd hop neighbors. The radar charts on the left and the right shows the results for the 3rd hop neighbors corresponding to each of the 2nd hop neighbors. It is seen that the onset of congestion can be identified accurately most of the time.

Figure III.14 represents the overall workflow of the congestion forecasting framework.

III.6 Outcome

In this chapter we looked at how to predict anomaly propagation path in advance in a connected cyber-physical system. For this a city-wide ensemble of intersection level connected

Long Short Term Memory (LSTM) network models has been proposed to analyze how the congestion created at a s ource will affect its neighbors and in what order. We validated our congestion forecasting framework on the real world traffic data of Nashville, TN, USA and achieved an average precision of 0.9269 and an average recall of 0.9118 in identifying the onset of congestion for the neighboring road segments of congestion sources tested on ten congestion events.

CHAPTER IV

Conclusions and Future Work

In this thesis we demonstrated mechanisms for spatiotemporal modelling of networked Cyber Physical Systems. As an example of large scale cyber-physical system we worked on transportation networks and modeled it as a directed connected graph. We proposed a robust traffic speed prediction architecture by learning the distribution of traffic speed of a road segment as a function of its neighboring segments exploiting the graphical interdependence in multi-dimensional space. Experiments show that we could identify malicious traffic sensor attacks, with a precision-recall curve having AUC of 0.8507, demonstrating the effectiveness of the approach in anomaly detection. Next, we analyzed cascading effect of traffic congestion using a traffic simulator SUMO and predicted its impact on the traffic speeds in the neighboring region of the source of congestion. The most interesting contribution of this work lies in formulating the cascading effect of congestion propagation problem as a Timed Failure Propagation Graph. We identified the source of congestion traversing through the TFPG on observation of congestion at any edge of the traffic network.

We also developed a traffic congestion forecasting framework based on city-level connected LSTM networks. We took into account the likelihood of congestion propagation for each of the neighboring segments of any congestion source and identified the onset of congestion at each of them with an average precision of 0.9269 and an average recall of 0.9118 tested on ten congestion events. This approach serves the purpose of forecasting the onset of congestion in advance, so that traffic routing algorithms can divert the traffic away from the roads to be congested in near future. In future, we plan to extend this framework to predict cascading effects of failure in other networked systems such as electrical grids and water networks using similar approach. We expect to use this generalized spatio-temporal data analysis framework in various related applications aimed at real-time decision support.

BIBLIOGRAPHY

(Accessed on Aug 2020). Here api, https://developer.here.com/.

(Accessed on Aug 2020). Openstreetmap, https://www.openstreetmap.org.

Abdelwahed, S., Karsai, G., Mahadevan, N., and Ofsthun, S. C. (2009). Practical implementation of diagnosis systems using timed failure propagation graph models. *IEEE Transactions on Instrumentation and Measurement*, 58(2):240–247.

Arnott, R. (2013). A bathtub model of downtown traffic congestion. *Journal of Urban Economics*, 76:110 – 121.

Barros, J., Araujo, M., and Rossetti, R. J. F. (2015). Short-term real-time traffic prediction methods: A survey. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 132–139.

Basak, S., Ayman, A., Laszka, A., Dubey, A., and Leao, B. P. (2019). Data-driven detection of anomalies and cascading failures in traffic networks. In *Proceedings of the Annual Conference of the PHM Society*.

Basak, S., Dubey, A., and Bruno, L. (2019a). Analyzing the cascading effect of traffic congestion using lstm networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2144–2153.

Basak, S., Sengupta, S., and Dubey, A. (2019b). Mechanisms for integrated feature normalization and remaining useful life estimation using lstms applied to hard-disks. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 208–216.

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo - simulation of urban mobility: An overview. In *in SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68.

Daqing, L., Yinan, J., Rui, K., and Havlin, S. (2014). Spatial correlation analysis of cascading failures: Congestions and blackouts. In *Scientific reports*.

Davis, B., Gera, R., Lazzaro, G., Lim, B. Y., and Rye, E. C. (2016). *The Marginal Benefit of Monitor Placement on Networks*, pages 93–104. Springer International Publishing, Cham.

Deka, L., Khan, S. M., Chowdhury, M., and Ayres, N. (2018). 1 - transportation cyber-physical system and its importance for future mobility. In Deka, L. and Chowdhury, M., editors, *Transportation Cyber-Physical Systems*, pages 1 – 20. Elsevier.

Fei, W., Song, G., Zang, J., Gao, Y., Sun, J., and Yu, L. (2017). Framework model for time-variant propagation speed and congestion boundary by incident on expressways. *IET Intelligent Transport Systems*, 11(1):10–17.

Gers, F. A. and Schmidhuber, E. (2001). Lstm recurrent networks learn simple context-free and context-sensitive languages. *Trans. Neur. Netw.*, 12(6):1333–1340.

Ghafouri, A., Laszka, A., Dubey, A., and Koutsoukos, X. (2017). Optimal detection of faulty traffic sensors used in route planning. In *Proceedings of the 2Nd International Workshop on Science of Smart City Operations and Platforms Engineering*, SCOPE '17, pages 1–6, New York, NY, USA. ACM.

Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552.

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

Hu, J., Gao, P., Yao, Y., and Xie, X. (2014). Traffic flow forecasting with particle swarm optimization and support vector regression. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2267–2268.

Liang, Y., Jiang, Z., and Zheng, Y. (2017). Inferring traffic cascading patterns. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '17, pages 2:1–2:10, New York, NY, USA. ACM.

Long, J., Gao, Z., Ren, H., and Lian, A. (2008). Urban traffic congestion propagation and bottleneck identification. *Science in China Series F: Information Sciences*, 51(7):948.

Lu, X.-Y., Varaiya, P., Horowitz, R., and Palen, J. (2008). Faulty loop data analysis/correction and loop fault detection. In *15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual MeetingITS AmericaERTICOITS JapanTransCore*.

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

Ma, X., Yu, H., Wang, Y., and Wang, Y. (2015b). Large-scale transportation network congestion evolution prediction using deep learning theory. *PloS one*, 10:e0119044.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1-2):100–115.

Pan, B., Demiryurek, U., Gupta, C., and Shahabi, C. (2015). Forecasting spatiotemporal impact of traffic incidents for next-generation navigation systems. *Knowl. Inf. Syst.*, 45(1):75–104.

Polson, N. G. and Sokolov, V. O. (2017a). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17.

Polson, N. G. and Sokolov, V. O. (2017b). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1 – 17.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Sengupta, S., Basak, S., and Peters, R. A. (2018). Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, 1(1):157–191.

Sengupta, S., Basak, S., and Peters, R. A. (2019a). Chaotic quantum double delta swarm algorithm using chebyshev maps: Theoretical foundations, performance analyses and convergence issues. *Journal of Sensor and Actuator Networks*, 8(1).

Sengupta, S., Basak, S., Saikia, P., Paul, S., Tsalavoutis, V., Atiah, F., Ravi, V., and Peters, R. A. (2019b). A review of deep learning with special emphasis on architectures, applications and recent trends. *CoRR*, abs/1905.13294.

Sengupta, S., Basak, S., Saikia, P., Paul, S., Tsalavoutis, V., Atiah, F. D., Ravi, V., and Peters, R. A. (2019c). A review of deep learning with special emphasis on architectures, applications and recent trends. *ArXiv*, abs/1905.13294.

Tian, Y. and Pan, L. (2015). Predicting short-term traffic flow by long short-term memory recurrent neural network. In *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, pages 153–158. IEEE.

Wei, Y. and Li, S. (2015). Water supply networks as cyber-physical systems and controllability analysis. *IEEE/CAA Journal of Automatica Sinica*, 2:313–319.

Wijegunawardana, P., Ojha, V., Gera, R., and Soundarajan, S. (2017). Seeing red: Locating people of interest in networks. In Gonçalves, B., Menezes, R., Sinatra, R., and Zlatic, V., editors, *Complex Networks VIII*, pages 141–150, Cham. Springer International Publishing.

Xiong, H., Vahedian, A., Zhou, X., Li, Y., and Luo, J. (2018). Predicting traffic congestion propagation patterns: A propagation graph approach. In *Proceedings of the 11th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, IWCTS'18, pages 60–69, New York, NY, USA. ACM.

Yu, X. and Xue, Y. (2016). Smart grids: A cyber–physical systems perspective. *Proceedings of the IEEE*, 104(5):1058–1070.

Zhang, S. L., Yao, Y. Z., Hu, J., Zhao, Y., Li, S., and Hu, J. (2019). Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks. In *Sensors*.

Zhang, Y., Lu, Y., Lu, G., Chen, P., and Ding, C. (2015). Analysis of road traffic network cascade failures with coupled map lattice method. *Mathematical Problems in Engineering*, 2015:1–8.

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

Zygouras, N., Panagiotou, N., Zacheilas, N., Boutsis, I., Kalogeraki, V., Katakis, I., and Gunopulos, D. (2015). Towards detection of faulty traffic sensors in real-time. In *MUD @ ICML*, pages 53–62.