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

Sanchita Basak[1], Afiya Ayman[2], Aron Laszka[3], Abhishek Dubey[4], Bruno Leao[5]

[1,4] *Vanderbilt University, Nashville, Tennessee, 37235, USA*
sanchita.basak@vanderbilt.edu, abhishek.dubey@vanderbilt.edu

[2,3] *University of Houston, Houston, Texas, 77004, USA*
alaszka@uh.edu, aaayman@uh.edu

[5] *Siemens CT, USA*
bruno.leao@siemens.com

ABSTRACT

Traffic networks are one of the most critical infrastructures for any community. The increasing integration of smart and connected sensors in traffic networks provides researchers with unique opportunities to study the dynamics of this critical community infrastructure. Our focus in this paper is on the failure dynamics of traffic networks. By failure, we mean in this domain the hindrance of the normal operation of a traffic network due to cyber anomalies or physical incidents that cause cascaded congestion throughout the network. We are specifically interested in analyzing the cascade effects of traffic congestion caused by physical incidents, focusing on developing mechanisms to isolate and identify the source of a congestion. To analyze failure propagation, it is crucial to develop (a) monitors that can identify an anomaly and (b) a model to capture the dynamics of anomaly propagation. In this paper, we use real traffic data from Nashville, TN to demonstrate a novel anomaly detector and a Timed Failure Propagation Graph based diagnostics mechanism. Our 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. Our results show that our LSTM-based traffic-speed predictors attain an average mean squared error of 6.55×10^{-4} on predicting normalized traffic speed, while Gaussian Process Regression based predictors attain a much higher average mean squared error of 1.78×10^{-2}. We are also able to detect anomalies with high precision and recall, resulting in an AUC (Area Under Curve) of 0.8507 for the precision-recall curve. To study physical traffic incidents, we augment the real data with simulated data generated using SUMO, a traffic simulator. Finally, we analyzed the cascading effect of the congestion propagation by formulating the problem as a Timed Failure Propagation Graph, which led us in identifying the source of a failure/congestion accurately.

1. INTRODUCTION

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 paper, we study the failure dynamics of traffic networks, focusing on the detection and diagnostics of traffic anomalies based on traffic-prediction models. Traffic predictions can be performed based on two different approaches: model-driven and data-driven [Barros, Araujo, and Rossetti (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 [*HERE Developer* (2019)]. 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) ["SUMO" (2019)] 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, Karsai, Mahadevan, and Ofsthun (2009)].

Contributions Our contributions in this paper 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.

Outline The rest of the content is organized as follows. Section 2 sets up the research problem that we solve. We provide an outline of the research approach and compare it to related works in Section 3. Next, we describe our main contributions. Section 4 presents the models that we use for traffic speed prediction. We discuss our approach to anomaly detection and its comparison with the classical Gaussian Process Regression based anomaly detection in Section 5. Then, we discuss the cascade analysis approach and root cause isolation in Section 6. Section 7 concludes the paper and discusses future research directions.

2. PROBLEM STATEMENT

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. To help setup our problem, we first present a number of definitions, which include the transportation network as a graph and certain operators on the graph that we use later in the paper.

2.1. Definitions

Definition 1 (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.*

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

Definition 3 (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 $|out(v)|$.*

Definition 4 (Traffic Message Channels (TMC)) *An edge is called a* traffic message channel *(TMC) if it has a speed*

and jam factor sensor associated with it. We denote the set of TMC as $E^S \subseteq E$.

Definition 5 (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.*

Definition 6 (k-hop incoming neighbors) *The* k-hop incoming neighbors *($\mathcal{N}_{in}^k : E \rightarrow 2^E$) of an edge are the k-nearest hops of the incoming edges that can feed the traffic into an edge. We define this function recursively. $\mathcal{N}_{in}^1(e) = \bigcup_{x \in in(src(e))}(in(src(x)))$. Given the set $\mathcal{N}_{in}^{k-1}$, the set $\mathcal{N}_{in}^k$ can be defined as $\mathcal{N}_{in}^k(e) = \bigcup_{x \in \mathcal{N}_{in}^{k-1}(e)}(in(src(x)))$.*

Definition 7 (k-hop outgoing neighbors) *The* k-hop outgoing neighbors *($\mathcal{N}_{out}^k : E \rightarrow 2^E$) to an edge are the k-nearest hops of edges that take traffic away from an edge via its out node. We define this function recursively as well. $\mathcal{N}_{out}^1(e) = \bigcup_{x \in out(dst(e))}(out(dst(x)))$. Given the set $\mathcal{N}_{out}^{k-1}$, the set $\mathcal{N}_{out}^k$ can be defined as $\mathcal{N}_{out}^k(e) = \bigcup_{x \in \mathcal{N}_{out}^{k-1}(e)}(out(dst(x)))$.*

Definition 8 (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.*

Definition 9 (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.*

Definition 10 (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 paper. The greater the AUC of the PR curve, the better is the detection model.*

2.2. Problem Definition and Dataset

In this paper, 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 paper, 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.

3. RELATED WORK

The approach that we will describe later in this paper 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. In this paper we make the assumption of single root failure (physical incident). The model of computation we use and describe later in this paper can support multi-failure hypothesis. But we leave that for our future work.

3.1. Existing Work on Traffic Forecast with Machine Learning

Ma, Tao, Wang, Yu, and Wang (2015) 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 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 and Sokolov (2017) 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, Basak, Saikia, et al. (2019). Prior work on traffic forecasting has also been carried out with multi agent based approaches. Hu, Gao, Yao, and Xie (2014) used Particle Swarm Optimization for traffic flow prediction. Some recent swarm-based algorithms listed in [Sengupta, Basak, and Peters (2019), Sengupta, Basak, and Peters (2018)] can also be used in this purpose.

For short term traffic volume forecasting, Zhao, Chen, Wu, Chen, and Liu (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, Sengupta, and Dubey (2018)].

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.

3.2. Existing Work on Traffic Anomaly Prediction

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; Pearsons 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, Varaiya, Horowitz, and Palen (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.

3.3. Existing Work on Cascading Failures

Daqing, Yinan, Rui, and Havlin (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, Lu, Lu, Chen, and Ding (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, Jiang, and Zheng (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 paper, other than analyzing the cascading effect of traffic congestion on the neighboring road segments of the network, we show that the source of congestion can be isolated by formulating the congestion propagation problem as a Timed Failure Propagation Graph.

4. 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* $\times$ *traf-*

fic 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 11). 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 11 (Normalized speed) *The normalized speed of a TMC (definition 4) is a ratio of its current speed with the average of speeds for times when the jam factor (definition 5) is zero.*

For each TMC, $\mathcal{N}^1_{in}(TMC)$ and $\mathcal{N}^1_{out}(TMC)$ 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 1 shows the average loss on test data calculated over normalized speeds for different timesteps produced by RNN and LSTM.

Table 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 $(Samples \times timesteps \times features)$ 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).

4.1. 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 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.

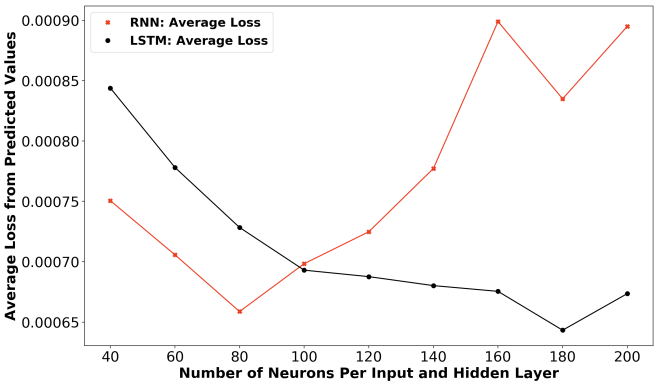

Figure 1. Average loss from predicted speed values by RNN and LSTM for different number of neurons

We have used *Mean Squared Error* as the loss function for RNN. For training the deep neural model, we have used *Adam* optimizer. Figure 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}.

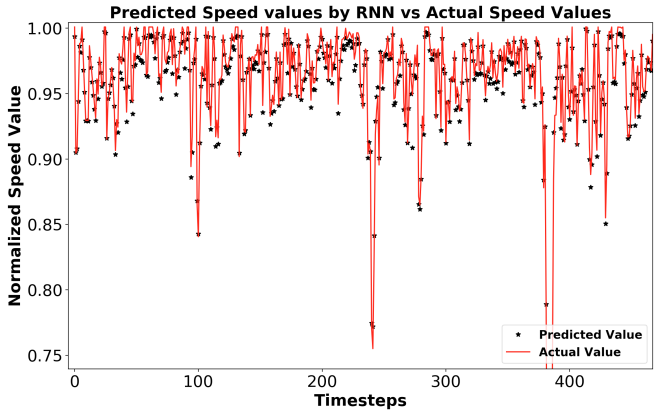

Figure 2. Predicted speed values by RNN vs actual speed values for the test data of first TMC

4.2. 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 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 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.

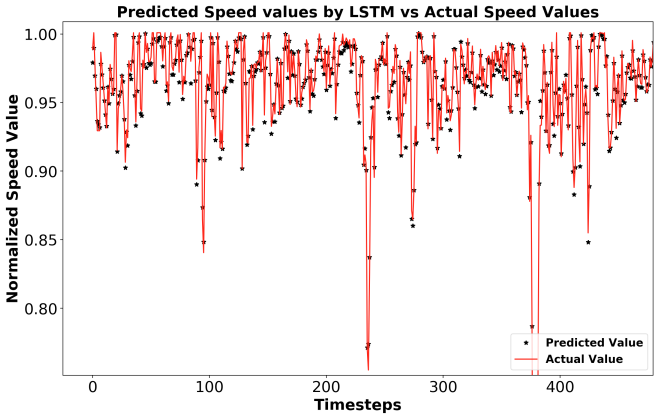

Figure 3. Predicted speed values by LSTM vs actual speed values for the test data of the first TMC

Figure 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}.

4.3. 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 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 best for this dataset.

4.4. 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

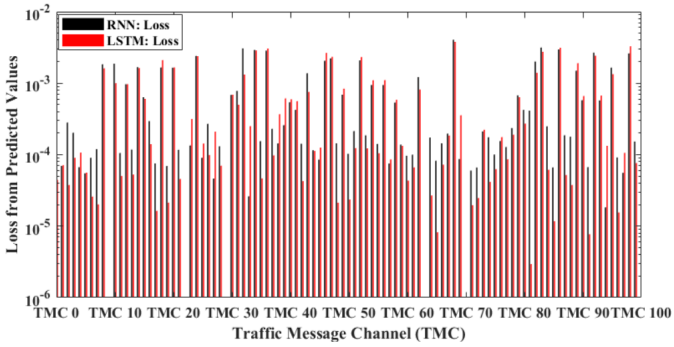

Figure 4. Loss from predicted speed values by LSTM and RNN for the first 100 TMC (difference is shown in Log scale)

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, Laszka, Dubey, and Koutsoukos (2017)]. In this work, we have used Radial Basis Function (RBF) as the kernel. Figure 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.

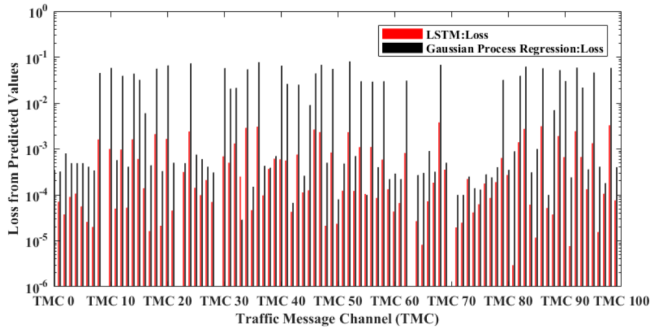

Figure 5. Comparison of LSTM and Gaussian Process Regression based on predicting the speed of a chosen TMC

5. 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 6 shows and example of differences between the predicted and the actual real-time sensor measurements during an additive sensor attack.

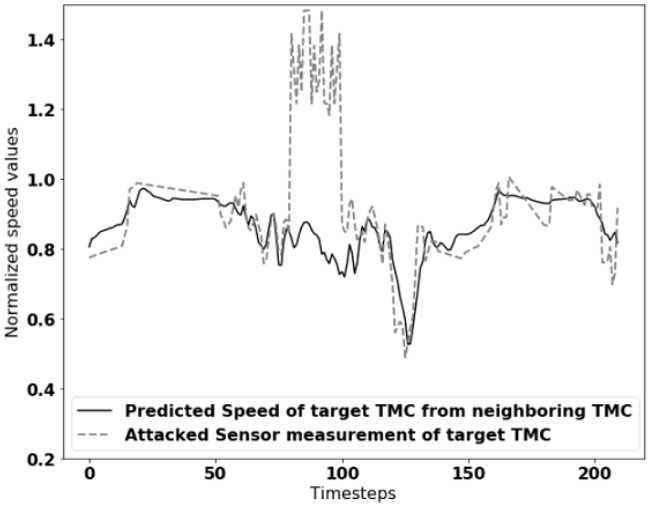

Figure 6. Introducing additive anomaly into sensor readings of a TMC

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:

$$usum_s^t = \max\{0, usum_s^{t-1} + x_s^t - \mu - k\} \quad (1)$$

$$lsum_s^t = \min\{0, lsum_s^{t-1} + x_s^t - \mu + k\} \quad (2)$$

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.

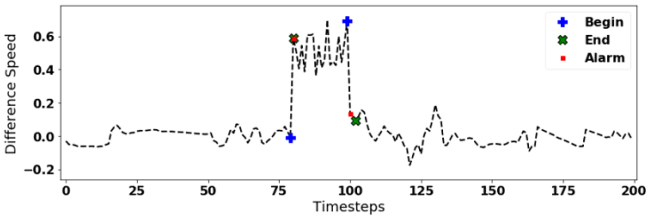

Figure 7. Detection of anomaly through CUSUM algorithm

Figure 7 shows the detection of anomaly for the case described in Figure 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.

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 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 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 6.

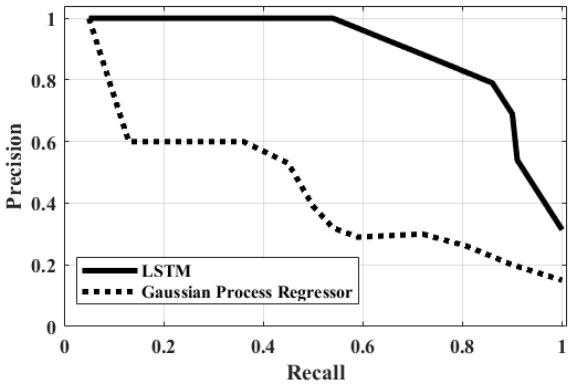

Figure 8. Precision Recall curves of LSTM and Gaussian Process Regression showing their comparative efficiency in identifying anomaly

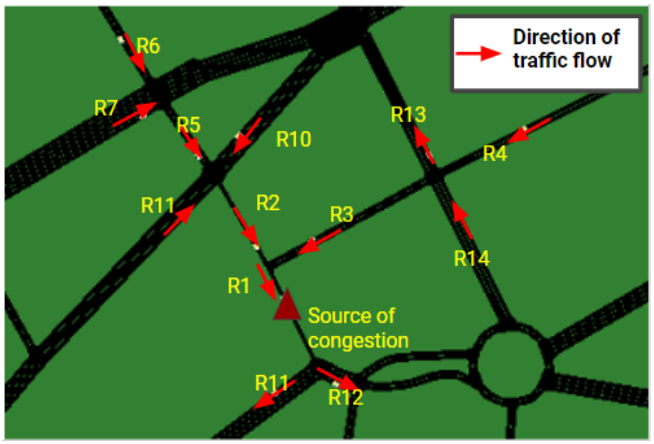

Figure 9. Part of the Nashville traffic network showing the source of congestion and the direction of traffic flow

6. 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 paper, 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 ["OpenStreetMap" (2019)]. Figure 9 shows the part of the road network that we used in our simulations. For our experiments, we introduced congestion at road segment R1.

6.1. Congestion Simulation

Figure 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.

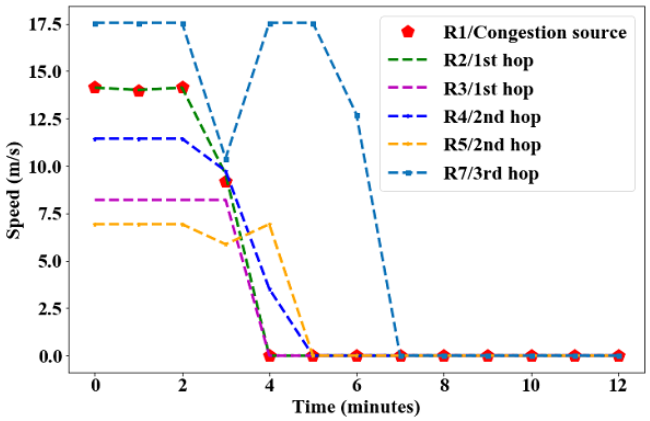

Figure 10. Congestion instance: vehicles at target road R1 completely stop due to some incident.

6.2. Effect of Physical Incidents

In section 5 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 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

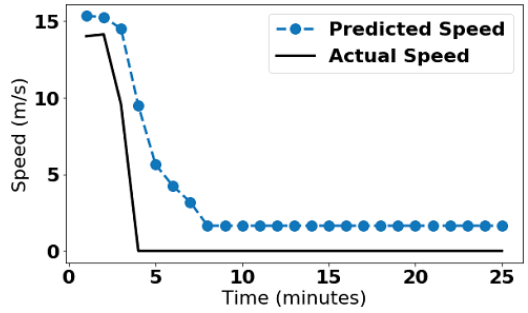

Figure 11. Prediction result for 1st hop neighbor R2

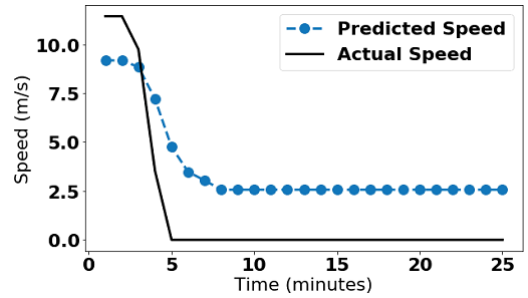

Figure 12. Prediction result for 2nd hop neighbor R4

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 11 and 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.

6.3. 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

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

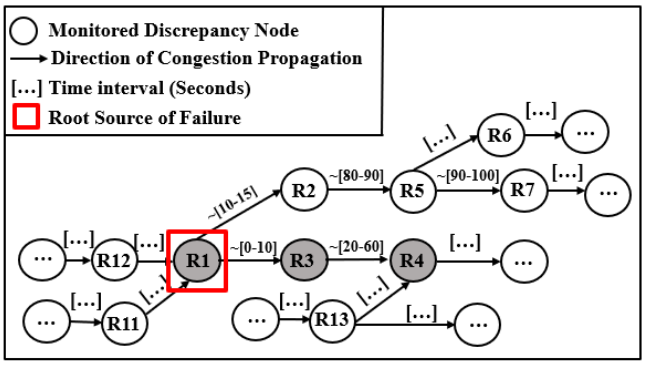

Figure 13. TFPG model of congestion propagation

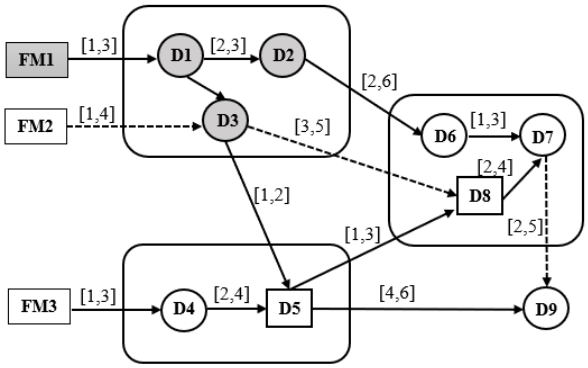

Figure 14. An illustration of a TFPG Model with Failure Modes (FM), Discrepancies (D), and fault propagation links. Labels on edges indicate delay in (min,max) values. For a detailed understanding please refer to Abdelwahed et al. (2009).

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 14).

Figure 13 shows a TFPG model capturing the propagation of congestion among the edges of the network described in Figure 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 simula-

tion, 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, Gera, Lazzaro, Lim, and Rye (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, Ojha, Gera, and Soundarajan (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.

6.4. 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.

6.5. Case Study

Here we present a case study, where we try to find the source of congestion for road segment $R4$ (see Figure 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 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.

7. CONCLUSIONS AND FUTURE WORK

We proposed a traffic prediction model considering a large-scale traffic network as a connected directed graph and compared two machine learning approaches, of which LSTM performed the best with an average loss of 6.55×10^{-4} on Nashville traffic data. We employed CUSUM along with the trained traffic predictor models to identify malicious sensor attacks, which achieved a precision-recall curve with AUC 0.8507, demonstrating the effectiveness of the approach in anomaly detection. Next, we analyzed cascading effect of traffic congestion using a traffic simulator and predicted its impact on the traffic speeds in the neighboring region of the source of congestion. The most interesting contribution of this paper 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. In future work, we will analyze cascading failures in other large-scale coupled systems, such as electrical grids and water networks, and identify the sources of failures using approaches that are similar to the ones introduced in this paper.

ACKNOWLEDGEMENTS

This research is funded in part by a grant from Siemens, CT and the following grants from National Science Foundation: 1818901 and 1647015.

REFERENCES

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

10.1109/TIM.2008.2005958

Barros, J., Araujo, M., & Rossetti, R. J. F. (2015, June). Short-term real-time traffic prediction methods: A survey. In *2015 international conference on models and technologies for intelligent transportation systems (mtits)* (p. 132-139). doi: 10.1109/MTITS.2015.7223248

Basak, S., Sengupta, S., & Dubey, A. (2018). Mechanisms for integrated feature normalization and remaining useful life estimation using lstms applied to hard-disks..

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

Davis, B., Gera, R., Lazzaro, G., Lim, B. Y., & Rye, E. C. (2016). The marginal benefit of monitor placement on networks. In H. Cherifi, B. Gonçalves, R. Menezes, & R. Sinatra (Eds.), *Complex networks vii: Proceedings of the 7th workshop on complex networks complenet 2016* (pp. 93–104). Cham: Springer International Publishing.

Gers, F. A., & Schmidhuber, E. (2001, November). Lstm recurrent networks learn simple context-free and context-sensitive languages. *Trans. Neur. Netw.*, *12*(6), 1333–1340. Retrieved from https://doi.org/10.1109/72.963769 doi: 10.1109/72.963769

Ghafouri, A., Laszka, A., Dubey, A., & 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* (pp. 1–6).

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

Here developer. (2019). Retrieved from https://developer.here.com/

Hu, J., Gao, P., Yao, Y., & Xie, X. (2014, Oct). Traffic flow forecasting with particle swarm optimization and support vector regression. In *17th international ieee conference on intelligent transportation systems (itsc)* (p. 2267-2268). doi: 10.1109/ITSC.2014.6958049

Liang, Y., Jiang, Z., & Zheng, Y. (2017). Inferring traffic cascading patterns. In *Proceedings of the 25th acm sigspatial international conference on advances in geographic information systems* (pp. 2:1–2:10).

Lu, X.-Y., Varaiya, P., Horowitz, R., & 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., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, *54*, 187–197.

Openstreetmap. (2019). Retrieved from https://www.openstreetmap.org/

Page, E. S. (1954, 06). Continuous inspection schemes. *Biometrika*, *41*(1-2), 100-115. doi: 10.1093/biomet/41.1-2.100

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

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning (adaptive computation and machine learning)*. The MIT Press.

Sengupta, S., Basak, S., & 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. Retrieved from https://www.mdpi.com/2504-4990/1/1/10 doi: 10.3390/make1010010

Sengupta, S., Basak, S., & Peters, R. A. (2019). Chaotic quantum double delta swarm algorithm using chebyshev maps: Theoretical foundations, performance analyses and convergence issues. *Journal of Sensor and Actuator Networks*, *8*(1). Retrieved from https://www.mdpi.com/2224-2708/8/1/9 doi: 10.3390/jsan8010009

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

Sumo. (2019). Retrieved from http://sumo.sourceforge.net/

Tian, Y., & 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)* (pp. 153–158).

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

Zhang, Y., Lu, Y., Lu, G., Chen, P., & Ding, C. (2015). Analysis of road traffic network cascade failures with coupled map lattice method..

Zhao, Z., Chen, W., Wu, X., Chen, P. C., & 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., & Gunopulos, D. (2015). Towards detection of faulty traffic sensors in real-time. In *MUD @ ICML* (pp. 53–62).

BIOGRAPHIES

Sanchita Basak is a graduate student in the Department of Electrical Engineering and Computer Science at Vanderbilt University and a research assistant at the Institute for Software Integrated Systems. She is currently working towards her Ph.D. at Vanderbilt University. Her current research interests include Neural Networks, Machine Learning, Big Data and Multiagent optimization with applications in smart city planning. She received her M.Tech degree in Electrical and Electronics Communication Engineering from Indian Institute of Technology, Kharagpur in India in May 2017 and B.Tech. degree from West Bengal University of Technology, India in 2015.

Afiya Ayman is a graduate student working towards her Ph.D. in the Department of Computer Science at the University of Houston. She is currently working as a research assistant with Dr. Aron Laszka in the Resilient Networks and Systems lab at UH. Her research interests include the application of Neural Networks and Machine Learning for solving security problems. She completed her undergraduate study in Computer Science and Engineering from Chittagong University of Engineering and Technology, Bangladesh in October 2015.

Dr. Aron Laszka is an Assistant Professor in the Department of Computer Science at the University of Houston. His research interests broadly revolve around the security and resilience of cyber-physical systems and the Internet of Things, the economics of cyber-security, and using game theory and artificial intelligence to solve security problems. Previously, he was a Research Assistant Professor at Vanderbilt University from 2016 to 2017, a Postdoctoral Scholar at the University of California, Berkeley from 2015 to 2016, and a Postdoctoral Research Scholar at Vanderbilt University from 2014 to 2015. He graduated summa cum laude with a Ph.D. in Computer Science from the Budapest University of Technology and Economics in 2014. In 2013, he was a Visiting Research Scholar at the Pennsylvania State University.

Dr. Abhishek Dubey is an Assistant Professor of Electrical Engineering and Computer Science at Vanderbilt University, Senior Research Scientist at the Institute for Software-Integrated Systems and co-lead for the Vanderbilt Initiative for Smart Cities Operations and Research (VISOR). His research interest is secure and resilient operation of cyber-physical systems with an emphasis on transportation and power networks. He is a senior member of IEEE and published several peer-reviewed articles. His key contributions include a robust software model for building cyber-physical applications, along with spatial and temporal separation among different system components, which guarantees fault isolation. He completed his PhD in Electrical Engineering from Vanderbilt University in 2009. He received his M.S. in Electrical Engineering from Vanderbilt University in August 2005 and completed his undergraduate studies in electrical engineering from the Indian Institute of Technology, Banaras Hindu University, India in May 2001.

Dr. Bruno Paes Leao is a Data Scientist with Siemens, CT, Princeton, New jersey. He obtained his D.Sc. in Electronics Engineering and Computer Science from ITA, Brazil in 2011, M.Eng. in Aeronautical and Mechanical Engineering from ITA, Brazil in 2006 and B.S. in Control and Automation Engineering from UFMG, Brazil in 2003.