ASTCN: An Attentive Spatial–Temporal Convolutional Network for Flow Prediction

Haizhou Guo, Dian Zhang, *Member, IEEE*, Landu Jiang, Kin-Wang Poon, and Kezhong Lu

***Abstract*—Flow prediction attracts intensive research interests, since it can offer essential support to many crucial problems in public safety and smart city, e.g., epidemic spread prediction and medical resource allocation optimization. Among all the models in flow prediction, deep learning models (e.g., convolutional neural networks, recurrent neural networks, and graph neural networks) are popular and outperform other statistics and machine learning models, since they can learn intrinsic structures and extract features from spatial–temporal (ST) data. However, most of them set strict temporal periods in the prediction or separate the interaction between spatial and temporal correlations. Therefore, the prediction accuracy is affected. To overcome the difficulties, we propose a flow prediction network attentive spatial–temporal convolutional network (ASTCN), which can effectively handle large-scale flow data and learn complex features. In ASTCN, we leverage an attention mechanism to overcome the previous problem of strict temporal periods, and can effectively fuse ST data with multiple factors from different time-series sources. Furthermore, we propose a causal 3-D convolutional layer based on temporal convolutional networks (TCNs). It can simultaneously extract both spatial and temporal features to improve the prediction accuracy. We comprehensively conducted our experiments based on real-world data sets. Experimental results show that ASTCN outperforms the state-of-the-art methods by at least 3.78% in root mean square error. Therefore, ASTCN is a potential solution to other large-scale ST problems.**

***Index Terms*—Flow prediction, neural networks, spatial–temporal (ST) data, time-series prediction.**

I. Introduction

With the rapid growth of city population, the study of flow prediction has attracted a lot of attention in academia and industry. The accurate flow prediction, such as inflow and outflow of an area at a specific time slot, is a spatial–temporal (ST) prediction problem, which plays a crucial role in traffic classification [1] and traffic control [2]. For example, the flow of population mobility and transportation between different cities can be used to predict and analyze the spread of epidemics [3], [4], and provide support for public health institutions to optimize medical resource allocation. Therefore, it is desirable to make the prediction from complex flow data, which can provide essential service for public safety and intelligent city [5].

The previous methods [6]–[8] aim to predict the human mobility or the taxi-passenger demand at different locations. Their models for time-series analysis are mainly based on the statistical model (e.g., autoregressive integrated moving average) or machine learning model. However, most of them ignore or only consider very limited spatial dependency between regions. For example, Hoang *et al.* [8] assumed the same transition probabilities among regions, it will be less practical assumption when facing a large scale of locations. If their models fail to consider the ST dependency, they are not able to make accurate flow prediction on complex spatial data.

Recently, deep learning become a promising tool [9], which revolutionizes various industries, such as traffic prediction [10]–[13] including the flow prediction. Deep models, such as convolutional neural networks (CNNs) [14]–[16], have great power to learn intrinsic structures and extract features from ST data. The main problem of the methods is that they set strict temporal periods to make prediction. Yao *et al.* [17] further combined recurrent neural networks (RNNs) with CNNs to model spatial dependency and temporal dependency. They use CNN to extract features for spatial representation, then feed the features into RNN to model the temporal correlation of the features. Recently, graph neural network models, e.g., STG2Seq [18], are proposed to consider spatial dependency from a global perspective. However, these methods separate the interaction between spatial correlation and temporal correlation [19], [20]. Therefore, the above problems reduce the performance of flow prediction.

To address the above issues, we propose attentive spatial–temporal convolutional network (ASTCN), a flow prediction network which effectively handles large-scale flow data and learns complex features. To overcome the problem of strict temporal periods, the system assigns weights to different time steps by applying the attention mechanism at long-term input. To simultaneously extract ST features, we propose a causal 3-D convolutional layer based on temporal convolutional networks (TCNs) [21], which is good at processing long-term data [22]–[24]. Thus, this structure of ASTCN is capable of learning complex features in long-term ST data. In the evaluations, we use three real-world data sets [15], [17] to examine the accuracy. The experimental results demonstrate that our model outperforms other state-of-the-art methods. For

Manuscript received January 12, 2021; revised April 17, 2021 and June 12, 2021; accepted June 29, 2021. Date of publication August 10, 2021; date of current version February 21, 2022. This work was supported in part by NSFC under Grant 61872247; in part by the Shenzhen Peacock Talent under Grant 827-000175; in part by the Guangdong Natural Science Fund under Grant 2019A1515011064; in part by the Lingnan University Direct under Grant DR21A6; and in part by the Lingnan Research Seed Fund under Grant 102363. *(Corresponding authors: Dian Zhang; Kezhong Lu.)*

The authors are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: 1800271045@email.szu.edu.cn; serena.dian@gmail.com; landu.jiang@mail.mcgill.ca; eric-poon@outlook.com; kzlu@szu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3100068

the large-scale data set TaxiBJ, our model can have 3.78% improvement than the well-designed STG2Seq on root mean square error (RMSE). For the small-scale data sets BikeNYC and TaxiNYC, our model can achieve better performance than the state-of-the-art methods.

The contributions of this article are summarized as follows.

1) We propose a causal 3-D convolutional layer, which can consider spatial and temporal dependency simultaneously, and efficiently extract the features from long-term ST data by adopting stride convolution and padding in the convolutional layer.
2) We design the attention mechanism, which takes both 3-D ST features and 1-D external data into account to assign different weight values at each time step (the higher value means more important). By applying the attention mechanism at long-term input, our model can focus on the important parts and avoid strict temporal periods usually adopted by traditional methods, to enhance the prediction performance.
3) We conduct experiments on two real-world data sets. The results show that our model achieves better performance than the state-of-the-art methods. Moreover, our model has better performance for long-term data.

The remainder of this article is organized as follows. Section II reviews the related studies of flow prediction. Section III provides the definitions and the problem of the flow prediction. Section IV presents our system design and implementation details of ASTCN. Section V shows the evaluation results and discussion and Section VI concludes this article.

II. Related Work

For the task of flow prediction, deep learning becomes a promising tool to model ST dependency. There are three main categories of the deep learning models: 1) CNNs-based models; 2) recurrent-neural-networks-based (RNNs-based) models; and 3) GNNs-based models.

Zhang *et al.* [14] proposed DeepST, which utilizes CNN to model both spatial and temporal closeness, period, and trend. To make efficient prediction, ST-ResNet [15] used three residual convolutional units to model temporal closeness, period, and trend. However, their studies took temporal dependency as the channel of convolutions and lack the modeling of temporal dimension. StepDeep [16] used 3-D CNN to process temporal dimension while also considering spatial dependencies. However, the above methods set a strict temporal period (such as daily, weekly) for temporal dependency while the periods are dynamic in practice.

RNNs-based models are also proposed to handle spatial and temporal dependency. The ConvLSTM network [25] is a novel framework to solve ST problems by designing the convolution operator inner LSTM. Yao *et al.* [26] proposed DMVST-Net, which uses local CNN to capture spatial correlation and LSTM to model temporal correlation. By considering the flow gating mechanism for CNN, STDN [17] was proposed to consider dynamic spatial similarity. STDN can learn long-term periodic dependency without setting strict temporal period like previous studies [14]–[16]. However, they used CNN to learn representation from spatial dependency, then fed to RNNs to model temporal dependency in flow prediction. These models separate the interaction between spatial correlation and temporal correlation and are limited in performance [19].

Recently, researchers have investigated graph neural network models for flow prediction. Bai *et al.* [18] proposed STG2Seq to capture spatial and temporal correlations simultaneously. Based on DCRNN [27], Zhang *et al.* [28] proposed ST-GDN to consider the global geographical contextual information. However, they concatenate features of each node in different time steps and separate ST correlations [20].

To make more accurate predictions, many models considered external context data (e.g., temperature and weather) because the data are important influence factors of flow prediction. For the studies mentioned above, some works [14], [15] fed external context data to fully connected layers, then fused with ST representation. Their work overlooks the temporal dependency of external context data. Yao *et al.* [26] concatenated context data with ST representation at the same timestamp as the input of LSTM. StepDeep [16] generated the attention mask from external context data, then used convolutional layer map external context data to new meaningful values. Our model is able to use causal multihead attention to fuse 3-D ST representation and 1-D external context data to generate more meaningful attention values.

III. Problem Definition

In this section, we will briefly introduce the definition of human mobility volume, inflow, and outflow, and then define the problem of flow prediction.

The goal of the flow prediction problem is to predict the inflow and outflow of human mobility volume at all locations of an area $\mathcal{A}$. We denote the locations (i, j) of the area as $\mathcal{A}_{i,j}$ by splitting $\mathcal{A}$ into $I \times J$ equal-sized grids. Then, we define human mobility as follows.

Mobility Volume: Mobility volume is the number of people in the location of an area over time. The volume of each location can be collected from the trajectories generated from vehicles or mobile phones. We denote the set of individuals (vehicles or mobile phones in our later scenarios) as $\mathcal{U}$. For each individual $u \in \mathcal{U}$, we use $\mathcal{T}_u$ to denote the trajectory of the individual over time. The geographic coordinates of u over time can be denoted as $p_{u,t}^{x,y} \in \mathcal{T}_u$. If the individual u at timestamp t does not exist, we set the geographic coordinates (x, y) to $(\texttt{null}, \texttt{null})$. The set of human traverse location (i, j) at the corresponding time slot can be defined as

$$\mathcal{V}_{\tau}^{i,j} = \left\{\mathcal{T}_u \mid \texttt{map}\left(p_{u,t}^{x,y}\right) \in \mathcal{A}_{i,j}\right\}, \tau - \delta \leq t \leq \tau \quad (1)$$

where $\texttt{map}$ is a function to map geographic coordinates to $\mathcal{A}_{i,j}$, δ is the time interval, and τ represents the time slot.

The human mobility total volume is denoted as $|\mathcal{V}_{\tau}^{i,j}|$. Having $\mathcal{V}_{\tau}^{i,j}$, inflow/outflow of mobility volume can be defined as follows.

Inflow/Outflow: For all locations $\{(i,j) \mid 1 \le i \le I, 1 \le j \le J\}$ of $\mathcal{A}_{i,j}$, we can define the inflow and outflow of mobility volume as follows:

$$F^{i,j}_{\tau,\texttt{in}} = \left|\left\{\mathcal{T}_u \mid \mathcal{T}_u \notin \mathcal{V}^{i,j}_{\tau-1} \wedge \mathcal{T}_u \in \mathcal{V}^{i,j}_{\tau}\right\}\right| \tag{2}$$

$$F^{i,j}_{\tau,\texttt{out}} = \left|\left\{\mathcal{T}_u \mid \mathcal{T}_u \in \mathcal{V}^{i,j}_{\tau} \wedge \mathcal{T}_u \notin \mathcal{V}^{i,j}_{\tau+1}\right\}\right|. \tag{3}$$

Then, the inflow and outflow of the area can be denoted as $\mathcal{F}^c_\tau \in \mathbb{R}^{I\times J\times 2}$, where $c = \texttt{in}$ means inflow $F^{i,j}_{\tau,\texttt{in}}$, $c = \texttt{out}$ means outflow $F^{i,j}_{\tau,\texttt{out}}$.

External Context Data: External context data are the factors (meteorological and event features) from daily life. Meteorological data, such as weather condition, temperature, and so on, are considered in external context data, which can influence the flow prediction and can be accurately obtained from public websites (such as, weather underground)[1]. To comprehensively consider the factors of human activities, event features occurring in holiday are considered. In our article, the weather condition and event features are processed as one-hot vectors, and the others are continuous value.

Then, we denote the external context data as $E_\tau \in \mathbb{R}^m$ at the same time slot, where m denotes the number of features of external context data.

As shown in Fig. 1, the data of history (l time steps) are the input, including $\mathcal{F}^c_t$ and E_t, where $t = \{\tau-l+1, \ldots, \tau-1, \tau\}$. Each grid in $\mathcal{F}^c_t$ corresponds to the inflow and outflow of a region in the real map. Given historical inflow/outflow data and external context data, we can predict future inflows and outflows. Therefore, Flow Prediction is formally defined as follows.

Problem 1: Given the history inflow and outflow of an area and external context data, our goal is to predict the inflow and outflow of the next time slot. We define our problem as follows: given a time slot τ and its previous history l steps, the inflow and outflow of an area can be represented as $\{\mathcal{F}^c_t \mid t = \tau-l+1, \ldots, \tau-1, \tau\}$, and external context data can be represented as $\{E_t \mid t = \tau-l+1, \ldots, \tau-1, \tau\}$. Our goal is to predict the inflow and outflow $\mathcal{F}^c_{\tau+1}$ of the next time step $\tau+1$.

Formally, our prediction problem is

$$\hat{\mathcal{F}}^c_{\tau+1} = \mathcal{G}\left(\mathcal{F}^c_t, E_t\right), t = \tau-l+1, \ldots, \tau-1, \tau \tag{4}$$

where $\hat{\mathcal{F}}^c_{\tau+1}$ is the prediction result and $\mathcal{G}$ is the function we need to build.

IV. Methodology

In this section, we will present the details of ASTCN in the flow prediction.

A. Overview

The network of ASTCN consists of three parts: 1) downsampling; 2) ST features extraction; and 3) upsampling, as shown in Fig. 2. The input data and target are shown in Fig. 1.

The first part is to model spatial dependency and reduce the spatial feature map. We perform downsampling through a convolution layer with stride. The spatial feature map will be smaller than what it was before, while the temporal dimension is still not changed.

[1]https://www.wunderground.com/

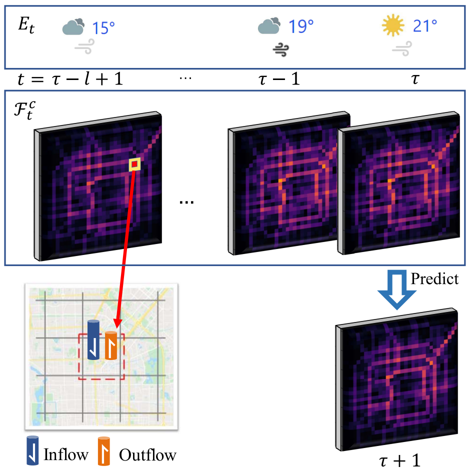


Fig. 1. Problem definition of flow prediction.

For the second part, we propose ST block to model ST dependency. ASTCN is stacked with a series of novel ST blocks. Each block consists of two main components.

1) One is *attention layer*, which combines spatial information and external context information to calculate important values for each time step. The important values multiplied with the spatial features at the corresponding time step will generate the ST features that have different importance distribution at the temporal dimension. The larger important values can increase the spatial representation values at specific time steps. Thus, our model can pay different attention to each time step.
2) The other one is *causal 3-D convolution*, which can efficiently capture ST dependency. We design a 3-D convolution, which can keep the causality character of TCNs and efficiently model the ST dependency. By applying causal 3-D convolution, the temporal features are further extracted and the length of temporal dimension will be reduced.

After a series of ST blocks, the temporal dimension will reduce to 1. Then, we can apply upsampling to expand the spatial feature map and use the convolutional layers to output our inflow and outflow prediction in the last part.

B. Downsampling

In the input stage, we use $\kappa_f \times \kappa_f$ convolution with stride of κ_s to extract spatial representation and reduce the computation by reducing the feature maps of spatial representation. For the input $\mathcal{F}^c_t \in \mathbb{R}^{I\times J\times 2}$, where $t = \tau-l+1, \ldots, \tau-1, \tau$. After the convolution layer, our spatial feature maps are reduced from $(I \times J)$ to $([I/2] \times [J/2])$.

We represent the feature maps as $\mathcal{X} \in \mathbb{R}^{[I/2]\times[J/2]\times C^0\times l}$, where C^0 represents the number of channels and l represents the dimension of temporal.

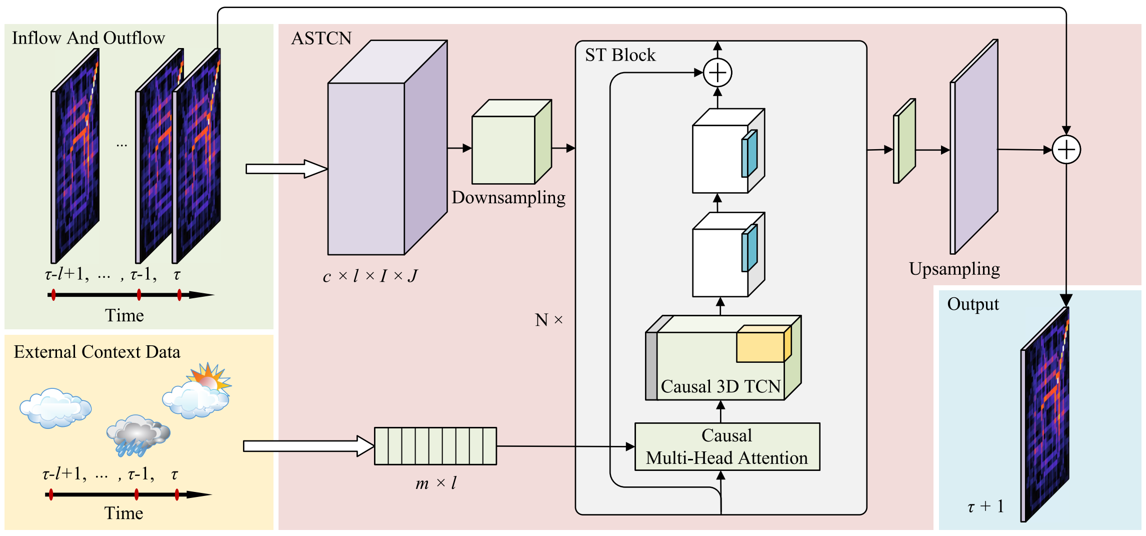


Fig. 2. Framework of ASTCN.

C. Spatial–Temporal Block

In order to reduce the complexity and achieve long-term memory, we propose ST block to extract spatial–temporal features and reduce the temporal dimension when facing long-term data. Moreover, we can rapidly merge the information of temporal by stacking several ST blocks. Letting K represent the number of ST blocks, the temporal dimension after the kth ST block can be reduced to l^k, where $l > l^1 > \cdots > l^k > \cdots > l^K = 1$.

For the first ST block, it takes the $\mathcal{X}$ as ST representation $\mathcal{X}^0$ and $\{E_t \mid t = \tau - l + 1, \ldots, \tau - 1, \tau\}$ as external input $\mathcal{E}^0$. Formally, the output of each ST block can be formulated as follows:

$$\mathcal{X}^k = \mathcal{B}^k\left(\mathcal{X}^{k-1}, \mathcal{E}^{k-1}\right) \tag{5}$$

where $\mathcal{X}^k \in \mathbb{R}^{[I/2]\times[J/2]\times C^k \times l^k}$ and $\mathcal{B}^k$ is the nonlinear function of the kth ST block.

The ST block consists of four layers: an attention layer, a causal 3-D convolution layer, and two convolutional layers. For the attention layer and causal 3-D convolution layer, we describe the details of them in the following section.

D. Attention Layer

For our model, we proposed an attention layer, which can combine both spatial information and external context information as shown in Fig. 3.

The layer takes the 3-D ST features and 1-D external context features as input. By fusing the two features, we can obtain the features containing both ST information and external context information. Then, we use the features to dynamically calculate the important values for the time steps of input. For the attention layer, the first step is the representation fusion, which aims to fuse the 3-D ST information and 1-D external context information. Then, the representations are used to calculate attention values by causal multihead attention, which consists of multiple self-attention layers.

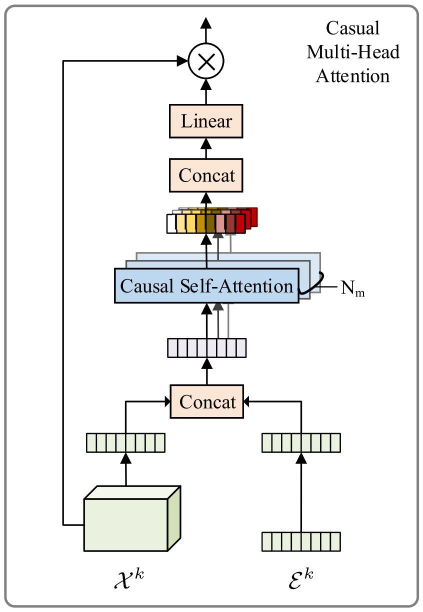


Fig. 3. Attention layer.

1) Representation Fusion: Because the inputs of attention layer are $\mathcal{X}^k \in \mathbb{R}^{[I/2]\times[J/2]\times C^k \times l^k}$ and $\mathcal{B}^k \in \mathbb{R}^{m \times l^k}$, these two kinds of data have different dimensions and cannot be fused them directly. Inspired by SENet [29], we squeeze the spatial dimensions ($[I/2] \times [J/2]$) to (1, 1) by adopting global average pooling, which are denoted as $\mathcal{X}_s^k$. For $\mathcal{B}^k$, we use two convolution layers to extract features, which are denoted as $\mathcal{X}_e^k$. Then, we concatenate $\mathcal{X}_s^k$ and $\mathcal{X}_e^k$ as $\mathcal{X}_F^k$, which contain both the ST information and external context information. The features can help the attention mechanism to calculate meaningful features.

2) Causal Multihead Attention: Causal multihead attention takes $\mathcal{X}_F^k$ as the input and aims to calculate the important values for each time step of $\mathcal{X}^k$. It consists multiple causal self-attention layers as shown in Fig. 3. Each causal self-attention

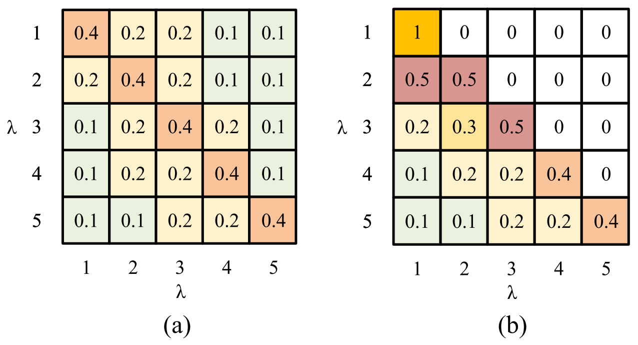


Fig. 4. Example of causal self-attention layer. (a) Anti-causal. (b) Causal.

layer can calculate attention values for different time steps. By stacking multiple layers, causal multihead attention can obtain comprehensive important values. Then, we use a matrix to fuse the important values of each self-attention layer and obtain the final attention values.

Causal self-attention takes $\mathcal{X}_F^k$ as the input and is multiplied by three same-size matrixes to form Q, K, and V, which represent *query*, *key*, and *value*, respectively, [30]. The process of self-attention can be defined as follows:

$$\texttt{Attention}(Q, K, V) = \texttt{softmax}\left(\frac{QK^T}{q_k}\right)V \qquad (6)$$

where QK^T is a square matrix and q_k is the temporal dimension of Q.

However, Attention invalidates causality of the network, because the operation involves the future temporal representation. To better understand the above problem, we show an example matrix generated from $\texttt{softmax}(QK^T/q_k)$, whose temporal dimension is 5 as illustrated in left subfigure of Fig. 4. The rows and columns represent the time step λ, and the number in the matrix represent the correlation between the two time steps. For example, the first row of the matrix shows the correlation values (0.4, 0.2, 0.2, 0.1, 0.1) of time step 1 at each time step (1, 2, 3, 4, 5). The summation of the correlation values equals 1. However, the process of the calculation is not causal because the operation of QK^T involves the future temporal representation. For instance, when the model is calculating at timestamp 1, the model involves the calculation of future timestamps 2, 3, 4, and 5. If the model is causality, it should not involve the future data. Apparently, the process of (6) can invalidate the causality of the network.

Thus, we apply a mask layer, which processes the values related to the future into zeros to make self-attention causal. The mask layer is defined as follows:

$$\texttt{Mask}(X) = \begin{cases} X_{(i,j)} = X_{(i,j)}, & \text{if } i \le j \\ X_{(i,j)} = -\texttt{inf}, & \text{otherwise.} \end{cases} \qquad (7)$$

Formally, we revise (6) as

$$\texttt{Attention}(Q, K, V) = \texttt{softmax}\left(\frac{\texttt{Mask}\big(QK^T\big)}{d_k}\right)V. \qquad (8)$$

The right figure of Fig. 4 shows the result of $\texttt{softmax}([\texttt{Mask}(QK^T)]/d_k)$.

As the figure shows, the model does not involve the calculation of future timestamp anymore. The correlation can considered as important values. The larger value means more attention the model will pay to the time step. For instance, the value of (5, 1) of the matrix is 0.1 and the value of (5, 5) is 0.4. When calculating the attention value of time step 5, the model will pay less attention on time step 1 and more attention on time step 5.

Multihead attention can merge the attention values from multiple causal self-attention layers. Here, we apply a matrix to fuse the attention values to final important values $A^k \in \mathbb{R}^{l^k}$ for each time step. Before multiplying the important values by $\mathcal{X}_F^k$, we extend A^k to $\mathcal{A}^k \in \mathbb{R}^{[I/2]\times[J/2]\times C^k \times l^k}$ by duplication. Then, we multiply $\mathcal{X}_F^k$ by $\mathcal{A}^k$ to form $\mathcal{X}_A^k$.

E. Causal 3-D Convolution Layer

After obtaining $\mathcal{X}_A^k$, the next is to learn ST representation by causal 3-D convolutions. As shown in Fig. 5(a), the temporal information can be merged by applying multilayer causal 3-D convolutions. The convolutions are the main components to reduce the time step l^k. Thus, our causal 3-D convolutions use stride convolutions, which are cost effective by gradually reducing the temporal dimension. This design can avoid the massive calculation of the traditional TCNs [21] which use dilated convolutions to learn temporal features and reduce the possibility of out of memory when facing long-term data.

To realize causal 3-D convolutions, we utilize two technologies: 1) strided convolutions and 2) padding. Strided convolutions can reduce the dimension of temporal representation while cannot align the time slot τ. Padding is one of the solution to keep causality by padding zeros to the head of temporal representation.

Strided Convolutions: Fig. 5(b) shows the illustration of normal strided convolution without padding. By setting the stride of convolutions as N_s, the dimension of temporal can be reduced as $l_k = \lfloor[l_{k-1} - N_k]/N_s\rfloor + 1$, where N_k is the kernel size of temporal and is set to 2 in this article.

In this way, the convolution operation can work efficiently while the representation near the time slot τ is overlooked as shown in Fig. 5(b).

Padding: To take the above representation into account, we use padding which pads zeros on the left of $\mathcal{X}_A^k$ as shown in Fig. 5(c). By padding the zeros, the kernel of convolutions can be aligned to the time slot τ. Algorithm 1 is the process of calculating the padding size Φ_p and the temporal dimension of output Φ_o for each ST block.

By combining both the strided convolutions and padding, we can obtain the output $\mathcal{X}_C^k$ of the causal 3-D convolution layer.

F. Residual Unit

In this section, we will show how we employ residual learning [31] to effectively simplify the training of our network.

The input of ST-block $\mathcal{X}_F^k$ cannot combine with $\mathcal{X}_C^k$, because they have different dimension of temporal. So, we select the spatial feature maps $\mathcal{X}_{sC}^k$ from $\mathcal{X}_F^k$ at every 2-s intervals from

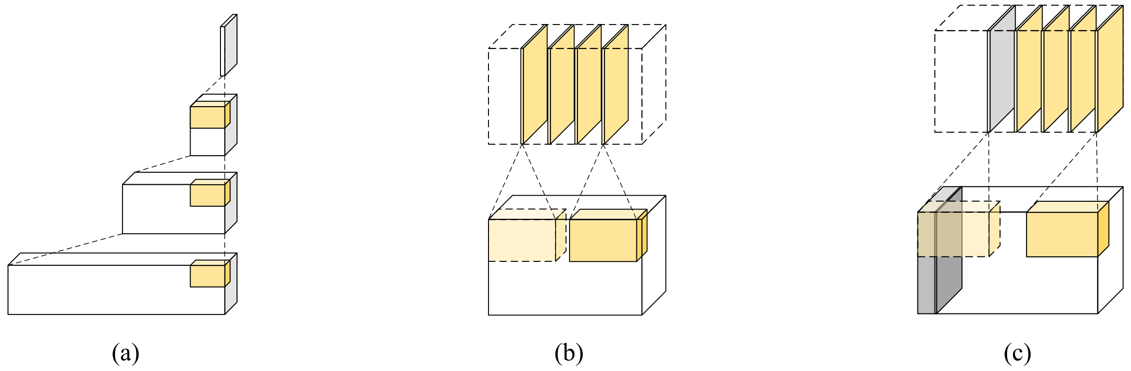


Fig. 5. Causal 3-D convolutions. (a) Stacking the causal 3-D convolutions. (b) Normal 3-D convolutions. (c) Causal 3-D convolutions.

Algorithm 1 Process of the Padding Size Φ_p and the Output Temporal Dimension Φ_o of Each ST Block

Input: The temporal dimension l of the $\mathcal{X}$, temporal kernel size k, stride s.

Output: The padding size Φ_p, the output temporal dimension Φ_o of each ST block.

```
1: while $l > 1$ do
2:     $count = \lceil \frac{l-k}{s} \rceil$
3:     if $count <= 0$ then
4:         $count = 0$
5:     end if
6:     Append $pad = count \times s + k - l$ to $\Phi_p$
7:     $l = \frac{l+pad-k}{s} + 1$
8:     Append $l$ to $\Phi_o$
9: end while
```

right to left. Then, we can produce the final representation of feature maps as follows:

$$\mathcal{X}_F^{k+1} = \texttt{ReLU}\left(\mathcal{X}_F^k + \mathcal{X}_{sC}^k\right) \tag{9}$$

where ReLU is a nonlinear activation function.

G. Upsampling

By stacking a specific number of ST-blocks, we can reduce the dimension of l to 1. To output the target of (4), we need to propagate spatial context information to higher representations, which have the same size with the input area. Thus, we upsample $\mathcal{X}^K \in \mathbb{R}^{[I/2]\times[J/2]\times C^K \times l^K}$ to $\mathbb{R}^{I\times J\times C^K \times l^K}$ by applying deconvolution as shown in (10)

$$\mathcal{X}_U^d = \texttt{Deconv}\left(\mathcal{X}_F^{k+1}\right) \tag{10}$$

$$\mathcal{F}_\tau^r = \texttt{Tanh}\left(\texttt{Conv}(\mathcal{X}_U^d) + \mathcal{F}_\tau^c\right). \tag{11}$$

Then, we use three convolutional layers and a residual connection as shown in Fig. 2, then use the Tanh function to produce $\mathcal{F}_\tau^r$. After that, our final prediction $\hat{\mathcal{F}}_{\tau+1}^c$ is obtained by three convolutional layers.

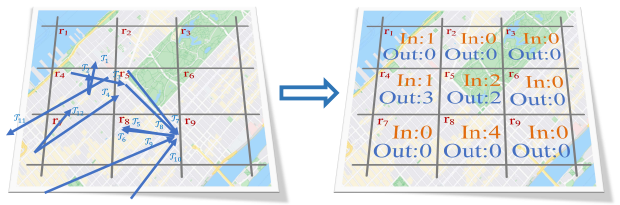


Fig. 6. Samples of BikeNYC.

V. Experiment and Evaluation

In this section, we will evaluate our ASTCN, based on three real-world inflow and outflow data sets. We also perform comprehensive experiments to show the impact of each component.

A. Data Sets

We performed experiments based on three data sets, which are generated from a real-world taxicab GPS data set of Beijing (referred to as TaxiBJ) [15], trajectory data set of the NYC Bike system (referred to as BikeNYC) [15], and the taxi trip records of NYC (referred to as TaxiNYC) [17]. Taxi data sets and Bike data sets are used in our experiments because they represent two kinds of typical flows (traffic flow and crowd flow).

The TaxiBJ data set contains more than 34 000 trajectories of taxicabs, which contain four time periods: 1) 07/01/2013 to 10/30/2013; 2) 03/01/2014 to 06/30/2014; 3) 03/01/2015 to 06/30/2015; and 4) 11/01/2015 to 04/10/2016. The trajectories are assigned to 32×32 grids and the time interval δ is set to 30 min. The external context data contain holidays (41 types), weather conditions (16 types, e.g., Cloudy, HeavyRain), temperature (°C), and wind speed (mph).

The BikeNYC data set contains more than 6800 trajectories from the NYC Bike system, whose period is from 04/01/2014 to 09/30/2014. The trajectories are split into 16×8 grids and the time interval δ is set to 1 h. The external context data include the features of holidays.

The TaxiNYC data set contains 22 349 900 taxi trip records from 01/01/2015 to 03/01/2015 of NYC. The records are assigned to 10×20 grids and the time interval δ is set to 30 min.

Fig. 6 shows that some real trajectories are converted into inflow/outflow for nine regions (r_1–r_9) from the BikeNYC data set. The arrows ($\mathcal{T}_1$–$\mathcal{T}_{12}$) in the figure are the real trajectories pointing from the user's start location to the end location. The inflow and outflow of each cell is the indegree and outdegree of each cell, respectively. For example, in cell r_8, we can see four trajectories arrows ($\mathcal{T}_7$, $\mathcal{T}_8$, $\mathcal{T}_9$, and $\mathcal{T}_{10}$) pointing to it from outside. Therefore, the inflow of cell r_8 is 4, as shown in the right subfigure of Fig. 6. The starting and ending points of arrows $\mathcal{T}_5$ and $\mathcal{T}_6$ are all in cell r_8, so these will not affect the inflow/outflow results of cell r_8. Accordingly, we may easily calculate the flow rates by using (2) and (3), for all the regions.

B. Baselines

To evaluate our model, we will compare ASTCN with the following baselines.

1) History average (HA) makes prediction using the historical inflow and outflow in the same weekday.
2) The least absolute shrinkage and selection operator (LASSO) is a linear regression method, which is used to estimate sparse parameters and can perform variable selection and regularization.
3) Gradient boost decision tree (GBDT) [32] is a widely used machine learning method in data mining, which uses the ensemble of decision trees.
4) Random forest (RF) [33] is an ensemble learning method applying the bagging technique and also a powerful method for classification and regression problems.
5) DeepST [14] is a deep model to predict crowd flows, which can combine ST data and external factors.
6) ST-ResNet [15] is a state-of-the-art model using a residual convolution unit for different temporal properties (closeness, period, and trend).
7) LSTN-PSAM [17] is an RNN-based model, which can learn long-term periodic dependency without setting the strict temporal period.
8) DCRNN [27] is a diffusion convolutional RNN, which is proposed to model spatiotemporal dependencies by the data-driven approach.
9) ST-GDN [28] is a graph neural network proposed to learn global geographical contextual information and encode the complex traffic transition regularities.
10) STG2Seq [18] is a hierarchical graph convolutional model, which is proposed to extract spatial and temporal correlations simultaneously.

C. Preprocessing and Parameters Setting

In our ASTCN, the numerical ratio of training, validation, and testing samples is 8:1:1. We use the Min–Max normalization to scale the continuous features (e.g., the inflow and outflow, temperature and wind speed) and one-hot encoding to transform discrete features (e.g., weather conditions).

We use PyTorch to build our model and train the network from scratch. Adam (Adaptive Moment Estimation) [34] is adopted as our optimizer. Learning rate is set to 6e-4 and batch size is set to 32. The κ_f and κ_s are set to 7 and 2 in our work. Our model takes one week data as input, because one week fits people's routine. Thus, we take history $l = 7 \times 48 = 336$ steps as input for TaxiBJ and TaxiNYC. For BikeNYC, we take history $l = 7 \times 24 = 168$ steps as input. For machine learning models (Lasso, GBDT, and RF), we implement the models with the scikit-learn library [35] and use the default setting. For the opensource models (DeepST, ST-Resnet, LSTN-PSAM, and STG2Seq), we performed related experiments according to the setting mentioned in their papers. For DCRNN and ST-GDN, we set the number of RNN unit to 128, the max diffusion step to 2, the learning rate is set to 0.01, and the batch size is set to 32.

For the performance metric, we use RMSE to evaluate our algorithm, which defined as follows:

$$\texttt{RMSE} = \sqrt{\frac{1}{Z}\sum_{z=1}^{Z}\left(\hat{\mathcal{F}}^{c}_{\tau+1} - \mathcal{F}^{c}_{\tau+1}\right)^2} \tag{12}$$

where Z represents the testing samples.

D. Performance Comparison

In this section, we will compare ASTCN with other baselines, based on the three data sets mentioned above: 1) TaxiBJ; 2) BikeNYC; and 3) TaxiNYC. Table I shows the performance of the proposed model and the other baseline models. We can see that our ASTCN has the best performance among the state-of-the-art models. The details are as follows.

When comparing our method with the traditional methods (HA, Lasso, GBDT, and RF), we can observe that ASTCN outperforms them by considering the more comprehensive ST dependency. In detail, taking the data set of TaxiBJ as an example, the results show that ASTCN is relatively 59.5% better than HA, 38.9% better than Lasso, 32.2% better than GBDT, and 31.3% better than RF.

When compared with the deep learning methods (DeepST, ST-ResNet, and LSTN-PSAM), ASTCN outperforms them. Especially, for the large-scale data set TaxiBJ, the ASTCN obtains highest accuracy on RMSE and has 6.76% improvement than the ST-ResNet. The reason why our model can have better performance than CNN-based models (DeepST and ST-ResNet) is because our model can focus on important time steps to avoid strict temporal periods. In detail, compared to DeepST and ST-ResNet, ASTCN has 10.15% and 6.76% improvement, respectively. The RNN-based method LSTN-PSAM considered the shifted temporal periods, but it splits the correlation between spatial dependency and temporal dependency. On the contrary, ASTCN can take both ST dependency and shifted temporal periods into consideration and improves the RNN-based model by 7.36%.

Compared to the state-of-the-art methods (DCRNN, ST-GDN, and STG2Seq), our model ASTCN also achieves better performance for all the three data sets. Specifically, both DCRNN and ST-GDN apply diffusion convolution to model the spatial dependency. ST-GDN obtains better than DCRNN by applying graph attention to model global spatial dependency. But the performance of DCRNN and ST-GDN varies a lot when they are applied in different data sets. When applying on the BikeNYC data set, ST-GDN and

TABLE I
COMPARISON WITH BASELINE METHODS ON TAXIBJ AND BIKENYC

Model	TaxiBJ	BikeNYC	TaxiNYC
	RMSE	RMSE	RMSE
HA	39.831	8.508	46.111
Lasso	26.411	12.411	17.084
GBDT	23.814	10.985	14.053
RF	23.497	9.859	13.322
DeepST	17.969	6.455	12.742
ST-ResNet	17.318	6.447	11.802
LSTN-PSAM	17.429	6.546	11.453
DCRNN	19.132	5.760	14.186
ST-GDN	18.226	5.711	13.199
STG2Seq	16.781	4.909	12.661
ASTCN	**16.146**	**4.651**	**10.948**

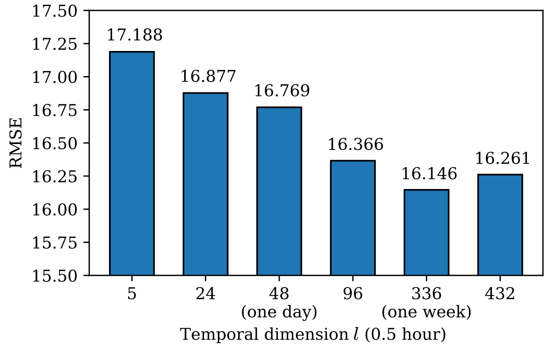


Fig. 7. Impact of input length.

DCRNN have better performance than ST-ResNet while applying on the TaxiBJ data set, they have worse performance than ST-ResNet. Among all the baselines, STG2Seq outperforms the other baselines on TaxiBJ and TaxiNYC since it is based on the graph convolutional structure. However, STG2Seq is obviously worse than LSTN-PSAM on TaxiNYC. STG2Seq still separates ST correlations, making its performance worse than our ASTCN model. Our ASTCN leverages both attention mechanism and casual 3-D convolutions to model the ST correlations. Therefore, our model has at least 3.78% improvement than STG2Seq based on all the data sets.

By comparing with many baselines, the above experiments and results show that ASTCN has the best performance on both data sets.

E. Impact of Input Length

In this section, we will investigate how the length of input will influence the prediction accuracy.

As shown in Fig. 7. The result shows that the performance of ASTCN is improved when it takes longer term data as input. It has the best performance when the temporal dimension l is 336. l is the length of one week data which represent the basic time unit of human routine. Its length is reasonable to contain the pattern of regular human mobility. Compared to the short-term data (l is 5), our model can bring 6% improvement when l is set to 336. It means that our model can work well and achieve better performance in longer term data. However, the performance of our ASTCN degrade when the temporal dimension becomes longer. This may because the most human behavior patterns are hidden in one week data, thus longer data cannot help improve performance and cause the model hard to learn.

TABLE II
MODEL ABLATION OF ASTCN

Model	RMSE
ASTCN	**16.146**
ASTCN-WE	16.465
ASTCN-WA	16.964

TABLE III
NUMBER OF HEADS OF ASTCN

HEAD	1	2	4	8
RMSE	16.475	16.437	**16.146**	16.432

F. Impact of the Attention Layer

To understand the effectiveness of attention in ASTCN, we conducted model ablation to explore the improvement of each component can bring, as shown in Table II. We tested ASTCN without external context data (ASTCN-WE), and ASTCN without attention (ASTCN-WA). It is noted that ASTCN-WA removes the component of attention, external data are also removed because it only feeds into the attention.

We can see that the error of ASTCN-WE is larger than ASTCN. It means that ASTCN can leverage the context data to make better prediction by fusing ST features and context data. Also, we find that the error of ASTCN-WA significantly increases. This is because our attention mechanism can make the model focus on the important part of input by dynamic calculating important values for each time step. In other words, our model can dynamic select the important time steps to avoid strict temporal periods.

In order to better understand the attention mechanism and external data can help predict flow prediction, we show our inflow and outflow prediction error, as shown in Fig. 8. The errors are calculated by $|\hat{\mathcal{F}}^c_{\tau+1} - \mathcal{F}^c_{\tau+1}|$ which represent the absolute errors of each location in an area. When we remove the external data, the errors become larger, e.g., the upper right corner part as shown in the box I of Fig. 8(b) and (e). Moreover, when removing the attention mechanism, the errors become much larger. For example, in Fig. 8(c) and (f), the errors at box II become larger.

Thus, the attention mechanism and the external data can significantly improve the prediction.

G. Impact of the Number of Heads

To enhance the self-attention, we usually apply multihead, which consists of multiple self-attention layers. In this section, we will investigate the number of heads that will affect the performance.

Table III shows the performance of ASTCN by varying the number of heads. We can find that using four heads in ASTCN has the best performance. Compared to the 1-head model, the 4-head model is the best in terms of the performance. The reason is possible that each self-attention layer focuses on different time steps. Thus, the model with more heads can reduce the error. We also find that when the number of head reaches

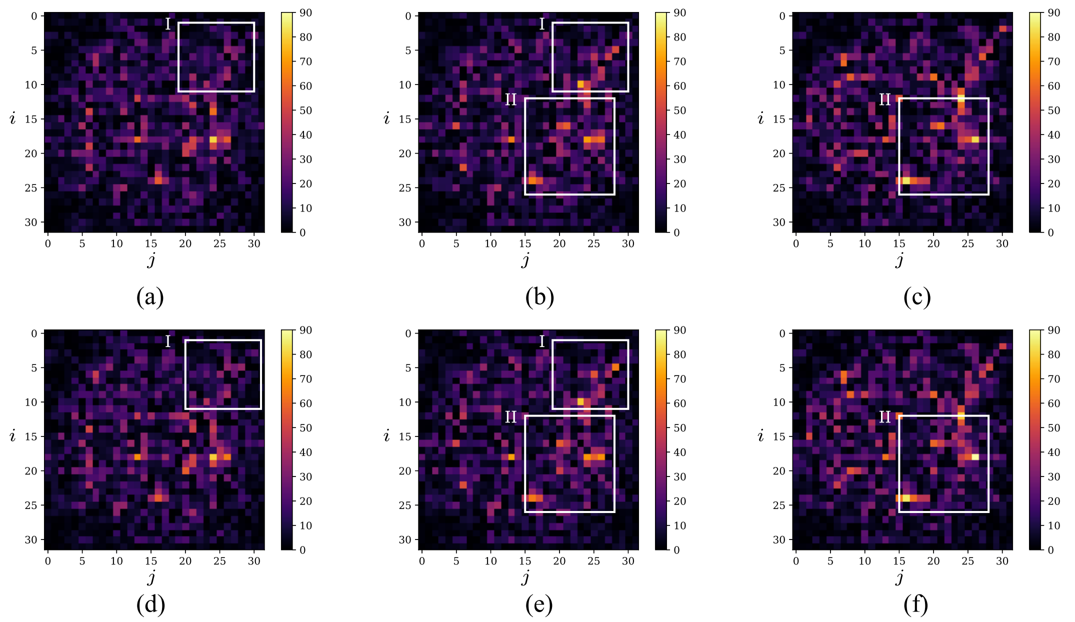


Fig. 8. Error of the inflow and outflow in 2016-02-22 13:30. (a) Inflow error of ASTCN. (b) Inflow error of ASTCN-WE. (c) Inflow error of ASTCN-WA. (d) Outflow error of ASTCN. (e) Outflow error of ASTCN-WE. (f) Outflow error of ASTCN-WA.

TABLE IV
IMPACT OF PADDING AND STRIDE

Model	Without-padding	Stride N_s		
		2	3	4
RMSE	19.843	**16.146**	19.834	18.473

8, the performance decreases. It may be caused by too many heads focusing on different time steps. So, it is apt to cause distraction. Therefore, in our experiment, we use the 4-head model by default.

H. Impact of the Causal 3-D Convolutional Layer

In this section, we will evaluate the stride convolution and padding of the ASTCN. As shown in Table IV, when we enlarge the stride N_s, the performance of ASTCN is significantly reduced. The reason is that the large N_s makes the model quickly reduce the dimension of temporal, and causes the ST features hard to be captured. By invalidating the padding operation, the operation of the convolutional layer work is as shown in Fig. 5(b). The result shows that the performance without padding is dramatically reduced by 22.9% because the time slots near τ are overlooked.

The experiments of the causal 3-D convolutional layer show that both the techniques of padding and stride convolution are important designs, which can help improve the performance of ASTCN.

I. Qualitative Analysis

To visualize how the self-attention allocates the important values, we show one of the attention values in Fig. 9. The attention values are chosen from the first attention layer. Because this layer contains all the time steps, we can obtain which time steps our model pays attention.

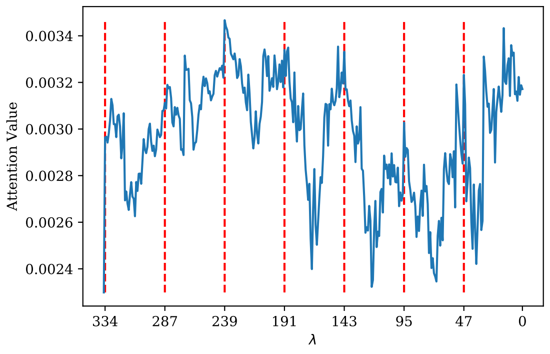


Fig. 9. Important values of the first attention layer.

We use λ to represent the number of time steps (here, each step is 0.5 h) before time τ. Red dashed lines are used to represent the same time slot of the day before $\tau + 1$. For example, the line at $\lambda = 47$ represents the same time slot before one day and $\lambda = 95$ represents the same time slot before two days. The attention value grows larger if our model pays attention to the time step. The figure shows that our model can focus more on the time steps near the same time slot of the days. This can help ASTCN dynamically set higher weight to those important time steps which will contribute to the final prediction.

VI. CONCLUSION

In this article, we proposed a novel deep learning model ASTCN for flow prediction. We designed a causal 3-D convolutional layer using strided convolutions and padding, which is able to efficiently extract the features from long-term ST data. Moreover, we designed an attention mechanism making ASTCN focus on important part by assigning higher weight. It also leverages multihead attention, which can fuse ST data and external context data for each time step. We evaluated our model based on three real-word data sets, and the experimental results showed that our model outperforms the state-of-art methods by at least 3.78%.

References

[1] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, and Z. Han, "Capsule network assisted IoT traffic classification mechanism for smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7515–7525, Oct. 2019.

[2] J. Li *et al.*, "An end-to-end load balancer based on deep learning for vehicular network traffic control," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 953–966, Feb. 2019.

[3] *Baidu Migration*. Accessed: Sep. 3, 2020. [Online]. Available: http://qianxi.baidu.com/

[4] *Tencent Migration*. Accessed: Sep. 3, 2020. [Online]. Available: https://heat.qq.com/qianxi.php

[5] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, pp. 1–55, 2014.

[6] X. Li *et al.*, "Prediction of urban human mobility using large-scale taxi traces and its applications," *Front. Comput. Sci.*, vol. 6, no. 1, pp. 111–121, 2012.

[7] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1393–1402, Sep. 2013.

[8] M. X. Hoang, Y. Zheng, and A. K. Singh, "FCCF: Forecasting citywide crowd flows based on big data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2016, pp. 1–10.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] X. Sun, G. Gui, Y. Li, R. P. Liu, and Y. An, "ResInNet: A novel deep neural network with feature reuse for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 679–691, 2018.

[11] F. Zhou, Q. Yang, K. Zhang, G. Trajcevski, T. Zhong, and A. Khokhar, "Reinforced spatio-temporal attentive graph neural networks for traffic forecasting," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6414–6428, Jul. 2020.

[12] R. Jia, P. Jiang, L. Liu, L. Cui, and Y. Shi, "Data driven congestion trends prediction of urban transportation," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 581–591, Apr. 2018.

[13] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An intelligent traffic load prediction-based adaptive channel assignment algorithm in SDN-IoT: A deep learning approach," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5141–5154, Dec. 2018.

[14] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2016, pp. 1–4.

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

[16] B. Shen, X. Liang, Y. Ouyang, M. Liu, W. Zheng, and K. M. Carley, "Stepdeep: A novel spatial-temporal mobility event prediction framework based on deep neural network," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2018, pp. 724–733.

[17] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 5668–5675.

[18] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, "STG2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 1981–1987.

[19] L. Kuang, X. Yan, X. Tan, S. Li, and X. Yang, "Predicting taxi demand based on 3D convolutional neural network and multi-task learning," *Remote Sens.*, vol. 11, no. 11, p. 1265, 2019.

[20] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, pp. 914–921, 2020.

[21] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018. [Online]. Available: arXiv:1803.01271.

[22] J. Hou, G. Wang, X. Chen, J.-H. Xue, R. Zhu, and H. Yang, "Spatial-temporal attention res-tcn for skeleton-based dynamic hand gesture recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 273–286.

[23] 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.*, 2017, pp. 156–165.

[24] G. Singh and F. Cuzzolin, "Recurrent convolutions for causal 3D CNNs," in *Proc. IEEE Int. Conf. Comput. Vision Workshops*, 2019, pp. 1456–1465.

[25] 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. Adv. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

[26] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2588–2595.

[27] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–16.

[28] X. Zhang *et al.*, "Traffic flow forecasting with spatial-temporal graph diffusion network," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 15008–15015.

[29] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[30] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[32] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[33] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: arXiv:1412.6980.

[35] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

Haizhou Guo received the Master degree from the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China, in 2021.

His research interests include machine learning, neural networks, and spatiotemporal data mining.

Dian Zhang (Member, IEEE) received the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2010.

She worked as a Research Assistant Professor with the Fok Ying Tung Graduate School, HKUST. She is currently an Associate Professor with Shenzhen University, Shenzhen, China. Her research interests include big data analytics and mobile computing.

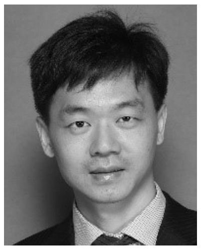
Dr. Zhang is a CCF member and an AAAI life member.

Landu Jiang received the B.Eng. degree in information security engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010, and the master's degree in computer science and minor in construction management from the University of Nebraska–Lincoln, Lincoln, NE, USA, in 2012, and the Ph.D. degree in computer science from McGill University, Montreal, QC, Canada, in 2018.

His research interests include ubiquitous computing, machine learning applications, mobile computing, computer vision, cyber-physical systems, and green energy solutions.

Kin-Wang Poon received the Bachelor degree from Shenzhen University, Shenzhen, China, in 2015.

He is a Research Assistant with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include deep learning and computer vision.



Kezhong Lu received the bachelor's and Ph.D. degrees in computer science from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He is a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China, where he is the Vice Dean of the College of Computer Science and Software Engineering. His research concerns big data, parallel and distributed computing, and wireless sensor network.