Analyzing the Cascading Effect of Traffic Congestion Using LSTM Networks

Sanchita Basak, Abhishek Dubey, Bruno P. Leao

***Abstract*—This paper presents a data-driven approach for predicting the propagation of traffic congestion at road segments as a function of the congestion in their neighboring segments. 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 LSTM 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, USA and identified the onset of congestion in each of the neighboring segments of any congestion source with an average precision of 0.9269 and an average recall of 0.9118 tested over ten congestion events.**

***Index Terms*—Cascades, Long Short Term Memory, traffic congestion, network graph, Forecasting,**

I. Introduction

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. A primary road congestion created at a source can trigger secondary and tertiary road congestion due to physical connectivity. It can cause severe operational problems including traffic delays and waste of time and energy. To mitigate such effects and build effective route guidance systems it is necessary to forecast the propagation of congestion in advance to predict when the neighboring road segments of a congestion source will be affected in the near future.

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. [1] modeled the traffic congestion inspired by shockwave theory, Arnott [2] 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 [3].

S. Basak and A. Dubey are with the Electrical Engineering and Computer Science Department at Vanderbilt University, USA. e-mail: sanchita.basak@vanderbilt.edu; abhishek.dubey@vanderbilt.edu

B.P. Leao is with Siemens Corporate Technology, Princeton, NJ, USA. e-mail: bruno.leao@siemens.com;

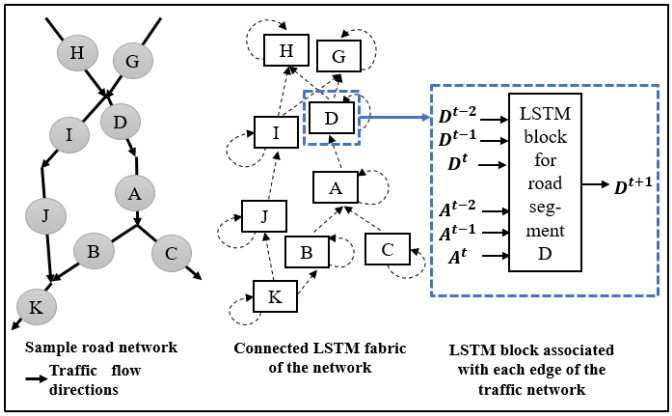


Fig. 1: A sample road network and corresponding connected LSTM networks.

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 [4] [3] in traffic prediction considered the traffic network as a homogeneous system and were mostly focused on applying 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 [5].

Figure 1 illustrates the representation of a sample road network with directions of traffic flow and its corresponding

978-1-7281-0858-2/19/$31.00 © 2019 IEEE

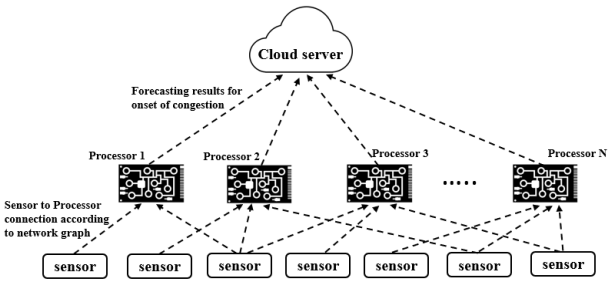


Fig. 2: Each computing processor associated with each road segment in the network collects the speed of the neighboring segments according to the graphical model of the network, process them to forecast the speed and send the results to a central cloud server which can be used for taking traffic routing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

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 upto current time to determine the future traffic state of the target edge. Figure 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 [6] 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 paper 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.

The rest of the paper is organized as follows: Section II walks through the problem formulation including graphical modelling of the network. Section III discusses the related research in traffic congestion forecasting. Section V discusses our proposed approach for the congestion forecasting framework. Section VI discusses the results and section VII concludes the paper with future directions.

II. System Model and Problem Formulation

We model the transportation network as a directed connected graph defined as $G = (V, E)$, where V is a set of nodes representing intersections. 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 the graph we apply in and out operator such that the operator $in : V \rightarrow 2^E$ gives all the edges for which this node v is the destination and the operator $out : V \rightarrow 2^E$ gives all the edges whose source is node v. The *indegree* of a node v is the number of road segments incoming to the node and can be calculated as $|in(v)|$, whereas, the *outdegree* of a node v is the number of road segments outgoing from the node and is calculated as $|out(v)|$. Similarly, the source and destination node of an edge can be accessed. Each node is associated with some static information. The attribute of a node $v \in V$ contains longitude and latitude. $v_{loc} : V \rightarrow \Re \times \Re$ provides location of node v as a tuple of two reals (latitude, longitude). Each edge contains the information of the geographical shape of the road segment as a sequence of latitude longitude tuples.

Definition 1 (Traffic Message Channel (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.

In addition to the traffic message channels, we define certain operations to provide us access to neighbors of a TMC. These operators are defined below.

Definition 2 (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 $N^k_{in}(e)$ can be defined recursively as $\bigcup_{x \in N^{k-1}_{in}(e)}(in(src(x)))$ using the definition of 1 hop neighbors, $N^1_{in}(e) = \bigcup_{x \in in(src(e))}(in(src(x)))$

Definition 3 (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. $N^1_{out}(e) = \bigcup_{x \in out(dst(e))}(out(dst(x)))$. Given the set N^{k-1}_{out}, the set N^k_{out} can be defined as $N^k_{out}(e) = \bigcup_{x \in N^{k-1}_{out}(e)}(out(dst(x)))$.

Figure 3 provides an illustrative example of k-hop incoming and outgoing neighbors.

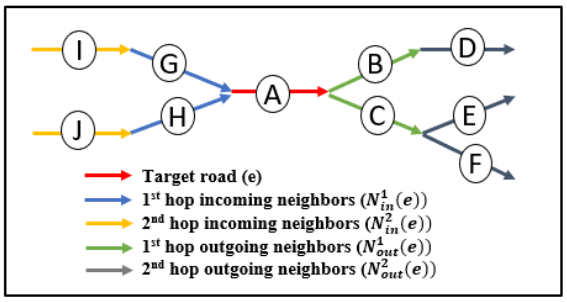


Fig. 3: An illustrative representation of k-hop incoming and outgoing neighbors. In this figure, 'A' is the target road, 'G' and 'H' are its 1st hop incoming neighbors, 'I' and 'J' are its 2nd hop incoming neighbors, 'B' and 'C' are its 1st hop outgoing neighbors and 'D','E' and 'F' are its 2nd hop outgoing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

Definition 4 (Jam Factor): The jam factor $JF : TMC \rightarrow [0, 1]$ is an indicator of number of cars over capacity on the road. Generally JF is from 0 to 10, but we normalized its value within 0 to 1, where $JF = 1$ indicates non-moving traffic and $JF = 0$ indicates free flowing traffic.

Definition 5 (Free flow speed): The free flow speed $FF : TMC \rightarrow \Re^+$ as the average of all maximum speeds observed when the jam factor observed on the TMC is 0.

A. Problem Statement

This paper 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.

Definition 6 (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. [7] used reduction of 50% speed compared to free flow speed as an indicator of congestion. We use the congestion criteria described as above.

Definition 7 ($\Delta-$ Cascade Event): The Delta Cascade Event is defined as a congestion event where more than 50% of first hop neighbors ($N^1_{in}(e)$) show 60% speed reduction with Δ time steps. We say e is the source of the cascade event.

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.

B. Dataset

In this paper we study the congestion forecasting problem on a real world traffic dataset. Particularly we use the traffic data of Nashville (USA) collected by our research team from HERE API [5]. In this dataset each road segment is expressed as a Traffic Message Channel (TMC) having a TMC ID. Each TMC ID has timestamped information consisting of traffic speed, jam factor, free flow speed collected over several months. We use the traffic data from January 1, 2018 to February 12, 2018. Each TMC ID signifies a specific road segment in the network and contains the sensor information for that particular segment. In Nashville we have a total of 3724 TMCs.

III. Related Work

Previous work on traffic congestion analysis have adopted several model-driven approaches. Fei et al. [1] proposed a time-variant model of congestion boundary founded on shock-wave 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 [2] 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. JianCheng et al. [8] proposed a congestion propagation framework inspired by the cell transmission phenomenon to identify network congestion bottleneck under various traffic demand scenarios.

Xiong et al. [7] 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 [9] are suitable to be applied to congestion forecasting. Zhang et al. [10] 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. [4] 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. [3] 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. [11]. Due to the exceptional capability of learning temporal sequence, LSTMs are used in various other domains including language learning [12], prognostics [13] as well as traffic prediction. Tian et al. [14] 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.

IV. 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 [15]. 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 [16], 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:

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

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 1 shows an example of a sample road network and its corresponding connected fabric of LSTM.

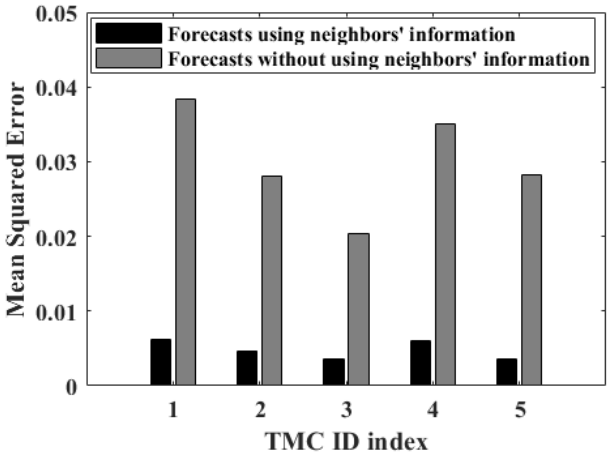


Fig. 4: Comparison of mean Squared error in traffic forecasting with and without using neighborhood information to solely evaluate the importance of using neighborhood information in the traffic prediction architecture.

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

A. 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 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.

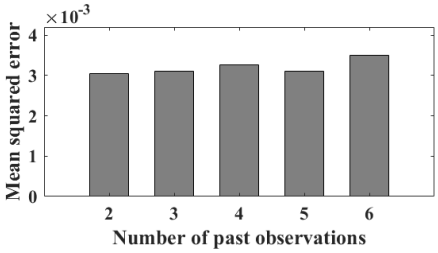


Fig. 5: Comparison of the mean squared errors of taking different number of past observations in predicting the future speed

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

C. 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.

D. 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

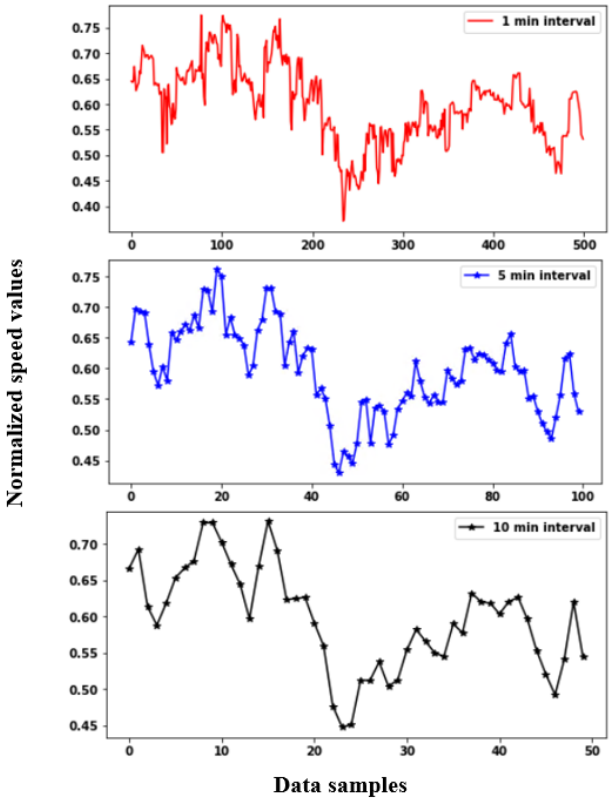


Fig. 6: Selecting hyper-parameter: time constant at which the data should be sampled.

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 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 7.

V. Congestion Prognostics

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

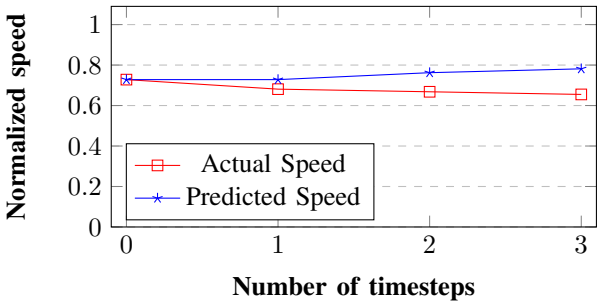


Fig. 7: Predicting normalized traffic speed of TMC '4424-0.12847' upto three timesteps, i.e., 30 minutes ahead from current time using the citywide connected LSTM fabric.

A. 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 IV to train the network. The trained model for each TMC is saved which is used again in the congestion forecasting phase. Figure 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.

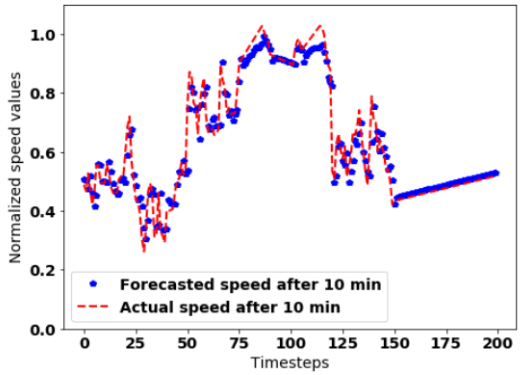


Fig. 8: Comparing the forecasted speed after 10 minutes and the actual observed speed after 10 minutes on a TMC ID having five neighbors.

In this work, we also compare the mean squared error (MSE) in predicting one vs. multiple timesteps ahead in future for multiple TMCs. Figure 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.

B. 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 $N^1_{in}(e)$, For each of those 1st hop neighbors, it finds the 2nd hop incoming neighbors denoted as $N^2_{in}(e)$. It repeats the process for 3rd hop incoming neighbors to find a set of

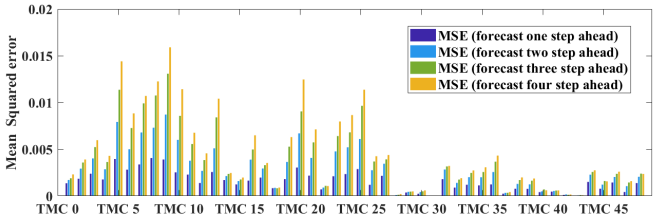


Fig. 9: Comparison of the mean squared errors among forecasting one, two, three and four timesteps ahead respectively for 45 TMC out of 3724 TMC in Nashville. The plot shows first 45 TMC only for brevity.

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

Input: Congestion event CE at TMC e at timestep n
$\tilde{N} = []$
$\tilde{N}.append(N^1_{in}(e))$
for each i in $N^1_{in}(e)$ **do**
  $\tilde{N}.append(N^2_{in}(e))$
  for each j in $N^2_{in}(e)$ **do**
    $\tilde{N}.append(N^3_{in}(e))$
  end for
end for
for timestep $n : n + 10$ **do**
  $flag = zeros(length(\tilde{N}))$
  for each i in $\tilde{N}$ **do**
    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**
      $flag[i] = 1$
      Output $\tilde{N}$ will have onset of congestion at timestep $(n + 1)$
    end if
  end for
  for each j in flag **do**
    if $flag[j] = 1$ **then**
      $\tilde{N}.delete(\tilde{N}[i])$
    end if
  end for
end for

it denoted as $N^3_{in}(e)$. 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 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

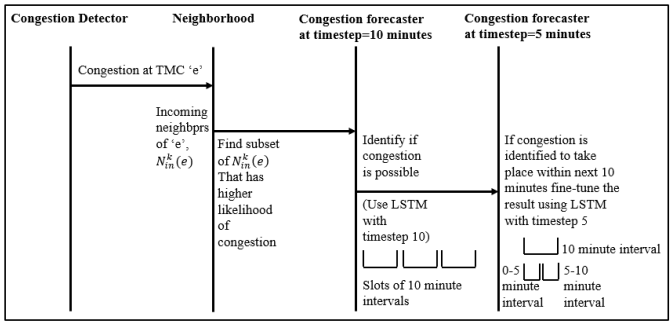


Fig. 10: An illustration of the overall congestion forecasting framework

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 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 10 shows a diagram explaining the overall congestion forecasting framework.

C. Identifying likelihood of congestion propagation

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 $ev1$ corresponds to the phenomenon where a significant speed decrease is observed at any target road. Event $ev2$ 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.

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.

$\hat{cn^1}(e) = []$
for i in $N_{in}^1(e)$ **do**
  $ev1 = 0$
  $ev2 = 0$
  if $s[e][n] < 0.6 * FF$ **then**
    $ev1 = ev1 + 1$
    $temp = 0$
    $speed\_array$ = $s[i][n : n + 4]$
    if any item in $speed\_array < 0.6 * FF$ **then**
      $temp = 1$
    end if
    if $temp = 1$ **then**
      $ev2 = ev2 + 1$
    end if
  end if
  $likelihood[i] = ev2/ev1$
  if $likelihood[i] > 0.5$ **then**
    $\hat{cn^1}(e).append(i)$
  end if
end for

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, $\hat{cn}^1(e) \subset N_{in}^1(e)$, 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 $N_{in}^1(e)$. Thus we are reducing the execution time of the overall congestion forecasting algorithm by an order of $\hat{cn}^1(e)/N_{in}^1(e)$ for each of the road segments. Xiong et al. [7] also used congestion propagation probabilities to construct propagation graphs from congestion matrices. But, in this work we use the likelihood of congestion propagation to

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_in^1(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 ≥ 0.5 * |N_in^1(e)|) then
        Output: TMC e has congestion at timestep n
      end if
    end if
  end for
end for
```

save the time complexity of the overall congestion forecasting algorithm. However, our algorithm can still be applied to edges that are left out.

VI. Validation

We validate our algorithm on the Nashville dataset described in Section II-B. The data from January 28, 2018 to February 12, 2018 was used for validation purposes. The outline of this section is as follows. We first identify the set of congestion events (definition 6). 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.

A. 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 $N_{in}^1(e)$ 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 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.

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

C. Fine tuning progression results Using LSTM with $timestep = 5$

Figure 12 shows the transportation network for congestion event index 10 described in table I. For better understanding of the result on the cascaded congestion prediction using LSTM

TABLE I: Summary of the congestion forecasting result for ten congestion events whose precision and recall values are shown in Figure 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 14:50	14:20 15:00	15:00	15:10	-	-
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 - 07:20	06:50 07:10 07:10	07:40 07:50	07:30 07:50	-	-
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 07:40	07:00 07:20	07:00 07:10 07:30 07:30 07:20 07:30	07:00 07:00 07:30 07:30 07:20 07:30

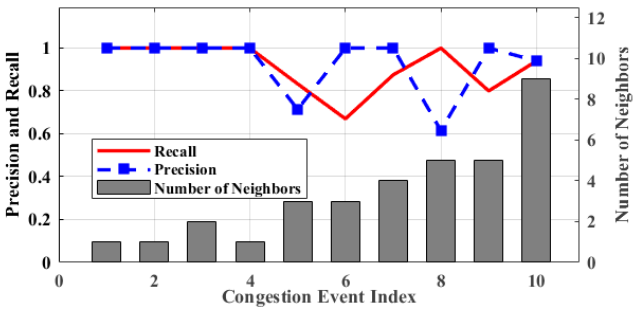


Fig. 11: Precision and recall values identifying the onset of congestion in all 1st, 2nd and 3rd hop neighboring segments of a congestion source tested over ten congestion events.

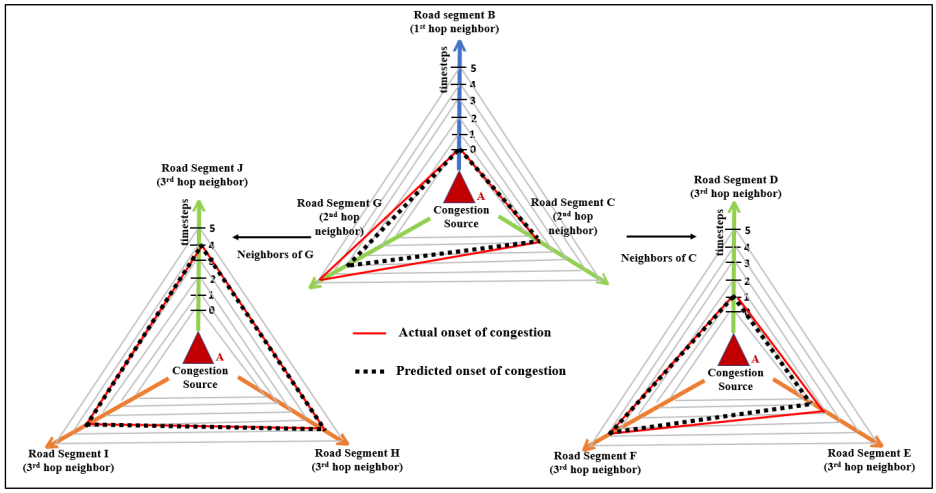
Fig. 12: Road segment for congestion event 10 in Table I. The source road of congestion is road segment 'A'. Following the congestion at the source road segment, the congestion propagates to the 1st ('B'), 2nd ('C', 'G') and 3rd hop ('D', 'E', 'F', 'H', 'I', 'J') incoming neighbors respectively.

with $timestep = 10$ we present Figure 13a 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.

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. Figure 13b 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 I 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.

VII. Conclusions and Future Work

In this paper, we demonstrated mechanisms for spatiotemporal modelling of traffic network and learned the distribution of traffic speed of a road segment as a function of its neighboring segments. We developed a traffic congestion forecasting framework based on city-level connected LSTM networks. Figure 14 summarizes the workflow of our approach. 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



(a)

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

(b)

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

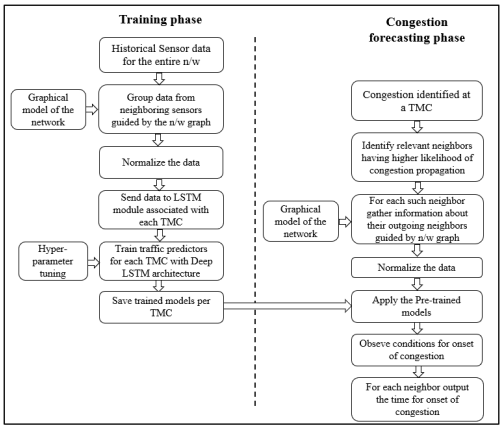


Fig. 14: Workflow of the congestion forecasting framework.

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.

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

[1] W. Fei, G. Song, J. Zang, Y. Gao, J. Sun, and L. Yu, "Framework model for time-variant propagation speed and congestion boundary by incident on expressways," *IET Intelligent Transport Systems*, vol. 11, no. 1, pp. 10–17, 2017.

[2] R. Arnott, "A bathtub model of downtown traffic congestion," *Journal of Urban Economics*, vol. 76, pp. 110 – 121, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094119013000107

[3] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PloS one*, vol. 10, p. e0119044, 03 2015.

[4] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1 – 17, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X17300633

[5] "Here developer," 2019. [Online]. Available: https://developer.here.com/

[6] B. Pan, U. Demiryurek, C. Gupta, and C. Shahabi, "Forecasting spatiotemporal impact of traffic incidents for next-generation navigation systems," *Knowl. Inf. Syst.*, vol. 45, no. 1, pp. 75–104, Oct. 2015. [Online]. Available: http://dx.doi.org/10.1007/s10115-014-0783-6

[7] H. Xiong, A. Vahedian, X. Zhou, Y. Li, and J. Luo, "Predicting traffic congestion propagation patterns: A propagation graph approach," in *Proceedings of the 11th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, ser. IWCTS'18. New York, NY, USA: ACM, 2018, pp. 60–69. [Online]. Available: http://doi.acm.org/10.1145/3283207.3283213

[8] J. Long, Z. Gao, H. Ren, and A. Lian, "Urban traffic congestion propagation and bottleneck identification," *Science in China Series F: Information Sciences*, vol. 51, no. 7, p. 948, Jun 2008. [Online]. Available: https://doi.org/10.1007/s11432-008-0038-9

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

[10] S. L. Zhang, Y. Z. Yao, J. Hu, Y. Zhao, S. Li, and J. Hu, "Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks," in *Sensors*, 2019.

[11] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "Lstm network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[12] F. A. Gers and E. Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *Trans. Neur. Netw.*, vol. 12, no. 6, pp. 1333–1340, Nov. 2001. [Online]. Available: https://doi.org/10.1109/72.963769

[13] S. Basak, S. Sengupta, and A. Dubey, "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)*, June 2019, pp. 208–216.

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

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[16] S. Basak, A. Ayman, A. Laszka, A. Dubey, and L. Bruno, "Data-driven detection of anomalies andcascading failures in traffic networks," *Annual Conference of the PHM Society*, 2019, in press.