PCNN: Deep Convolutional Networks for Short-Term Traffic Congestion Prediction

Meng Chen, Xiaohui Yu, *Member, IEEE*, and Yang Liu, *Member, IEEE*

***Abstract*— Traffic problems have seriously affected people's life quality and urban development, and forecasting short-term traffic congestion is of great importance to both individuals and governments. However, understanding and modeling the traffic conditions can be extremely difficult, and our observations from real traffic data reveal that: 1) similar traffic congestion patterns exist in the neighboring time slots and on consecutive workdays and 2) the levels of traffic congestion have clear multiscale properties. To capture these characteristics, we propose a novel method named PCNN, which is based on a deep convolutional neural network, modeling periodic traffic data for short-term traffic congestion prediction. PCNN has two pivotal procedures: time series folding and multi-grained learning. It first temporally folds the time series and constructs a 2-D matrix as the network input, such that both the real-time traffic conditions and past traffic patterns are well considered; then, with a series of convolutions over the input matrix, it is able to model the local temporal dependency and multiscale traffic patterns. In particular, the global trend of congestion can be addressed at the macroscale, whereas more details and variations of the congestion can be captured at the microscale. Experimental results on a real-world urban traffic data set confirm that folding time series data into a 2-D matrix is effective and PCNN outperforms the baselines significantly for the task of short-term congestion prediction.**

***Index Terms*— Traffic congestion prediction, periodic traffic data, convolutional neural network.**

I. Introduction

PEOPLE are getting increasingly concerned about traffic congestion, which has seriously affected their life quality and urban development. To monitor real-time traffic conditions, many cities around the world have deployed embedding sensors, e.g., inductive-loop detectors and video image processors, in road networks [1], and GPS (Global Position System)-based services such as Google Maps have been developed to show traffic conditions and even details regarding individual vehicles [2]. The increasing availability of data from such devices and services has created unique opportunities to predict traffic conditions (e.g., predicting travel speed and traffic volume [3]–[5], predicting city-scale traffic flow [2], [6]), benefiting the decision making of individuals and governments. For example, people can adjust their driving routes dynamically and authorities can optimize traffic signal time according to predicted traffic conditions.

Most existing work on predicting traffic conditions has focused on predicting future traffic flows at a given location [7], [8] or the travel time on a given road segment [9]. In this paper, we target instead at directly forecasting short-term traffic congestion levels for road segments in urban road networks. The reason is that very often people would just like to know how "jammed" the traffic is going to be in the next minutes or hours on the road segments of their interest, rather than the actual traffic flow values or travel time. In this paper, we define the *congestion level* c for a road segment during a given time slot as $c = \max[0, (t - \overline{t})/\overline{t}]$, where t is the average travel time of vehicles for that segment in that time slot, and $\overline{t}$ is the baseline travel time for the same road segment in ideal traffic conditions. Congestion level is an intuitive way to depict the traffic condition and is suitable for visualization, as it resembles an evaluative meter that is understandable to most people.

Forecasting traffic congestion levels, however, is filled with challenges, because of a series of complex factors. To demonstrate this, we carefully depict the congestion time series with an urban traffic dataset. We randomly choose two road segments (1 and 2) in Jinan and show their traffic congestion levels during one week in Fig. 1(a), and further plot the congestion levels of road segment 1 from 6:30 am to 8:00 am on workdays in Fig. 1(b).

- **Local coherence**. The traffic congestion level in a time slot has a strong correlation with those in the neighboring time slots, and the correlation diminishes as the temporal distance increases. For example, the traffic conditions of 6 pm may be affected by the congestion occurring at 5 pm, but can be considered free from the influence of the traffic at 8 am of the same day.
- **Periodicity**. Traffic congestion levels on different workdays exhibit a temporal periodicity, i.e., repeating a similar pattern roughly every 24 hours. For example, as shown in Fig. 1, traffic congestion levels during the same time slots (e.g., morning rush hours) are similar on consecutive workdays, but are different from those in other time periods, e.g., from 11 am to 1 pm, of the same day. Further, our analysis of the real traffic data during a span of six weeks reveals that the congestion levels of a given workday are more similar to those of adjacent

Manuscript received September 10, 2017; revised February 23, 2018; accepted May 6, 2018. Date of publication June 21, 2018; date of current version November 9, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61572289 and in part by the NSERC Discovery Grants. The Associate Editor for this paper was V. Punzo. *(Corresponding author: Xiaohui Yu.)*

M. Chen and X. Yu are with York University, Toronto, ON M3J 1P3, Canada (e-mail: xhyu@yorku.ca).

Y. Liu is with Wilfrid Laurier University, Waterloo, ON N2L 3C5, Canada.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2018.2835523

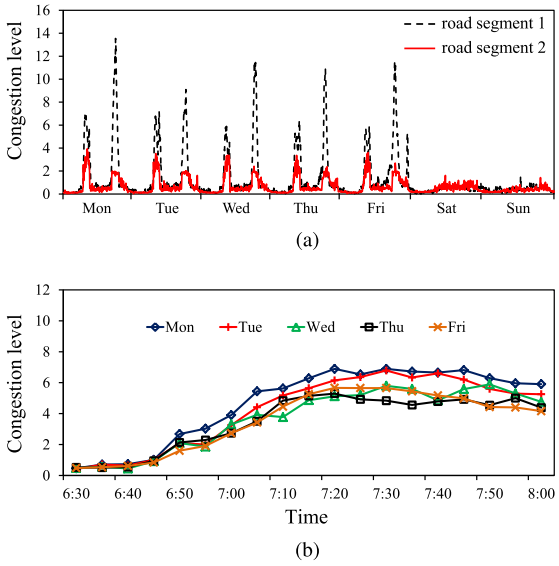


Fig. 1. Examples of traffic congestion levels of two road segments. (a) Traffic congestion levels of two road segments during one week. (b) Traffic congestion levels of road segment 1 from 6:30 am to 8:00 am.

days, rather than the same days in other weeks.

- **Multiscale property**. Traffic congestion levels have clear multiscale properties. At the microscale, the variation of congestion levels can be observed with precise details, while it is hard to discover the global trend of large temporal scope. In contrast, at the macroscale, the global trend of congestion levels can be easily revealed, while many details are lost. Thus, the traffic congestion level in a given time slot is the result of both global and local effects, and a combination of global trend and local fluctuation may help make better prediction.

The properties of local coherence and periodicity imply that the traffic congestion level in a time slot is related to those in the neighboring slots of both the same day and previous days. However, most existing methods [8], [10] predict future traffic congestion $c_{m,n}$ in the time slot n of day m, by taking just the immediately preceding t values and/or the values of the same slot in previous d days into consideration. They fail to consider the similar patterns on preceding workdays, which may degrade the prediction accuracy. In addition, to the best of our knowledge, none of these existing methods consider the multiscale property in making short-term traffic prediction.

A. Present Work

To capture the similar traffic patterns and multiscale congestion properties, we propose **PCNN**, a **C**onvolution-based deep **N**eural **N**etwork modeling **P**eriodic traffic data, which converts the one-dimensional data into an image-like input matrix and applies a series of convolutions on it. Specifically, PCNN has two pivotal procedures: time series folding and multi-grained learning. To predict future traffic congestion level $c_{m,n}$, we fold the time series of congestion levels based on the period (i.e., 24 hours), and combine the $2t$ values around the current time slot n in the previous d days with the immediately preceding t values (replicating once) to generate the input matrix with size $(d+1) \times 2t$. Consequently, the two-dimensional matrix contains both the traffic conditions in the immediate past and a large volume of similar historical patterns; thus both local coherence and periodicity are taken into consideration.

Another key contribution of PCNN is in learning a set of multi-grained features. By performing an array of convolutions over the input matrix, PCNN could capture the local temporal dependency and numerous higher-level features. Then, these features are transmitted to the output layer to predict the future traffic congestion levels. Finally, the objective can be efficiently optimized with stochastic gradient descent (SGD) akin to back propagation on the deep convolutional networks.

Our experiments focus on short-term traffic congestion prediction with the real vehicle passage records data in Jinan, China. We contrast the performance of PCNN with state-of-the-art traffic forecasting methods, including regressive models (e.g., ARIMA (autoregressive integrated moving average) [11]), pattern recognition methods (e.g., K-NN (K-nearest neighbors) [7]), and neural networks (e.g., MLP (multilayer perceptrons) [12] and LSTM (long short-term memory) [13]). Experiments show that PCNN has smaller forecast errors. In addition, we also apply the two-dimensional input matrix to some baselines, and the results demonstrate that the methods with the two-dimensional input performs better than those with original one-dimensional input.

The main contributions can be summarized as follows:

- Different from existing methods that model traffic patterns with one-dimensional time-series, we propose to fold the traffic data based on the period and model them as a two-dimensional matrix, which considers the traffic conditions in the immediate past and similar historical patterns simultaneously.
- We propose to apply a series of convolutions on the two-dimensional input matrix to model local temporal dependencies and multi-grained features. We are thus able to estimate the approximate range of future congestion levels at different scales. To the best of our knowledge, this is the first time to apply convolutions on the periodic traffic data.
- We conduct extensive experiments with the real vehicle passage records in an urban road network to investigate the effectiveness of the proposed PCNN. For the task of congestion prediction, the results demonstrate that the methods with two-dimensional input perform better, and PCNN shows remarkable improvement compared with several baselines.

The rest of this paper is organized as follows. Section II reviews the studies on short-term traffic prediction. Section III introduces the definition of congestion level and the problem solved in this paper. Section IV presents our deep convolutional networks for short-term traffic congestion level prediction. The experimental results are discussed in Section V. Section VI concludes this paper.

II. Related Work

Traffic congestion prediction can be considered as an extension of short-term traffic forecasting, which is a pivotal application in intelligent transportation systems. See Bolshinsky

and Freidman [14] for a thorough survey on different techniques (e.g., time series models, Markov chain models, non-parametric methods) used for traffic forecasting. Here we only focus on summarizing existing works that are directly related to our study. These works fall into three broad categories, i.e., regressive models, pattern recognition methods and neural networks (NN).

A. Regressive Models

Regressive models are a type of general methods for forecasting time series data. As most traffic data tend to be closely related to their previous values, a special group of regressive models named autoregressive integrated moving average (ARIMA) are usually adopted for short-term traffic prediction. ARIMA is parameterized by three non-negative integers, commonly represented as ARIMA$(p; d; n)$, where p is the number of autoregressive terms, d is the number of nonseasonal differences, and n is the number of lagged forecast errors in the prediction equation. Ahmed and Cook first introduce the model to predict the freeway traffic volume and occupancy time series [11]. After that, numerous ARIMA-based variants have been proposed in traffic time series prediction, e.g., seasonal ARIMA [15] and space-time ARIMA [16]. Chung and Rosalion [17] systemically compare ARIMA and its variants with some alternative solutions including regression, historical average, etc. Their results reveal that the above strategies perform reasonably well under normal conditions, but less satisfactory when external changes (e.g., weather, special events) happen.

B. Pattern Recognition Methods

Pattern recognition methods have also been applied to short-term traffic forecast, e.g., support vector machines (SVM) [18], and K-nearest neighbors [7], [19], [20]. Wang and Shi [18] integrate Wavelet-Chaos Analysis and SVM regression theory, and construct a new kernel function to capture the non-stationary characteristics of the short-term traffic speed data for prediction. Considering the time-varying and continuous characteristic of traffic flow, Yu *et al.* [20] propose a multi-time-step prediction model based on the K-nearest neighbors (K-NN) algorithm for short-term traffic condition prediction. Further, Habtemichael and Cetin [7] present an enhanced K-NN method using weighted Euclidean distance to identify similar traffic patterns for short-term traffic forecast. In addition, Xia *et al.* [19] propose a K-NN model in a general MapReduce framework on a Hadoop platform to enhance the efficiency of short-term traffic flow forecasting. However, the pattern recognition methods cannot work well when the number of historical data exhibiting similar patterns is limited, for example, the time slots with extreme traffic congestion are rare, and these methods fail to identify similar patterns for prediction in this case.

C. Neural Networks

Most early studies along this line exploit feed-forward multilayer perceptrons (MLP) [12], [21], in which the temporal relationships are augmented in the input data during pre-processing. Besides, there are approaches [22], [23] adopting dynamic neural networks, e.g., Lingras and Mountford [22] use a genetic algorithm to optimize the connections between inputs and hidden layers for traffic volume estimation.

Deep neural network (DNN), which refers to a feed forward neural network with more than one hidden layers, has recently revolutionized the machine learning society, and achieved great success in natural language processing, computer vision, etc. Convolutional neural network (CNN) and recurrent neural network (RNN) are two main types of DNN architectures. In general, CNN is hierarchical, and originally applied to capture spatial features in image classification [24]. RNN exhibits a sequential architecture, and is intuitively plausible for sequence modeling tasks, e.g., language modeling [25]. In practice, both kinds of neural networks have been explored to capture spatial and temporal dependencies, and sometimes are even applied simultaneously [26]. Despite the great success of DNN, only few efforts have been made to use it for traffic forecasting. As a representative piece of work, Tian and Pan [13] adopt the LSTM (long short-term memory method) which could determine the optimal length of the input historical data dynamically to make short-term traffic flow prediction.

In this paper, we work on the specific problem of short-term traffic congestion prediction, and introduce a novel convolutional neural network to model the intricate natures of temporal features, including periodicity, local coherence, etc., which have been rarely considered in previous studies. In the same vein, some recent studies attempt to perform the city-scale crowd flow prediction with DNN [6], [27], whose objective is to estimate the total traffic of crowds entering/leaving a region during a given time interval. Nonetheless, in contrast to our work that aims to explain the temporal dependencies, they tend to focus on capturing spatial dependencies, as the inflow of one region is affected by outflows of nearby regions in their applications.

III. Preliminaries

We first introduce the definition of congestion level and then formally define the problem to be addressed in this paper.

Definition 1 (Congestion Level): Given a road segment, we define the **congestion level** $c_{i,j}$ based on the average travel time $t_{i,j}$ for that segment in time slot j of day i and the baseline travel time $\bar{t}$ for the same road segment, formally as $c_{i,j} = \max(0, (t_{i,j} - \bar{t})/\bar{t})$, where $\bar{t}$ can be estimated using data from periods of light traffic (e.g., after midnight).

Problem 1 (Short-Term Traffic Congestion Forecasting): For a specific road segment, given a sequence of observed congestion data $\{c_{i,j}\}$, $i = 0, 1, \ldots, m$, $j = 0, 1, \ldots, n-1$, where i represents the index of day and j is the index of time slot in day i, the problem is to predict anticipated traffic congestion level $c_{m,n}$ in the next time slot of day m.

IV. Deep Convolutional Networks

In this section, we first elaborate the architecture of the proposed neural network, and then provide details of each

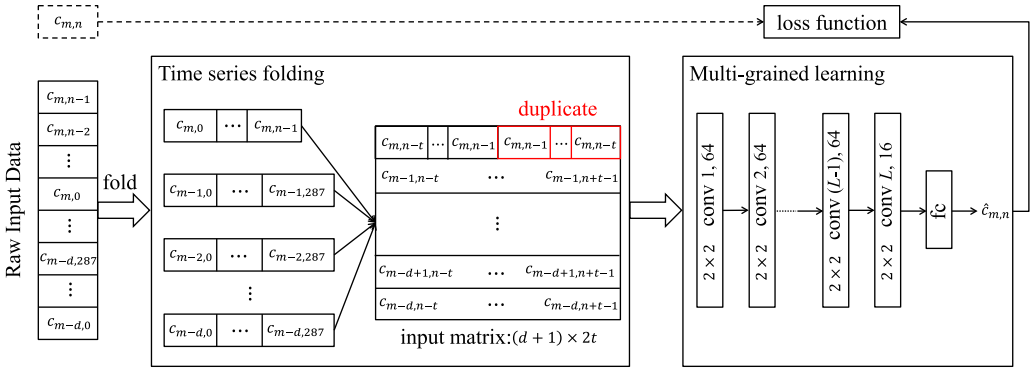


Fig. 2. Architecture of PCNN. conv: convolutional layer, fc: fully-connected.

component. Finally, we introduce the objective function and the method for parameter learning.

A. Overview

Intuitively, due to local coherence and periodicity, the variations of congestion levels in the neighboring time slots on consecutive workdays are similar. Besides, traffic congestion levels have shown distinct multiscale features. These properties can be effectively handled by the convolutions that have proved effective in capturing the local structural information and multiscale features from pixel-level raw images [24]. Inspired by this idea, we propose a tailored method named **PCNN** to capture the periodic traffic congestion patterns at different scales for predicting short-term traffic congestion levels. Fig. 2 presents the architecture of PCNN, which consists of two major components: time series folding and multi-grained learning. As illustrated in the left part of Fig. 2, for the raw input data, we take 5 minutes as the size of time slots as an example, and obtain 288 slots in one day. We first choose the historical data from the previous d days and fold them into a two-dimensional matrix. The input matrix is then fed into multi-grained learning component, capturing multi-scale features by a series of convolutions. Finally, the output of the final convolutional layer is transmitted to the output layer, yielding the predicted value.

B. Time Series Folding

We have observed local coherence and periodicity, but there is no observed evidence that the congestion levels of a particular workday (say, Tuesday) bears higher similarity with those of the same days in previous weeks (past Tuesdays), according to our traffic data. Thus, we only consider the historical values in the preceding d days and fold them into d vectors. Furthermore, the traffic congestion level in a specific time slot has a strong correlation with those in the neighboring time periods. Therefore, to predict the traffic congestion level $c_{m,n}$, where m and n are the indexes of day and time slot, we take the $2t$ values ($\langle c_{m-i,n-t},\ldots,c_{m-i,n+t-1}\rangle, i \in \{1,2,\ldots,d\}$) around the current slot n in every day into consideration. Note that, it is superior to only using the d values of slot n in previous d days, as the $2t$ time slots exhibit similar traffic congestion patterns in the d days because of local coherence and periodicity. Further, the traffic conditions in the immediate past are pretty important, and we duplicate the congestion levels of the recent t slot to get a vector $\langle c_{m,n-t},\ldots,c_{m,n-1},c_{m,n-1},\ldots,c_{m,n-t}\rangle$ of length $2t$. Afterwards, these vectors are integrated to yield the $(d+1)\times 2t$ input matrix $\mathbf{X}_{m,n}$:

$$\mathbf{X}_{m,n}=\begin{bmatrix} c_{m,n-t} & \cdots & c_{m,n-t} \\ c_{m-1,n-t} & \cdots & c_{m-1,n+t-1} \\ \vdots & \vdots & \vdots \\ c_{m-d,n-t} & \cdots & c_{m-d,n+t-1} \end{bmatrix}. \tag{1}$$

In our study, d and t are data-independent, and we will evaluate their effects in the experiments. Further, we use $c_{i,0}(m-d\le i\le m)$ to pad those elements of each row vector whose index is less than 0 under condition of $n<t$; similarly, we use $c_{0,j}(n-t\le j\le n+t-1)$ to pad each column vector under condition of $m<d$.

C. Multi-Grained Learning

In order to capture the multiscale congestion patterns, we decide to apply a series of convolutions on the input matrix. As one convolution only accounts for near dependencies, limited by the size of their kernels, we need to use multiple convolutional layers to model the dependency over a greater time range. Here we do not use pooling operations, but only convolutions, following the suggestion in [6] and [28].

Given the input matrix $\mathbf{X}_{m,n}$, we apply the convolutional operation (i.e., conv 1 in Fig. 2) on it:

$$\mathbf{H}^{(1)}=f(\mathbf{W}^{(1)}*\mathbf{X}_{m,n}+b^{(1)}), \tag{2}$$

where $*$ denotes the convolution, and f is an activation function, e.g., the rectified linear function (ReLU) [24]. $\mathbf{W}^{(1)}$ and $b^{(1)}\in\mathbb{R}$ are the learnable parameters in the first convolutional layer. We then feed $\mathbf{H}^{(1)}$ to the next layer, until the L-th layer:

$$\mathbf{H}^{(l)}=f(\mathbf{W}^{(l)}*\mathbf{H}^{l-1}+b^{(l)}),\quad l=2,\ldots,L. \tag{3}$$

For each one in the first $(L-1)$ layers, we use 64 filter maps of size 2×2 at a stride of 1 over the input data. Note that each filter map is replicated across the entire input matrix, and a unit in the filter map has 4 inputs connected to a 2×2 area in the input matrix, called the receptive field of the unit. Therefore, each unit has 4 trainable coefficients $\mathbf{W}^{(l)}$ plus a trainable

bias $b^{(l)}$, and all the units in a filter map share the same set of weights. A complete convolutional layer is composed of 64 filter maps, and each map uses different sets of weights and biases, thereby extracting different types of local features. These features are then combined by the subsequent layers in order to capture higher order features. For the last convolution, we take 16 filters to reduce the dimension of output, and the experimental results prove that it is superior to 64 filters.

With these convolutional operations, PCNN is able to extract the local temporal dependency among neighboring days and time slots and learn multi-grained features. Then we transmit these features $\mathbf{H}^{(L)}$ to the output layer to generate the predicted congestion level $\hat{c}_{m,n}$. Here we use the identity function as the activation function,

$$\hat{c}_{m,n} = \mathbf{W}^o \cdot \mathbf{H}^{(L)} + b^o, \tag{4}$$

where $\mathbf{W}^o$ is a weight term and b^o is a bias term in the layer.

D. Loss Function

We use the square error between the predicted congestion levels and the observed values to define the objective function, i.e.,

$$\ell = \min \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{1}{2} \|\hat{c}_{m,n} - c_{m,n}\|^2 + \frac{1}{2} \lambda \|\Theta\|^2, \tag{5}$$

where M is the number of days in the training set and N is the number of time slots in a day. Θ represents the whole parameters in PCNN, and λ is the regularization coefficient.

Note that, our proposed PCNN is able to make not only one-step ahead predictions, but also multi-step ahead predictions. Given a sequence of observed congestion data $\{c_{i,j}\}$, $i = 0, 1, \ldots, m$, $j = 0, 1, \ldots, n-1$, when predicting u-step ahead congestion level $c_{m,n+u-1}$, we just take the $2t \times d$ values ($\langle c_{m-i,n+u-1-t}, \ldots, c_{m-i,n+u-2+t} \rangle, i \in \{1, 2, \ldots, d\}$) of the past d days around the time slot $n+u-1$ and the $2t$ values $\langle c_{m,n-t}, \ldots, c_{m,n-1}, c_{m,n-1}, \ldots, c_{m,n-t} \rangle$ as the input of PCNN. Then we compute $\hat{c}_{m,n+u-1}$ with PCNN and define the objective function according to Equation (5).

E. Algorithm and Optimization

We then use the stochastic gradient descent (SGD) method with the RMSprop update rule [29] to minimize the square errors between our predictions and the actual congestion levels. Algorithm 1 outlines the training process of PCNN. We first construct the training instances from the original traffic congestion level data (lines 1-5), i.e., building the input matrix $\mathbf{X}_{m,n}$ for each predicted traffic congestion level $c_{m,n}$ in time slot n of day m. Then, PCNN is trained via back propagation (lines 6-10).

Algorithm 1 PCNN Training Algorithm

Input: historical congestion levels $\mathcal{C}$, the size of the input matrix, the number of convolutions;
Output: the learned model;
// construct training instances
1: $\mathcal{D} \leftarrow \emptyset$;
2: **for** $c_{m,n} \in \mathcal{C}$ **do**
3: build the input matrix $\mathbf{X}_{m,n}$ according to Equation (1);
4: put a training instance $(\mathbf{X}_{m,n}, c_{m,n})$ into $\mathcal{D}$;
5: **end for**
// train the model
6: initialize all parameters Θ;
7: **repeat**
8: randomly select a batch of instances $\mathcal{D}_b$ from $\mathcal{D}$;
9: update Θ by minimizing the objective (5) with $\mathcal{D}_b$;
10: **until** *stopping criteria is met*

V. Experiments

In this section, to evaluate the effectiveness of PCNN, we first introduce our dataset and basic settings, and then demonstrate the performances evaluated with different parameters. Finally, we show the experimental results compared with several baselines.

TABLE I
Definition of Traffic Conditions

group	description
normal traffic	$c \leq 1$
light congestion	$1 < c \leq 3$
heavy congestion	$c > 3$

A. Dataset and Settings

With the deployment of surveillance cameras on road networks, vehicles are photographed when they pass by, and structured vehicle passage records (VPRs) containing vehicle ID, location, and timestamp can be subsequently extracted from pictures using optical character recognition (OCR) [1]. The accuracy of recognizing the plate number by OCR could reach 97% in ideal weather/lighting conditions. In our experiments, we collect six weeks of VPRs from 614 road segments in Jinan, China.

1) Preprocessing: In this study, we notice that traffic conditions on weekends clearly differ from those on workdays, and traffic jam rarely occurs on weekends (as indicated in Fig. 1); therefore we only use the traffic data on workdays (30 days in total) in our experiments. Further, as few people drive late at night, we only keep those records captured from 6:00 to 24:00 everyday. Here we first take 5 minutes as the size of time slot (later we compare different methods with various sizes of slot), and compute the traffic congestion levels for all the slots based on Definition 1. Thus, we have 3,978,720 ($614 \times 30 \times 18 \times (60/5)$) values of traffic congestion level. As forecast of congestion level is more important when traffic is heavy, we differentiate the traffic conditions based on the value of congestion level c, and define the traffic condition as *congested* if c is larger than 1, as shown in Table I.

In order to understand the traffic congestion data better, we compute the cumulative distribution functions (cdf) of the congestion levels and the distribution of *congested* traffic by time of day, as shown in Fig. 3. Clearly, *congested* traffic occurs in about 36% time slots, and is mainly concentrated around the morning peak and the evening peak.

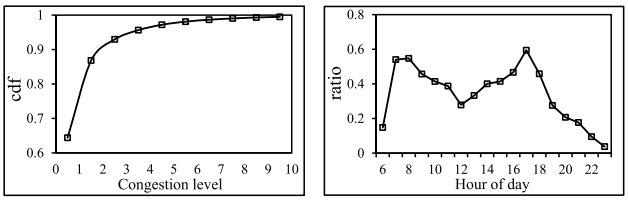


Fig. 3. Characteristics of the traffic dataset. (a) Cdf of congestion levels. (b) Distribution of *congested* traffic.

In the training process, we use the Min-Max normalization method to scale the whole dataset into the range [0, 1]. In the evaluation, we re-scale the predicted values back to the normal values, to compare with the ground truth.

2) Hyperparameters: We use the open-source deep learning library, deeplearning4j,[1] to build our models. The first $(L-1)$ convolutions use 64 filters of size 2×2, and the last one uses a convolution with 16 filters of size 2×2. We use ReLU as the activation function, and fix the learning rate at 0.005. We set the l_2 regularization parameter at 0.001, and the batch size at 128. These values are selected via a grid search on our dataset. We take the traffic congestion values of the first 20 days as the training set, and the next 5 days' data as the validation set for tuning parameters. Afterwards, we continue to train the model on the full dataset for a fixed number of epochs (e.g., 10 epochs), and compare the performance on the last 5 days' data with baselines.

3) Evaluation Metrics: To evaluate the effectiveness of PCNN, we use three performance metrics, namely, the mean absolute error (MAE), the root-mean-square error (RMSE), and the mean relative error (MRE), which are defined as

$$MAE = \frac{1}{M' \times N} \sum_{m=1}^{M'} \sum_{n=1}^{N} \|\hat{c}_{m,n} - c_{m,n}\|,$$

$$RMSE = \left[\frac{1}{M' \times N} \sum_{m=1}^{M'} \sum_{n=1}^{N} \|\hat{c}_{m,n} - c_{m,n}\|^2 \right]^{\frac{1}{2}},$$

$$MRE = \frac{1}{M' \times N} \sum_{m=1}^{M'} \sum_{n=1}^{N} \frac{\|\hat{c}_{m,n} - c_{m,n}\|}{c_{m,n}}, \tag{6}$$

where $\hat{c}_{m,n}$ is the predicted congestion level, and $c_{m,n}$ is the observed value. M' is the number of days in the test set and N is the number of time slots in a single day.

B. Performance of PCNN

In this section, we first evaluate the performance of PCNN with different parameters, namely, the size of the input matrix (the number of days, d and the number of time slots, t), and the number of convolutional layers (L), and tune them one by one on the validation set. Then we show the detailed forecast performance with respect to varying traffic conditions and time of day.

[1]https://deeplearning4j.org/

1) Identifying a Suitable Size of the Input Matrix: On one hand, we know that the variations of congestion level in one day are similar to those in the preceding days. On the other hand, the congestion level is closely related to those in the adjacent slots. Thus, we set t and d at 3, 6, 9, 12 respectively in this case, and choose the optimal number of convolutions for the models with different input matrix sizes. We first evaluate the effect of t with defined d on forecast accuracy, as shown in Fig. 4 (a), (b) and (c). It can be observed that (1) with increase in the number of training epochs, the prediction errors (including MAE, RMSE, and MRE) start to decline, and remain stable after about 8 epochs; (2) the model with $t = 3$ performs relatively poor, as it only considers the traffic conditions in the neighboring 15 minutes, without taking enough related values into consideration; (3) the model with $t = 6$ performs the best, indicating that exploiting traffic conditions in half an hour around the current slot is the most suitable in our case. The impact of t with other d (i.e., $d = 3, 6, 12$) on forecast accuracy is similar, and we set t at 6 in the following experiments.

We then measure the impact of d on forecast accuracy in terms of the three error criteria, as shown in Fig. 4 (d), (e) and (f). Similarly, for different d, the prediction errors decrease when the number of epoches increases, and the model with $d = 9$ obtains the best performance. In addition, the model involving the larger input matrix contains more parameters, and it needs more time to complete the training procedure. Therefore, we choose $t = 6, d = 9$ and 10 training epochs as our default setting, and the size of the input matrix is 10×12.

2) Identifying the Number of Convolutional Layers: The number of convolutional layers determines the depth of PCNN, and we need to validate whether deep networks are more effective than the shallow ones. With the 10×12 input matrix, we consider a series of L (the number of convolutions) values in this study, ranging from 1 to 9, and train the models with the same parameter setting. The experimental results are demonstrated in Fig. 5. Clearly, a consistent improvement in forecast accuracy is observed with an increase in the number of convolutional layers, as the proposed model cannot capture enough multi-grained features with the shallow networks (e.g., 1 or 2 convolutions); on the other hand, the models with very deep networks (e.g., 8 and 9 convolutions) also get relatively large forecast errors, and take more time in the training process to train. Thus, we set the number of convolutional layers at 5 in our study based on the performance of prediction.

3) Accuracy of Forecast by Different Traffic Conditions and Time of Day: To evaluate the effectiveness of the proposed model further, we examine the performance by different traffic conditions and time of day. Fig. 6 shows the performance of short-term congestion forecast using PCNN in terms of MAE, RMSE and MRE by various traffic conditions and hour of day. The box plots show the spread of the forecast errors and the red solid line represents the mean of the errors.

At forecast time, MAE and RMSE have consistent improvements when the traffic conditions change from normal to congested, as both MAE and RMSE consider only the magnitude of deviations of the forecasted values from the observed

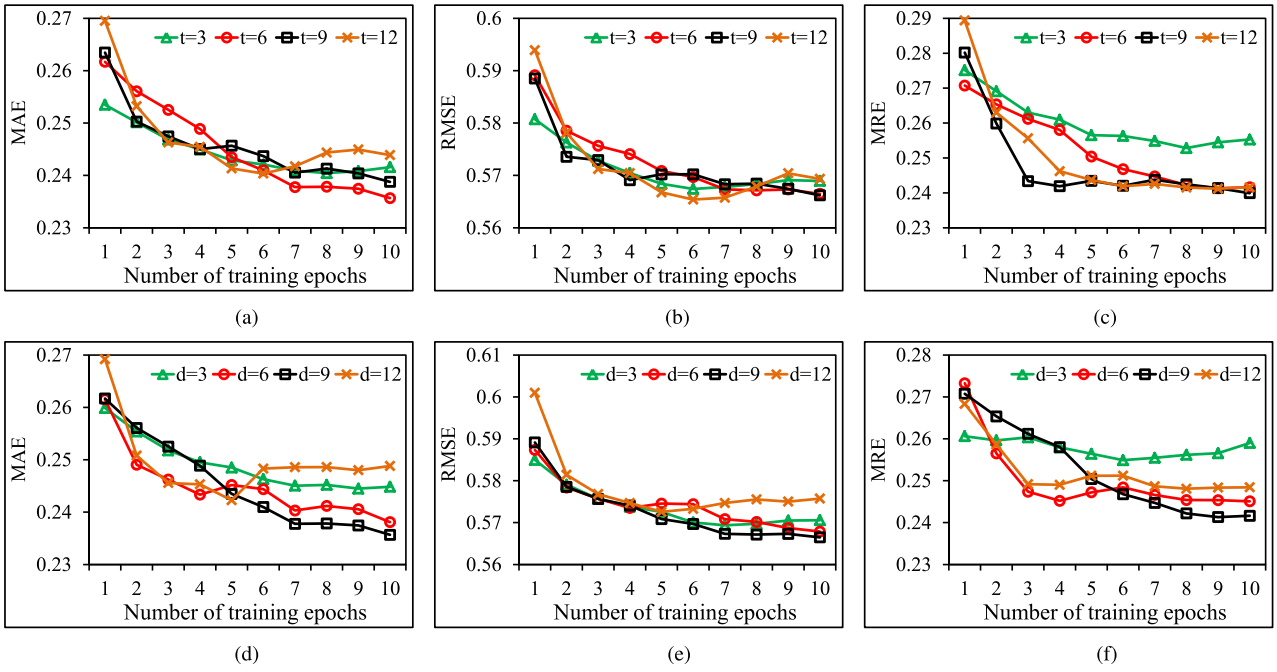


Fig. 4. Effect of size of input matrix. (a) MAE ($d = 9$). (b) RMSE ($d = 9$). (c) MRE ($d = 9$). (d) MAE ($t = 6$). (e) RMSE ($t = 6$). (f) MRE ($t = 6$).

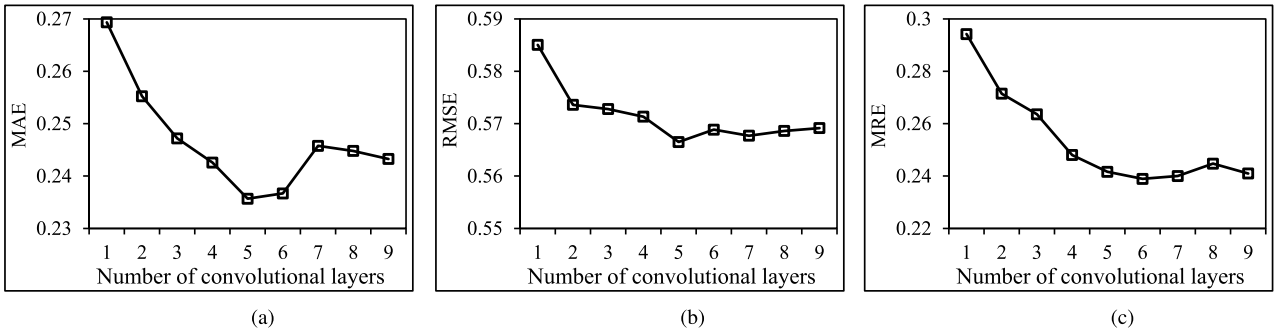


Fig. 5. Effect of number of convolutional layers. (a) MAE. (b) RMSE. (c) MRE.

ones. Meanwhile, MRE provides a better sense of forecast accuracy as the errors are examined in terms of percentage deviations from the observed value, and it has consistent reduction evidently. Similarly, when the nature of the errors corresponding to the hour of day is examined, the forecast accuracy during peak-hours (7:00 - 9:00 and 17:00 - 18:00) is relatively lower when compared with off-peak hours. This is because of the fact that the patterns are more complicated and the values of congestion level are larger during peak-hours, as depicted in Fig. 3.

Moreover, examining the mean errors and the third quartiles (i.e., the top of the box), we find that the mean values are always greater than or approximately equal to the third quartiles, indicating that a few extremely large forecast errors exist in this case. To explore such cases, we investigate the distributions of errors, i.e., the cdf (cumulative distribution function) of MRE, as shown in Fig. 7. Clearly, about 74% of the MREs are less than 0.3, and only 1.6% of them are greater than 2. The reason may be that traffic accidents occur frequently, which often lead to a sudden surge in congestion levels within a short period of time. As instances with sudden change are rare, a general statistic model will be dominated by normal instances, and is difficult to capture the special patterns.

Generally speaking, according to the spread of MRE (the mean error is around 20%) in Fig. 6 (c), it can be said that the proposed PCNN provides reliable and accurate forecasts of traffic congestion levels.

C. Comparisons With State-of-the-Art Methods

To evaluate the performance of our proposal, we compare PCNN with several state-of-the-art methods for predicting short-term traffic congestion levels. To ensure a fair comparison, a common dataset and measure of performance are used.

- **HA:** This is probably the most straightforward method which assumes that the future value $\hat{c}_{m,n}$ is the average of the historical data.
- **LR:** Weuse linear function on the input data to minimize the square error between our predictions and the actual values.
- **ARIMA:** This is a general method for forecasting a time series, illustrated in detail in Section II.

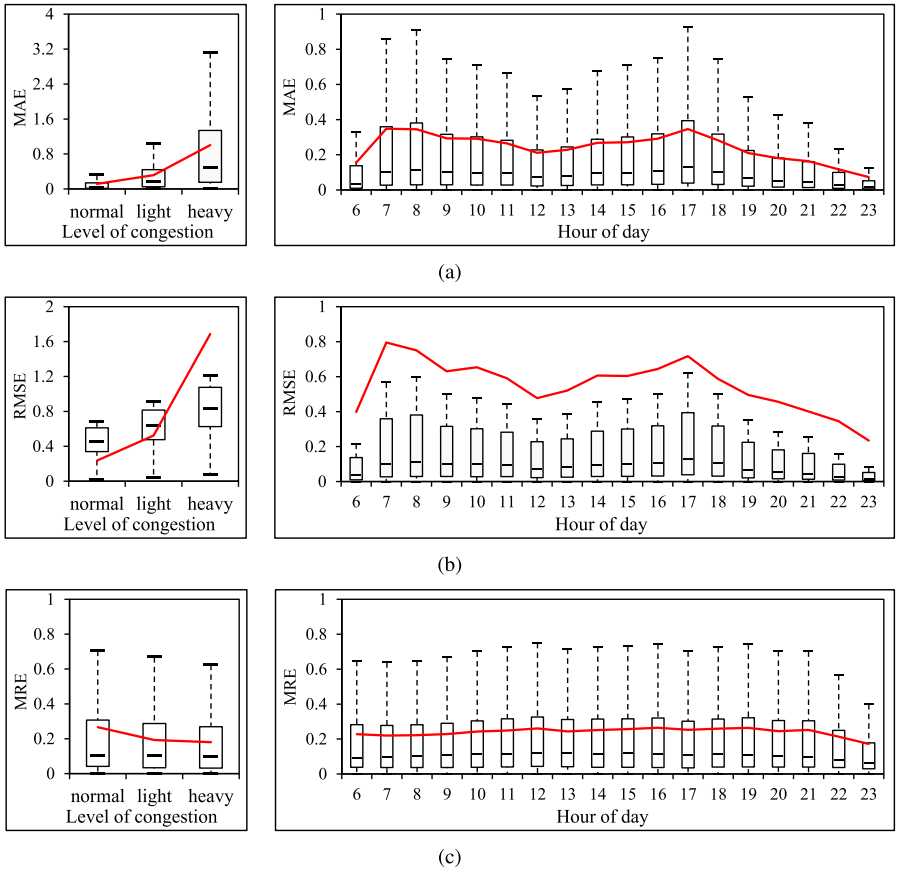


Fig. 6. Forecast errors by level of traffic and time of day. (a) MAE by level of traffic and hour of day. (b) RMSE by level of traffic and hour of day. (c) MRE by level of traffic and hour of day.

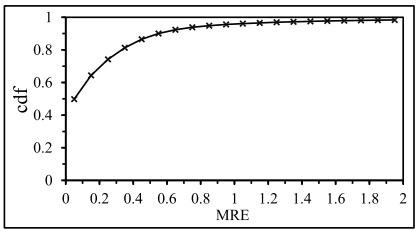


Fig. 7. Cdf of MRE.

- **SARIMA:** We compare with the Seasonal ARIMA [15], as traffic congestion levels have temporal periodicity.
- **K-NN:** This is an enhanced K-nearest neighbors (K-NN) algorithm for short-term traffic forecasting based on identifying similar traffic patterns [7].
- **MLP:** We construct many MLP structures with different numbers of hidden layers to predict traffic congestion values.
- **LSTM:** This is a long short-term memory network which shows superior capability for time series prediction with long temporal dependency [13].

To validate the necessity of folding the temporal data into a two-dimensional matrix, we experiment with both vectors and matrices as the inputs of HA, LR and MLP, denoted as HA(1), HA(2), LR(1), LR(2), MLP(1) and MLP(2). When predicting future congestion level $c_{m,n}$, the data in previous t slots of day m and in the n-th slot of preceding d days are used to construct the one-dimensional vectors $\langle c_{m,n-t}, \ldots, c_{m,n-1}, c_{m-1,n}, \ldots, c_{m-d,n} \rangle$ of length $(t + d)$ as the inputs of HA(1), LR(1) and MLP(1). For the matrices, we flatten them and construct the new vectors $\langle c_{m-d,n-t}, \ldots, c_{m-d,n+t-1}, c_{m-d+1,n-t}, \ldots, c_{m,n-t} \rangle$ of length $(d+1) \times 2t$ as the inputs of HA(2), LR(2) and MLP(2).

For all the baselines, the system configuration parameters are optimized by a grid search. For instance, we vary K from 5 to 50 at a step of 5 for K-NN, and find that it obtains the best performance when K is 15; as it is very tricky to set up configuration options for MLP, we train a group of MLPs with the number of hidden layers varying from 1 to 9 and the number of neurons in each layer varying from 50 to 500, and the MLP structure with the best performance we can come up with contains 8 hidden layers with 150 neurons in each layer for the two-dimensional input and 5 layers with 200 neurons for the one-dimensional input.

To compare the proposed approach (PCNN) with the baselines, we not only evaluate the overall prediction performance, but also measure it under different traffic conditions. We demonstrate the forecast errors on testing data in Table II, and our evaluation on the proposed method is 3-fold.

1) Forone-dimensional input, we measure all baselines, including the naive method (HA), the regressive models (LR, ARIMA and SARIMA), the pattern recognition method (K-NN), and the neural networks (MLP and LSTM). Specifically, HA treats

TABLE II
RESULTS OF METHODS

method	normal			light			heavy			overall		
	MAE	RMSE	MRE	MAE	RMSE	MRE	MAE	RMSE	MRE	MAE	RMSE	MRE
HA(1)	0.194	0.453	0.469	0.449	0.781	0.279	1.289	1.984	0.239	0.346	0.770	0.399
LR(1)	0.192	0.359	0.455	0.466	0.720	0.287	1.421	2.072	0.265	0.359	0.738	0.394
ARIMA	0.152	0.306	0.368	0.407	0.613	0.241	1.196	1.887	0.215	0.299	0.654	0.321
SARIMA	0.149	0.298	0.361	0.392	0.601	0.233	1.180	1.854	0.212	0.293	0.636	0.313
K-NN	0.173	0.304	0.416	0.411	0.613	0.255	1.350	1.971	0.248	0.326	0.672	0.359
MLP(1)	0.147	0.276	0.340	0.369	0.571	0.226	1.141	1.730	0.210	0.282	0.600	0.299
LSTM	0.149	0.285	0.346	0.362	0.573	0.232	1.124	1.692	0.205	0.280	0.596	0.304
HA(2)	0.252	0.536	0.634	0.551	0.915	0.343	1.629	2.339	0.298	0.436	0.907	0.528
LR(2)	0.187	0.344	0.440	0.461	0.709	0.285	1.344	1.996	0.251	0.348	0.714	0.383
MLP(2)	0.139	0.259	0.321	0.351	0.533	0.217	1.102	1.676	0.197	0.268	0.573	0.283
PCNN	**0.119**	**0.241**	**0.269**	**0.307**	**0.523**	**0.190**	**0.947**	**1.636**	**0.172**	**0.232**	**0.557**	**0.240**

TABLE III
RESULTS OF METHODS WITH DIFFERENT TIME GRANULARITY

method	10-min			15-min			30-min			60-min		
	MAE	RMSE	MRE	MAE	RMSE	MRE	MAE	RMSE	MRE	MAE	RMSE	MRE
LSTM	0.263	0.596	0.268	0.259	0.601	0.273	0.254	0.592	0.264	0.253	0.586	0.259
MLP(1)	0.267	0.604	0.270	0.266	0.606	0.271	0.260	0.595	0.267	0.253	0.585	0.252
MLP(2)	0.256	0.571	0.261	0.252	0.567	0.257	0.244	0.562	0.252	0.231	0.557	0.243
PCNN	**0.228**	**0.551**	**0.225**	**0.221**	**0.543**	**0.217**	**0.214**	**0.541**	**0.212**	**0.210**	**0.536**	**0.207**

each element equally and LR assigns the elements with different weights, so LR performs better than the naive HA; ARIMA and SARIMA use the differencing step to eliminate the non-stationarity, and their forecast errors are smaller than LR's; SARIMA considers the temporal periodicity, and it performs better than ARIMA; different from the regressive models, K-NN identifies the similar patterns and predicts future value based on them, but it does not obtain decent performances, especially in the heavy congestion condition, as there is only 7% of data in that case (as depicted in Fig. 3); MLP uses many non-linear functions to model the relationship between the predicted values and the actual congestion levels, and performs better than K-NN; LSTM has an advantage of memorizing long historical data and can achieve lower prediction errors.

2) We then analyze the effect of introducing the two-dimensional input. HA averages more weakly related values when using the two-dimensional input, so it performs worse than with the one-dimensional input. LR and MLP with the two-dimensional perform better than those with the one-dimensional (according to the MRE metric, the error rates of LR(2) and MLP(2) decline by 2.8% and 5.4% respectively), as they could extract more effective features through complex linear or non-linear functions from the input matrix, validating the effectiveness of folding periodic time series data and constructing the two-dimensional input.
3) Our proposed method, PCNN, is clearly superior to the baselines based on the experimental results. Taking the overall prediction as an example, compared with the LSTM that has the best performance with one-dimensional input in the baselines, the MRE rate drops by 21.1%, which is a considerable improvement; even compared with MLP(2) that has the same input matrix, the MRE of PCNN still drops by 15.2%. The reason is two-fold: on one hand, the two-dimensional input matrix takes both the real-time traffic conditions and the historical similar traffic patterns into consideration; on the other hand, a series of convolutions over the input matrix could capture the local temporal dependency and model multiscale traffic congestion features.

Further, we set the size of time slot at 10, 15, 30, and 60 minutes respectively, and compare the proposed PCNN with the baselines (as MLP and LSTM outperform other baselines obviously, we only compare PCNN with them). As shown in Table III, as the size of time slot increases, the prediction errors decline. It is evident because the congestion levels become smoother when we consider larger time slot. Furthermore, the proposed PCNN performs better than these baselines under the circumstances of different time granularity, further validating the robustness of our model.

VI. CONCLUSION

In this paper, we have proposed a novel method (**PCNN**) based on the convolution-based deep neural network modeling periodic traffic data to make short-term traffic congestion prediction. The accurate forecast could be used as a decision support tool for traffic operators to design an alternative traffic management strategy to avoid traffic jams. Considering the characteristics of urban traffic congestion data, we fold the time series data and construct a two-dimensional matrix. PCNN takes the matrix as its input, and models the local

temporal dependency and multiscale traffic patterns through multiple convolutional operations. Finally, we evaluate the performances of PCNN on a real traffic dataset, and experimental results show that the proposed method outperforms state-of-the-art baselines significantly.

REFERENCES

[1] M. Chen, X. Yu, and Y. Liu, "Mining moving patterns for predicting next location," *Inf. Syst.*, vol. 54, pp. 156–168, Dec. 2015.

[2] X. Zhan, Y. Zheng, X. Yi, and S. V. Ukkusuri, "Citywide traffic volume estimation using trajectory data," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 272–285, Feb. 2017.

[3] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 653–662, Apr. 2015.

[4] Y. Xu, Q.-J. Kong, R. Klette, and Y. Liu, "Accurate and interpretable Bayesian MARS for traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2457–2469, Dec. 2014.

[5] R. Silva, S. M. Kang, and E. M. Airoldi, "Predicting traffic volumes and estimating the effects of shocks in massive transportation systems," *Proc. Nat. Acad. Sci. USA*, vol. 112, no. 18, pp. 5643–5648, 2015.

[6] J. Zhang, Y. Zheng, and D. Qi. (2016). "Deep spatio-temporal residual networks for citywide crowd flows prediction." [Online]. Available: https://arxiv.org/abs/1610.00081

[7] F. G. Habtemichael and M. Cetin, "Short-term traffic flow rate forecasting based on identifying similar traffic patterns," *Transp. Res. C, Emerg. Technol.*, vol. 66, pp. 61–78, May 2015.

[8] L. Lv, M. Chen, Y. Liu, and X. Yu, "A plane moving average algorithm for short-term traffic flow prediction," in *Proc. PAKDD*. New York, NY, USA: Springer, 2015, pp. 357–369.

[9] M. Yildirimoglu and N. Geroliminis, "Experienced travel time prediction for congested freeways," *Transp. Res. B, Methodol.*, vol. 53, pp. 45–63, Jul. 2013.

[10] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[11] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using box-jenkins techniques," *Transp. Res. Rec.*, vol. 722, no. 1, pp. 1–9, 1979.

[12] M. S. Dougherty and M. R. Cobbett, "Short-term inter-urban traffic forecasts using neural networks," *Int. J. Forecasting*, vol. 13, no. 1, pp. 21–31, 1997.

[13] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *Proc. IEEE SmartCity*, Dec. 2015, pp. 153–158.

[14] E. Bolshinsky and R. Freidman, "Traffic flow forecast survey," Tech. Rep. CS-2012-06, 2012.

[15] B. Williams, P. Durvasula, and D. Brown, "Urban freeway traffic flow prediction: Application of seasonal autoregressive integrated moving average and exponential smoothing models," *Transp. Res. Rec.*, vol. 1644, pp. 132–141, Jan. 1998.

[16] Y. Kamarianakis and P. Prastacos, "Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches," *Transp. Res. Rec.*, vol. 1857, pp. 74–84, Jan. 2003.

[17] E. Chung and N. Rosalion, "Short term traffic flow prediction," in *Proc. ATRF*, 2001, p. 16.

[18] J. Wang and Q. Shi, "Short-term traffic speed forecasting hybrid model based on Chaos–Wavelet Analysis-Support Vector Machine theory," *Transp. Res. C, Emerg. Technol.*, vol. 27, pp. 219–232, Feb. 2013.

[19] D. Xia, B. Wang, H. Li, Y. Li, and Z. Zhang, "A distributed spatial-temporal weighted model on MapReduce for short-term traffic flow forecasting," *Neurocomputing*, vol. 179, pp. 246–263, Feb. 2016.

[20] B. Yu, X. Song, F. Guan, Z. Yang, and B. Yao, "*k*-nearest neighbor model for multiple-time-step prediction of short-term traffic condition," *J Transp. Eng.*, vol. 142, no. 6, p. 04016018, 2016.

[21] S. D. Clark, M. S. Dougherty, and H. R. Kirby, "The use of neural networks and time series models for short term traffic forecasting: A comparative study," in *Proc. PTRC SAM*, 1993, p. 363.

[22] P. Lingras and P. Mountford, "Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting," in *Proc. IEA/AIE*. New York, NY, USA: Springer, 2001, pp. 290–299.

[23] B. Abdulhai, H. Porwal, and W. Recker, "Short-term traffic flow prediction using neuro-genetic algorithms," *J. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 3–41, 2002.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

[26] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. NIPS*, 2015, pp. 802–810.

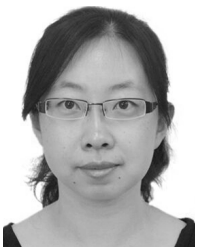
[27] Y. Li, Y. Zheng, H. Zhang, and L. Chen, "Traffic prediction in a bike-sharing system," in *Proc. SIGSPATIAL*, 2015, Art. no. 33.

[28] V. Jain *et al.*, "Supervised learning of image restoration with convolutional networks," in *Proc. IEEE ICCV*, Oct. 2007, pp. 1–8.

[29] T. Tieleman and G. Hinton, "Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

Meng Chen received the Ph.D. degree in computer science and technology from Shandong University, China, in 2016. He is currently a Post-Doctoral Fellow with the School of Information Technology, York University, Canada. His research interest is in the area of data mining.

Xiaohui Yu received the Ph.D. degree in computer science from University of Toronto, Canada, in 2006. He is currently an Associate Professor with the School of Information Technology, York University, Canada. His research interests are in the areas of database systems and data mining.



Yang Liu received the Ph.D. degree in computer science and engineering from York University, Canada, in 2008. She is currently an Associate Professor with the Department of Physics and Computer Science, Wilfrid Laurier University, Canada. Her main areas of research are data mining and information retrieval.