Accident Analysis and Prevention 135 (2020) 105371

Contents lists available at ScienceDirect

Accident Analysis and Prevention

journal homepage: www.elsevier.com/locate/aap

# Real-time crash risk prediction on arterials based on LSTM-CNN*

Pei Li*, Mohamed Abdel-Aty, Jinghui Yuan

*Department of Civil, Environmental & Construction Engineering, University of Central Florida, Orlando, FL 32816, United States*

ARTICLE INFO

*Keywords:*
Real-time crash risk
Urban arterials
Recurrent neural network
Deep learning

ABSTRACT

Real-time crash risk prediction is expected to play a crucial role in preventing traffic accidents. However, most existing studies only focus on freeways rather than urban arterials. This paper proposes a real-time crash risk prediction model on arterials using a long short-term memory convolutional neural network (LSTM-CNN). This model can explicitly learn from the various features, such as traffic flow characteristics, signal timing, and weather conditions. Specifically, LSTM captures the long-term dependency while CNN extracts the time-invariant features. The synthetic minority over-sampling technique (SMOTE) is used for resampling the training dataset. Five common models are developed to compare the results with the proposed model, such as the XGBoost, Bayesian Logistics Regression, LSTM, etc. Experiments suggest that the proposed model outperforms others in terms of Area Under the Curve (AUC) value, sensitivity, and false alarm rate. The findings of this paper indicate the promising performance of using LSTM-CNN to predict real-time crash risk on arterials.

## 1. Introduction

In 2016, 7.4 % of the total deaths was caused by traffic accidents in the USA (Ritchie, 2019). To enhance traffic safety, many studies have been conducted to develop advanced traffic management system. One of its crucial components is the real-time crash risk prediction. Different from the traditional crash frequency prediction based on aggregated data, real-time crash risk prediction aims to predict crash probabilities during a short period (e.g. 15 min or 20 min). Recently, with the implementation of various intelligent transportation technologies, extensive real-time traffic data became available through fixed sensors and automated vehicle detection devices. All of these provide massive potential for predicting crash risk in real-time.

Among the existing studies about real-time crash risk prediction, most are limited to freeways (Oh et al., 2005; Abdel-Aty et al., 2012; Ahmed et al., 2012; Xu et al., 2013; Yu and Abdel-Aty, 2014) rather than urban arterials (Yuan and Abdel-Aty, 2018; Yuan et al., 2018). Traffic environment of arterials is much more complicated than freeways since the existence of the intersections. Thus, it is difficult to predict arterial crash risk only based on traffic flow parameters, signal timing need to be considered as well (Yuan et al., 2018). Furthermore, crash risk prediction is a typical binary classification problem since its output is a categorical event (crash or non-crash). One of the big challenges is the rareness of crashes. In the real life, non-crash events are much more common than crash events, which generates an extremely imbalanced dataset. Previous studies applied different resampling techniques to solve this problem. Matched case-control design is a traditional under-sampling method (Abdel-Aty et al., 2004). However, since the test data are created manually, it could somehow impair the model's performance on real-world data. Additionally, some valuable information of non-crash events may be lost during the training process. Synthetic minority over-sampling technique (SMOTE) is another method which does not suffer from the above problems (Yuan et al., 2019), since it is only applied to the training dataset. The test dataset can still reflect the real-world information. Besides, the characteristics of non-crash events are remained during the training procedure. In general, several types of SMOTE are available currently, including regular SMOTE (Basso et al., 2018; Parsa et al., 2019; Yuan et al., 2019), SVM SMOTE (Shi et al., 2019), and SMOTE + ENN (Batista et al., 2004). This paper selects regular SMOTE since its simplicity and good performance on the real-time crash risk prediction.

In general, two kinds of methods are available for real-time crash risk prediction, statistical and machine learning methods. Specifically, statistical methods include conditional logit model, log-linear model, logistic regression, etc. These models are usually built on matched-case control data and have strong assumptions (Shi and Abdel-Aty, 2015; Wang et al., 2015; Yu et al., 2016). Considering these limitations, machine learning methods become popular, such as Support Vector Machine (SVM) (Yu and Abdel-Aty, 2013), Random Forest (Lin et al., 2015), etc. Recently, with the rapid development of deep learning, it

* This paper has been handled by associate editor Tony Sze.

* Corresponding author.
*E-mail addresses:* peili@knights.ucf.edu (P. Li), M.Aty@ucf.edu (M. Abdel-Aty), jinghuiyuan@knights.ucf.edu (J. Yuan).

https://doi.org/10.1016/j.aap.2019.105371
Received 30 July 2019; Received in revised form 27 October 2019; Accepted 17 November 2019
Available online 26 November 2019
0001-4575/ Published by Elsevier Ltd.

was implemented to solve various transportation problems. Chen et al. (2016) developed a deep stack denoise autoencoder model to learn from hierarchical features of human mobility and predict traffic accident in aggregated way. Ma et al. (2017) utilized a deep CNN model to predict traffic speed in Beijing. Spatial and temporal traffic dynamic are converted to images describing the time and space relations of traffic flow. Results shown CNN outperformed other methods such as Random Forest and K-Nearest Neighbor. Moreover, RNN was proved to be especially useful for learning time-series data (Tian and Pan, 2015; Zhao et al., 2017; Zheng et al., 2019). Different from the traditional neural network that only maps the current input vector to output vector (Tian and Pan, 2015), RNN introduces recurrent connections, which allow information to persist. However, one drawback of the RNN is it cannot capture long-term dependency (Bengio et al., 1994). Thus, long short-term memory neural network (LSTM), was invented by Hochreiter and Schmidhuber (1997). LSTM improves the performance of RNN by including memory cells and gates, which preserve the information for a long period.

There are several existing studies that applied LSTM in the transportation field. Zhao et al. (2017) investigated short-term traffic forecast of Beijing based on LSTM, which considers temporal–spatial correlation in the traffic system via a two-dimensional network. The results of LSTM are better than other methods, such as autoregressive integrated moving average model and normal RNN. Similarly, Tian and Pan (2015) utilized LSTM to predict short-term traffic flow. With the ability to memorize long historical data and automatically determine the optimal time lags, LSTM outperformed others such as SVM and single layer feed forward neural network. There are currently few papers used LSTM for real-time crash risk prediction. For example, Yuan et al. (2019) utilized LSTM to predict crash risk in real-time, the authors claimed that their models achieved 60.67 % sensitivity, which was much better than the conditional logistic model.

Furthermore, hybrid neural network, which combines the strengths of different neural networks, draws more and more attention recently. Several previous studies combined LSTM and CNN to address temporal and spatial relationships, respectively. For example, Bao et al. (2019) implemented a spatiotemporal convolutional long short-term memory network to predict the citywide crash frequency based on multiple data sources, such as taxi trip data, road network attributes, and land use features. Similarly, Liu et al. (2017) used a LSTM-CNN model to predict short-term traffic flow. Ma et al. (2017) applied a convolutional LSTM network for traffic speed prediction in Beijing. Results from these studies proved the combination of LSTM and CNN achieved better results compared to single LSTM and CNN. However, the exploitation of LSTM-CNN is limited for time-series data, which has no spatial relationship (Lin et al. 2017). The LSTM and CNN can learn time series data in different ways. LSTM is specialized for learning long-term dependencies and sequential correlations (Zhou et al., 2015), while CNN can extract patterns of local trend and the same patterns which appears in different time (Lin et al., 2017; Tian et al., 2018). There are only few studies about the application of LSTM-CNN on time series data. Karim et al. (2018) designed a LSTM-CNN model for time series classification and test its performance on 85 UCR datasets. Results showed this model outperforms other benchmark methods with higher classification accuracy. Furthermore, Lin et al. (2017) proposed the TreNet model combined LSTM and CNN. The CNN model is used to extract salient features from local raw data, while LSTM is used to capture long-term dependency. TreNet also achieved better results for time series prediction compared to single CNN and LSTM. To conclude, the good performance of LSTM-CNN on time-series data is achieved by learning local trend and long-term dependency separately, which could complement each other. Real-time crash risk prediction involves lots of time-series data, such as traffic volume, signal timing, traffic speed, etc. The unique architecture of the LSTM-CNN can better capture the long-term and short-term characteristics of these data. Thus, it is promising to investigate the performance of LSTM-CNN on real-time crash risk prediction.

This paper aims to predict real-time crash risk on arterials through LSTM-CNN, the contributions of this paper can be summarized as below:

- As far as the authors' knowledge, this is one of the pioneer studies which explores the application of LSTM-CNN on real-time crash risk prediction.
- The possibilities of using various data sources for real-time crash prediction are explored. Such as Bluetooth data, detector data and weather data. One year's data are analyzed extensively. Different data preparation techniques are used.
- SMOTE is utilized to address the imbalance problem. It has been proven to be an efficient re-sampling method, which enables the LSTM-CNN to achieve better results.
- The performance of LSTM-CNN is compared with other benchmark approaches on the same dataset. Results suggest that the LSTM-CNN outperforms the others with various evaluation metrics, i.e., AUC, sensitivity and false alarm rate.

## 2. Methodologies

Three methods are introduced in this part, including LSTM, CNN, and LSTM-CNN. The characteristics of each method will be explained and compared. These models are trained based on the training data after over-sampling and evaluated through test data.

### 2.1. LSTM

LSTM is one kind of RNN. It is a powerful deep learning method for time series data since its unique design (Hochreiter and Schmidhuber, 1997). Generally, RNN suffers from the problem of vanishing gradients, which happens to learning of long data sequences (Aebel, 2018). LSTM can solve it by introducing the memory cell to determine when to forget certain information. A LSTM network is composed of the input layer, the hidden layers, and the output layer. The main characteristics of the LSTM are the memory cells in its hidden layers, which contain memory blocks rather than traditional neuron nodes (Olah, 2015). Each block has several self-connected memory cells and three multiplicative units, input, output and forget gates. These gates provide continuous analogues of write, read and reset operations on the cells. The structure of a LSTM unit at each time step is shown in Fig. 1.

The LSTM generates a mapping from an input sequence vectors $X = (X_1, X_2, \cdots, X_N)$ to an output probability vector by calculating the network unit activations using the following equations (Graves et al., 2013), iterated from $t = 1$toN:

$$i_t = \sigma(W_{ix}X_t + W_{ih}h_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}X_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$o_t = \sigma(W_{ox}X_t + W_{oh}h_{t-1} + W_{oc}c_t + b_o) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cx}X_t + W_{ch}h_{t-1} + b_c) \quad (4)$$

$$h_t = o_t \odot tanh(c_t) \quad (5)$$

$$y_t = W_{yh}h_{t-1} + b_y \quad (6)$$

Where $W$ represents weight matrices, for example, $W_{ix}$ denotes the weight matrix from the input gate to the input, $\sigma$ is the logistic sigmoid function and $\odot$ indicates elementwise product of the vectors. The forget gate $f_t$ controls the extent to which the previous step memory cell is forgotten, the input gate $i_t$ determines how much to update for each unit, and the output gate $o_t$ controls the exposure of the internal memory state. Since the values of all the gating variables vary for each time step, the model could learn how to represent information over multiple time steps.

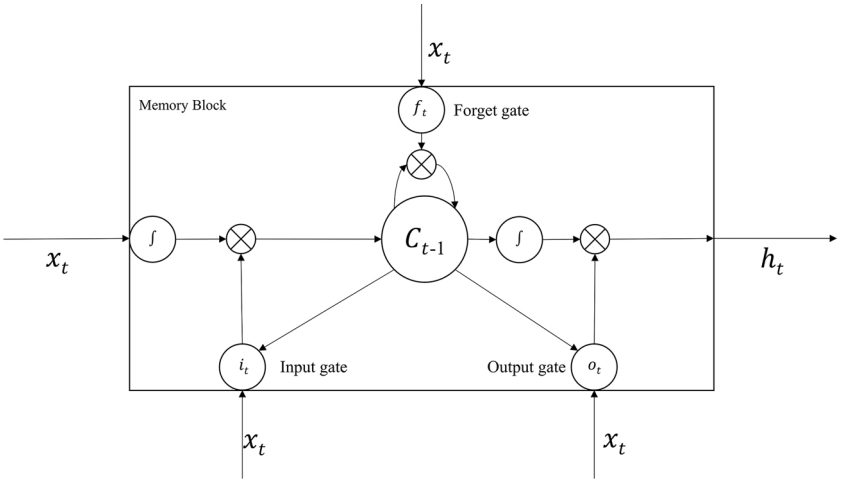



**Fig. 1.** LSTM unit structure (Graves et al. 2013).

### 2.2. CNN

Convolutional neural network (CNN) was originally developed for image classification problems, where the model learns an internal representation of a two-dimensional input, in a process referred to as feature learning (Krizhevsky et al., 2012). The same process can also be utilized for time series classification tasks and achieve good results (Cui et al., 2016; Wang et al., 2017).

The key component for CNN is its convolution layer, where CNN applies a filter to extract features through the input data. Since time series data have one dimension (time), the filter of CNN has also one dimension instead of two dimensions for image (width and height) (Fawaz et al., 2019). Features generated by a filter usually go through an activation function, such as Rectified Linear Unit (ReLU). The purpose is to introduce non-linearity since most of the real-world data do not always have the linear relationship. After these two procedures, the new features can go through multiple filters and activate functions. The generated features contain valuable information for the prediction or classification problem. There are other useful layers for CNN. For instance, pooling layer is used to reduce the number of parameters, while dropout layer can prevent the over-fitting problem. Overall, one big advantage for the application of CNN on time series data is its convolutional layer, where the filter is applied to the data across all the time stamps. This allows CNN to learn features that are invariant across the time dimension.

### 2.3. LSTM-CNN

Both LSTM and CNN have its unique characteristics. Thus, it becomes reasonable to combine them for better results. Previous studies proved the performance of LSTM could be improved by augmenting it with the CNN for time series classification problem. Karim et al. (2018) designed a LSTM-CNN network and applied it on University of California Riverside (UCR) Benchmark datasets. Results indicated that it can achieve state-of-the-art performance compared with other methods. The LSTM and CNN receive the same time-series data as input and their results are concatenated to generate output. The reason is LSTM is capable to learn long-term dependency and CNN can extract time-invariant features. Thus, the combination of them improve the overall accuracy.

The proposed model in this paper follows the similar logic of the previous studies. It has one LSTM part and one CNN part, the features extracted from them are combined to generate the final results. Its structure is elaborated in the following sections.

## 3. Data description and preparation

### 3.1. Data description

This paper focuses on the real-time crash risk prediction on urban arterials, more specifically, road segments. A segment is defined as the road facility between two consecutive intersections with the certain direction. Considering the availability of data, eight miles of arterials (38 segments, 21 intersections) are selected from four urban arterials in Orlando, Florida, as shown in Fig. 2. Four types of data are collected from September 2017 to September 2018, including crash data provided by Signal Four Analytics (S4A), signal timing, queue length, waiting time and traffic volume provided by the adaptive signal controllers (InSync), vehicle speed data collected through Bluetooth detectors (BlueMAC), and weather data of Orlando International Airport archived by the National Oceanic Atmospheric Administration (NOAA).

S4A provides detailed information for every crash event, including crash time, crash location, crash severity, and crash type. Since this paper only focuses on segment crashes, thus, the crashes within the intersection influence area (within 250 feet of intersection) are excluded from the original dataset. In addition, crashes that resulted from drugs and alcohol are also deleted, since these kinds of crashes are usually not attributed to real-time traffic and signal characteristics which are the focus of this study. After these processes, there are 110 crashes in total. They are allocated to the corresponding segments considering the location of the crash and direction of the segment.

Vehicle speed data are provided by BlueMAC. BlueMAC devices detect the vehicles equipped with Bluetooth devices which are working on discoverable mode, as shown in Fig. 3. The individual vehicular speed on a specific segment is calculated as the segment length divided by the travel time of each detected vehicle on this segment. In this paper, the Bluetooth penetration rate is 3.69 %, which is higher than the threshold suggested by the previous studies (Chen and Chien, 2000; Long Cheu et al., 2002). Also, the validity of Bluetooth detectors for measuring individual vehicular speed on urban arterials has been proven by previous research (Yuan and Abdel-Aty, 2018; Yuan et al., 2018).

InSync controller archived the real-time signal timing and lane-specific 15-minute aggregate traffic volume data. The lane-specific 15-minute aggregated traffic volume data are collected by the video detectors, which are installed for the adaptive signal controller to detect the real-time volume, queue length and waiting time. The lane-specific traffic volume for each minute is calculated based on the assumption that the traffic volume within the 15-minute interval is evenly distributed. Furthermore, since a segment has its direction, it has the upstream and the downstream intersections. Traffic features of the two

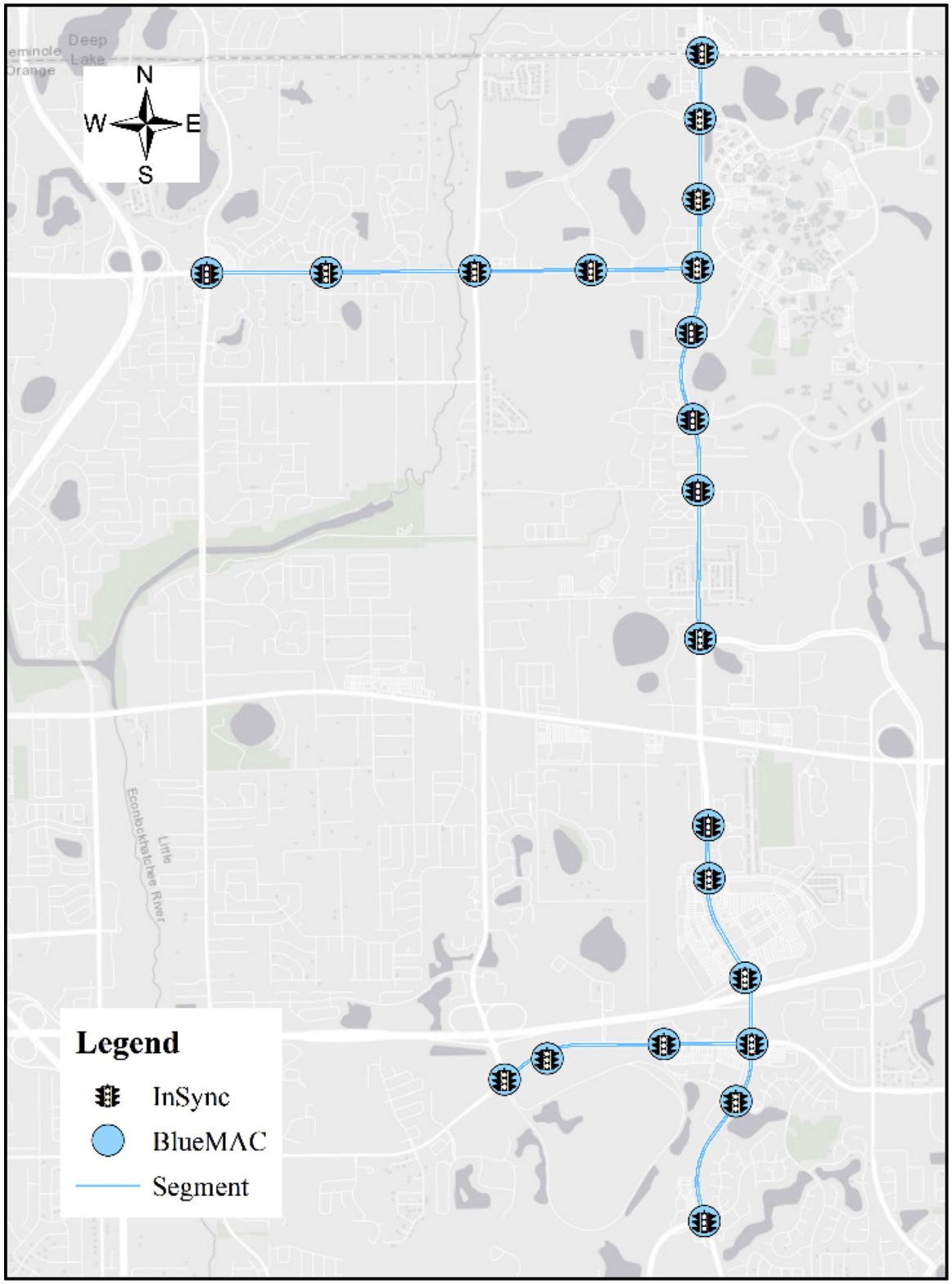



**Fig. 2.** Layout of Selected Segments.

intersections are included in the features of this segment.

Five weather-related variables are collected through NOAA. Since the study area is within 20 miles of the selected airport, the weather data is valid according to the previous research (Chung et al., 2018). In total, each segment has 24 features. Since this paper aims to predict real-time crash risk in five-minute time-slice and update every minute. All the statistical features are generated in five-minute interval, which are shown in Table 1. Overall, one segment has roughly 525,600 samples (365 days * 24 h * 60 min). It should be noted that this number is overestimated since some segments do not have the whole year's data.

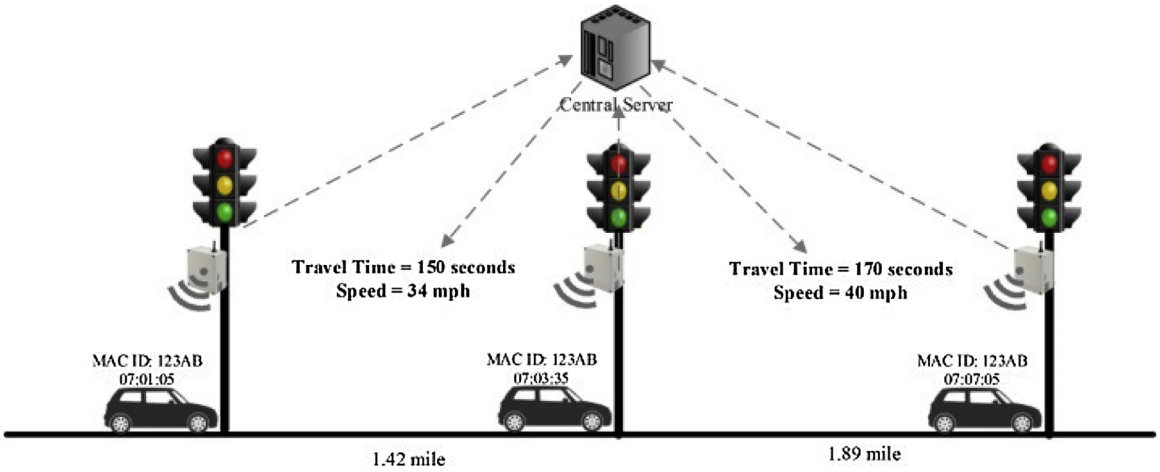



**Fig. 3.** Illustration of Bluetooth Data Collection (Yuan et al., 2019).

**Table 1**
Feature Description.

| Type | Feature | Description | Mean (Std) | (Min Max) |
|---|---|---|---|---|
| Traffic Data | speed_avg | Average speed on the segment in mph | 31.45 (11.60) | (0.66 80.00) |
| | speed_std | Speed standard deviation on the segment | 4.35 (5.85) | (0.00 69.38) |
| | up_lt_volume | Average left turn volume of upstream intersection | 4.18 (9.94) | (0.00 309.25) |
| | up_th_volume | Average through volume of upstream intersection | 45.94 (60.65) | (0.00 1363.00) |
| | down_lt_volume | Average left turn volume of downstream intersection | 6.93 (13.38) | (0.00 684.00) |
| | down_th_volume | Average through volume of downstream intersection | 43.74 (56.32) | (0.00 1208.00) |
| | up_lt_queue | Average maximum left turn queue length of upstream intersection | 2.76 (4.87) | (0.00 99.00) |
| | up_th_queue | Average maximum through queue length of upstream intersection | 1.30 (3.97) | (0.00 40.00) |
| | down_lt_queue | Average maximum left turn queue length of downstream intersection | 2.65 (4.92) | (0.00 99.00) |
| | down_th_queue | Average maximum through queue length of downstream intersection | 1.65 (3.68) | (0.00 40.09) |
| | up_lt_wait | Average maximum left turn wait time of upstream intersection | 9.97 (14.29) | (0.00 55.50) |
| | up_th_wait | Average maximum through wait time of upstream intersection | 8.71 (20.71) | (0.00 38.46) |
| | down_lt_wait | Average maximum left turn wait time of downstream intersection | 9.46 (13.91) | (0.00 63.96) |
| | down_th_wait | Average maximum through wait time of downstream intersection | 15.50 (17.79) | (0.00 37.72) |
| Signal Timing Data | up_lt_green_time_ratio | Ratio of left turn green time of upstream intersection | 5.39 (6.86) | (0.00 30.00) |
| | up_th_green_time_ratio | Ratio of through green time of upstream intersection | 33.49 (30.96) | (0.00 100.00) |
| | down_lt_green_time_ratio | Ratio of left turn green time of downstream intersection | 2.77 (5.37) | (0.00 40.00) |
| | down_lt_green_time_ratio | Ratio of through green time of downstream intersection | 32.52 (30.26) | (0.00 100.00) |
| Weather Data | visibility | Horizontal distance an object can be seen and identified given in miles | 9.64 (1.53) | (0.00 10.00) |
| | weathertype | Normal weather: 0. Abnormal weather data: 1 | 0.06 (0.24) | (0.00 1.00) |
| | humidity | Relative humidity given in percentage | 74.13 (20.54) | (9.00 100.00) |
| | precipitation | Amount of precipitation in inches to hundredths | 0.00 (0.04) | (0.00 1.99) |
| | temperature | Dry-bulb temperature in whole degrees Fahrenheit | 70.72 (11.45) | (0.00 94.00) |
| | windspeed | Speed of the wind at the time of observation given in mph | 8.47 (4.87) | (0.00 56.00) |

### 3.2. Data preparation

#### 3.2.1. Feature selection

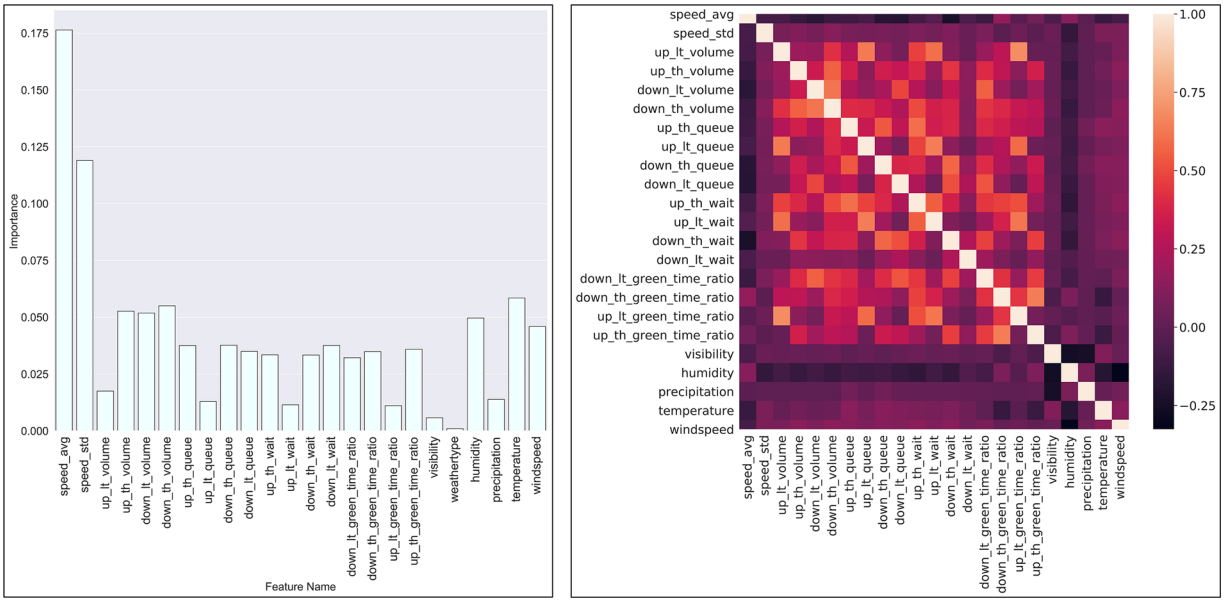
Feature importance and correlation were addressed based on extra-tree (Geurts et al., 2006) and Pearson correlation coefficient (Benesty et al., 2009). After normalizing all the features according to the maximum and minimum values, the results of feature importance and feature correlation are shown in Fig. 4, respectively. Extra-tree classifier implements a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting (Pedregosa et al., 2011). Pearson correlation coefficient measures the correlation between every pair of features from -1 to 1. It reflects the strength of the relationship between two variables. In general, a correlation is treated as strong if its absolute value is larger than 0.5 (Cohen, 1992).

Several features have high correlations with others, such as 'visibility' and 'humidity'. Moreover, some features have extremely lower importance than others, such as 'weather type' and 'visibility. A feature selection rule is made according to these two findings. A feature will be eliminated if it is highly correlated with another feature but are less important than the other one. In the end, 11 features were included in the final dataset, including speed_avg, speed_std, up_th_volume, down_th_volume, down_th_queue, down_lt_queue, down_lt_wait, up_th_green_time_ratio, humidity, temperature, windspeed.

#### 3.2.2. Crash labeling

In this paper, data of three time-slice are stacked to predict the crash risk during next 5 − 10 min based on LSTM-CNN. To create 'crash' label for samples in every minute, they are labeled according to actual crash events (Fig. 5). For instance, if a crash happens at 00:10, the time interval from 00:05 to 00:15 will be considered as the 'crash region'. Five crash regions can be generated for one crash, which correspond to five data samples, i.e., from 00:00 to 00:05. Then, the labels from 00:00 to 00:05 are defined as '1', the rest are labeled as '0' for non-crash.

After data preparation and crash labeling, there are 7098269 non-crash events and 432 crash events. The crash to non-crash ratio is around 1:16,431, indicating that the data are extremely imbalanced. There are many methods to solve this problem. For example, coping more minority classes or adjusting the cost function to make misclassification of minority classes more important than misclassification of majority instances. In this paper, a resampling technique is employed to tackle this problem. More specifically, over-sampling, which creates




**(a) Feature Importance** **(b) Feature Correlation**

**Fig. 4.** Feature importance and correlation.

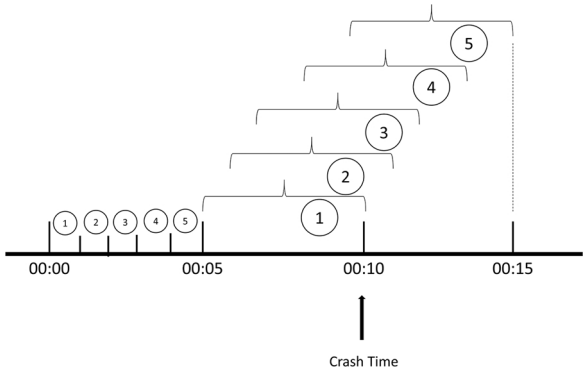



**Fig. 5.** Crash labeling illustration.

more samples for the minority classes. Synthetic Minority Over-sampling Technique (SMOTE) is a powerful method among existing studies (Chawla et al., 2002). SMOTE uses a nearest neighbors' algorithm to generate new and synthetic data. It synthesizes new minority instances between real minority instances. In this paper, we use SMOTE to generate synthetic samples of crash events to create an equal number between crash and non-crash events. Besides, SMOTE is only applied to the training data, while the test data are still the real-world data.

## 4. Experiments and results

### 4.1. Network architecture

The network architecture of our LSTM-CNN is shown in Fig. 6. It has two LSTM layers and two CNN layers. Moreover, dropout layers are added to prevent over-fitting, while average pooling layer is included for reducing the number of parameters. The LSTM and CNN are then combined together by a concatenate layer.

The input of the network is prepared with the shape as (S, T, F), where S is the sample size, T is the time-slice and F is the number of features. In this paper, we prepared the dataset as (S, 3, 11) since we stack the data of three time-slice and have 11 features. The CNN and LSTM modules receive data simultaneously and learned it separately. However, there are also several papers that indicated that the LSTM-CNN can be prepared in a sequential way (Zhang et al., 2018; Ma et al., 2019). Which means that the CNN module receives the input first and its results are then passed to the LSTM module. The performance of parallel and sequential LSTM-CNN would be compared in the following sections.

### 4.2. Experiment design and results

The experiment procedure is shown in. Firstly, the data are divided into training (75 %) and test (25 %). Secondly, the proposed model is trained based on training data after over-sampling. In the end, the trained model is evaluated according to the test data. Regarding model evaluation, AUC (Hanley and McNeil, 1982) is adopted in this paper, which is the area under the receiver operating characteristics (ROC) curve. Specifically, A ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: true positive rate and false positive rate, which can be found from Eqs. (7) and (8), respectively. AUC measures the entire two-dimensional area underneath the entire ROC curve from (0, 0) to (1, 1). In addition, sensitivity and false alarm rate are also included as other two metrics.

$$TPR = \frac{TP}{TP + FN} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

The model is implemented based on Keras (Chollet, 2015) using NVIDIA GTX 1050 4 G GPU. During the training process, one crucial step is to select the proper hyperparameters and optimization functions to achieve the best result. There are several optimization functions are available, such as Stochastic Gradient Descent (SGD), RMSProp, and Adaptive Moment Estimation (Adam). Besides, five common hyperparameters are tuned, including the filter size of CNN module, unit number of LSTM module, learning rate, epoch number, and batch size, which are summarized in Table 2.

After the model is fine-tuned. Its best results are shown in Table 3. To estimate the sensitivity and false alarm rate, the threshold of ROC curve is chosen as the point where sensitivity (true positive rate) equals to specificity (true negative rate). The proposed model achieves state-of-the-art results with high AUC value, high sensitivity, and low false alarm rate.

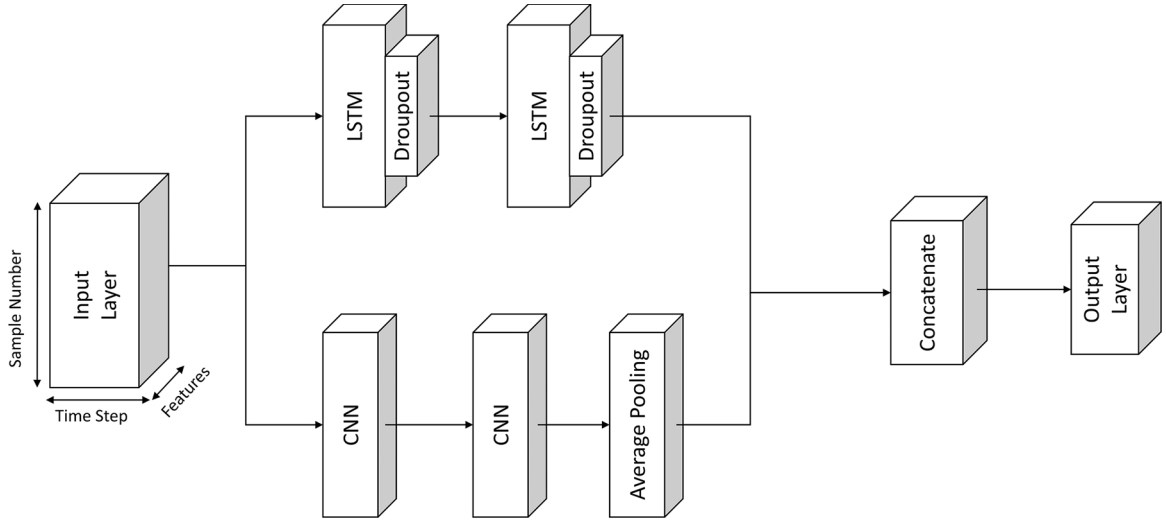



**Fig. 6.** Network architecture.

**Table 2**
Hyperparameters Tuning.

| Hyperparameter | Range | Value |
|---|---|---|
| Learning Rate | 0.0001, 0.001, 0.01 | 0.01 |
| Epoch Number | 50, 100, 150, 200, 500 | 100 |
| Batch Size | 1000, 5000, 10,000, 20,000 | 10,000 |
| CNN Filter Size | 128, 64, 32, 16 | 64 |
| LSTM Unit Number | 128, 64, 32, 16 | 16 |
| Optimization Function | Adam, RMSprop, SGD | Adam |

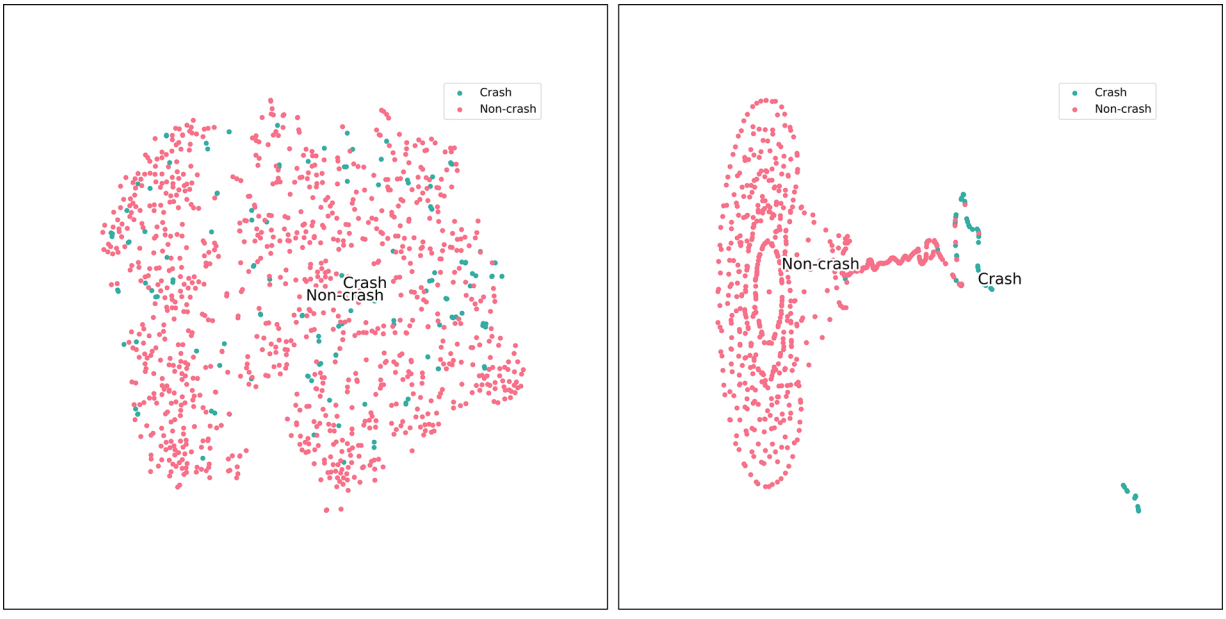
**Table 3**
Experiment results.

| Metric Name | AUC | False Alarm Rate | Sensitivity |
|---|---|---|---|
| **Value** | 0.932 | 0.132 | 0.868 |

Furthermore, to illustrate the model's capacity of distinguishing crash and non-crash events, t-Distributed Stochastic Neighbor Embedding (t-SNE) is introduced to visualize features for crash and non-crash events. t-SNE is a powerful tool for high-dimensional data visualization (Maaten and Hinton, 2008). It maps multi-dimensional data to two or more dimensions suitable for human observation. Specifically, t-SNE of the raw features is shown in Fig. 7(a), and t-SNE of the extracted features from the last layer of LSTM-CNN is shown in Fig. 7(b). Obviously, these two events are difficult to distinguish in the raw features, their patterns are critically tangled together (Fig. 7 (a)). However, the features extracted from LSTM-CNN successfully divide them. The two events become almost separable. But it is also worth to notice that some non-crash events still mix with crash events. The performance of the model can be improved in the future.

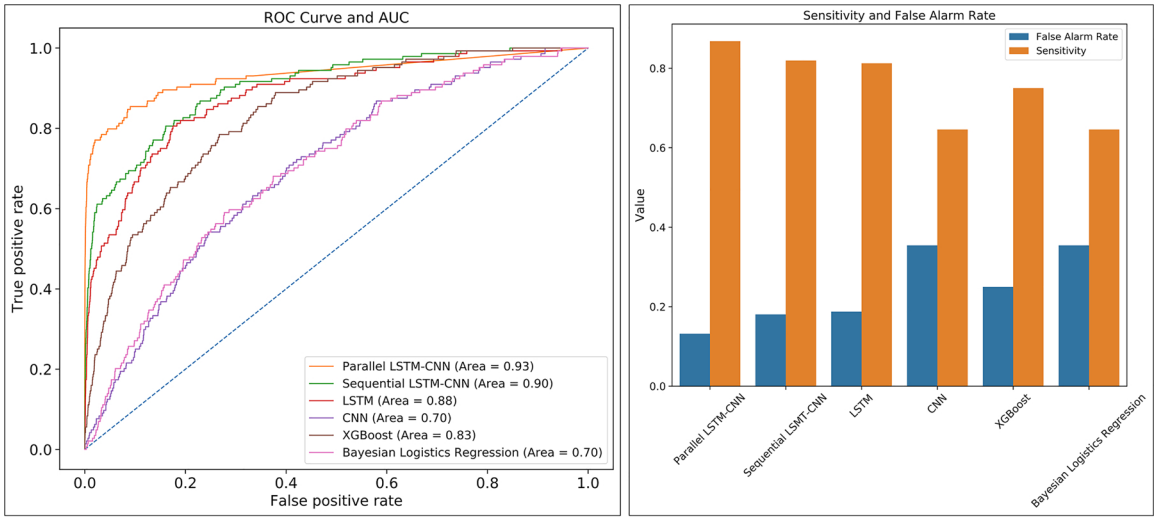
### 4.3. Model comparison

Five benchmark models are selected to compare the performance




**(a) Raw Features** **(b) Extracted Features**

**Fig. 7.** t-SNEs of raw features and extracted features.




**(a) ROC Curve and AUC** **(b) Sensitivity and False Alarm Rate**

**Fig. 8.** Model Comparison Results.

with the proposed LSTM-CNN, including Bayesian Logistic Regression, XGBoost, LSTM, CNN, and Sequential LSTM-CNN. These five models are trained and tested on the same dataset as the proposed model. However, some models cannot accept stacked data as input, such as the XGBoost and Bayesian Logistic Regression. Thus, the dimension of the input data is reshaped to (S, 3 × 11). In addition, feature selection is applied for Bayesian logistics regression considering the correlations between them. All these models are fine-tuned accordingly. Fig. 8(a) shows the ROC curves and AUC values of all the six models. The proposed LSTM-CNN has the highest AUC values of 0.93, which is better than the Sequential LSTM-CNN. Besides, Fig. 8(b) indicates the proposed LSTM-CNN has the highest sensitivity and lowest false alarm rate, which proves its good performance from another aspect. From the model comparison, several conclusions can be summarized as following:

(a) In general, the proposed LSTM-CNN outperforms the other methods in terms of AUC, sensitivity, and false alarm rate. The result confirms the feasibility and superiority of the proposed method, which can accurately predict the real-time crash risk on arterials. In addition, low false alarm rate is one of the most unique advantages of the proposed model.
(b) The performance of the LSTM can be enhanced by augmenting it with the CNN component. It is suggested to prepare the LSTM-CNN in a parallel way rather than a sequential way. Besides, CNN cannot accurately predict the real-time crash alone since it fails to capture the long-term dependency.
(c) Deep learning and machine learning models tend to have better results than statistical models. XGBoost reaches a relatively decent accuracy. Its performance on the real-time crash risk prediction can be exploited in the future.

## 5. Conclusions

This study applied LSTM-CNN to predict real-time crash risk on urban arterials. First, different data were collected for one year, such as traffic flow, signal timing and weather conditions. Second, raw data were cleaned and pre-possessed according to correlation and importance. Third, SMOTE was applied as an over-sampling technique to solve the imbalanced data problem. In the end, a LSTM-CNN model was built to predict real-time crash risk. Several benchmark models are implemented for comparisons with various evaluation metrics.

The results suggest that the proposed LSTM-CNN outperforms the others from several aspects. First, it achieves the highest sensitivity as 88 % and lowest false alarm rate as 12 %. Second, it has the highest AUC value as 0.93, which is much higher than other methods, including LSTM, CNN, XGBoost, Bayesian Logistics regression, etc. In the end, the t-SNE plot indicates the features extracted by the LSTM-CNN successfully distinguish crash and non-crash events, which illustrate the reason for its good performance.

Overall, this paper succeeds in verifying the possibilities to predict real-time crash risk based on deep neural network. Besides, the performance of LSTM can be enhanced by introducing CNN to extract features differently. SMOTE is proved as a useful over-sample method for large imbalanced crash dataset. The results of this paper can be used for the implementation of an advanced traffic management system, which has the potential to reduce crashes. However, there are still several limitations in the current study. First, the performance of LSTM-CNN can be improved by adding more layers or trying more combinations of different hyperparameters. In addition, geographical features can also be considered as possible inputs for the CNN part, which may help us improve the model's performance. Second, the impact of different time-slice can be tested, the way to prepare the data may influence the model's performance. Third, SMOTE is only one method for over-sampling, there are various methods for the same purpose, such as adaptive boosting and gradient tree boosting. The results of different over-sampling methods are worth to investigate. In the end, our dataset lack driver characteristics. Augmenting this dataset can significantly improve the performance of our model. However, it is hard to get such data. Connected vehicle technology may help in providing more comprehensive driving data in the future (Yue et al., 2018).

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

Abdel-Aty, M., Uddin, N., Pande, A., Abdalla, M.F., Hsia, L., 2004. Predicting freeway crashes from loop detector data by matched case-control logistic regression. Transp. Res. Rec. 1897 (1), 88–95.
Abdel-Aty, M.A., Hassan, H.M., Ahmed, M., Al-Ghamdi, A.S., 2012. Real-time prediction of visibility related crashes. Transp. Res. Part C Emerg. Technol. 24, 288–298.
Aebel, N., 2018. How lstm networks solve the problem of vanishing gradients. Medium.
Ahmed, M.M., Abdel-Aty, M., Yu, R., 2012. Assessment of interaction of crash occurrence, mountainous freeway geometry, real-time weather, and traffic data. Transp. Res. Rec.

2280 (1), 51–59.
Bao, J., Liu, P., Ukkusuri, S.V., 2019. A spatiotemporal deep learning approach for city-wide short-term crash risk prediction with multi-source data. Accid. Anal. Prev. 122, 239–254.
Basso, F., Basso, L.J., Bravo, F., Pezoa, R., 2018. Real-time crash prediction in an urban expressway using disaggregated data. Transp. Res. Part C Emerg. Technol. 86, 202–219.
Batista, G.E.A., Prati, R.C., Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. 6 (1), 20–29.
Benesty, J., Chen, J., Huang, Y., Cohen, I., 2009. Pearson correlation coefficient. Noise Reduction in Speech Processing. Springer, pp. 1–4.
Bengio, Y., Simard, P., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. 5 (2), 157–166.
Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.
Chen, M., Chien, S.I.J., 2000. Determining the number of probe vehicles for freeway travel time estimation by microscopic simulation. Transp. Res. Rec. 1719 (1), 61–68.
Chen, Q., Song, X., Yamada, H., Shibasaki, R., 2016. Learning deep representation from big and heterogeneous data for traffic accident inference. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, Phoenix, Arizona, pp. 338–344.
Chollet, F., 2015. Keras.
Chung, W., Abdel-Aty, M., Lee, J., 2018. Spatial analysis of the effective coverage of land-based weather stations for traffic crashes. Appl. Geogr. 90, 17–27.
Cohen, J., 1992. A power primer. Psychol. Bull. 112 (1), 155.
Cui, Z., Chen, W., Chen, Y., 2016. Multi-scale Convolutional Neural Networks for Time Series Classification.
Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.-A., 2019. Deep learning for time series classification: a review. Data Min. Knowl. Discov. 33 (4), 917–963.
Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Mach. Learn. 63 (1), 3–42.
Graves, A., Mohamed, A., Hinton, G., 2013. Speech recognition with deep recurrent neural networks. Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing 6645–6649.
Hanley, J.A., Mcneil, B.J., 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology 143 (1), 29–36.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.
Karim, F., Majumdar, S., Darabi, H., Chen, S., 2018. Lstm fully convolutional networks for time series classification. IEEE Access 6, 1662–1669.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Proceedings of the Advances in Neural Information Processing Systems 1097–1105.
Lin, L., Wang, Q., Sadek, A.W., 2015. A novel variable selection method based on frequent pattern tree for real-time traffic accident risk prediction. Transp. Res. Part C Emerg. Technol. 55, 444–459.
Lin, T., Guo, T., Aberer, K., 2017. Hybrid neural networks for learning the trend in time series. Proceedings of the Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence 2273–2279.
Liu, Y., Zheng, H., Feng, X., Chen, Z., 2017. Short-term traffic flow prediction with conv-lstm. Proceedings of the 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP) 1–6.
Long Cheu, R., Xie, C., Lee, D.-H., 2002. Probe vehicle population and sample size for arterial speed estimation. Comput. Civ. Infrastruct. Eng. 17 (1), 53–60.
Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., Wang, Y., 2017. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. Sensors 17 (4).
Ma, X., Zhang, J., Du, B., Ding, C., Sun, L., 2019. Parallel architecture of convolutional bi-directional lstm neural networks for network-wide metro ridership prediction. Ieee Trans. Intell. Transp. Syst. 20 (6), 2278–2288.
Maaten, L.V.D., Hinton, G., 2008. Visualizing data using t-sne. J. Mach. Learn. Res. 9 (Nov), 2579–2605.
Oh, J.-S., Oh, C., Ritchie Stephen, G., Chang, M., 2005. Real-time estimation of accident likelihood for safety enhancement. J. Transp. Eng. 131 (5), 358–363.
Olah, C., 2015. Understanding lstm networks.
Parsa, A.B., Taghipour, H., Derrible, S., Mohammadian, A., 2019. Real-time accident detection: coping with imbalanced data. Accid. Anal. Prev. 129, 202–210.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., 2011. Scikit-learn: machine learning in python. J. Mach. Learn. Res. 12 (Oct), 2825–2830.
Ritchie, H., 2019. Does the news reflect what we die from? Our World In Data.
Shi, Q., Abdel-Aty, M., 2015. Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. Transp. Res. Part C Emerg. Technol. 58, 380–394.
Shi, X., Wong, Y.D., Li, M.Z.-F., Palanisamy, C., Chai, C., 2019. A feature learning approach based on xgboost for driving assessment and risk prediction. Accid. Anal. Prev. 129, 170–179.
Tian, C., Ma, J., Zhang, C., Zhan, P., 2018. A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. Energies 11 (12), 3493.
Tian, Y., Pan, L., 2015. Predicting short-term traffic flow by long short-term memory recurrent neural network. Proceedings of the 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity) 153–158.
Wang, L., Abdel-Aty, M., Shi, Q., Park, J., 2015. Real-time crash prediction for expressway weaving segments. Transp. Res. Part C Emerg. Technol. 61, 1–10.
Wang, Z., Yan, W., Oates, T., 2017. Time series classification from scratch with deep neural networks: a strong baseline. Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN) 1578–1585.
Xu, C., Tarko, A.P., Wang, W., Liu, P., 2013. Predicting crash likelihood and severity on freeways with real-time loop detector data. Accid. Anal. Prev. 57, 30–39.
Yu, R., Abdel-Aty, M., 2013. Utilizing support vector machine in real-time crash risk evaluation. Accid. Anal. Prev. 51, 252–259.
Yu, R., Abdel-Aty, M., 2014. Analyzing crash injury severity for a mountainous freeway incorporating real-time traffic and weather data. Saf. Sci. 63, 50–56.
Yu, R., Wang, X., Yang, K., Abdel-Aty, M., 2016. Crash risk analysis for shanghai urban expressways: a bayesian semi-parametric modeling approach. Accid. Anal. Prev. 95, 495–502.
Yuan, J., Abdel-Aty, M., 2018. Approach-level real-time crash risk analysis for signalized intersections. Accid. Anal. Prev. 119, 274–289.
Yuan, J., Abdel-Aty, M., Gong, Y., Cai, Q., 2019. Real-time crash risk prediction using long short-term memory recurrent neural network. Transp. Res. Rec. 2673 (4), 314–326.
Yuan, J., Abdel-Aty, M., Wang, L., Lee, J., Yu, R., Wang, X., 2018. Utilizing bluetooth and adaptive signal control data for real-time safety analysis on urban arterials. Transp. Res. Part C Emerg. Technol. 97, 114–127.
Yue, L., Abdel-Aty, M., Wu, Y., Wang, L., 2018. Assessment of the safety benefits of vehicles' advanced driver assistance, connectivity and low level automation systems. Accid. Anal. Prev. 117, 55–64.
Zhang, J., Ma, X., Ding, C., Wang, Y., Liu, J., 2018. Forecasting Subway Demand in Large-scale Networks: a Deep Learning Approach.
Zhao, Z., Chen, W., Wu, X., Chen, P.C.Y., Liu, J., 2017. Lstm network: a deep learning approach for short-term traffic forecast. Iet Intell. Transp. Syst. 11 (2), 68–75.
Zheng, Z., Yang, Y., Liu, J., Dai, H., Zhang, Y., 2019. Deep and embedded learning approach for traffic flow prediction in urban informatics. Ieee Trans. Intell. Transp. Syst. 1–13.
Zhou, C., Sun, C., Liu, Z., Lau, F., 2015. A c-lstm Neural Network for Text Classification. arXiv Preprint arXiv:1511.08630.