IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 21, 2019, accepted August 11, 2019, date of publication August 15, 2019, date of current version August 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935504

Deep Temporal Convolutional Networks for Short-Term Traffic Flow Forecasting

WENTIAN ZHAO[1], YANYUN GAO[2], TINGXIANG JI[1], XILI WAN[1], FENG YE[3], AND GUANGWEI BAI[1]

[1]School of Computer Science and Technology, Nanjing Tech University, Nanjing 211816, China
[2]School of Electrical Engineering and Electronic Information, Xihua University, Chengdu 610039, China
[3]Department of Electrical and Computer Engineering, University of Dayton, Dayton, OH 45469, USA

Corresponding author: Xili Wan (xiliwan@njtech.edu.cn)

This work was supported in part by the National Nature Science Foundation of China under Grant 61602235 and Grant 61802176, and in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20161007.

ABSTRACT To reduce the increasingly congestion in cities, it is essential for intelligent transportation system (ITS) to accurately forecast the short-term traffic flow to identify the potential congestion sites. In recent years, the emerging deep learning method has been introduced to design traffic flow predictors, such as recurrent neural network (RNN) and long short-term memory (LSTM), which has demonstrated its promising results. In this paper, different from existing work, we study the temporal convolutional network (TCN) and propose a deep learning framework based on TCN model for short-term city-wide traffic forecast to accurately capture the temporal and spatial evolution of traffic flow. Moreover, we design the model with the Taguchi method to develop an optimized structure of the TCN model, which not only reduces the number of experiments, but also yields high accuracy of forecasting results. With the real-world traffic flow data collected from highways in Birmingham City of U.K., we compare our model with four deep learning based models including LSTM models, GRU models, SAE models, DeepTrend and CNN-LSTM models in terms of the mean absolute error (MAE) and mean relative error (MRE) regarding the actual flow data. The experimental results demonstrate that our framework achieves the state-of-art performance with superior accuracy in short-term traffic flow forecasting.

INDEX TERMS Deep learning, temporal convolutional networks, short-term forecasting.

I. INTRODUCTION

Intelligent Traffic System (ITS) [1] has become a powerful approach for traffic control in cities. The accurate and timely traffic condition information generated from ITS not only allows residents in cities to make smart commuting decisions, but also benefits traffic congestion relief in cities [2]. It is essential to develop a predictor to forecast accurate traffic flow condition for smart traffic operations and controls in cities.

In general, traffic flow forecasting can be classified into two categories, i.e., short-term and long-term traffic flow forecasting. The long-term forecasting focuses on monthly or yearly traffic flow forecasting which are useful for city construction or transportation programming. The short-term forecasting dedicates to real-time forecast of the potential traffic flow for a short period time ahead, e.g. a few minutes. Its forecasting results are broadly used in traffic control, congestion anticipation and so on. Since the performance of ITS largely depends on the accuracy of real-time short-term traffic flow prediction [3], short-term traffic flow forecasting has been playing a significant role in ITS. In this paper, we focuses on developing an accurate short-term traffic flow predictor, which receives past traffic flow data within the past short time and outputs the coming future traffic flow information.

To achieve the high accuracy of traffic flow forecasting, much work has been done on developing various predicting methods and models for the traffic flow forecasting. In general, previous studies on short-term traffic prediction can be roughly divided into two categories, namely parametric approaches and non-parametric approaches. Autoregressive integrated moving average (ARIMA) model is one of the typical parametric approach which is currently widely recognized framework for traffic forecasting [4]. Ding *et al.* proposed a space-time autogressive integrated

The associate editor coordinating the review of this article and approving it for publication was Dezhong Peng.

moving average (STARIMA) model to forecast the traffic volume in urban areas [5]. Chen *et al.* proposed an autoregressive integrated moving average with generalized autoregressive conditional heteroscedasticity (ARIMA-GARCH) model for traffic flow forecasting [6]. However, this type of model is limited by its stationary assumption of time sequences, i.e., it does not take the spatio-temporal correlation into account for traffic flow prediction. Inevitably, this method may lead to less accuracy when applied for short-term traffic flow forecasting.

Another typical parametric method is to apply Kalman filtering model. Adaptive Kalman filtering model has been demonstrated to have the improved ability to adapt the volatile traffic [3]. In fact, parametric approaches can achieve good performance on traffic flow with regular variations. However, the prediction bias becomes obvious when the traffic shows irregular variations [7].

To avoid the drawback of the parametric approach, researchers tried to seek non-parametric approaches for the short-term traffic flow forecasting. Many techniques have been applied or adapted from different disciplines, including Support vector machine (SVM) [8], k-nearest neighbors (K-NN) algorithm [9], neural network prediction [10]. Many researchers have attempted to change the choices of the kernel function and parameters for optimizing the SVMs [11], such as chaos wavelet analysis SVMs [12], genetic algorithm SVMs [13], novel wavelet-SVM [14] and single-step prediction SVMs [15]. The results of these methods have shown the superior performance of the non-parametric methods, compared to traditional parametric methods.

In recent years, the emerging deep learning method, especially deep Convolutional Neural Networks (CNN), has achieved great successful applications in traditionally hard or intractable tasks, including object classification, image recognition and natural language processing. Also, in the domain of ITS, attentions have started to shift to utilize deep learning based methods for traffic flow forecasting [16].

Some deep learning based methods have been proposed on the short-term traffic flow forecasting and promising results have been demonstrated. Lv *et al.* [17] first applied a deep architecture model with stacked auto-encoder (SAE) as building blocks to capture the nonlinear spatio-temporal effects. Polson and Sokolov [16] proposed an deep learning architecture by combing a linear model fitted with l_1 regularization and a sequence of tanh layers to predict the short-term traffic flow. Although the deep learning technique is applied in the above mentioned work, their experimental results shows that the temporal-spatial correlation is not well captured [16], [17]. Other variants of neural network have been proposed for traffic forecasting, including feed forward neural network [18], recurrent neural network (RNN) [19] and long short-term memory (LSTM) [7]. Among these neural networks, LSTM [7], [20] has achieved the state-of-art performance of short-term traffic flow forecasting in various conditions [16], [21]. Dai et al. [22] proposed a DeepTrend method which was improved by LSTM. Duan *et al.* [23] proposed a deep hybrid CNN-LSTM model which can achieve higher prediction accuracy.

By the internal gating method, RNNs and its variants rely on gated nonlinearities (e.g., LSTM, GRU) to model deterministic hidden state which actually act as an internal memory in the model. Information are then propagated through these hidden states. By this way, RNNs can represent long-term dependencies in sequential data and demonstrated their excellent performance in traffic flow forecasting.

However, RNNs and its variants, which are built on recurrent architectures, have been found empirically to have a limited span of attention and are difficult to interpret, although they can capture latent temporal patterns. As shown from both the empirical evaluation in [24] and theoretical understanding work in [25], RNNs and its variants are hard to introspect and difficult to correctly train. Moreover, empirical exploration work on RNNs has showed that a network architecture for better results is not trivial to find [26]. These issues inevitably affect RNNs and its variants to further improve their performance when applying on traffic flow forecasting.

Recent research results suggest that TCNs convincingly outperform baseline recurrent architectures over a broad range of sequence modeling tasks, including action segmentation [27] and speech analysis and synthesis tasks [28], [29]. Our work is motivated by the recent success of temporal convolutional network (TCN) on for these sequence modelling tasks, since traffic flow forecasting falls into the task of sequence modelling. Without utilizing recurrence architectures, these work on TCNs [27]–[29] has demonstrated that TCNs not only achieve better performance and but also reduce the computational cost for training, compared to that of RNNs and its variants.

Distinct from previous convolutional architectures for sequential data [28], [30]–[32], TCNs have the following distinguishing characteristics:

1) By stacking casual dilated convolutions, TCN is able to have flexible receptive fields which can be scaled up as large as the entire sequence length. Moreover, by augmenting with residual layers, dilated convolutions can have longer effective history sizes. Combining residual layers and dilated convolutions, TCNs can have much longer memory and model longer time scales up to the entire sequence [24].
2) Unlike RNNs, there is no explicit temporal dependency between predictions for adjacent timesteps and hence the convolutions can be performed in parallel. Thus, no matter for training or evaluation, TCNs can process a long input as a whole sequence.
3) By introducing a hierarchy of temporal convolutional filters, referred the temporal hierarchy, TCNs can capture long-range patterns. In particular, the sequence information learnt from local layers is propagated through the temporal hierarchy by residual block, while the upper layers learn representations at a larger time scale with access to longer input sub-sequences [33].

The above summarized characteristics enable the TCNs to have longer memory, large receptive ïĄeld size and the ability to process the input sequence as a whole such that the long-range patterns for traffic prediction with high accuracy can be captured. Thus, different from previous work, we develop the TCN model for capturing the temporal and spatial evolution features of traffic flow, aiming to build short-term traffic flow predictor for cities. The main contributions of this paper are summarized as the following:

1) We present a deep learning framework based on the emerging temporal convolutional network without relying on the recurrent architecture, where the stacked casual dilated convolutions and the hierarchy of temporal convolutional filter are used to enable much larger receptive fields and model longer time scales up to the entire sequence such that the long-range patterns can be captured. The framework not only achieves high accurate forecasting result but also simpler and clearer, compared to the state-of-art performance and neural architecture.
2) Traditionally, the topology of a neural network is determined by trial-and-error method, which is usually time-consuming and not efficient in yielding high accuracy. To address this problem, we introduce the Taguchi method in our framework as a new means of determining optimal topologies of the Temporal Convolutional Networks (TCN), by utilizing an orthogonal array to simultaneously study the significance of the design factors of the proposed TCN model. Compared with the commonly used trial-and-error network training method, the introduced Taguchi method not only reduces the number of experiments to find an optimized structure, but also yields high accuracy of forecasting results.
3) Extensive experiments on real-world traffic flow data trace collected from highways in Birmingham City of U.K. are done to validate the effectiveness of our deep learning framework for short-term traffic flow forecasting. The experimental results shows that our framework achieves the state-of-art performance with superior accuracy in short-term traffic flow forecasting.

The remainder of this paper is organized as follows. Section II describes the details of our proposed framework, including the data acquisition, model training, and model evaluation. Section III introduces the source of the traffic flow data and explains the methodologies and designs used in this paper. Section IV describes the experimental results and evaluates the performance of the TCN model with an optimized structure. Finally the conclusion is given in Section V.

II. THE OVERALL FRAMEWORK

The architecture of our deep learning framework is presented in Figure 1. To successfully apply the deep learning model, three phases are included in our framework, i.e., data acquisition, model training, and model evaluation.

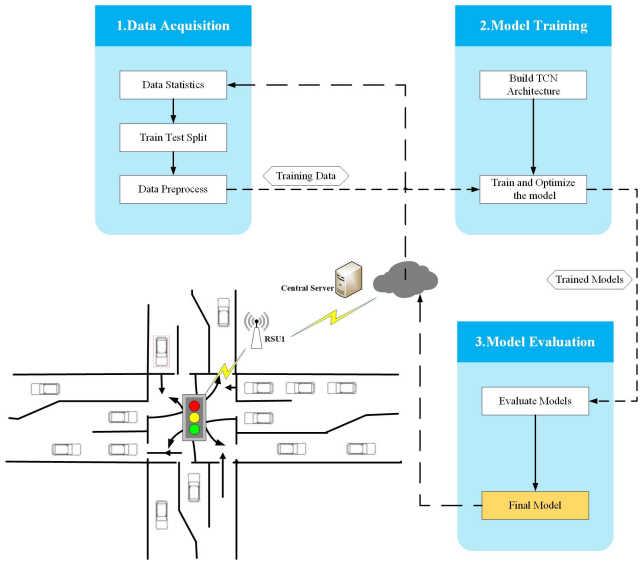


FIGURE 1. The Architecture of the deep learning framework for short-term traffic flow forecasting.

In our proposed framework, the traffic flow data is first obtained and reprocessed to train the TCN model. The traffic flow data is divided into two parts, training data and testing data, which will be used for training and evaluation of the TCN network, respectively. Moreover, the format of the traffic flow data must be transformed to the input format of TCN network. The TCN model is first built and trained with the prepared training data. Then it is further optimized by utilizing Taguchi method. In this phase, multiple modes of the TCN model are trained by adjusting various configurations of neural network parameters. Finally, the trained and optimized network is evaluated by the testing data. With the predefined evaluation metrics, the TCN model with the best performance is selected for our framework.

A. DATA ACQUISITION AND PREPROCESSING

The dataset of the traffic flow for training the TCN model in the propposed framework is obtained from the Road Side Units (RSUs) [34] deployed in Birmingham City of United Kingdom. According to the vehicle information obtained by a RSU, the total number of vehicles at the intersection is achieved every 15 minutes. As shown in Figure 2, each rectangle represents the sum of vehicles passing through the intersection in 15 minutes. Then the data preprocessing is done as follows. We can divide the data set into several groups, each of which contains five ground truths and a predicted value. As shown in Figure 3, the list {21,18,35,31,39,47,36,39,33,31,30,27, ...} is the initial traffic flow data. The value of elements in this list represents the total number of vehicles arriving at the intersection every 15 minutes. We set the first five value {21,18,35,31,39} in the list as a group of ground truth and we will use this group of ground truth to predict the sixth value of 47 in the list. Similarly, the next group of ground truth is {36,39,33,31,30} and the predicted value is 27. Then we can divide the processed

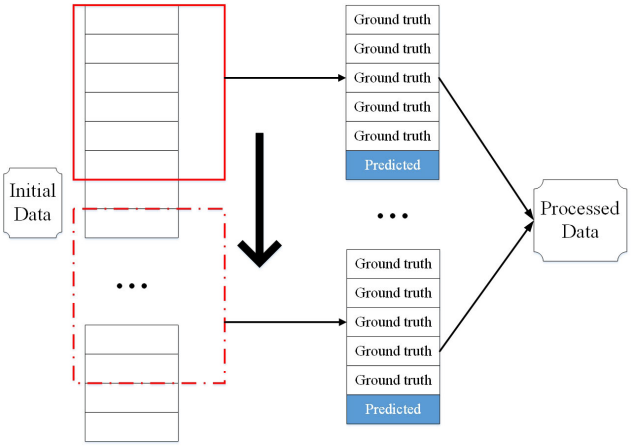


FIGURE 2. The process of data processing.

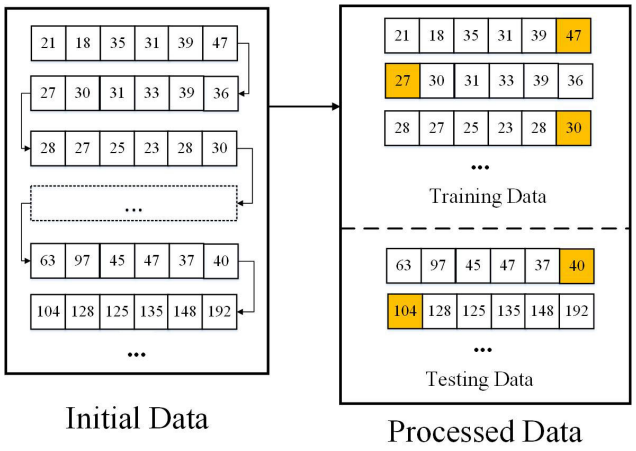


FIGURE 3. An example of the data processing.

data into two parts, training data and testing data. Finally, the training data can be sent to the TCN framework for training and the testing data can be sent to the TCN framework for testing.

B. TEMPORAL CONVOLUTIONAL NETWORK MODEL

Before defining the network structure, we first describe a generic architecture for convolutional sequence prediction which is the key part of network. Suppose that we are given a series of traffic flow data $\{x_0, \ldots, x_T\}$ and apply these data to predict the traffic flow data $\{y_0, \ldots, y_T\}$ at the next period time. Note that we are constrained to utilize the existing observed data $\{x_0, \ldots, x_{t-1}\}$ as the inputs to predict the outputs y_t for some time t. A sequence modeling network can be expressed as a function $f: X^T \rightarrow Y^T$ that defines the following mapping

$$\{\hat{y_0}, \ldots, \hat{y_T}\} = f(x_0, \ldots, x_T) \tag{1}$$

If the function f satisfies the causal condition that y_t depends only on $\{x_0, \ldots, x_{t-1}\}$ rather than any "future" inputs $\{x_{t+1}, \ldots, x_T\}$, we have the function f if we want to predict the traffic flow y_t at time t,

$$\hat{y_t} = f(x_0, \ldots, x_{t-1}) \tag{2}$$

In fact, the short-term traffic flow forecasting can be considered as the sequence modeling task. The goal of learning the setting for the sequence modeling is to find a network f that can minimize the expected loss between the actual data and the prediction, i.e., $L(\hat{y_t}, \mathrm{f}(x_0, \ldots, x_{t-1}))$.

The temporal convolutional network (TCN) serves as the core building block of the proposed deep learning framework. Our TCN model is inspired and adapted from [27] which was originally designed for action segmentation and detection. Different from existing neural networks for short-term traffic flow forecasting, the TCN has its outstanding characteristics. First, the causal convolutions is utilized in the TCN, where an output at time t is convoluted only with the elements from time t and earlier in the previous layer. This characteristic naturally corresponds the sequence prediction described in above. The causal convolutions actually play a role of a filter at time t which can only see inputs no later than t. This leads to no information leakage from future to past [24]. Second, the TCN uses a 1D fully-convolutional network (FCN) architecture [35] to map an input sequence of any length to an output sequence of the same length, where each hidden layer is the same length as the input layer and zero padding is added to keep the same length for the subsequent layers. This gives our framework to handle a data sequence input with any length.

However, to enable accurate the short-term traffic flow prediction, it needs very deep networks and a long effective history size. This will definitely lead to a complicated network structure and inevitably heavy computation. Alternatively, dilated convolutions and residual layers are integrated in the proposed TCN architecture. Particularly, dilated convolutions enable an exponentially large receptive field [24]. Formally, for a 1-D sequence input $x \in R$ and a filter $f : \{0, \ldots, k-1\} \rightarrow R$, the traffic flow F at time t is defined as

$$F(t) = (x *_d f)(t) = \sum_{i=0}^{k-1} f(i) \cdot x_{t-d \cdot i} \tag{3}$$

where d is the dilation factor, k is the filter size, and $t - d \cdot i$ indicates the direction of the past. The dilation factor is a fixed step between every two adjacent filter taps. A dilated convolution with dilation factor $d = 1$ actually is a regular convolution. The receptive field of the TCN is adjusted by the dilation factor. In our framework, d is adjusted exponentially with the depth of the network, which ensures an extremely large effective history. Consequently, the increase of dilation enables the increase of the receptive field. This leads the output at the top level to represent a wider range of inputs. Note that filter size k can also be adjusted to increase the receptive field of the TCN. An illustration from [27] is given in Figure 4.

Another important component of TCN is the residual connection [36], which contains a branch leading out to a series of transformations $\mathcal{F}$, whose outputs are added to the

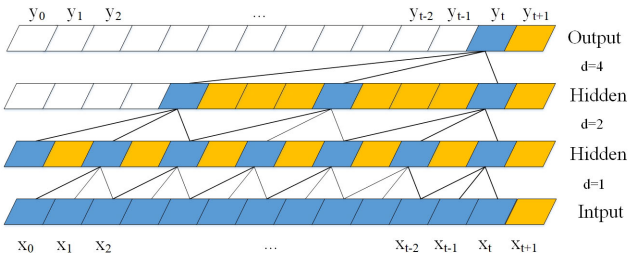


FIGURE 4. Architectural elements in a TCN:dilation factors d = 1,2,4 and filter size l = 3.

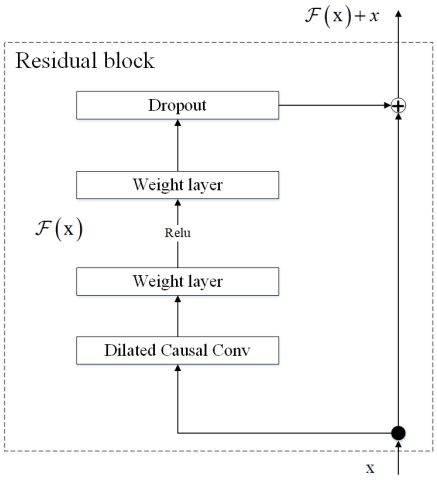


FIGURE 5. TCN residual block.

input x of the block. The residual block for our TCN is shown in Figure 5. Within a residual block, the TCN has two weight layers, for which we used the rectified linear unit (ReLU) [37]. In addition, a spatial dropout [38] is added after the last weight layer for regularization. Formally, in this paper we consider the residual block defined as:

$$y = \mathcal{F}(x, W_i) + x \tag{4}$$

Here y is the output vector of the layer considered. The function $\mathcal{F}(x, W_i)$ represents the residual mapping to be learned, where W_i represents the weights of layer i. For the example in Figure 5 which has two layers, $\mathcal{F} = W_2\sigma(W_1x) + e$ in which σ denotes ReLU [37] and e is the bias.

Following the original TCN setting, the TCN architecture for our framework is constructed based on [27] in this work. It is composited by a series of blocks, each of which contains a sequence of L convolutional layers. Each layer is composited by dilated convolutions, each of which is associated with a dilation factor d, a non-linear activation $f(.)$. Also, a residual connection is added into each dilated convolution to integrate the convolution result with the layer's input. Let the activations for the ith layer and jth block be $S^{(i,j)} \in R^{F_w \times T}$. Note the number of filters F_w in each layer i is the same. Let $\hat{S}_t^{(j,l)}$ and $S_t^{(j,l)}$ be the output of the dilated convolution at time t and the convolution result after the residual connection, respectively.

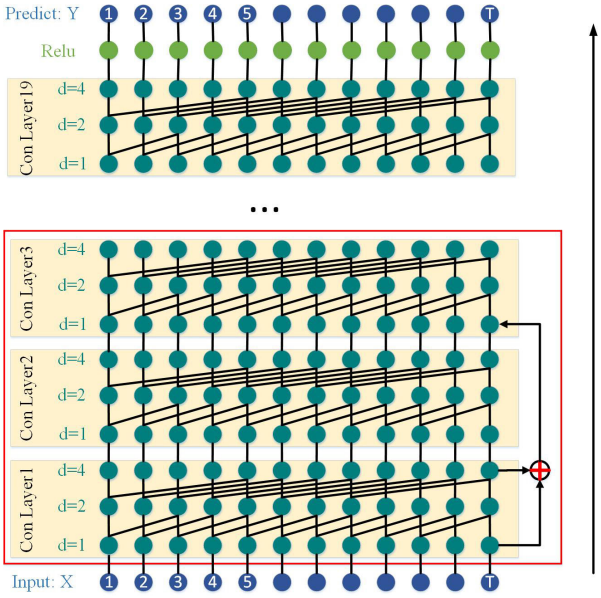


FIGURE 6. The architecture of TCN for traffic flow prediction.

Then we have

$$\hat{S}_t^{(j,l)} = f(W^1 S_{t-d}^{(j,l-1)} + W^2 S_t^{(j,l-1)} + b)$$
$$S^{(j,l)} = S_t^{(j,l)} + V\hat{S}_t^{(j,l)} + e. \tag{5}$$

W^1 and W^2 are the weight parameters. $V \in R^{F_w \times F_w}$ and $e \in R^{F_w}$ are a set of weights and biases for the residual, respectively.

The output of each block is summed by a group of skip connection with $Z_t^{(0)} \in R^{T \times F_w}$ satisfying $Z_t^{(0)} = ReLU(\sum_{j=1}^{B} S_t^{(j,L)})$,

The latent result $Z_t^{(1)}$ is $ReLU(V_r Z_t^{(0)} + e_r)$ for the weight matrix V_r and the bias e_r. Then the final forecasting result $\hat{y}_t$ for each time t is

$$\hat{y}_t = softmax(UZ_t^{(1)} + c). \tag{6}$$

with weight matrix $U \in R^{(C \times F_w)}$ and bias $c \in R^C$.

We choose the equation (7) as our model's objective function. We keep training the data to minimize the value of L. The value of $\hat{y}_i$ represents the forecasting result at time i. The value of $F(i)$ represents the initial traffic flow at time i. When we train the TCN model, we set the value of batch size to 128, the value of epoch to 30, the value of dropout rate to 0.5 and the value of initial learning rate to 0.002. In the course of training, we adopt the method of stochastic gradient descent to reduce the learning rate.

$$L = \frac{1}{t}\sum_{i=0}^{t}(\hat{y}_i - F(i))^2. \tag{7}$$

The overall model architecture follows nineteen convolutional layers and one fully connected layer which is shown in Figure 6. The rectangular area with yellow backgroud in Figure 6 represents the convolutional layer which is shown in Figure 4. The red rectangle in Figure 6 contains three

FIGURE 7. The selected sub-area (marked by a red circle) road network of Birmingham, U.K.

TABLE 1. Model architectures.

Layer(type)	Output Shape	Connected to
input_layer	(None, 6, 1)	
initial_conv	(None, 6, 12)	input_layer[0][0]
dilated_conv_1	(None, 6, 12)	initial_conv[0][0]
activation_1	(None, 6, 12)	dilated_conv_1[0][0]
lambda_1	(None, 6, 12)	activation_1[0][0]
spatial_dropout1d_1	(None, 6, 12)	lambda_1[0][0]
conv1d_1	(None, 6, 12)	spatial_dropout1d_1[0][0]
add_1	(None, 6, 12)	conv1d_1[0][0] initial_conv[0][0]
dilated_conv_2	(None, 6, 12)	add_1[0][0]
...	...	...
add_19	(None, 6, 12)	conv1d_1[0][0] conv1d_2[0][0] conv1d_3[0][0] ... conv1d_18[0][0]
activation_19	(None, 6, 12)	add_19[0][0]
lambda_19	(None, 6)	activation_19[0][0]
dense_1	(None, 1)	lambda_19[0][0]
output_dense	(None, 1)	dense_1[0][0]

yellow rectangle areas which represents the input of the convolutional layer1. Moreover, the output of the convolutional layer1 can be used as the input of the convolution layer3 directly through the residual structure which is shown in Figure 5. For all convolutional layers, the kernel size is 10 and the filter size is 24. Detailed information is shown in Table 1. In the next section, we will introduce a way to vary the factors of the TCN model based on the Taguchi method [39], to find the optimized structure of TCN model for building short-term traffic forecasting predictor.

After building the TCN model, we can train the model with the processed training data. The final TCN with the optimized structure will be send to the central server, and predict the total number of vehicles at the intersection in the next 15 minutes based on the input of the historical traffic flow data sequence. The central server then sends the signal timing to the traffic signal control system through road side units, as shown in the Figure 1.

III. DATA DESCRIPTION AND STRUCTURE OPTIMIZATION

In this section, we first describe the traffic flow data used for our modeling training and evaluation in our framework. Then we introduce the Taguchi method to our framework and show how it is applied to optimize the structure of our TCN model.

A. DATA DESCRIPTION

We use the traffic flow data collected by loop detectors at Birmingham City of United Kingdom, which is publicly available on the website (https://data.gov.uk/). We extract the traffic volume of days from 01 January 2014 to 31 December 2014 as the original data set. The traffic data are deployed within the site M42 between M42 J1 and M5 J4A of England with a frequency of 15 minutes, as shown in Figure 7 (the intersection marked by the red circle). There are 35040 validated records in total, which are divided into two subsets: data from the first 9 months are used as the training dataset, and the others are used as the test dataset.

B. TCN STRUCTURE OPTIMIZATION WITH TAGUCHI METHOD

To apply the TCN for traffic flow forecasting, the key part is to search the best topology of the neural network, i.e., parameters combinations, including the number of filters, the size of the kernel, the list of the dilations and the number of stacks of residual blocks. Traditionally, the topology of a neural network is determined by trial-and-error method, which is

TABLE 2. Identified design factors for structure optimization.

Design factors		Level		
		1	2	3
i	The number of filters	6	12	24
ii	The size of the kernel	10	15	20
iii	The list of the dilations	list1	list2	list3
iv	The number of stacks of residual blocks to use	2	6	9

usually time-consuming and not efficient in yielding high accuracy, since it may involve a large number of trail experiments. This motivated us to seek alternative efficient ways rather than using traditional trial-and-error method.

Instead, The Taguchi method [39] has been successfully applied for the robust design of high-quality products at low cost for various manufacturing processes. Actually, we can consider the determination of topologies of TCN model with high accuracy as the product design with high quality at low cost. Driven by this view, we apply the taguchi method as a means of determining optimal topologies of our neural network, by utilizing an orthogonal array to simultaneously study the significance of the design factors of the proposed TCN model. With this way, a small number of trials are only conducted to optimize the structure of the proposed TCN model which is trained with the training dataset. The trial results indicates the difference between the predicted output of our TCN model and the practical value. With these trial results, we can efficiently evaluate the appropriate values of the design factors with the designed performance metrics. Then we can finally determine the topology of our TCN model according to these evaluation results of the design factors. The structure optimization by Taguchi method involves the following three parts, i.e., identifying the design factors, determining performance metrics, trial design and performance analysis.

1) IDENTIFICATION OF DESIGN FACTORS

The design factors identified for our TCN model and their corresponding levels are presented in Table 2. These design factors are selected since they largely involves the optimal topology determination of the TCN model.

1) Design Factor *i*: The number of filters used in the convolutional layers. The most commonly used number of filters recommended by [24] for deep structure are 6, 12, and 24. In our work, we follow the practice of using the same number of filters. 6, 12, and 24 are set as Levels 1, 2, and 3, respectively.
2) Design Factor *ii*: The size of the kernel used in each convolutional layer. In this work, we consider the size of 10, 15, and 20 as Levels 1, 2, and 3, respectively.
3) Design Factor *iii*: The number of filters and the list of the dilation. They are important since they determine the size of the TCN model. The list1 recommended by [24] is $\{2^0,2^1,2^2,2^3,2^4\}$. In addition, we set the list2 as $\{2^0,2^1,2^2,2^3,\ldots,2^9\}$ and the list3 as $\{1,2,3,\ldots,9\}$. Therefore, we set list1, list2, and list3 as Level 1, Level 2, and Level 3, respectively.
4) Design Factor *iv*: For the number of stacks of residual blocks, the minimum number of stacks is set as 2 by [40] and the maximum number of stacks suggested by [41] is 9. Therefore, the number of 2, 6, and 9 are set as Levels 1, 2, and 3, respectively.

2) SPECIFICATION OF PERFORMANCE MEASUREMENT

To evaluate the effectiveness of the TCN model for traffic forecasting, we use two performance metrics, the mean absolute error (MAE) and mean relative error (MRE), which indicate the absolute and mean of the difference between the actual traffic flow conditions and the outputs of the prediction from the proposed framework, respectively. MAE and MRE are formally defined as

$$MAE = \frac{1}{n}\sum_{i=0}^{n}\left|\hat{\ell}_i - \ell_i\right| \tag{8}$$

$$MRE = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{\hat{\ell}_i-\ell_i}{\ell_i}\right| \tag{9}$$

where $\hat{\ell}_i$ denotes the observed actual traffic flow condition, ℓ_i denotes the predicted traffic flow condition, and n is the number of forecasted points. The lower value of MAE and MRE indicates less difference between the actual flow and the prediction result, i.e., the better accuracy of the prediction result.

3) TRIAL DESIGN AND RESULTS ANALYSIS

Since there are four design factors, each of which has three levels, we use an orthogonal array $L_{16}(3^4)$ for the trial design. 16 trails from the the combination of the design factors and corresponding levels are executed to optimize the structure of the proposed TCN. Each row of the adopted orthogonal array $L_{16}(3^4)$ corresponds to a main trail, in which the training set of the collected traffic data is used to develop the TCN model. With respect to the 16 main trails, the results for three random days (Sept. 1, 2014; Sept. 2, 2014; Sept. 4, 2014) and the average results of the testing set are shown in Table 3.

By utilizing the Taguchi method, there are 81 trials and 16 main trails executed over 5 days with the collected traffic data, to assessment the performance of the TCN model. As shown in Table 3, the 16th main trial (shown in bold) with 24 filters, 2 stacks, a 15-size kernel and the dilation with list1 achieves the smallest MAE and MRE value. The top three ranked trails are the 7th, 11th, and 16th main trials, all the dilation of which are configured with list1 configuration. This indicates that the TCN model with the list1 dilation can achieve better predicted results than other two lists. Moreover, the MAE and MRE results of 1st, 4th, and 5th main trials perform poorly, which indicates that configuration with 6 filters and little size kernel cannot achieve accurate predication results. Note that the prediction accuracy is affected by the topology of the neural network. i.e., the combination of the various tested parameters, although for some parameters it

TABLE 3. Orthogonal array $L_{16}(3^4)$ and experimental results.

Main Trials	Level				Day	MAE	MRE	Average results of 5 days(1^{st} to 5^{th} September 2014)			
	i	ii	iii	iv				MAE	Rank	MRE	Rank
1	1	1	1	1	1^{st}	68.713	0.799	60.532	16	0.448	16
					2^{nd}	18.120	0.235				
					4^{th}	45.082	0.122				
2	1	2	2	2	1^{st}	21.999	0.256	22.732	7	0.101	5
					2^{nd}	0.129	0.003				
					4^{th}	3.682	0.048				
3	1	3	3	3	1^{st}	24.323	0.283	25.258	10	0.111	8
					2^{nd}	1.397	0.031				
					4^{th}	1.482	0.019				
4	1	1	2	3	1^{st}	293602	0.344	30.729	15	0.166	14
					2^{nd}	5.967	0.133				
					4^{th}	8.239	0.107				
5	1	1	3	1	1^{st}	34.211	0.398	29.502	14	0.164	13
					2^{nd}	7.522	0.167				
					4^{th}	2.617	0.034				
6	2	1	2	3	1^{st}	23.394	0.272	20.937	4	0.105	7
					2^{nd}	1.874	0.042				
					4^{th}	1.996	0.026				
7	2	2	1	3	1^{st}	16.302	0.190	20.327	3	0.094	3
					2^{nd}	1.248	0.028				
					4^{th}	4.959	0.064				
8	2	3	3	1	1^{st}	34.157	0.397	28.398	11	0.147	11
					2^{nd}	1.771	0.039				
					4^{th}	5.595	0.073				
9	2	1	3	3	1^{st}	29.517	0.343	28.499	12	0.163	12
					2^{nd}	4.285	0.095				
					4^{th}	11.494	0.149				
10	2	3	1	3	1^{st}	19.632	0.228	21.555	5	0.101	5
					2^{nd}	1.915	0.043				
					4^{th}	3.415	0.044				
11	3	3	1	2	1^{st}	13.915	0.162	20.028	2	0.093	2
					2^{nd}	2.837	0.063				
					4^{th}	4.821	0.063				
12	3	3	2	1	1^{st}	39.960	0.465	28.754	13	0.198	15
					2^{nd}	9.739	0.216				
					4^{th}	9.857	0.128				
13	3	1	1	1	1^{st}	1.487	0.017	21.784	6	0.095	4
					2^{nd}	5.183	0.115				
					4^{th}	9.102	0.118				
14	3	1	2	1	1^{st}	22.775	0.265	24.962	9	0.116	9
					2^{nd}	0.436	0.010				
					4^{th}	5.315	0.069				
15	3	2	2	2	1^{st}	33.133	0.385	24.833	8	0.144	10
					2^{nd}	4.309	0.096				
					4^{th}	2.674	0.035				
16	**3**	**2**	**1**	**1**	$\mathbf{1^{st}}$	**20.201**	**0.235**	**16.157**	**1**	**0.082**	**1**
					$\mathbf{2^{nd}}$	**0.301**	**0.007**				
					$\mathbf{4^{th}}$	**1.816**	**0.024**				

seems reducing the sample numbers can decrease the prediction accuracy in the trial result.

Since the combinations under the Taguchi method in each trail are orthogonal, the effect for each design factor can be separately evaluated [42]. For each design factor at a given level, its effect can be calculated by taking the average of the corresponding values from Table 3. For example, the Level 3 of the design factor *ii* is in the 3rd, 8th, 10th, 11th, and 12th main trails, and its average effects of this level are 24.7986 (MAE) and 0.13 (MRE). The effects of each design factor at each of the three levels are displayed in Table 4.

In addition, range analysis is applied to indicate the sensitivity of the design factors to the performance values, which is denoted by the difference between the largest and the smallest performance values for each design factor. The sensitivity results are shown in Table 4. The corresponding order of the sensitivity of the four design factors is $i > ii > iv > iii$ in terms of MAE and $i > ii > iv > iii$ in terms of MRE. The above orders indicate that the design factor *i* is always the primary factors in the proposed TCN forecasting model no matter in terms of MAE or MRE, since the lower effect value indicates better performance.

TABLE 4. Effects and the sensitivity of each design factor.

Design factor	MAE				MRE			
	i	ii	iii	iv	i	ii	iii	iv
Level1	33.7506	30.9921	26.7305	30.0127	0.198	0.1796	0.1522	0.1786
Level2	23.9432	21.0123	25.4912	22.531	0.122	0.1053	0.1383	0.1127
Level3	22.753	24.7986	27.9143	24.5508	0.1213	0.13	0.1463	0.1233
Sensitivity	10.9976	9.9798	2.4231	7.4817	0.0767	0.0743	0.0139	0.0659

TABLE 5. Performance of different methods.

Algorithm	MAE	MRE	Forecasting Accuracy
TCN	8.4257	0.0458	95.42%
LSTM	29.6075	0.1964	80.36%
GRU	36.0862	0.1922	80.78%
SAE	33.4104	0.1681	83.19%
DeepTrend	21.4055	0.1381	86.19%
CNN-LSTM	24.5798	0.1403	85.97%

The results shown in Table 3 and Table 4 indicate the optimized structure of the TCN model for short-term traffic flow forecasting consists of the design factor *i* with Level 3, design factor *ii* with Level 2, design factor *iii* with Level 2, and design factor *iv* with Level 2. That is, the TCN model with 24 filters, 6 residual blocks' stacks, the kernel size of 15 and the list of the dilation is $\{2^0,2^1,2^2,2^3,\ldots,2^9\}$ generates the most accurate forecasting results among all the trail experiments.

IV. PERFORMANCE EVALUATION OF THE TCN MODEL WITH AN OPTIMIZED STRUCTURE

In this section, we evaluate our TCN model with the structure optimized by the Taguchi method for short-term traffic flow forecasting.We conduct extensive experiments based on the prepared traffic data set. To demonstrate the accuracy of the TCN model, we compare it in terms of the MAE and MRE performance metrics with another five widely applied traffic flow predictors, which are the long short-term memory (LSTM), the Gated Recurrent Units (GRU), the Stacked Auto-encoder (SAE), DeepTrend and the CNN-LSTM. Note that the LSTM model achieves the state-of-the-art forecasting result. Also, we calculate the forecasting accuracy rate based on the the actual traffic flow data. The comparison methods (or models) use their best performance. In detail, in our experiments, we use the same training data to train the comparison models. Also, the same testing data is applied on the comparison methods to do the comparison on the performance of the comparison models. All of these methods are implemented in Python environment on a PC with Intel(R) Core(TM) E5-2620 CPU, 62 GB memory and NVIDIAGTX 1080 GPU.

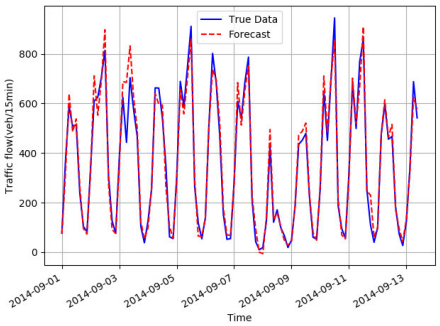
Table 5 shows the MAE and the forecasting accuracy rate of the traffic forecasting with the TCN, LSTM, GRU, SAE, DeepTrend, and CNN-LSTM models. The results shows that the proposed TCN model generates much more accuracy results than that of the other models for the prediction of the traffic flow. It is worth to mention that our TCN model with the optimized structure is able to generate the forecasting results with approximately 95% forecasting accuracy rate, which is 15% higher compared with the LSTM and GRU models and 10% higher compared with the SAE, CNN-LSTM and DeepTrend models.

Moreover, we can conclude that our TCN model achieves much lower value in terms of the MAE and MRE performance metrics, which indicates the superior forecasting performance. In detail, the MAE result of 8.4257 from our TCN model is even less than half of the MAE result of 29.6075 from the LSTM model, 24.5798 from the CNN-LSTM model and 21.4055 from the DeepTrend model. What's more, the MAE result from our TCN model is even less than one-fourth of the MAE result of 36.0862 from the GRU model and 33.4104 from SAE model. From the MRE result, similar conclusion can be made. The MRE result of 0.0458 from our TCN model is even less than one-third of the MRE result of 0.1681 from the SAE model, 0.1403 from the CNN-LSTM model and 0.1381 from the DeepTrend model. In addition, the MRE result our TCN model is even less than one-fourth of the MRE result of 0.1964 from the LSTM model and 0.1922 from the GRU model.

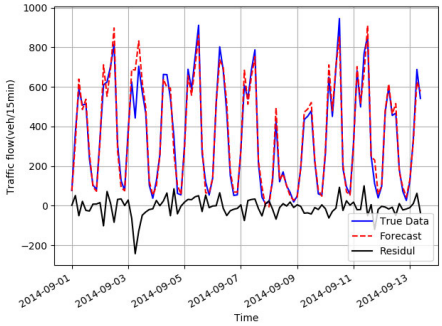
Figure 9 presents the output of the TCN, LSTM, GRU, SAE, DeepTrend, and CNN-LSTM models for traffic flow forecasting for the day of 1^{st} September 2014. The actual traffic flow is also shown in Figure 9 with the blue line. As can be seen in Figure 9, the forecasting traffic flow from our TCN model is highly approximate to the actual data line, which means the forecasting traffic flow result achieves similar traffic pattern as the observed traffic flow behaves on that day. In contrast, the forecasting performance of the LSTM, GRU, SAE, DeepTrend, and CNN-LSTM shows large difference and fluctuation from the actual traffic flow, compared to the TCN model.

To further demonstrate the performance of our TCN model, we present the difference between the actual data and forecasting output of the traffic flow over various days which begin from Sept. 1th, Sept. 21th and Oct. 15th, 2014, and the error percentage of the forecasting traffic flow results of our TCN model, as illustrated in Figure 8 (a), (b), (c), (d), (e) and (f), respectively. From the Figure 8 (a), (c) and (e), we can see that the forecasting traffic flow line has almost approximate with the actual data line except for a few individual points, such as the point at 4^{th} September 2014. The reason for this phenomenon is that the traffic flow at that day has changed dramatically, which may be caused by the weather or the holiday.

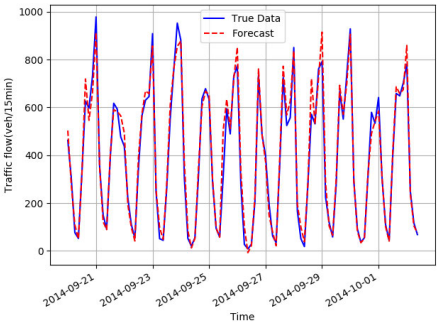
Figure 8 (b), (d) and (f) further indicates the difference between the actual data and our forecasting output. From the Figure 8 (b), (d) and (f), we can see that most of the



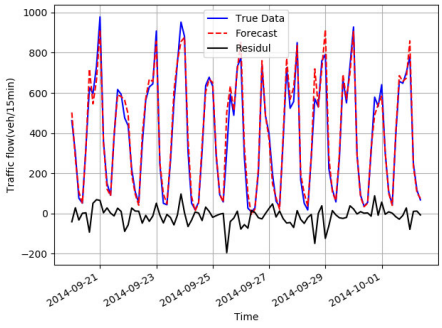
(a) Comparison of the actual and forecasting output



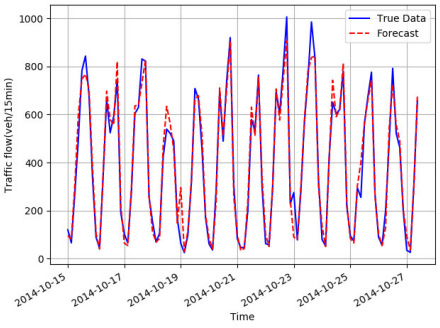
(b) Error percentage of traffic flow forecasting results



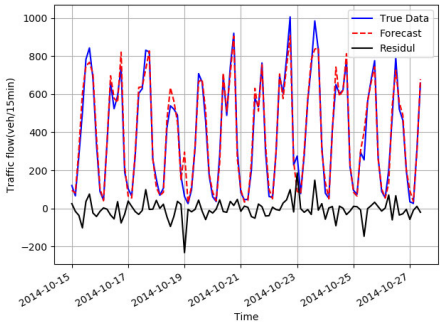
(c) Comparison of the actual and forecasting output



(d) Error percentage of traffic flow forecasting results



(e) Comparison of the actual and forecasting output



(f) Error percentage of traffic flow forecasting results

FIGURE 8. Performance evaluation of the TCN model.

percentage error of the forecasting traffic flow is limited in about 5% except for the a few points with the dramatic data fluctuation which may be caused by the weather or the holiday. Since the severe weather conditions may cause the burst (or unexpected) traffic flow, this experimental result inspires us to consider the prediction with weather conditions into our framework to address this issue. We leave this as our future work and some discussion about it is presented in next section.

In addition, we report the training time and the time for prediction in practice. With our proposed Taguchi method, we use almost half an hour to train the model to find the best parameter configurations. Given all previous input sequential data, it only takes less than one second to output the forecasting data, when applying the trained model for traffic flow prediction for the next time period.

To sum up, the proposed TCN model based deep learning framework can achieve higher accuracy and much less forecasting difference, compared with the state-of-the-art LSTM and typical GRU models. Therefore, with the evaluation on the real data trace, our framework is demonstrated to be more effective and promising for the short-term traffic flow forecasting in practice than the existing forecasting models.

Driven by this, we will consider this as our future work which is to design a traffic flow predictor with considering these unusual situations on the basis of the proposed TCN based prediction model in this work. We have noticed that it is not common to have the unusual situations (e.g. severe

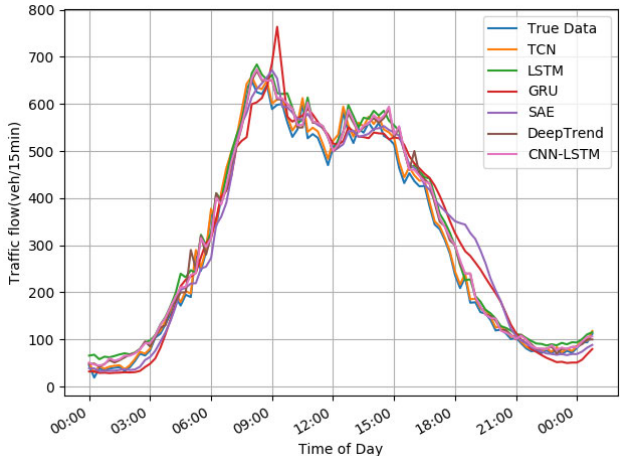


FIGURE 9. Forecasting results obtained by different methods.

weather conditions causes burst traffic flow) such that the data-driven method used in this work may not perform very well on the prediction in this kind of scenario, since less unusual situations means less available data for training.

V. CONCLUSION AND FUTURE WORK

The short-term traffic flow forecasting is a critical problem in Intelligent Traffic System. In this paper, different from previous work, we propose a deep learning framework based on the Temporal Convolutional Network. Moreover, the Taguchi method is adopted to improve the effectiveness of the design of short-term traffic flow forecasting model. With the real data trace, we compare our model with the LSTM model, the GRU model, the SAE model, the DeepTrend model and the CNN-LSTM model to validate that our model can achieve superior forecasting results. The optimized structure of the TCN found by the Taguchi method is demonstrated to have much more improved performance over other methods. The accuracy rate can reach as high as 95%. Our work demonstrates that the TCN network can be served as a effective tool for short-term traffic prediction in cities.

Driven by the experimental result in last section, we will further improve our model on the basis of the proposed TCN based prediction model in this work by considering unusual situations such as severe weather conditions as our future work. We have noticed that it is not common to have the unusual situations (e.g. severe weather conditions causes burst traffic flow) such that the data-driven method used in this work may not perform very well on the prediction in this kind of scenario, since less unusual situations means less available data for training. Our basic idea to address this issue is to build the predictor with the power of the unsupervised learning techniques. Moreover, to design a comprehensive traffic forecast, it can include travel time, traffic speed and occupancy [7]. As a future work, we will consider the further design of the TCN network to adapt the different format of traffic data, and the heavy traffic flow fluctuation influenced by the weather and holidays factors.

REFERENCES

[1] Y. Wang, N. Geroliminis, and L. Leclercq, "Recent advances in ITS, traffic flow theory, and network operations," *Transp. Res. C, Emerg. Technol.*, vol. 100, no. 68, pp. 507–508, 2016.
[2] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.
[3] J. Guo, W. Huang, and B. M. Williams, "Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 50–64, Jun. 2014.
[4] M. Levin and Y. D. Tsao, "On forecasting freeway occupancies and volumes (abridgment)," *Transp. Res. Rec.*, vol. 722, pp. 47–49, 1980.
[5] Q. Y. Ding, X. F. Wang, X. Y. Zhang, and Z. Q. Sun, "Forecasting traffic volume with space-time ARIMA model," *Adv. Mater. Res.*, vols. 156–157, pp. 979–983, Oct. 2010.
[6] C. Chen, J. Hu, Q. Meng, and Y. Zhang, "Short-time traffic flow prediction with ARIMA-GARCH model," in *Proc. Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 607–612.
[7] 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, 2017.
[8] Y. Zhang and Y. Liu, "Traffic forecasting using least squares support vector machines," *Transportmetrica*, vol. 5, no. 3, pp. 193–213, 2009.
[9] Y. Chen, Y. Zhang, and J. Hu, "Multi-dimensional traffic flow time series analysis with self-organizing maps," *Tsinghua Sci. Technol.*, vol. 13, no. 2, pp. 220–228, Apr. 2008.
[10] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transp. Res. C*, vol. 19, no. 3, pp. 387–399, Jun. 2011.
[11] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, p. 1501, 2017.
[12] 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.
[13] Z. Yang, D. Mei, Q. Yang, H. Zhou, and X. Li, "Traffic flow prediction model for large-scale road network based on cloud computing," *Math. Problems Eng.*, vol. 2014, Aug. 2014, Art. no. 926251.
[14] Y. Sun, B. Leng, and W. Guan, "A novel wavelet-SVM short-time passenger flow prediction in Beijing subway system," *Neurocomputing*, vol. 166, pp. 109–121, Oct. 2015.
[15] B. Yao, C. Chen, Q. Cao, L. Jin, M. Zhang, H. Zhu, and B. Yu, "Short-term traffic speed prediction for an urban corridor," *Comput.-Aided Civil Inf.*, vol. 32, no. 2, pp. 154–169, 2017.
[16] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transp. Res. C, Emerg. Technol.*, vol. 79, pp. 1–17, Jun. 2017.
[17] 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.
[18] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach," *Transp. Res. C, Emerg. Technol.*, vol. 13, no. 3, pp. 211–234, 2005.
[19] P. Lingras, S. Sharma, and M. Zhong, "Prediction of recreational travel using genetically designed regression and time-delay neural network models," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1805, no. 1, pp. 16–24, 2002.
[20] 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. 2015, pp. 153–158.
[21] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2017, pp. 777–785.
[22] X. Dai, R. Fu, Y. Lin, L. Li, and F.-Y. Wang, "DeepTrend: A deep hierarchical neural network for traffic flow prediction," 2017, *arXiv:1707.03213*. [Online]. Available: https://arxiv.org/abs/1707.03213
[23] Z. Duan, Y. Yang, K. Zhang, Y. Ni, and S. Bajgain, "Improved deep hybrid networks for urban traffic flow prediction using trajectory data," *IEEE Access*, vol. 6, pp. 31820–31827, 2018.
[24] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*. [Online]. Available: https://arxiv.org/abs/1803.01271

[25] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Int. Conf. Mach. Learn. (ICML)*, vol. 28, 2013, pp. III-1310–III-1318.
[26] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," *J. Mach. Learn. Res.*, vol. 37, pp. 2342–2350, Jul. 2015.
[27] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 156–165.
[28] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*. [Online]. Available: https://arxiv.org/abs/1609.03499
[29] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: Modelling raw audio at scale," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 8000–8010.
[30] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1961–1970.
[31] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, Vancouver, BC, Canada: Association for Computational Linguistics, Jul. 2017, pp. 123–135.
[32] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," 2016, *abs/1610.10099*. [Online]. Available: https://arxiv.org/abs/1610.10099
[33] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Computer Vision—ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Cham, Switzerland: Springer, 2016, pp. 47–54.
[34] A. Abdrabou and W. Zhuang, "Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity," *IEEE J. Sel.Areas Commun.*, vol. 29, no. 1, pp. 129–139, Jan. 2011.
[35] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[37] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.
[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[39] G. S. Peace, *Taguchi Methods: A Hands-on Approach*. Reading, MA, USA: Addison-Wesley, 1993.
[40] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, vol. 4, 2017, p. 12.
[41] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.
[42] M. S. Phadke, *Quality Engineering Using Robust Design*. Upper Saddle River, NJ, USA: Prentice-Hall, 1995.

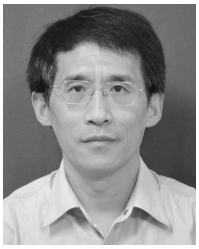
WENTIAN ZHAO received the B.S. degree in computer science from the Nanjing University of Science and Technology, in 2017. He is currently pursuing the master's degree with Nanjing Tech University. His research interests include optimization in intelligent transportation and algorithm design.

YANYUN GAO is currently pursuing the bachelor's degree with the School of Electrical Engineering and Electronic Information, Xihua University, Chengdu, China. Her research interests include artificial intelligence and cloud computing.

TINGXIANG JI is currently pursuing the bachelor's degree with the School of Computer Science and Technology, Nanjing Tech University, Nanjing, China. Her research interests include artificial intelligence and cloud computing.

XILI WAN received the B.S. and M.S. degrees in computer science from the Nanjing University of Science and Technology, and the Ph.D. degree in computer science from the University of Missouri, Kansas, USA, in 2014. He was a Researcher with the Huawei USA Research Center. Since 2016, he has been an Assistant Professor with the School of Computer Science and Technology, Nanjing Tech University, China. His research interests include optimization in wireless sensor networks, design and analysis of approximation algorithms, and cloud computing. He served as the Poster Co-Chair for the IEEE SECON 2018. He is also a Reviewer of journals and conferences, including the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE INTERNET OF THINGS JOURNAL, Springer *Telecommunications Systems* journal, the IEEE INFOCOM, the IEEE GLOBECOM, the IEEE ICC, the IEEE SECON, the IEEE ICNC, and the IEEE ISC2.

FENG YE received the B.S. degree from the Department of Electronics Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2011, and the Ph.D. degree in electrical and computer engineering from the University of Nebraska–Lincoln (UNL), NE, USA, in 2015. He was with the Department of Electrical and Computer Engineering, UNL, as an Instructor and a Researcher from 2015 to 2016. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Dayton (UD), Dayton, OH, USA. His research interests include cyber security and communication network security, wireless communications and networks, green ICT, smart grid communications and energy optimization, and big data analytics and its applications. He serves as a TPC Member for numerous international conferences, including INFOCOM, GLOBECOM, VTC, and ICC. He was a recipient of the 2015 Top Reviewer Award from the IEEE Vehicular Technology Society. He served as the Co-Chair for the IEEE ICC 2018 and the Cognitive Radio and Networking Symposium. He also served as the Publicity Co-Chair for the IEEE CBDCom 2018, and the Co-Chair for the Signal Processing for Communications Symposium, ICNC 2019. He is currently an Associate Editor of *Security and Privacy* (Wiley) and *China Communications*. He is also a Reviewer for several IEEE journals, including the IEEE TRANSACTIONS ON BIGDATA, the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, the IEEE TRANSACTIONS ON SMART GRID, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He serves as the Secretary for the IEEE Technical Committee on Green Communications and Computing.



GUANGWEI BAI received the B.Eng. and M.Eng. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in computer science from the University of Hamburg, Hamburg, Germany, in 1999. From 1999 to 2001, he was a Research Scientist with the German National Research Center for Information Technology, Germany. In 2001, he joined the University of Calgary, Canada, as a Research Associate. Since 2005, he has been a Professor of computer science with Nanjing Tech University, Nanjing, China. In 2011, he was a Visiting Professor with the University of Waterloo, Canada. His research interests include wireless networks, multimedia, edge computing, and location-based services. He is a member of the Association for Computing Machinery.