A Data-driven Approach to Estimate and Predict the Traffic Incidents' Queue Length

Banishree Ghosh[1], Justin Dauwels[2]

***Abstract*— Traffic incidents often lead to the closure of lanes, causing a reduction in road capacity. To handle such situations, Intelligent Transport Systems (ITS) are commonly employed to maximize the utilization of the remaining capacity. By leveraging data mining and deep learning techniques, our objective is to anticipate congestion patterns during such incidents and develop a real-time guidance system to aid drivers. To accomplish this, we build a Long Short-Term Memory neural network-based prediction model which incorporates up-to-date traffic speed-flow data and a range of spatiotemporal features as inputs and predicts the queue length of the incidents. Moreover, the model continuously predicts the queue length throughout the duration of the incidents, capturing the temporal dynamics of the situation. Initially, our proposed model exhibits an average error of 52.54%, which improves to 18.5% over the course of one hour of prediction.**

I. INTRODUCTION

The occurrence of traffic incidents is on the rise in urban areas worldwide. To mitigate the consequences of these incidents, Intelligent Transportation Systems (ITS) are increasingly employed for various purposes, such as predicting incident duration, modeling, and congestion avoidance. These systems gather real-time traffic data from multiple sources, providing drivers with timely information about the state of the road network [1]. In our research, we integrate the geographical and spatio-temporal features with historical traffic data to forecast the queue length, representing the extent of congestion. This predictive capability enables drivers to make informed route choices. Additionally, our model incorporates multi-step prediction to account for evolving future traffic conditions. We also delve into existing studies concerning incident impact prediction in different cities, identifying research gaps that we seek to address in our work.

The prediction of the impact of incidents has been an active research area in the field of transportation in the last two decades [1]. In the past, several studies explored the simulation-based theoretical model-driven approach for analyzing the impact of the incidents which are built on a prior knowledge of the traffic dynamics instead of dealing with historical or real-time traffic data [2].

Another approach for congestion prediction is based on machine learning techniques. These techniques have been widely used in recent years due to their ability to handle large amounts of data and learn complex patterns. However, there are certain shortcomings of the data-driven studies as well. For example, Miller *et al.*'s analysis focused solely on a single upstream-downstream link pair near the incident location, potentially overlooking the complete propagation of congestion [3]. In real-life situations, congestion can expand beyond a single upstream link. Moreover, a number of studies analyzed one of the three traffic parameters, i.e., speed, flow or density in their analysis. For example, Previous studies conducted by Shahabi *et al.* and Lin *et al.* solely examined traffic speed data to predict accident impacts [4] [5]. However, incorporating multiple traffic variables allows us to mitigate false alarms caused by sensor noise. In order to strengthen the robustness of our approach to outliers, we take into account both the recorded traffic flow and speed measurements from the sensors to estimate and predict the queue length. Besides, while performing the prediction of incident duration in multiple phases, Shang *et al.* did not incorporate the real-time traffic data in their analysis [6]. However, predicting the temporal dynamicity of the incidents' impact is challenging without up-to-date traffic data because the extent of congestion at any given moment is influenced by the instantaneous traffic conditions on the road. Therefore, we perform a dynamic prediction of the impact of the incidents using the online streaming traffic speed and flow data in our analysis.

With the advent of deep learning architectures, numerous studies have been conducted focusing on the utilization of deep neural networks to capture the spatiotemporal dynamics of the road networks for various purposes, such as traffic prediction [7] or detection of incidents [8]. For example, Ren *et al.* developed an LSTM network to predict citywide accident risk [9]. However, their model makes predictions at a coarser level because their data-set has lower spatial and temporal resolution. In that study, the temporal resolution is 1 hour, whereas our data is recorded at 5-min intervals. Similarly, the spatial resolution in their research is 1000 meters compared to our finer resolution of 100 meters. In other recent works related to congestion detection, Ali *et al.* built a bi-LSTM model based on the data collected from social networking sites for accident detection and condition analysis [8], whereas Sun *et al.* built a CNN model to detect the non-recurrent congestion caused by different scenarios [10]. Nevertheless, these studies did not specifically address incidents-induced congestion.

Apart from that, some studies have also explored the use of hybrid networks for traffic congestion prediction [11] [12]. In these studies, the extent of congestion is categorized in different levels and classification is performed to predict the congestion level at different grids of the roads. However, the

[1]Banishree Ghosh is with the Institute for Infocomm Research (I^{2}R), Agency for Science, Technology and Research (A*STAR), Singapore. Email: ghosh.banishree2008@gmail.com.

[2]Justin Dauwels is with the Department of Microelectronics at TU Delft, Netherlands. Email: J.H.G.Dauwels@tudelft.nl.

results have only been reported for short-term prediction. Also, the models did not incorporate external features or spatio-temporal features in their models.

In the following, we highlight the main contributions of our study:

1) We opt for a rigorous data-driven approach to estimate the length of congestion during the incidents based on their spatio-temporal impact on the traffic parameters (speed and flow) of neighboring upstream links.
2) By feeding the LSTM network with a comprehensive feature set that includes up-to-date traffic data as well as incidents' features, we aim to perform the post-impact prediction of traffic incidents. This approach factors in the present traffic condition and the daily average traffic state, facilitating more accurate forecasting and management of congestion.
3) Since the historical traffic data used in our work gets updated at every 5 minutes, our proposed prediction model forecasts the queue length dynamically till the incident ends. To this end, We construct a single LSTM network that can handle inputs with different numbers of time-steps during the training phase, and subsequently use this model for predicting the queue length at various time instants.

The remainder of this paper is organized as follows. In the following section, we provide a detailed description of our data-set. Section III explains the steps for determining the queue lengths at different instants for the incidents obtained from historical data. These queue lengths serve as the target variables during the prediction. In Section IV, we provide a detailed description of our queue length prediction model and demonstrate the experimental setup for this model. In Section V, we provide a comparative analysis of the prediction errors obtained by different methods. Next, we evaluate the prediction performance of the model using the LSTM network for different classes of incidents in Section VI. Finally, in Section VII we offer concluding remarks.

II. Description of the Data

For this study, the data-set utilized comprises traffic speed and flow data collected between August 2016 and January 2017, with a temporal resolution of 5 minutes. Traffic flow is defined as the number of vehicles passing through a specific point per unit time. In Singapore, traffic flow data is gathered by sensors positioned at the end of each road segment. Consequently, the recorded readings indicate the total number of vehicles exiting the road link at 5-minute intervals, measured in vehicles per hour. Meanwhile, traffic speed denotes the average speed of vehicles passing through a specific point, measured in kilometers per hour. The expressways in Singapore generally have speed limit of 100 km/h. The speed values are recorded by the sensors in terms of ten discrete speed bands, each spanning 10 km/h. Thus, the recorded speed values range from a minimum of 0 to a maximum of 10.

Apart from that, the data-set comprises the details of $15,000$ incidents that occurred on the expressways of Singapore. The incidents recorded over the span of those six months have the following features: (1) the incident ID, which is a unique serial number, (2) the coordinates and the upstream link IDs of the incident location, (3) the expressway and direction where the incident occurred, (4) the incident's start and end time, (5) information regarding the closure of adjacent shoulder lane and main carriageway lanes, and (6) the type of incident (vehicle breakdown, accident, etc.). The categorical features within our feature set are transformed into binary values using the one-hot assignment method, while the numerical features are converted from decimal numbers to binary representation. Additionally, the data includes an extra feature called "queue length," which reports the maximum length of congestion observed by the Land Transport Authority (LTA) during these incidents. However, as congestion length varies over time, a more rigorous approach is necessary to measure the queue length accurately for individual incidents. Therefore, the queue length is only available for $1,209$ incidents in our data-set.

III. Queue Length Estimation using Traffic Data

There is no standardized definition for the queue length of incidents. In our analysis, we define the incident queue length or congestion length as the distance along the road where delays occur due to the incident. Consequently, we consider a link or road segment impacted by an incident if the average vehicle speed significantly decreases compared to the free flow speed, and there is a sudden decline in traffic flow following the incident. In this section, we elaborate on our presumptions regarding the traffic data and establish criteria for identifying congested links. Using these assumptions, we calculate the queue lengths during incidents.

Step 1: We first explain the experimental set-up to figure out how long the congestion has spread during the incidents.

We begin with the initial assumption that an incident is reported to take place at link ℓ for a duration of $T_{\text{rep\_start}} - T_{\text{rep\_end}}$. The length of individual link ℓ varies in the range of $20 - 200$ meters. Given that the traffic data is recorded at 5-minute intervals, we compute the queue length every 5 minute. Hence, we estimate the nearest multiple of 5 as the approximate start and end time of the incident, which can be expressed by:

$$t_0 = \left\lfloor \frac{T_{\text{rep\_start}}}{5} \right\rfloor * 5, t_L = \left\lceil \frac{T_{\text{rep\_end}}}{5} \right\rceil * 5. \tag{1}$$

Next, we need to establish the upper limit for the number of upstream links to be considered when calculating the queue lengths. To ensure comprehensive analysis and prevent any loss of data, we investigate all the upstream links starting from the incident location and extending all the way to the farthest point on the expressway to determine if they experienced congestion or not.

Step 2: As traffic incidents are not a regular occurrence unlike the typical congestion experienced during peak hours, we analyze the traffic data recorded at each specific link on the incident day and compare it with the data from non-incident days. This allows us to comprehend the impact of the incidents in the traffic.

We use the notation ℓ_x to represent the consecutive upstream links of ℓ, where $x \in \{1,2,3,...\}$. The time instants are denoted as t_y at 5-minute intervals, where $y \in \{0,1,2,...,T\}$. From the traffic data-set, we extract the traffic speed $V_{\text{inc}}(\ell_x,t_y)$ and flow value $F_{\text{inc}}(\ell_x,t_y)$ for each time instant t_y on the day of the incident at link ℓ_x. Moreover, the average speed and flow value of non-incident days are denoted as $V_{\text{avg}}(\ell_x,t_y)$ and $F_{\text{avg}}(\ell_x,t_y)$, respectively. These values are computed using the traffic data from the same day but from different weeks when no incidents occurred.
Therefore, the flow deviation is defined by:

$$d_{F(\ell_x,t_y)} = \frac{(F_{\text{inc}}(\ell_x,t_y) - F_{\text{avg}}(\ell_x,t_y))}{F_{\text{avg}}(\ell_x,t_y)}, \quad (2)$$

and the difference in speed can be expressed as:

$$d_{V(\ell_x,t_y)} = \frac{(V_{\text{inc}}(\ell_x,t_y) - V_{\text{avg}}(\ell_x,t_y))}{V_{\text{avg}}(\ell_x,t_y)}. \quad (3)$$

Thus, we compute the values of $d_{V(\ell_x,t_y)}$ and $d_{F(\ell_x,t_y)}$ for individual link ℓ_x at each time instant t_y, where $x \in \{1,2,3,...\}$ and $y \in \{0,1,2,\dots,T-1,T\}$.

Step 3: Our next objective is to determine whether the individual links were congested or not at a specific time due to the incident, based on their speed-flow profiles. We assume that if the link ℓ_x is congested, there will be a sudden and substantial decrease in both traffic speed and flow, indicating the onset of congestion in that particular link. Considering the inherent randomness of traffic data, the drop in values must be significant to ensure that it is not merely a result of data randomness, rather a consequence of the incident itself. To achieve this, we introduce the Queue Length Estimation (QLE) Algorithm (Algorithm 1), which outlines our definition of congestion for a given link at a particular instant. The cutoffs P_{10_V} and P_{10_F} in Algorithm 1 are selected by trial and error method. We compute the correlation coefficient of the reported and estimated queue-lengths for different cutoffs. We plot the values of the coefficients with different cutoff values in Fig. 1 and choose the best one as the optimum cutoff (10$^{\text{th}}$ percentile).

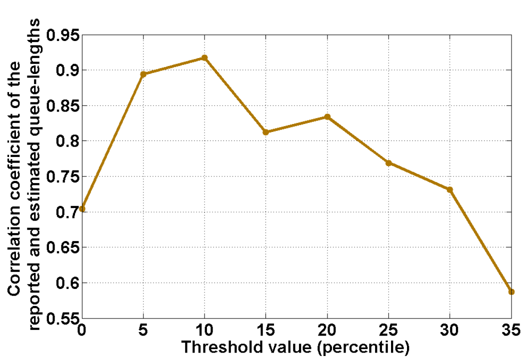

Fig. 1: Variation of correlation coefficient of the reported and estimated queue-lengths for different cutoff values.

Further details of Algorithm 1 are explained in [13].

Step 4: In the final step, we calculate the queue length at each time instant. Following the previous steps, we can identify the congested links at each instant t_y. By summing up the lengths of these congested links, we determine the queue length at each time instant t_y.
Thus, we calculate the queue length or congestion length at a 5-minute intervals throughout the entire duration of the incidents.

Algorithm 1 Queue Length Estimation (QLE) Algorithm

INPUT: 1) Speed and flow values $V(\ell_x,t_y)$ and $F(\ell_x,t_y)$ for each link ℓ_x at every instant t_y, 2) the length of the links ℓ_x.
OUTPUT: Queue length $Q(t_y)$ at each instant t_y.

for each link ℓ_x **do**
  for $y = 1: T-1$ **do**
    Compute $\Delta_{V(\ell_x,t_y)} = d_{V(\ell_x,t_{y+1})} - d_{V(\ell_x,t_y,1)}$
    Compute $\Delta_{F(\ell_x,t_y)} = d_{F(\ell_x,t_{y+1})} - d_{F(\ell_x,t_y)}$
  end for
  $m_{1_{V(\ell_x)}} \leftarrow \min(\Delta_{V(\ell_x,t)}, 1)$ ▷ where min(E,i) represents the i^{th} smallest element of E.
  $m_{2_{V(\ell_x)}} \leftarrow \min(\Delta_{V(\ell_x,t)}, 2)$
  $m_{1_{F(\ell_x)}} \leftarrow \min(\Delta_{F(\ell_x,t)}, 1)$
  $m_{2_{F(\ell_x)}} \leftarrow \min(\Delta_{F(\ell_x,t)}, 2)$
  if $m_{1_{V(\ell_x)}} < 0$ and $m_{1_{F(\ell_x)}} < 0$ **then**
    for $y = 1: T$ **do**
      if $\Delta_{V(\ell_x,t_y)} = m_{1_{V(\ell_x)}}$ **then**
        $t_{\text{min}_{V(\ell_x)}} \leftarrow t_y$
      end if
      if $\Delta_{F(\ell_x,t_y)} = m_{1_{F(\ell_x)}}$ **then**
        $t_{\text{min}_{F(\ell_x)}} \leftarrow t_y$
      end if
    end for
    $T_{\text{start}}(\ell_x) \leftarrow \min(t_{\text{min}_{V(\ell_x)}}, t_{\text{min}_{F(\ell_x)}})$
    if $t_y > T_{\text{start}}(\ell_x)$ and $d_{V(\ell_x,t_y)} > 0$ and $d_{F(\ell_x,t_y)} > 0$ **then**
      $T_{\text{end}}(\ell_x) = t_y$
    end if
    Compute $\delta_{V(\ell_x)} = m_{2_{V(\ell_x)}} - m_{1_{V(\ell_x)}}$
    Compute $\delta_{F(\ell_x)} = m_{2_{F(\ell_x)}} - m_{1_{F(\ell_x)}}$
  end if
end for
$P_{10_V} \leftarrow \text{PCTL}_{10\%}(\delta_V)$
$P_{10_F} \leftarrow \text{PCTL}_{10\%}(\delta_F)$
for each link ℓ_x **do**
  if $\delta_{V(\ell_x)} > P_{10_V}$ and $\delta_{F(\ell_x)} > P_{10_F}$ **then**
    for $y = 1: T$ **do**
      if $t_y \geq T_{\text{start}}$ and $t_y \leq T_{\text{end}}$ **then**
        $Q(t_y) \leftarrow Q(t_y) + \ell_x$
      end if
    end for
  end if
end for

IV. Prediction Methods and Model Development

In this section, we present a comprehensive analysis of our predictive model which dynamically forecasts the queue length at intervals of 10 minutes from the beginning of the incidents. We explain the experimental setup and discuss how the inputs are fed to the LSTM model. Later, we mention the evaluation metric which we choose for reporting the prediction errors.

A. *Experimental Setup*

In this subsection, we first present the model architecture employing the LSTM network in Fig. 2. Our feature set

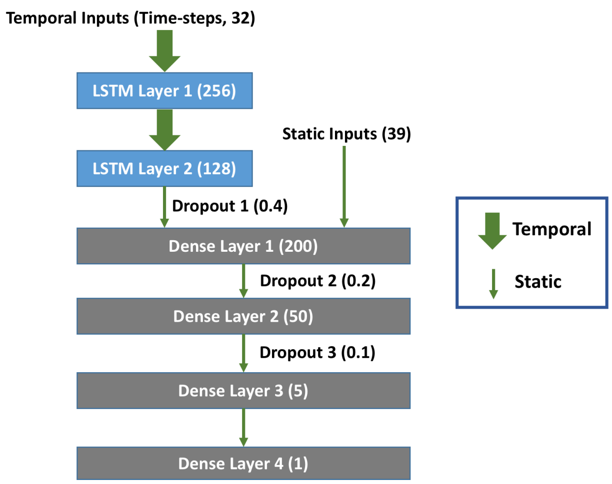

Fig. 2: The schematic diagram illustrating the LSTM network architecture in our proposed model.

comprises two types of features, static (expressway, direction, etc.) and temporal (speed, flow, etc.). Given the LSTM network's inherent ability to learn the temporal patterns in the data-set, we feed the LSTM layers with the time-varying inputs only, as can be seen in Fig. 2. In contrast, we feed the static features directly into the dense layers which are fully connected layers. The LSTM network's hidden layers consist of two LSTM layers followed by four dense layers in sequence. The output generated by the final LSTM layer is combined with the static inputs and fed into the first dense layer. The final dense layer serves as the output layer, providing the predicted queue length. To prevent overfitting, we introduce three dropout layers at different stages with dropout rates of 40%, 20%, and 10%, respectively. The first dropout layer is in between the last LSTM and the first dense layer, while the remaining two dropout layers are added between the fully connected layers. Rectified Linear Units (RELU) is employed as the activation function for the dense layers.

Next, we illustrate how we combine the input features to train the prediction model. The LSTM layers require a three-dimensional input structure, as depicted in Fig. 3. The dimensions are as follows: 1) Samples - This refers to the number of samples in each batch, which can vary across batches. In our work, each individual incident represents a sample within a batch. 2) Time-Steps - Time-steps indicate the number of previous time instants considered for each sample within a batch. The time-steps should be consistent for samples within a single batch. However, it is not necessary to impose this constraint across different batches. Given

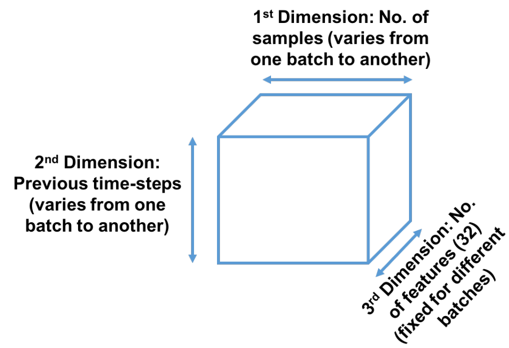

Fig. 3: Description of the three dimensions of the input fed into the LSTM network.

that our traffic data has a resolution of 5 minutes and we forecast the queue length at 10-minute intervals, we can take into account 2 previous time steps since the beginning of the incidents for a 10-minute prediction (0 min and 5 min). In the similar way, we can utilize the traffic data from 4 previous time-steps for 20-minute prediction (0 min, 5 min, 10 min, and 15 min), and so on. 3) Features - The time-dependent attributes are referred to as features. The number of features must be consistent across all samples in all batches. In our work, we generate a feature set of size 32 by analyzing various combinations of speed-flow data from the neighboring links. We refer to the notations and definitions described in Section III for explaining these features. For each upstream link ℓ_x at each time-instant t_y, we consider the traffic speed $V_{\text{inc}}(\ell_x, t_y)$, traffic flow $F_{\text{inc}}(\ell_x, t_y)$, speed-difference $d_{V_{(\ell_x,t_y)}}$, and flow-difference $d_{F_{(\ell_x,t_y)}}$ of the day of the incident. Furthermore, as the number of upstream neighboring links varies for each incident while the number of features should remain constant, we adopt a different approach. Instead of treating each upstream link as a separate feature, we combine the upstream links in a weighted manner to construct the features. For instance, the feature associated with the traffic speed values can be expressed as follows:

$$\mathcal{F}_{V_{\text{inc}}} = \sum_i w_{\ell_x}.V_{\text{inc}}(\ell_x, t_y), \tag{4}$$

where w_{ℓ_x} is the weight of the upstream link ℓ_x. To ensure that the weights are inversely proportional to the distance between the incident location and the upstream links, we select two inverse-distance weighting functions as follows:

$$w_{\ell_x} = \frac{1}{\text{dist}(\ell, \ell_x)^p}, \tag{5}$$

and

$$w_{\ell_x} = e^{-q.\text{dist}(\ell,\ell_x)}, \tag{6}$$

where $p \in \{0.05, 0.1, 0.2, 0.4, 0.8\}$, and $q \in \{0.02, 0.05, 0.08\}$. Therefore, we create eight features for each attribute, considering different weightage to the upstream links based on five values of p and three values of q. Considering the four attributes we have ($V_{inc}(\ell_x, t_y)$, $F_{inc}(\ell_x, t_y)$, $d_{V_{(\ell_x,t_y)}}$, and $d_{F_{(\ell_x,t_y)}}$) at each time-step, we obtain a total of 32 features for each sample. The spatiotemporal features are reshaped as a three-dimensional tensor, where we incorporate the temporal variation in the second dimension (time-steps) and the spatial variation in the third dimension. This approach enables us to provide the LSTM layers with distinct batches of three-dimensional input data, where each batch corresponds to different elapsed time. As time elapses, the second dimension (representing previous time instants) expands, while the size of the first dimension (representing the sample size) diminishes as predictions are no longer performed for incidents that have already concluded. Consequently, the first two dimensions differ among various batches. We train the same network using inputs at different time-steps. Table I lists the hyper-parameters employed during the training phase.

TABLE I: The hyper-parameters utilized during the training phase of LSTM and GRU networks.

Parameter	Value
Optimizer	'adam'
Batch size	150
Activation function	'relu'
Loss function	mean squared error
Learning rate	optimized on training set
Training epochs	optimized on training set

B. Evaluation Metric

To evaluate the performance of our prediction model, we select the Mean Absolute Percentage Error (MAPE) value as the error metric, which can be defined by the following equation:

$$\mathrm{MAPE} = \frac{100}{N} . \sum_{i=1}^{N} \left| \frac{Q_i - \hat{Q}_i}{Q_i} \right|, \tag{7}$$

where the actual and predicted queue lengths are Q_i and $\hat{Q}_i$ respectively, and N is the number of incidents.

V. Performance Evaluation of Different Regression Methods

In this section, we compare the performance of the queue length prediction model for the LSTM network with traditional methods, like Treebagger [14], Multi-Layer Perceptron (MLP) [15], and Support Vector Regression (SVR) [16] and another deep learning framework named Gated Recurrent Unit (GRU) [17]. We follow the approach of cluster-specific modeling for the traditional methods as explained in one of our previous works [18]. We employ nested cross-validation for model selection and hyper-parameter optimization. We opt for 3-fold CV in the outer loop (train-test split) and 10-fold CV in the inner loop (hyper-parameter optimization). On the other hand, to construct the GRU network, we substitute the LSTM layers with GRU layers. One key difference between the traditional methods and LSTM/GRU networks is the input configuration. In the former, all features are collectively provided as input to the model, whereas deep learning networks handle temporal and static features separately in two distinct stages. All experiments are conducted on a PC with the following specifications: CPU- Intel(R) Core(TM) i7-4770 CPU @3.40 GHz, and 8 GB RAM. We build our deep learning architectures using Keras framework [19] with Tensorflow backend [20].

We calculate the average MAPE values for all incidents at 10-minute intervals using the above-mentioned methods and enlist them in Table II. We observe that among the traditional machine learning methods SVR performs the best. However, the deep learning methods LSTM and GRU outperform the traditional methods significantly. Moreover, on an average LSTM exhibits slightly better performance compared to GRU because the simpler network structure of GRU results in a lower number of trainable parameters. Overall, we observe a non-monotonic variation of MAPE values with time for all methods. By incorporating additional features related to traffic flow and speed data over time, we observe a substantial improvement in prediction error. However, as time progresses, many incidents get cleared resulting in a much smaller number of samples, especially after 60 minutes. Consequently, the training data size becomes small, leading to a declining prediction accuracy.

TABLE II: Variation of MAPE values (in percentage) attained by different regression methods over time.

Methods	10 min	20 min	30 min	40 min	50 min	60 min	80 min	100 min	120 min
Treebagger	98.2	92.45	87.9	83.1	77.26	69.9	73.14	78	85.28
MLP	84.6	79.13	73.9	67	61.3	56.75	60.8	63.5	71.67
SVR	96.13	90	83.87	78.14	71.52	65.66	69.37	75.88	83.27
LSTM	52.54	42.4	34.1	26.7	25.9	18.5	20.9	25.7	29.03
GRU	60.7	54.57	47.2	39.51	32.3	26.58	27.3	32.65	40.2

VI. Prediction Performance for Different Categories of Incidents

In this section, we study the variation of MAPE values with the duration of incidents. For this purpose, the incidents are classified into various categories depending on their durations, such as the first class comprises incidents with durations in the range of 11 to 20 minutes, the second class includes incidents that lasted for 21 to 30 minutes, and so on. Subsequently, we assess the performance of our proposed model by comparing the error values obtained for different incident classes, as shown in Table III. The rows in the table correspond to the incident classes, while the columns represent the elapsed time. The MAPE values are enlisted for individual categories separately at 10-minute intervals. Table III indicates that the error values are relatively high

TABLE III: Variation of MAPE values (in percentage) generated by the LSTM network for different classes of incidents, categorized by their duration.

	10 min	20 min	30 min	40 min	50 min	60 min	80 min	100 min	120 min
11-20 min	67.17	–	–	–	–	–	–	–	–
21-30 min	63.45	54.1	–	–	–	–	–	–	–
31-40 min	59.5	50.2	41.6	–	–	–	–	–	–
41-50 min	53.3	47.8	39.15	34.5	–	–	–	–	–
51-60 min	51.75	44.32	37.8	33.6	36.9	–	–	–	–
61-80 min	48	40.45	33.36	29.06	25.26	28.37	–	–	–
81-100 min	45.69	37.3	29.5	25.32	22.4	16	23.6	–	–
101-120 min	42.4	35.07	28.6	21.56	18.97	14.7	20.99	28.87	–
Rest	37.5	32.2	24.8	19.4	15.36	11.3	15.67	21.8	29.03

(above 55%) for the first three categories which correspond to shorter duration incidents. For the remaining incidents, the initial errors fall within the range of $35\% - 55\%$, which gradually improve to $15\% - 35\%$ as the prediction progresses. Thus, we observe a significant improvement in error values, especially for incidents with durations longer than 40 minutes. The improved performance can be attributed to the utilization of a larger set of features extracted from the up-to-date traffic data, enabling the LSTM network to make more accurate predictions as time passes.

Moreover, at the onset of prediction, we notice higher errors for the first few categories of incidents, despite all classes having the same number of features at a given instant. This discrepancy arises because these incidents have an impact that lasts approximately half an hour, encompassing the propagation and dissipation of the congestion. From the perspective of queuing theory, if we consider the queue's formation and dissipation as a birth-death process, the model needs to learn the entire process within such a short time-frame. However, due to the distinct dynamics of the birth process and the death process, the model struggles to forecast the queue length accurately for incidents of shorter duration. Conversely, for incidents lasting longer than 40 minutes, there is sufficient time available for the model to learn the birth and death processes individually, which results in a significant improvement in error values for these incident classes. Nevertheless, in each category, we notice a non-monotonic variation in error during the transition from the birth to death process. For instance, the MAPE value in the fifth class ($51 - 60$ minute) initially improves from 51.75% to 33.6%, but experiences an unexpected increase after the 40^{th} minute. This can be attributed to most incidents in this category, where the queue develops for roughly 40 min, reaching the maximum spread, and then subsequently dissipates. Similarly, in the seventh class ($81 - 100$ minute) also, the error becomes smaller till the 40^{th} minute, followed by the onset of the death process for the queue. Furthermore, for the last two classes, the number of samples is relatively low, resulting in a degradation of the LSTM model's performance towards the end of the prediction.

VII. Conclusion

This study introduces a dynamic model for predicting the queue length of non-recurring road incidents on Singapore's expressways. Our approach involves two key steps. Firstly, we analyze the traffic data to estimate the queue length at different instants until the end of the incidents. Subsequently, we construct a Long Short-Term Memory network, trained on historical data, to predict the incidents' queue lengths. The model continuously updates its predictions throughout the duration of the incidents. Our model demonstrates significant improvements in prediction accuracy for medium-duration incidents (31-80 minutes), reducing errors from $45\% - 60\%$ to $25\% - 40\%$. For the longer-duration incidents, the prediction performance improves from around 40% to 15%. Overall, the model performs better for the incidents lasting over 30 minutes. However, as deep learning networks require substantial training data, we intend to gather a larger data-set for better training of our model.

Acknowledgment

The authors wish to thank the Land Transport Authority of Singapore for providing the incidents and traffic data from the expressways of Singapore.

References

[1] N. Kumar and M. Raubal, "Applications of deep learning in congestion detection, prediction and alleviation: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 133, p. 103432, 2021.

[2] C. Liu, S. Zhang, H. Wu, and Q. Fu, "A dynamic spatiotemporal analysis model for traffic incident influence prediction on urban road networks," *ISPRS International Journal of Geo-Information*, vol. 6, no. 11, p. 362, 2017.

[3] M. Miller and C. Gupta, "Mining traffic incidents to forecast impact," in *Proceedings of the ACM SIGKDD international workshop on urban computing*. ACM, 2012, pp. 33–40.

[4] C. Shahabi and B. Pan, "Accident impact prediction," *Encyclopedia of GIS*, pp. 1–9, 2015.

[5] Y. Lin and R. Li, "Real-time traffic accidents post-impact prediction: Based on crowdsourcing data," *Accident Analysis & Prevention*, vol. 145, p. 105696, 2020.

[6] Q. Shang, T. Xie, and Y. Yu, "Prediction of duration of traffic incidents by hybrid deep learning based on multi-source incomplete data," *International journal of environmental research and public health*, vol. 19, no. 17, p. 10903, 2022.

[7] M. Chen, G. Yu, P. Chen, and Y. Wang, "Traffic congestion prediction based on long-short term memory neural network models," in *CICTP 2017: Transportation Reform and Change—Equity, Inclusiveness, Sharing, and Innovation*. American Society of Civil Engineers Reston, VA, 2018, pp. 673–681.

[8] F. Ali, A. Ali, M. Imran, R. A. Naqvi, M. H. Siddiqi, and K.-S. Kwak, "Traffic accident detection and condition analysis based on social networking data," *Accident Analysis & Prevention*, vol. 151, p. 105973, 2021.

[9] H. Ren, Y. Song, J. Wang, Y. Hu, and J. Lei, "A deep learning approach to the citywide traffic accident risk prediction," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3346–3351.

[10] F. Sun, A. Dubey, and J. White, "Dxnat—deep neural networks for explaining non-recurring traffic congestion," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 2141–2150.

[11] X. Di, Y. Xiao, C. Zhu, Y. Deng, Q. Zhao, and W. Rao, "Traffic congestion prediction by spatiotemporal propagation patterns," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2019, pp. 298–303.

[12] N. Ranjan, S. Bhandari, H. P. Zhao, H. Kim, and P. Khan, "City-wide traffic congestion prediction based on cnn, lstm and transpose cnn," *IEEE Access*, vol. 8, pp. 81 606–81 620, 2020.

[13] B. Ghosh, "Predicting the duration and impact of the non-recurring road incidents on the transportation network," Ph.D. dissertation, 2019.

[14] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*. Springer, 2000, pp. 1–15.

[15] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms," *Computer Standards & Interfaces*, vol. 16, no. 3, pp. 265–278, 1994.

[16] W. Wu, S.-y. Chen, and C.-j. Zheng, "Traffic incident duration prediction based on support vector regression," *Proceedings of the ICCTP*, pp. 2412–2421, 2011.

[17] S. Kostadinov, "Understanding gru networks – towards data science," Dec 2017. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[18] B. Ghosh, M. T. Asif, J. Dauwels, W. Cai, H. Guo, and U. Fastenrath, "Predicting the duration of non-recurring road incidents by cluster-specific models," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1522–1527.

[19] "Keras: Deep learning library for theano and tensorflow." [Online]. Available: https://faroit.github.io/keras-docs/1.2.2/

[20] Tensorflow, "tensorflow/tensorflow," Sep 2018. [Online]. Available: https://github.com/tensorflow/tensorflow