Multi-Lane Short-Term Traffic Forecasting With Convolutional LSTM Network

YIXUAN MA\textsuperscript{1,2,3}, (Student Member, IEEE), ZHENJI ZHANG\textsuperscript{1,2}, (Member, IEEE), AND ALEXANDER IHLER\textsuperscript{3}, (Member, IEEE)

\textsuperscript{1}School of Management and Economics, Beijing Jiaotong University, Beijing 100044, China  
\textsuperscript{2}International Center for Informatics Research, Beijing Jiaotong University, Beijing 100044, China  
\textsuperscript{3}School of Information and Computer Science, University of California at Irvine, Irvine, CA 92617, USA  

Corresponding author: Zhenji Zhang (zhjzhang@bjtu.edu.cn)

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2018YJS052, in part by the Defense Advanced Research Projects Agency under Grant W911NF-18-C-0015, in part by the National Science Foundation under Grant IIS-1254071, and in part by the Beijing Social Science Fund under Grant 16JDGLA010.

\textbf{ABSTRACT}  
Short-term traffic prediction consists a crucial component in intelligent transportation systems. With the explosion of automated traffic monitoring sensors and the flourishing of deep learning techniques, a growing body of deep neural network models have been employed to tackle this problem. In particular, convolutional neural networks (CNN) and long short-term memory (LSTM) recurrent networks have demonstrated their advantages in modeling and predicting the spatiotemporal evolution of traffic flows. In this paper, we propose a novel Convolutional LSTM neural network architecture for multi-lane short-term traffic prediction. Compared to existing methods, we highlight the importance of (1) applying multiple features to characterize traffic conditions; (2) explicitly considering the routing between neighbouring lanes and downstream/upstream traffics; and (3) predicting multiple time-step traffic in a rolling-prediction manner. Experiments on 10 months 5-minute interval observations of the US I-101 Northern freeway at California Bay Area verify the proposed model. The results show that our model has considerable advantages in predicting multi-lane short-term traffic flow.

\textbf{INDEX TERMS}  
Short-term traffic flow prediction, multi-lane traffic flow, convolutional LSTM.

\section{I. INTRODUCTION}
Accurate, in-time, and detailed traffic prediction plays a critical role in intelligent transportation systems (ITS). Such predictions help to plan and guide vehicle routing, increase the reliability and efficiency in road networks, and alleviate congestion and their related problems [1]. Consider a multi-lane freeway, which is a relatively simple subsystem of a traffic network, the spatiotemporal evolution of its traffic status results from individual traveler’s routing option and their dynamical interactions. Although it is difficult to predict traffic evolution at long range due to the highly variant traffic loading and releasing, it is feasible to make short-term traffic predictions based on current traffic status. Typically, the short-term forecasts are within the range of 5 to 30 minutes.

Short-term traffic forecast has been an important task in the transportation research community since 1970s.

Traffic flow data are often treated as time series or signal data with implantation of conventional time series models, such as autoregressive (AR) model, moving average (MA) model, auto regressive integrated moving average (ARIMA) model [2]–[4], Hidden Markov Model [5], and Bayesian network [6], [7]; or signal processing model, such as Kalman filter [8]. Some variant of these models and their combinations have also been applied for short-term traffic predictions, such as seasonal ARIMA model [9], KARIMA model [10], and ARIMAX model [11]. Other time series models can also be applied in this problem [12], [13]. While these classical models make strong assumptions of traffic evolution, such as stationarity and linearity, observation-based investigations have found that real-world traffic flows exhibit complex nonlinear behavior, and are usually non-stationary regarding the geographical locations, traffic hours, weather, and unexpected incidents [14].

The recent decades have witnessed a flourishing of automated traffic monitoring sensors. These sensors provide detailed and comprehensive picture of the traffic status.
The massive traffic observation data offer unprecedented opportunity to examine and analyze the traffic system, and make it possible to build more sophisticated, data-driven models for more accurate predictions.

Machine learning algorithms build empirical functions by learning from data. Provided with large amount of training data, these models usually show advanced prediction capacity compared to classical statistical models [15], [16]. Many machine learning techniques have been employed for short-term traffic prediction, such as support vector regression (SVR) [17], [18], k-nearest neighbor algorithm (KNN) [19], fuzzy logic [20], [21], and artificial neural network (ANN) models [14], [22], [23].

Compared with classical machine learning algorithms, deep learning models have shown advantage in scaling well with data availability, i.e., deep neural networks usually make better use of massive amount of data by learning customized feature representations. In particular, convolutional neural networks (CNN) and long short-term memory (LSTM) recurrent neural networks (RNNs) have demonstrated their peculiar advantages in modeling and predicting spatiotemporal data. Recently, such models have received extensive attention in the field of traffic flow prediction [24]–[31].

Although many researchers have noticed the advantage of the above mentioned modeling techniques, most studies limit on traffic prediction with single input of lane-average information. In traffic flow theory, traffic flow, speed and occupancy are correlated [32]. It is hard to learn and model traffic flow without speed or occupancy information. Besides, traffic patterns can be distinct for different lanes [33]: traffic flow tends to be high in the median lane of the freeway, compared to shoulder lanes [34]. Drivers may frequently route between neighbouring lanes to alleviate traffic over-headings. The local interactions can aggregate to influence the overall evolution of traffic flows. By explicitly considering the local traffic pattern between neighbouring lanes and along traffic streams, we may better capture traffic evolution patterns and make more accurate predictions. These understandings bring potential benefits for better modeling of traffic evolution, but there still lies a considerable gap in utilizing these understandings for better traffic predictions.

To fill the gap, we propose a novel convolutional LSTM recurrent neural network architecture for multi-lane short-term traffic flow prediction. The model uses CNN to extract routing patterns of neighbouring lanes and downstream/upstream traffics. The extracted features at each time step are connected using a LSTM module, allowing the model to simulate the temporal dynamic behavior of traffic flows.

This model allows using multiple features spatiotemporal input to characterize traffic conditions at lane scale. The key contributions of this paper could be summarized as follows:

- To the best of our knowledge, this paper is the first attempt to introduce convolutional LSTM architecture for multi-lane short-term traffic prediction.
- Multiple features are used in the model to characterize traffic conditions with a high prediction accuracy.
- Learning traffic patterns from neighbouring lanes and downstream/upstream traffics can provide better results in a transportation system. It is demonstrated that predicting multi-lane traffic flow, outperforms predicting average flow of all the lanes.
- The proposed model can make multi-step traffic prediction in a recursive manner.

The rest of this paper proceeds as follows. Section II reviews deep learning based traffic prediction works. Section III presents our proposed model. Section IV shows the experiments and results. Discussion and conclusion are drawn in Section V.

### II. RELATED WORKS

Short-term traffic flow prediction has a long history in the transportation literature. Many existing works have made comprehensive review of classical parametric modeling methods and the relatively novel machine and deep learning methods. The latter has demonstrated their advantage in leveraging large observation data for accurate predictions. Here, we only review the most relevant deep learning based short-term traffic prediction works.

The stacked autoencoder (SAEs) model was used to predict short-term traffic flow and achieve better results than parametric models such as ARIMA [23]. Similarly, a deep model of stack denoise autoencoder was developed to learn hierarchical feature representation [36]. A deep belief network (DBN) was used to capture the spatial-temporal features of traffic flow and a multi-task learning architecture to perform exit station flow and road flow forecasting [37]. In [38], two DBN architectures have been compared for short-term traffic flow forecasting, and they show that DBN based on restricted Boltzmann machines (RBMs) that have Gaussian visible units and binary hidden units outperforms the DBN based on RBMs with all units being binary. In [39], DBN has also been explored for longer term (i.e., day ahead) traffic prediction. Furthermore, Generative Adversarial Networks (GANs) have been used to estimate trip travel time distribution [40]. A deep neural network was used to capture nonlinear spatiotemporal effects [14]. These models can make accurately future traffic flow prediction to some degree, but they did not exploit long-term memory of traffic flow, which hindered their predictive power.

RNNs are neural networks with hidden states, which are well suited for modeling sequential data with dynamic behavior. In the context of traffic prediction, researchers in ITS treat the traffic evolution as a dynamical system, and apply RNNs to model this system [41]. However, the training of a vanilla RNN has been demonstrated to be difficult due to the vanishing/exploding gradient problem [42], [43]. LSTM architecture with recurrent gates called “forget” gates has made it possible to train deep recurrent neural networks efficiently, and shows excellent ability with regard to traffic prediction tasks [24], [44]–[49]. In addition, LSTM has been combined with DBN to model lane changing process [48]. Other gating mechanisms, such as gated recurrent units mechanism, are
adopted to predict network wide traffic [50]. Even though the studies above claimed that LSTM can characterize the time correlation of the traffic data and capture long-term memory of traffic flow, the spatial dependency is not fully utilized.

CNNs have succeed in many research field, including image recognition [51], [52], video analysis [53], [54], natural language processing [55], [56], and weather prediction [57]–[59]. Owing to the good performance of the CNN in capturing spatial features, some researchers in ITS use CNN for traffic forecasting task [27], [29], [31], [60], [61]. In most of their model architectures, the input traffic data have the spatial correlation in the column, and the temporal correlation in the row. In [29], multiple features are considered as channels of input, and output is 1D traffic speed at each station. In [31], multi-lanes are seen as channels of the CNN input, and output is 2D traffic speed at each station at each lane.

The combination of CNNs and LSTM RNNs model can effectively improve the accuracy of traffic prediction tasks, where CNN can capture spatial dependencies and LSTM can learn temporal features. There are different variations in formulating the problem, customizing and applying these modules. For instance, the model structure in [25] includes a 1D CNN for capturing the spatial features of traffic flow, two LSTM RNNS for learning temporal features, and fully connected layers for combining the output features. The model takes in average flow of all the lanes and treats time steps as CNN channels. In their following work [28], speed data were used to learn the weights of the models for flow prediction. Compared these two studies on freeway, Yu et al. [35] focused on regional network-wide traffic with hundreds of interchanges and intersections. The architecture they proposed consists a 2D CNN, two LSTMs, and a fully connected layer. The model inputs 2D average speed networks which is represented by grid-based transportation network segmentation methods, and outputs 1D traffic speed at each links.

The works that most related to us are summarized in Table 1. The table lists predictand, data source, samples, input variable, input dimension, output dimension, lane and model of each work. Most existing models limit on one step traffic prediction with single input/output of lane-average information.

Our approach is different from all those methods due to both the problem settings and the formulation of the combination of CNN and LSTM models. We model different attributes of the lane-level traffic at the same time. We apply convolution operator to explicitly seek spatial invariant patterns from the array-shaped data. In the context of traffic prediction, where we treat multi-lane traffic status as a 2D matrix, such local features are preferably to be particular traffic flow patterns that involve the interactions of several neighbouring vehicles. Besides, the sequence to sequence learning framework is different than previous studies. As shown in Figure 1, almost existing research adopted sequence to sequence LSTM model (diagram (a)), which use observations at sequence time step to make a multi-step forecast. The hidden state is accumulated with each input value before the first output value is produced. However, we use synced sequence to sequence LSTM (diagram (b)) which produces one time step forecast for each observed time step received as input, and the information are passed from one step to the next.

III. METHODOLOGY
A. PROBLEM STATEMENT
Consider a traffic monitoring system that observes the traffic flow along multiple lanes of a freeway with a fixed observing frequency. At each observation time step, we monitor \( c \) attributes of the traffic condition at \( h \) observation stations along the considered freeway. Each observation is made for all the \( w \) lanes. We treat the spatial and temporal dependent data as a sequence of spatial signals. Specifically, we apply a \( n \times c \times h \times w \) tensor \( T_n \) to represent the traffic data, where \( n \) denotes total observing time steps, \( c \) denotes observed attributes, \( h \) denotes number of stations, and \( w \) denotes number of lanes:

\[
T_n = [X_1, \ X_2, \ldots, \ X_n] 
\]  

where \( X_i \) represents the traffic status at time step \( i \). \( X_i \) is a \( c \times h \times w \) tensor:

\[
X_i = \begin{bmatrix}
  x_{i1} & x_{i2} & \ldots & x_{1w} \\
  x_{i1} & x_{i2} & \ldots & x_{2w} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{ih1} & x_{ih2} & \ldots & x_{ihw}
\end{bmatrix}
\]  

where each \( x_{ij}^{hk} \) is a \( c \) dimensional vector, representing \( c \) traffic attributes at time step \( i \), observation station \( j \), and lane \( k \).
TABLE 1. Summary of related works regarding deep neural networks for learning the spatial-temporal patterns for traffic prediction. The models include CNN, RNN or their combinations. The predictand of each work is flow/speed for freeway/urban traffic with single/multiple variable input(s). The input variable considers average value of all the lanes or distinct value for each lane. The input dimension of most models is 1D (number of stations) or 2D (number of stations/links by time step). The output dimension of most work is 1D (number of stations/links). The data source and data samples of each work are listed as well.

| Predictand | Data Source | Samples | Input Variable | Input Dimension | Output Dimension | Lane (average/multiple) | Model | Type | Ref. |
|------------|-------------|---------|----------------|-----------------|-----------------|------------------------|-------|------|------|
| flow       | PeMS a      | 1-405N  | flow           | # stations (1D) time step as channel | # stations average | CNN&LSTM | freeway | [25] |
| flow       | PeMS        | 1-405N  | flow speed     | # stations (1D) time step as channel | # stations average | CNN&GRU | freeway | [28] |
| speed      | Beijing traffic GPS networks data | 278 links 2-min interval 06/01/15-08/31/15 44,230 samples | speed | grid-based network segmentation (2D) | # links average | CNN&LSTM | urban | [35] |
| speed      | KDD Cup 2017 dataset | 20-min interval 07/19/16-10/17/16 1,527,136 samples | flow weather | # links x time step (2D) | # links average | embedding CNN&LSTM | urban | [30] |
| speed      | Beijing taxi GPS networks data | 2-min interval 05/01/06-06/06/2015 | speed | # stations x time step (2D) | # stations average | CNN | urban | [27] |
| speed      | PeMS        | 1-5     | flow speed occupancy | # stations x time step x # attributes (channel) | #stations average | attention CNN | freeway | [29] |
| speed      | Seattle freeway four lanes | 40 loop detectors 5-min interval 105,000 samples | flow speed | # stations x time step x # lanes (channel) | # stations x # lanes multiple | CNN | freeway | [31] |
| speed      | Beijing microwave sensor data | 06/01/13-06/30/13 2-min interval 42,387 record | speed volume | speed time series volume time series | speed time series average | LSTM | urban | [24] |
| speed      | GCM b       | 1-35N   | speed          | # stations x time step (2D) | # stations average | multilayer perceptron | freeway | [14] |

* Caltrans Performance Measurement System
b The Gary-Chicago-Milwaukee Corridor, http://www.travelmidwest.com

The goal of our short-term traffic flow forecasting task is to predict the one-step future traffic flow spatial signals based on the current and historic spatial signals. Each signal consists multiple aspect of the traffic, such as traffic flow and speed. We formalize forecasting problem as learning a function \( h(\cdot) \) that maps \( n \) historical spatial signals to \( (n+1) \)th step future spatial signal, as Equation 3.

\[
T_n h(\cdot) \rightarrow X_{n+1} \tag{3}
\]

With \( X_{n+1} \) being predicted, we could have \( T_{n+1} \) estimation by appending \( X_{n+1} \) to \( T_n \). Thus, we can iteratively apply Equation 3 to make multiple time step traffic predictions.

B. BUILDING BLOCK I: CONVOLUTIONAL NEURAL NETWORKS

This section describes the convolutional neural network that searches, extracts, and synthesizes spatial features from the traffic input data for traffic flow estimation. The model here makes explicit use of the traffic neighborhood lanes and upstream/downstream spatial structure but does not consider temporal evolution of the traffic. Details about the convolutional neural network module are described as follows.

The traffic input is denoted with a \( n \times c \times h \times w \) tensor. Here \( n \) is the duration time of traffic information, \( c = 2 \) represents 2 variables (flow and speed), \( h = 4 \) is the four lanes of the freeway, and \( w = 125 \) is the detectors span of the predictor.
field. The CNN model operates on the 3-D traffic data at each
time step. As illustrated in Figure 2, the CNN applies a set
of convolution kernels to go through the $c \times h \times w$ input
snapshot. The kernels are $c' \times c \times a \times b$ tensors composed of
 trainable parameters. Each convolution operation is carried
out by computing the dot product between the kernel and a
tensor with a receptive field of $a \times b \times 1$, resulting in $c_2$ local features. Higher level convolutions operate on the output from lower level convolutions,
enabling the model to capture multi-scale spatial features at one time step. 

A LSTM recurrent module is applied to join the extracted spatial features,
as shown by the green arrows. At each time step, the model predicts 5-minute ahead traffic flow and speed, based on (1) current traffic flow and speed, (2) hidden state $h_t$ from the LSTM module.

**C. BUILDING BLOCK II: RECURRENT NEURAL NETWORK**

The temporal connection of traffic information can be extracted using a recurrent neural network (RNN) module after spatial features extraction from CNN. RNN can take advantage of all the available input information up to the current time. It can be seen that the final output depends not only on the current input but also on the output of previous hidden layer. Basics about RNN are briefly introduced below.

RNNs are neural networks with hidden state variables:

$$h_t = f(h_{t-1}, x_t, \theta)$$

where $h_t$ denotes the hidden state variable at time $t$, $f$ is the transition function that updates the hidden state $h_t$ based on previous state $h_{t-1}$ and $t$-step input $x_t$. $\theta$ denotes parameter vector. For instance, in a vanilla RNN, the state transition function takes the following parametric form:

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$

We can unfold Equation 5 along time by iteratively applying Equation 5 on itself:

$$h_t = f(h_{t-1}, x_t, \theta) = f(f(h_{t-2}, x_{t-1}, \theta), x_t, \theta) = f(f(f(h_{t-3}, x_{t-2}, \theta), x_{t-1}, \theta), x_t, \theta)$$

Gradient-based training of RNN has been recognized difficult for basic recurrent architectures, such as for Equation 6. The difficulty originates from the fact that, the partial derivatives of the loss function $L$ with respect to the parameters tend to vanish or blow up as error signals flow backwards in time [42], [43]. Although it is possible to clip the gradients as they blow up [62], a vanishing gradient prevents effective
learning of long-term dependencies. Here we adopt the Long Short-Term Memory (LSTM) recurrent network [63] to mitigate the vanishing gradient problem.

A typical LSTM unit contains an input gate $i$, a forget gate $f$, an output gate $o$, and two hidden states in its computational graph, one is the conventional hidden state vector $h_t$, the other is a memory vector $c_t$. LSTM adopts four interacting modules to read from, write to, or reset the memory vector $c_t$:

$$
\begin{pmatrix}
  i \\
  f \\
  o \\
  g
\end{pmatrix}
= 
\begin{pmatrix}
  \sigma \\
  \sigma \\
  \sigma \\
  \tanh
\end{pmatrix} W \begin{pmatrix}
  h_{t-1} \\
  x_t
\end{pmatrix}
$$

where $\sigma / \tanh$ is element-wise sigmoid/hyperbolic-tangent function that squashes each element of a vector to $(0, 1)/(-1, 1)$. Assume the input vector $x$ of dimension $p$, and hidden state vector $h$ as well as memory vector $c$ of dimension $q$, $W$ is thus a $(4q, p+q)$-dimensional transition matrix, $i$, $f$, $o$, and $g$ are $q$-dimensional vectors. With these modules defined, the memory vector $c_t$ and $h_t$ are updated based on the following equations:

$$
c_t = f \odot c_{t-1} + i \odot g
$$

$$
h_t = o \odot \tanh(c_t)
$$

where $\odot$ indicates element-wise product. Equation 9 and 10 tells how $i/g, f$, and $o$ work as binary gates that control whether memory cell $c_t$ is updated, whether it is reset to zero, and whether its local state is revealed in the hidden vector, respectively [64]. The memory cell $c_t$ is updated based on two mechanisms that cooperates in an additive manner. The first mechanism tells how $c_t$ maintains memory from past state, which is described by $f \odot c_{t-1}$; the second mechanism tells how to update $c_t$ based on input $x_t$ and past hidden state $h_{t-1}$, which is described by $i \odot g$. The additive interaction of the two mechanisms allows error gradient on $c_t$ to be distributed through time without suffering from vanishing/blowing up gradient, thus enabling learning long-term dependencies [64].

We use LSTM to relate the extracted spatial feature to the corresponding traffic flow time sequences. Specifically, We start from mapping a CNN to the time series of predictors. The extracted spatial feature time series are then used as input for the LSTM for flow estimation.

D. ENTIRE ARCHITECTURE: CNN-LSTM

The high-level architecture of our model is shown in Figure 2. The model works by channeling various modules introduced in the previous sections. Specifically, at each 5-minute time interval (one time step), we have an observation represented as a $c \times h \times w$ tensor. $c$ denotes the features that are applied to characterize the traffic status, here 2 features are adopted, namely the traffic speed and flow; $h$ denotes the station number, $w$ denotes the lane number. To extract the spatiotemporal features, a convolution kernel of size $c_2 \times c_1 \times a_1 \times b_1$ is applied to scan through the observation tensor with a receptive field of $a_1 \times b_1$. Each convolution operation results in $c_2$ local features. The output maintains the spatial alignment of the lane and downstream/upstream traffic. Multiple convolutional layers are stacked. Higher level convolutions operate on the output from lower level convolutions, enabling the model to capture multi-scale spatial patterns. A LSTM recurrent module is applied to chain the extracted spatial features, as shown by the green arrows. At each time step, the model predicts 5-minute ahead traffic flow and speed, based on (1) current traffic flow and speed, (2) hidden state $h_t$ from the LSTM module.

IV. EXPERIMENTS

A. DATASET

We use traffic data collected from the Caltrans Performance Measurement System (PeMS). The data are collected every 30 seconds from over 35,000 individual single lane loop detectors, which are deployed statewide in freeway systems across California. The collected data are aggregated to 5-minute interval each for each detector station, and one detector preserves 288 data points per day. We select District 4 (US I-101 North) from 00:00 January 1st, to 23:55 January 31st, and 00:00 October 1st, 2018 to 23:55 June 30, 2019 by 125 sensors (detectors) as experiment samples. We select average flow and average speed of each lane for our experiments. The flow is the number of vehicles that crossed the detector during the time period. The speed is the average speed (km/h) that traffic is traveling, which calculated using g-factor (effective vehicle length) from the flow and occupancy data. The detailed algorithm for speed estimation can be found in [65].

To offer an impression of the data, the average traffic flow and speed for each lane of 125 stations of freeway US I-101 North during October 1st, 2018 to June 30th, 2019 are given in Figure 3. The average flow/speed are labeled with circles of corresponding size. The figure shows that outer lane has relative low flow and low speed, and lane 2 has relative high flow and inner lane has relative high speed, indicating that different lanes show distinct traffic behavior and it is reasonable to forecast traffic on lane-level.

B. MODEL IMPLEMENTATION

1) DATA PREPARATION

We start by structuring the observational data into the format of $n \times c \times h \times w$, where $n$ denotes total time steps, $c$ denotes observed attributes, $h$ denotes station numbers, and $w$ denotes lane numbers. It is worth noticing that the station sequence of the data should be rearranged based on the real-world station alignment. After data re-aligning, the flow and speed data are scaled to $[0, 1]$ using a min-max normalization. Thereafter, we divide the data into non-overlapping training/validation/test sets. Data from October 2018 to May 2019 are used for model training and validation(128 days × 288 5-min-observations/day). Data for January 2018 and June 2019 are used for test.

http://pems.dot.ca.gov
2) MODEL ARCHITECTURE AND HYPERPARAMETERS

Figure 2 offers a high-level illustration of the network model. The network architecture and model hyperparameters should be further specified. Here we rely on empirical experiments to determine network architecture and hyperparameters. For implementation convenience, we only consider equal convolution kernels (same channel size and receptive field) for CNN. We try different network architecture, learning rate, and training iterations to decide the optimal setting. We admit that the result could be significantly different by adopting alternative architectures. The specific network architecture and hyper-parameters options are listed in Table 2.

3) MODEL TRAINING

The model is trained using mini-batch gradient descent strategy. Each mini-batch is composed of $B$ truncated traffic sequences. $B$ denotes batch size, which is set to 10 here. We apply a truncate time of 1 day (288 5-minute steps). For each sequence in a mini-batch, the model error is calculated at each time step based on (1) the state updated from previous step (for the first time step, the state is randomly initialized), and (2) network activation for the current step. Then, we calculate and back-propagate the model errors gradients for each time step, based on the current step error and next step gradient [66]. The gradients for each time step are averaged. This calculation is carried out for $B$ sequences, and the averaged gradients is accumulated to update the model parameters.
The learning rate $r$ is used to control how much the parameters are adjusted along the gradient direction. The considered $r$ options are listed in Table 2.

4) MODEL IMPLEMENTATION DETAILS

We apply the Wolfram Mathematica 12.0 [67] deep learning library to implement the model. This library offers off-the-shelf network building blocks, and can automatic differentiate the network, enabling efficient and effective computing. We split the training process across 3 NVIDIA Titan X (Pascal) GPUs using the Common Unified Device Architecture (CUDA) library [68].

C. PERFORMANCE METRICS

The flowing four widely used performance metrics are used to evaluate the models:

\[
\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_{\text{obser}}^i - y_{\text{simu}}^i| \tag{11}
\]

\[
\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_{\text{obser}}^i - y_{\text{simu}}^i)^2} \tag{12}
\]

\[
\text{NSE} = 1 - \frac{\sum_{i=1}^{N} (y_{\text{obser}}^i - y_{\text{simu}}^i)^2}{\sum_{i=1}^{N} (y_{\text{obser}}^i - y_{\text{mean}}^i)^2} \tag{13}
\]

\[
\text{CORR} = \frac{\text{cov}(y_{\text{obser}}, y_{\text{simu}})}{\sigma_{y_{\text{obser}}} \sigma_{y_{\text{simu}}}} \tag{14}
\]

Here $y_{\text{obser}}$ is observation and $y_{\text{simu}}$ is simulation. MAE denotes mean absolute error. RMSE denotes root mean square error. Nash Sutcliffe Efficiency (NSE) denotes normalized square error, it expresses the square error relative to an average value guess. NSE = 1 for perfect predictions, and NSE = 0 for non-informative predictions. CORR is Pearson Correlation Coefficient. We carry out the evaluation at different spatiotemporal scales, as will be clarified in the following sections.

D. RESULTS

Figure 4 shows the training progress. Results here are based on the model configuration that achieved optimal performance for the validation set. The red/green/blue line represents the mean square error as function of training epoch for the training/mini-batch/validation set.

Figure 5 summarizes the prediction performances for all the four lanes and all the 125 observing stations. The computation is based on 5-minute traffic flow/speed time series for the all the entire test set.

Results here show that the model has similar satisfying prediction skill for most stations, with NSE skill above 0.9 and CORR skill above 0.95. We also noticed that intervals around Station 40 and Station 90 show poorer performance compared to the rest stations. For these two intervals, the outer lane is worse predicted than the inner lanes. By projecting the stations back to the geographical map, we found that these regions are where considerable traffic flows merge into I-101 North from the two bay bridges: Station 40 connects US I-101 and Dumbarton Bridge, Station 90 connects US I-101 and San Mateo–Hayward Bridge. We attribute the poor performance to the fact that the model is targeted for long homogeneous freeway sections, and do not explicitly consider the traffic merges from outside traffic flows. The analysis above is based on results for entire test period. Below we inspect model’s performance under typical traffic scenarios. Extreme weather, especially flash flood related with land-falling atmospheric river event, may pose severe challenge to the local traffic in our study area [69]. Besides, we may expect distinct traffic patterns in holidays and weekends, as compared to weekdays [70]. Figure 6 and 7 show the traffic flow and speed predictions for the following four scenarios: (1) a typical weekday (25 June 2019), (2) a typical weekend (23 June 2019, Sunday), (3) rainy day (08 January 2018, with a record of 80 mm heavy precipitation reported by National Weather Service), and (4) a national holiday (01 January 2018, new year).

Figure 6 and 7 show the 5-minute traffic flow and speed prediction for the four scenarios at an arbitrarily selected station (Station 400010). The first four columns for each feature show the results for Lane 1 (inner lane) to Lane 4 (outer lane). The last column summarizes the performance at each lane using MAE, RMSE, NSE, and CORR.

Considering the prediction for different lanes, both the traffic flow and speed prediction for the Lane 4 (outer lane) show slightly worse performance compared to the predictions for the rest lanes. The correlation skill difference is significant at 95% confidence level, based on a z statistic test. We believe the relatively poor performance for the outer lane is due to the frequent traffic loading and releasing from outside traffic systems.

Considering the difference for different scenarios, we find that weekdays are featured by heavy morning traffic, weekends and holidays show slower increase of traffic load in morning, and stronger persistence throughout the daytime. The heavy rain on January 8th, 2018 significantly slowed down the traffic speed and flow. Considering the model performance for these scenarios, we find that traffic flow predictions show no significant correlation skill difference between weekdays and weekends/rainy days/holidays at 95% confidence level, based on a z statistic test.
E. APPLYING MULTIPLE FEATURES TO CHARACTERIZE TRAFFIC DYNAMICS

In this part, we verify the benefits of considering multiple aspects of traffic conditions for characterizing traffic dynamics. Ma et al. [27] have demonstrated the advantage of including traffic volume information as a supplementary predictor to improve traffic speed prediction. However, their models only predict traffic speed but not traffic volume. With no traffic volume being predicted, the models do not have sufficient input to support predictions beyond one time step. Here we highlight the following two aspects:

1) The inclusion of traffic flow/speed information as predictors enhances the prediction accuracy for both of them.

2) To predict flow/speed simultaneously helps to regularize the model, and may enhance the prediction accuracy for each of these two variables.

To testify these two points, we design the following two control experiments:

Experiment 1.1 We build a Convolutional LSTM neural network model that uses only traffic flow or speed as predictor and predictand. The results are compared against the model proposed in Section III, which takes input of both variables.

Experiment 1.2 We build a Convolutional LSTM neural network model that uses traffic flow and speed as predictors, but predict only traffic flow or speed. The results are compared against the model proposed in Section III, which predicts both variables simultaneously.

We follow similar architecture and hyperparameter settings as the benchmark model in Section III. Model modifications are only made to highlight the control experiment settings described above.

Figure 8 shows the training progress for models in Experiment 1.1 and 1.2. For comparison purpose, we draw the validation loss for our benchmark model along with the training, mini-batch, and validation loss of the control models. Our benchmark model outperforms the comparison model in Experiment 1.1 for both speed and flow prediction. In Experiment 1.2, the benchmark model and comparison model show neglectable difference in...
speed prediction, however, the benchmark model significantly outperforms the comparison model for flow prediction.

The model performances at lane scale and multi-lane average scale are summarized in Table 3. The benchmark model outperforms the control model in Experiment 1.1 for predicting flow and speed at both lane scale and lane-average scale. This evidence suggests that by including extra traffic features, we can increase the prediction accuracy for both features. The skill improvement is more evident for flow prediction compared to speed prediction. For Experiment 1.2, the benchmark model outperforms the control model in predicting flow. This suggests that to predict traffic speed helps to regularize the traffic flow prediction task. On the other hand, the control model usually predicts speed with higher accuracy, indicating that to predict traffic flow as a regularization method does not contribute consistently in predicting traffic speed.

Besides regularization, another benefit for making traffic flow/speed prediction simultaneously is that we can make $n$ time step ahead predictions based on 1 up to $(n-1)$ step predictions, $n$ starts from 2. This is achieved by sequentially feeding 1 up to $(n-1)$ step predictions as input to the model.
FIGURE 8. Mean square error for speed (Row 1)/flow (Row 2) prediction as function of training epoch for the control models in Experiment 1.1 and 1.2. Results for Experiment 1.1/1.2 is shown in Column 1/2. Red /green /blue curves represents the mean error for training /mini-batch/validation set. For comparison, the validation loss for our benchmark model is shown with cyan and dashed line.

FIGURE 9. Prediction skill as function of prediction lead time. The models considered here are the benchmark model (red), control model in Experiment 1.1 (green), control model in Experiment 1.2 (blue), and persistency (black). Row 1/2 shows results for flow/speed prediction. Column 1/2 shows results evaluated using correlation skill score/normalized square error skill score.

Below we draw the prediction skill-lead time relation based on our model’s rolling predictions. The models considered here are the benchmark model (red), control model in Experiment 1.1 (green), control model in Experiment 1.2 (blue), and persistency (black). The persistency prediction is made using current step status as predictions.

As can be expected, all models show decrease of prediction skill with prediction lead time. For both flow and speed predictions, the benchmark model and the control model in Experiment 1.2 show advantageous performances for lead time from 10 minutes up to 55 minutes. It is worth noticing that the control model in Experiment 1.2 requires both traffic flow and speed as predictors, while only outputs one of them, the rolling prediction is achieved by combing two models, each predicting flow or speed separately. The control model in Experiment 1.1 makes slightly worse flow predictions, but still outperforms persistency for all the considered lead time periods. For speed prediction, the disadvantage of the control model in Experiment 1.1 is more obvious, with skills falling far behind the benchmark model and control model in Experiment 1.2 after lead time of 30 minutes. Besides, the skill for the control model in Experiment 1.1 is even worse than persistency after around 45 minutes. This quick diminish of prediction skill is attributed to the fact that prediction error may grow fast in a non-linear model. Overall, the results here suggests that the proposed model can make satisfying multiple step rolling predictions.

F. USING INFORMATION FROM NEIGHBOURING LANES

In this part, we investigate how traffic evolution is influenced by neighbouring lanes. Considering the traffic condition for different lanes at a same freeway cross-section, researchers have long noticed that drivers tend to make use of neighbouring lanes to buffer traffic congestions. This evidence suggests that to accurate predict traffic, we need to carefully model the routing between neighbouring lanes. Here we design the following experiment to highlight the importance of considering neighbouring lane for traffic prediction:

Experiment 2 We build a 1-D Convolutional LSTM neural network model that takes into input and predicts the traffic flow and speed at lane-average scale. By averaging the flow and speed for the four lanes, the model does not consider the routing between neighbouring lanes. The results are compared against the model proposed in Section III, which explicitly predicts the traffic evolution for all the four lanes.

As for Experiment 1.1 and 1.2, we follow similar architecture and hyperparameter settings as the benchmark model in Section III. The only difference is that the input/output are at lane average scale rather than for each lane. Figure 10 shows the training progress for the control model in Experiment 2. The flow and speed prediction skill are summarized in Table 3. For comparison, the lane-level predictions from the rest models are averaged and evaluated. We found that the benchmark model and the control model in Experiment 1.2 offer better flow and speed predictions, as compared to the control model in Experiment 2 here. This evidence suggests that by explicitly modeling the routing between neighbouring lanes using the proposed model architecture, we can improve the accuracy for traffic prediction.

G. USING DYNAMICAL TRAFFIC INFORMATION

In this part, we testify the importance of applying dynamical traffic information for short term traffic prediction. To achieve this target, we carry out a control experiment that makes prediction based solely on current time step traffic status. Also, bearing in mind that there are various ways for encoding dynamical traffic information besides the LSTM recurrent architecture employed here, we implement a comparison 3-D convolution neural network model that apply...
TABLE 3. Summary of prediction performances for the benchmark model proposed in Section III, and Experiment 1.1, 1.2, 2, 3.1, and 3.2. The lane-level evaluation is based on all stations, all time step predictions and observations. The average evaluation is based on all lanes, all stations, and all time step predictions and observations. The best performance are underlined and labeled bold.

| Evaluation Scale | Performance Metrics | Benchmark Model | Experiment 1.1 | Experiment 1.2 | Experiment 2 | Experiment 3.1 | Experiment 3.2 |
|------------------|---------------------|-----------------|----------------|---------------|--------------|---------------|---------------|
| Flow             | MAE                 | 9.426           | 9.635          | 9.618         | 14.188       | 15.843        |
|                  | RMSE                | 13.9            | 14.128         | 14.119        | 27.772       | 28.812        |
|                  | NSE                 | 0.926           | 0.924          | 0.924         | 0.706        | 0.705         |
|                  | CORR                | 0.963           | 0.962          | 0.962         | 0.862        | 0.861         |
| Speed            | MAE                 | 1.983           | 1.983          | 1.983         | 2.313        | 2.317         |
|                  | RMSE                | 3.32            | 3.359          | 3.303         | 3.959        | 3.964         |
|                  | NSE                 | 0.938           | 0.936          | 0.938         | 0.911        | 0.911         |
|                  | CORR                | 0.969           | 0.968          | 0.969         | 0.955        | 0.955         |

FIGURE 10. Mean error for speed(left)/flow(right) prediction as function of training epoch for the control model in Experiment 2. Red/green/blue curves represents the mean error for training/mini-batch/validation set.

Both spatial and temporal convolution. The specific experiment settings are described as follows:

**Experiment 3.1** We build a convolutional neural network model that uses only current step traffic status (flow and speed) as predictor and predictand. The difference with the benchmark model in Section III is that we drop off the LSTM recurrent structure here. The results are compared against the benchmark model.

**Experiment 3.2** We build a 3-D Convolutional neural network model that first apply convolution along both spatial and temporal domain. The results for neighbouring time steps are later synthesized using a temporal convolution kernel. The results are compared against the model proposed in Section III.

Figure 11 shows the training progress for models in Experiment 3.1 and 3.2. We draw the validation loss for our benchmark model along with the training, mini-batch, and validation loss of the control models. Performances of the models are summarized in Table 3.

The benchmark model proposed in Section III, which considers previous step traffic data using LSTM recurrent network, outperforms both control models by a large margin. These results suggest that the Convolution LSTM network architecture can effectively utilize dynamical traffic information for better predictions.

**H. COMPARING WITH CLASSICAL TIME SERIES MODEL**

The previous sections have used several baseline models to highlight the importance of explicitly considering local traffic patterns and traffic dynamics for traffic prediction. This is achieved using a novel composition of convolution...
FIGURE 11. Mean square error for speed (Row 1)/flow (Row 2) prediction as function of training epoch for the control models in Experiment 3.1 and 3.2. Results for Experiment 3.1/3.2 is shown in Column 1/2. Red /green /blue curves represents the mean error for training /mini-batch /validation set. For comparison, the validation loss for our benchmark model is shown with cyan and dashed line.

and recurrent neural network modules. At the beginning of this paper, we argued that classical time series models usually make strong assumptions of traffic evolution, such as stationarity and linearity. Such assumptions may not be valid for real-world traffic patterns. To justify this criticism, we compare our model with well-established time series model. Here we apply Auto Regressive Integrated Moving Average (ARIMA) model as the baseline.

ARIMA model can be understood by outlining each of its three components:

1) **Autoregression (AR)** Autoregression refers to time series in which variable depends linearly on its own previous values. The lagging dependency length is denoted with $p$.

2) **Moving average (MA)** Moving average refers to time series in which variable depends linearly on the current and various past values of a stochastic terms. The lagging dependency length is denoted with $q$.

3) **Integration (I)** Integration is the key component that differentiates ARIMA with autoregressive-moving-average (ARMA) models, which assumes auto-regressive and moving average properties of a time series. Integration replace data values with the difference between current and previous values, allowing the time series to be non-stationary. The degree of differencing is denoted with $d$. A detailed explanation can be found in [71].

We test and compare the performance of ARIMA for the same station (Station 400010) as investigated in previous sections. Different combinations of $p$, $q$, and $d$ parameters are tested. The ARIMA model is trained using bootstrapped data from the training set, and tested for the same test set. The comparison results are shown in Table 4.

Results in Table 4 shows that the CNN-LSTM model outperforms the ARIMA for traffic flow and speed predictions. We attribute this advantage to the fact that the deep neural network model can better capture the non-linear evolution of traffic flow by explicitly exploring local traffic patterns and traffic dynamics.

**TABLE 4.** Prediction performance for Station 400010 using different ARIMA settings. The last column shows results for the CNN-LSTM model. The best performance are underlined and labeled bold.

| ARIMA | CNN-LSTM |
|-------|-----------|
| Flow  | Speed     | Flow  | Speed     |
| (1,1) | (1,1)     | (1,1) | (1,1)     |
| MAE   | RMSE      | MAE   | RMSE      |
| 11.224 | 15.796 | 10.356 | 10.331 |
| 11.092 | 15.733 | 10.351 | 10.331 |
| 11.121 | 15.72  | 10.326 | 10.331 |
| 11.084 | 15.712 | 10.311 | 10.311 |

**V. CONCLUSION**

In this paper, we propose a novel Convolutional LSTM neural network architecture for multi-lane short-term traffic prediction. Compared to existing methods, we highlight the following aspects:

- The inclusion of multiple features for characterizing traffic condition helps to improve prediction accuracy.
- By explicitly modeling the routing between neighboring lanes and downstream/upstream traffics, we can significantly enhance traffic prediction.
- By applying synced sequence to sequence recurrent architecture, we can satisfactorily predict multiple time-step traffic in a rolling-prediction manner.

We carry out experiments based on PeMS dataset. The results show that our model has considerable advantages in predicting multi-lane short-term traffic.

**ACKNOWLEDGMENT**

The authors would like to make a grateful acknowledgement to the Caltrans Performance Measurement System for providing valuable daily traffic flow data, and the anonymous reviewers whose comments and suggestions helped to improve and clarify this manuscript.
REFERENCES

[1] C. Van Hinsbergen, J. Van Lint, and F. Sanders, “Short-term traffic prediction models,” in Proc. 14th World Congr. Intell. Transp. Syst. (ITS), Beijing, China, Oct. 2007, pp. 5013–5030.

[2] M. S. Ahmed and A. R. Cook, “Analysis of freeway traffic-volume-time-series data by using box-jenkins techniques,” Transp. Res. Rec., vol. 722, pp. 1–9, 1979.

[3] M. Levin and Y.-D. Tsao, “On forecasting freeway occupancies and volumes (abridgment),” Transp. Res. Rec., vol. 773, pp. 47–49, 1980.

[4] M. M. Hamed, H. R. Al-Masaeid, and Z. M. B. Said, “Short-term prediction of traffic volume in urban arterials,” J. Transp. Eng., vol. 121, no. 3, pp. 249–254, 1995.

[5] Y. Qi and S. Ishak, “A hidden Markov model for short term prediction of traffic conditions on freeways,” Transp. Res. C, Emerg. Technol., vol. 43, pp. 1–11, 2014.

[6] B. Ghosh, B. Basu, and M. O’Mahony, “Random process model for urban traffic flow using a Wavelet-Bayesian hierarchical technique,” Comput-Aided Civil Infrastruct. Eng., vol. 25, no. 8, pp. 613–624, Nov. 2010.

[7] J. Wang, W. Deng, and Y. Gao, “New Bayesian combination method for short-term traffic flow forecasting,” Transp. Res. C, Emerg. Technol., vol. 43, pp. 79–94, Jun. 2014.

[8] S. V. Kumar, “Traffic flow prediction using Kalman filtering technique,” Comput. Eng., vol. 5, pp. 582–587, Jan. 2017.

[9] B. M. Williams and L. A. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results,” J. Transp. Eng., vol. 129, no. 6, pp. 664–672, Nov. 2003.

[10] M. Van Der Voort, M. Doughearty, and S. Watson, “Combining Kohonen maps with Arima time series models to forecast traffic flow,” Transp. Res. C, Emerg. Technol., vol. 4, no. 5, pp. 307–318, Oct. 1996.

[11] B. M. Williams, “Multivariate vehicular traffic flow prediction: Evaluation of ARIMAX modeling,” Transp. Res. Rec., vol. 1776, no. 1, pp. 194–200, Jan. 2001.

[12] M. Yixuan, G. Xuedong, and P. Baoxiang, “Trend feature-based cluster-temoral correlation in a hybrid deep learning framework,” 2016, arXiv:1612.01022. [Online]. Available: http://arxiv.org/abs/1612.01022

[13] Y. Ma, Z. Zhang, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” 2017, arXiv:1709.04875. [Online]. Available: http://arxiv.org/abs/1709.04875

[14] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, “Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction,” Sensors, vol. 17, no. 4, p. 818, Apr. 2017.

[15] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang, “A hybrid deep learning based traffic flow prediction method and its understanding,” Transp. Res. C, Emerg. Technol., vol. 90, pp. 166–180, May 2018.

[16] Q. Liu, B. Wang, and Y. Zhu, “Short-term traffic speed forecasting based on attention convolutional neural network for arterials,” Comput.-Aided Civil Infrastruct. Eng., vol. 33, no. 11, pp. 999–1016, Nov. 2018.

[17] Z. Zheng, Y. Yang, J. Liu, H-N. Dai, and Y. Zhang, “Deep and embedded learning approach for traffic flow prediction in urban informatics,” IEEE Trans. Intell. Transp. Syst., vol. 20, no. 10, pp. 3927–3939, Oct. 2019.

[18] R. Ke, W. Li, Z. Cui, and Y. Wang, “Two-stream multi-channel convolutional neural network (TM-CNN) for multi-lane traffic speed prediction considering traffic volume impact,” 2019, arXiv:1903.01678. [Online]. Available: http://arxiv.org/abs/1903.01678

[19] D. L. Gerlough and M. J. Huber, “Traffic flow theory,” Transp. Res. Board Special Rep., vol. 165, p. 222, 1976.

[20] P. G. Michalopoulos, D. E. Beskos, and Y. Yamauchi, “Multilane traffic flow dynamics: Some macroscopic considerations,” Transp. Res. B, Methodol., vol. 18, nos. 4–5, pp. 377–395, Aug. 1984.

[21] C. F. Daganzo, “A behavioral theory of multi-lane traffic flow. Part I: Long homogeneous freeway sections,” Transp. Res. B, Methodol., vol. 36, no. 2, pp. 131–158, Feb. 2002.

[22] H. Yu, Z. Wu, S. Wang, Yang, and X. Ma, “Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks,” Sensors, vol. 17, no. 7, p. 1501, Jun. 2017.

[23] Q. Chen, X. Song, H. Yamada, and R. Shihabasi, “Learning deep representation from big and heterogeneous data for traffic accident inference,” in Proc. 13th AAAI Conf. Artif. Intell., Feb. 2016, pp. 338–344.

[24] H. Huang, G. Song, H. Hong, and K. Xie, “Deep architecture for traffic flow prediction: Deep belief networks with multitask learning,” IEEE Trans. Intell. Transp. Syst., vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[25] H. Tan, X. Xuan, Y. Wu, Z. Zhong, and B. Ran, “A comparison of traffic flow prediction methods based on DBN,” in Proc. CICTP, Jul. 2016, pp. 273–283.

[26] L. Li, L. Qin, X. Qu, J. Zhang, Y. Wang, and B. Ran, “Day-ahead traffic flow forecasting based on a deep belief network optimized by the multi-objective particle swarm algorithm,” Knowl.-Based Syst., vol. 172, pp. 1–14, May 2019.

[27] K. Zhang, N. Jia, L. Zheng, and Z. Liu, “A novel generative adversarial network for estimation of trip travel time distribution with trajectory data,” Transp. Res. C, Emerg. Technol., vol. 108, pp. 223–244, Nov. 2019.

[28] J. W. C. van Lint, S. P. Hoogendoorn, and H. J. van Zuylen, “Freeway travel time prediction with state-space neural networks: Modeling state-space dynamics with recurrent neural networks,” Transp. Res. Rec., vol. 1811, no. 1, pp. 30–39, Jan. 2002.

[29] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 157–166, Mar. 1994.

[30] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: The difficulty of learning long-term dependencies,” in A Field Guide to Dynamical Recurrent Networks, S. C. Kremer and J. F. Kolen, Eds. Piscataway, NJ, USA: IEEE Press, 2001.

[31] Y. Tian and L. Pan, “Predicting short-term traffic flow by long short-term memory recurrent neural network,” in Proc. IEEE Int. Conf. Smart City/SocialCom/SustainCom (SmartCity), Dec. 2017, pp. 153–158.

[32] Y.-y. Chen, Y. Lv, Z. Li, and F.-Y. Wang, “Long short-term memory model for traffic congestion prediction with online open data,” in Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC), Nov. 2016, pp. 132–137.

[33] R. Fu, Z. Zhang, and L. Li, “Using LSTM and GRU neural network methods for traffic flow prediction,” in Proc. 31st Youth Academic Ann. Conf. Chin. Assoc. Autom. (YAC), Nov. 2016, pp. 324–328.

[34] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, “LSTM network: A deep learning approach for short-term traffic forecast,” IET Intell. Transp. Syst., vol. 11, no. 2, pp. 68–75, Mar. 2017.

[35] D.-F. Xie, Z.-Z. Fang, B. Jia, and Z. He, “A data-driven lane-changing model based on deep learning,” Transp. Res. C, Emerg. Technol., vol. 106, pp. 41–60, Sep. 2019.
[49] Y. Gu, W. Lu, L. Qin, M. Li, and Z. Shao, “Short-term prediction of lane-level traffic speeds: A fusion deep learning model,” Transp. Res. C, Emerg. Technol., vol. 106, pp. 1–16, Sep. 2019.

[50] K. Zhang, L. Zheng, Z. Liu, and N. Jia, “A deep learning based multitask model for network-wide traffic speed prediction,” Neurocomputing, to be published. [Online]. Available: https://authors.elsevier.com/tracking/article/details?doi=206464&jid=NEU/COM&surname=Zhang.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017.

[52] S. J. Russell and P. Norvig, Artif. Intelligence: A Modern Approach. Kuala Lumpur, Malaysia: Pearson, 2016.

[53] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 1725–1732.

[54] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 568–576.

[55] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in Proc. 25th Int. Conf. Mach. Learn. (ICML), 2008, pp. 160–167.

[56] C. Dos Santos and M. Gatti, “Deep convolutional neural networks for sentiment analysis of short texts,” in Proc. 25th Int. Conf. Comput. Linguistics, Tech. Papers, 2014, pp. 69–78.

[57] B. Pan, K. Hsu, A. AghaKouchak, and S. Sorooshian, “Improving precipitation estimation using convolutional neural network,” Water Resour. Res., vol. 55, no. 3, pp. 2301–2321, Mar. 2019.

[58] Q. Miao, B. Pan, H. Wang, K. Hsu, and S. Sorooshian, “Improving monsoon precipitation prediction using combined convolutional and long short term memory neural network,” Water, vol. 11, no. 5, p. 977, May 2019.

[59] B. Pan, K. Hsu, A. AghaKouchak, and S. Sorooshian, “The use of convolutional neural network in relating precipitation to circulation,” in AGU Fall Meeting Abstracts. Louisiana, NO, USA: American Geophysical Union, Dec. 2017.

[60] L. Lin, Z. He, and S. Peeta, “Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach,” Transp. Res. C, Emerg. Technol., vol. 97, pp. 258–276, Dec. 2018.

[61] K. Zhang, Z. Liu, and L. Zheng, “Short-term prediction of passenger demand in multi-zone level: Temporal convolutional neural network with multi-task learning,” IEEE Trans. Intell. Transp. Syst., to be published.

[62] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in Proc. Int. Conf. Mach. Learn., 2013, pp. 1310–1318.

[63] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[64] A. Karpathy, J. Johnson, and L. Fei-Fei, “Visualizing and understanding recurrent networks,” 2015, arXiv:1506.02078. [Online]. Available: http://arxiv.org/abs/1506.02078.

[65] C. Chen, Z. Jia, J. Kwon, and E. van Zwet, “Statistical methods for estimating speed using single-loop detectors,” in Proc. 82nd Annu. Meeting Transp. Res. Board, 2002. [Online]. Available: https://drive.google.com/file/d/0B5wZ4dLpgONnT1NxbG9SQ21rR1k/view?usp=sharing.

[66] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition,” 2014, arXiv:1402.1128. [Online]. Available: http://arxiv.org/abs/1402.1128.

[67] Wolfram Inc., Champaign, IL, USA. (2019). Mathematica. Version 12.0. [Online]. Available: https://www.wolfram.com/mathematica.

[68] C. Nvidua, “Nvidia cuda c programming guide,” Nvidia Corp., vol. 120, no. 18, p. 8, 2011.

[69] B. Pan, K. Hsu, A. AghaKouchak, S. Sorooshian, and W. Higgins, “Precipitation prediction skill for the west coast united states: From short to extended range,” J. Climate, vol. 32, no. 1, pp. 161–182, Jan. 2019.

[70] J. Zhang, Y. Zheng, and D. Qi, “Deep spatio-temporal residual networks for citywide crowd flows prediction,” in Proc. 31st AAAI Conf. Artif. Intell., Feb. 2017, pp. 1655–1661.

[71] R. F. Nau. (2005). Introduction to Arima: Nonseasonal Models. Accessed: Jul. 5, 2018. [Online]. Available: https://people.duke.edu/~rnau/411arim.htm.