Applied Intelligence (2023) 53:8994–9008
https://doi.org/10.1007/s10489-022-03980-9

Multi-perspective convolutional neural networks for citywide crowd flow prediction

Genan Dai[1] · Weiyang Kong[1] · Yubao Liu[1,2] · Youming Ge[1] · Sen Zhang[1]

Accepted: 7 July 2022 / Published online: 5 August 2022

Abstract
Crowd flow prediction is an important problem of urban computing with many applications, such as public security. Inspired by the success of deep learning, various deep learning models have been proposed to solve this problem. Although existing methods have achieved good prediction performance, they cannot effectively capture richer spatial-temporal correlations that are important for crowd flow prediction. To address the limitation of existing methods, we propose a novel 2D CNN-based (convolutional neural networks) model via multiple perspectives called the MPCNN to capture richer spatial-temporal correlations. In particular, three perspective CNNs are included in the MPCNN: the front CNN, the side CNN and the top CNN. Then, we propose a fusion layer to combine the results of the three CNNs. In addition, in the MPCNN, we use external factors to enhance prediction performance. Based on four real-world datasets, we performed a series of experiments to compare the proposed method with existing methods, and experimental results demonstrate the effectiveness and efficiency of the proposed method.

Keywords Crowd flow prediction · Multi-perspective · Spatial-temporal data · Deep learning

1 Introduction

Crowd flow prediction is an important problem of urban computing with many applications, such as public security and urban management [8, 27, 36, 37]. For example, a stampede disaster occurred in Shanghai on New Year's Eve in 2015, when a large crowd of people gathered for the celebration of the New Year. Thirty-six people were killed, and forty-nine people were injured, making this incident one of the largest disasters in China in recent years. If crowd flows can be predicted, early warnings and advance measurements can be taken, and accidents can be prevented. Thus, public security requires a fast response because time is critical to effective application of preventive measures in public security.

Crowd flow prediction is challenging because it is affected by many complex factors, such as the spatial-temporal correlations among different regions of a city and external factors, including weather conditions and holidays.

Inspired by the success of deep learning, various deep learning models have been proposed for the problem. Existing methods can be grouped into two categories: graph-based methods and CNN-based (convolutional neural networks) methods. Graph-based methods [6, 12, 20] model a city map as a graph and apply graph neural networks (GNN) from a spatial perspective and recurrent neural networks (RNN) from a temporal perspective to capture spatial-temporal correlations. However, GNN and RNN capture spatial and temporal correlations separately and neglect the correlations between different types of flows. In addition, these methods are often time-consuming to perform. Thus, in this study, we focus on CNN-based

✉ Genan Dai
daign@mail2.sysu.edu.cn

✉ Yubao Liu
liuyubao@mail.sysu.edu.cn

Weiyang Kong
kongwy3@mail2.sysu.edu.cn

Youming Ge
geym@mail2.sysu.edu.cn

Sen Zhang
zhangs7@mail2.sysu.edu.cn

1 School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, 510006, China

2 Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, 510006, China

methods to simultaneously capture the spatial-temporal correlations of crowd flows with balancing effectiveness and efficiency.

There are many CNN-based methods that achieve good prediction performance, such as the 2D CNN-based methods ST-ResNet [37], DeepLGR [14], and the 3D CNN-based methods MST3D [2]. However, these methods cannot efficiently capture the richer spatial-temporal correlations, which are important for crowd flows prediction. Below, we use an example to show the limitations of the existing methods.

We now consider a map that is divided into 3×3 regions of equal size. As shown in Fig. 1, objects are located in regions. For example, there is a stadium (**S**) in region r_1 and a taxi stand (**T**) in region r_3. Each region is associated with two types of crowd flows, new-flow and end-flow [7]. Specifically, the new-flow is the crowd flows starting from a given region (e.g., people start driving from a parking spot). The end-flow is the crowd flow that arrives at a given region (e.g., people stop driving and park their cars). Each curve between two objects denotes a spatial trajectory (i.e., the movement of an individual). For example, the curve between **S** in region r_1 and **T** in region r_3 shows the movement from the stadium to the taxi stand.

We assume that there is a football game held in the stadium in the region r_1. When the football game ends at 9:00 P.M. (t_1), people rush out of the stadium, and then, the new-flow of r_1 grows. By 9:30 P.M. (t_2), many people have reached nearby regions r_2 and r_3. Thus, the end-flows of r_2 and r_3 become larger at t_2, and the new-flow of r_1 at t_1 has a strong influence on the end-flows of its nearby regions r_2 and r_3 at t_2. Effectively capturing such spatial-temporal correlations among different regions is important when predicting crowd flows.

However, existing methods cannot effectively capture such spatial-temporal correlations. Specifically, the existing

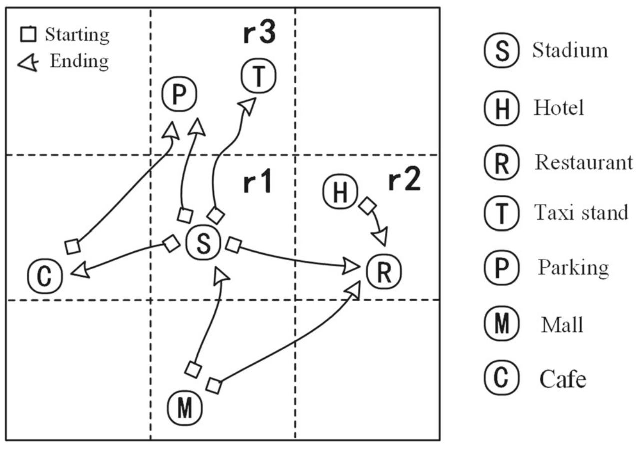

Fig. 1 Example of complex spatial-temporal correlation of crowd flows among different regions

3D CNN-based methods can only capture the spatial-temporal correlations among regions with the same type of flows. For example, using existing methods, we know that the new-flow of r_1 at t_1 will increase the new-flow of r_2 at t_2, but we cannot know that the new-flow of r_1 at t_1 will also increase the end-flow of r_2 at t_2. Existing 2D CNN methods lack an explicit way to capture the correlations with different types of flows. More specifically, they make a weighted summation of all time channels, which cannot effectively capture complex correlations.

To address the limitations of the existing methods, we propose a novel 2D CNN-based model via multiple perspectives called MPCNN to explicitly capture the richer spatial-temporal correlations among different regions (e.g., the correlations with different types of flows). In particular, Fig. 2 shows an example of three perspectives for the proposed model. In this figure, crowd flow data are modeled as a cube, including 3×3 regions and their new-flow at t_1 and t_2 and their end-flow at t_2. Each grid in the cube represents the crowd flow of a region, and each slice represents the crowd flow of the entire region at a time step. As shown in Fig. 2, we consider three perspectives, the front perspective, the side perspective, and the top perspective. The front perspective corresponds to the front view of the cube slices, the three slices with the new-flow at t_1 and the end-flow and new-flow at t_2. Similarly, as shown in Fig. 2, the side perspective corresponds to the side view of the cube slices, and the top perspective corresponds to the top view of the cube slices. The front view of cube slices contains the spatial information of different regions, and the side and top views of the cube slices contain the spatial-temporal information of the new-flow and end-flow. The front view of cube slices is also identical to the back view of cube slices; the left side view is identical to the right side view; and the top view is identical to the bottom view. Thus, we propose three perspective CNNs: the front CNN, the side CNN, and the top CNN.

In the front CNN, we use a 2D CNN to capture the spatial-temporal correlations on the front view of the cube slices. The correlations among the new flows of r_1, r_2, and r_3 can be captured on the slice with t_1, as shown in Fig. 2. Similarly, in the side CNN, we can capture the spatial-temporal correlations with different types of flows (e.g., the new-flow of r_1 and the end-flow of r_3). In the top CNN, we can capture the spatial-temporal correlations among the new-flow of r_1 and the end-flow of r_2 and the new-flow of r_2. In the following experiments, we present a real case study to show that the MPCNN can effectively capture correlations among the regions with different types of flows for crowd flow prediction.

In general, the contributions of this study can be summarized as follows:

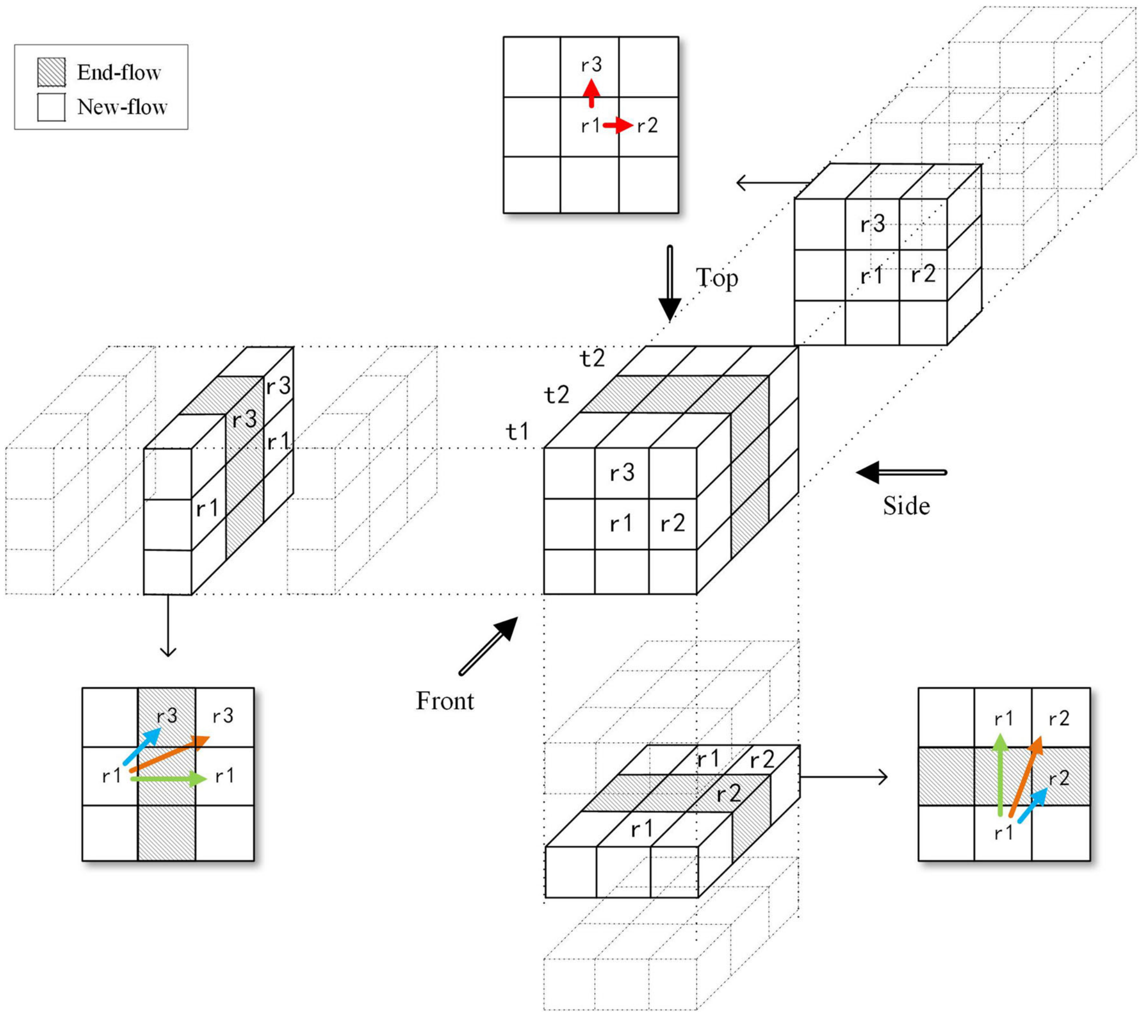

Fig. 2 Example of three perspectives for the proposed model

- Considering existing methods' inability to fully exploit the spatial-temporal correlations in crowd flow data, we propose a novel 2D CNN-based model called MPCNN for crowd flow prediction. In MPCNN, we propose the front CNN, the side CNN, and the top CNN to capture the richer spatial-temporal correlations hidden in the crowd flows. Then, we design a fusion layer to combine the results of the three CNNs. In addition, we use external factors such as metadata and weather condition to enhance prediction performance.
- We perform extensive experimental evaluations on four real-world datasets. Experimental results have demonstrated that the proposed MPCNN can achieve the best prediction performance and is the most efficient among all methods with high-quality performance.

2 Related work

Crowd flow prediction Crowd flow prediction has been extensively studied in recent decades. In general, there are two types of methods: micro-level and macro-level. The former focuses on the prediction of each individual movement based on their history trajectories [30], and traffic conditions on roadsegments [24]. Auto-regressive integrated moving average (ARIMA) is a well-known method that is commonly used to predict future values in time series. Kumar proposed a method [10] based on the Kalman filtering technique to predict future traffic conditions. However, these methods are not suitable for public security, which requires a citywide perspective. The latter [7, 17, 36] focuses on the prediction of citywide crowd flows. Thus, in this study, we investigate the macro-level view of crowd flow prediction by predicting two types of crowd flows in every region of a city: the new-flow and the-end flow [7]. These two flows summarize the movements of a crowd and are sufficient for traffic management and social safety precaution.

Recently, some researchers have used deep learning methods [1, 3, 5, 13, 15–19, 23, 25, 26, 28, 35, 38, 39, 41] to solve this problem. Deep learning methods have a strong ability of model expression due to nonlinear activation. Recurrent neural networks (RNNs) and their variants, including long short-term memory (LSTM) [31–33] have been proposed to capture the temporal features of flows. Poon et al. [18] proposed a time-series method with LSTM for crowd prediction with a long time gap. Singh [19] proposed an LSTM-based forecasting model on WiFi data for crowd forecasting. However, RNN-based methods have certain disadvantages, such as time-consuming iterations and complex gate mechanisms [34]. Compared with RNN-based methods, CNNs [11] are characterized by fast training times, simple structures, and excellent performance when extracting spatial features. Recently, graph neural networks [4, 6, 9, 12, 20, 28] have been proposed and have achieved good performance. Zhou et al. [40] proposed a multi-graph convolution operator to capture multiple spatial dependencies of urban crowd flow and used RNN to model the temporal dependency. MVGCN [22] was proposed to forecast crowd flows in irregular regions. MVGCN is a variant of GNN and a fully connected neural network that can effectively capture spatial correlations and global information. However, these methods are designed for graph-based data and ignore correlations between regions with different types of flows.

CNN-based methods In general, there are two types of CNN-based methods: 2D CNN-based methods and 3D CNN-based methods. For example, Zhang et al. [36] presented a method based on CNNs for crowd flow prediction that uses several CNN layers to capture distant spatial correlations. The performance of the method is limited by the number of CNN layers. However, too many layers will cause the gradient to disappear. To overcome this limitation, Zhang et al. [37] introduced residual learning into the model. To solve the inefficiency in capturing the long-distant spatial correlation of CNN methods, Liang et al. [14] combines CNNs and spatial pyramid pooling. Yao et al. [32] proposed a local CNN model that captures local characteristics of regions in relation to their neighbors. MST3D [2] uses 3D CNN to learn spatial-temporal correlations for crowd flows prediction. Different from the above CNN-based methods, the MPCNN captures the richer spatial-temporal correlations via multiple perspective CNNs. In addition, multi-view CNNs [21, 29] are proposed for 3D shape recognition. In essence, they still use the additional 2D CNN, which is different from the proposed model.

3 Preliminaries

In this section, we describe related concepts and provide the problem definition investigated in this study, which are similar to [7].

Definition 1 (Region) Given a map of a city, we can divide the map into $N \times M$ equal-size disjoint grid cells, where N and M are given positive integers. Each cell $r(i, j)$ denotes a region, where $1 \leq i \leq N$ and $1 \leq j \leq M$.

Definition 2 (New-flow and end-flow) New-flow and end-flow are two types of crowd flows. The movement of an individual can be recorded as a spatial trajectory tr, which is a sequence of time-ordered points, $tr : p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_{|tr|}$, where each point $p_i = (a_i, b_i, t_i)$ has a geospatial coordinate position (a_i, b_i) and a timestamp t_i, and $|tr|$ is the number of points in tr. Given a set of trajectories $\mathcal{P}$ at the t-th time interval, we split time into n nonoverlapping time intervals with the same length (e.g., an hour). For a region r, the new-flow and end-flow at the t-th time interval are defined as follows:

$$x_t^{new,r} = |tr \in \mathcal{P} : (a_1, b_1) \in r, t_1 = t| \tag{1}$$

$$x_t^{end,r} = |tr \in \mathcal{P} : (a_{|tr|}, b_{|tr|}) \in r, t_{|tr|} = t| \tag{2}$$

where $(a_1, b_1) \in r$ means that point lies within region r.

For example, as shown in Fig. 1, the map is divided into 3×3 regions. Because there are four trajectories starting from the stadium (**S**) in r_1, namely, **S**→**T**, **S**→**P**, **S**→**C** and **S**→**R**, the new-flow of r_1 is equal to 4. Similarly, the end-flow of r_1 is equal to 1 because there is one trajectory ending at r_1, that is, **M**→**S**. In particular, the new-flow and end-flow of r_2 are equal to 1 and 3, respectively. The trajectory **H**→**R** in r_2 is counted as the new-flow and end-flow of r_2. In general, the end-flow and new-flow in all $N \times M$ regions at the t-th time interval can be denoted as a cube $X_t \in \mathbb{R}^{2 \times N \times M}$.

Definition 3 (Problem definition) Given the historical observations $\{X_t | t = 0, \ldots, n\}$, we try to predict $\{X_t | t = n + 1, \ldots, n + k\}$ for the next k time intervals.

4 Method

In this section, we first provide an overview of the proposed framework and then present the key components. Then,

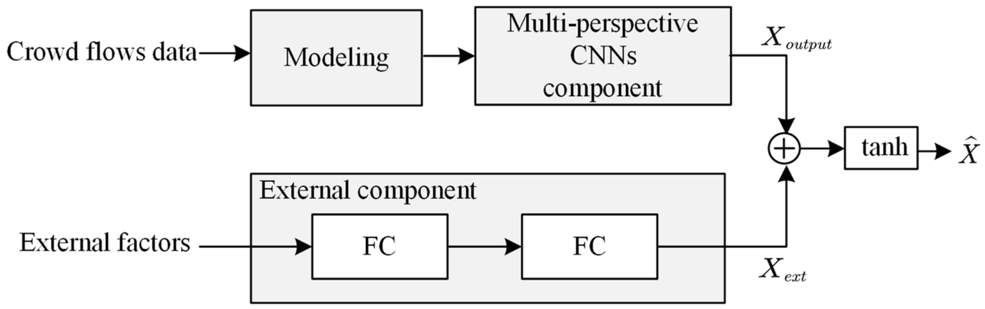

Fig. 3 Overview of the proposed MPCNN model includes three major components. FC denotes the fully connected layer

we compare the proposed method with existing methods and describe the results of the complexity analysis of the proposed model.

4.1 Overview

The overview of the proposed MPCNN model is shown in Fig. 3, which consists of three major components, the component of modeling, the component of multi-perspective CNNs, and the external component. Historical crowd flow data are first fed into the modeling component. In this component, we try to capture the temporal properties of the crowd flows by three temporal cubes (i.e., *closeness* cube, *period* cube and *trend* cube). These flows can be extracted from the historical flow data according to different time granularity requirements (e.g., a day or a week). Then, the three temporal cubes are combined into one cube and are fed into the component of multi-perspective CNNs. This component contains three perspective CNNs: the front CNN, the top CNN, and the side CNN. Using these perspective CNNs, we can explicitly capture spatial-temporal correlations with different types of flows, as shown in Fig. 2. In addition, we propose a fusion layer to combine the results of the three perspective CNNs. Then, we use an external component to combine the external factors to enhance the prediction performance.

4.2 Modeling

As shown in Section 3, we can compute the new-flow and end-flow in all $N \times M$ regions at the t-th time interval (i.e., the cube X_t with the size of $2 \times N \times M$). Similar to [37], we try to capture the temporal properties (i.e.,*closeness*, *period*, *trend*) hidden in the crowd flows by the temporal cubes with three time granularities (i.e., recent, near, distant), denoted as I_c, I_p and I_q, respectively.

We extract some cubes that can represent the three temporal properties. Specifically, crowd flows at adjacent time intervals can be input to simulate *closeness*; flows at the same time intervals every day can be input to simulate *period*; and flows concurrently intervals every week can be input to simulate *trend*. The *closeness* cube $I_c = [X_{t-l_c}, X_{t-(l_c-1)}, ..., X_{t-1}]$ is extracted as the *closeness* sequence, where l_c is the length of the *closeness* sequence. The *period* cube $I_p = [X_{t-l_p \dots p}, X_{t-(l_p-1) dotsp}, \dots, X_{t-p}]$ is extracted as the *period* sequence, where l_p is the length of the *period* sequence and p is the period span. The *trend* cube $I_q = [X_{t-l_q \dots q}, X_{t-(l_q-1) \dots q}, \dots, X_{t-q}]$ is extracted as the *trend* sequence, where l_q is the length of the *trend* sequence and q is the trend span. The period span p and trend span q can be customized as required. In the proposed method, we empirically set p as one day and q as one week. Then, we combine I_c, I_p and I_q as the input cube

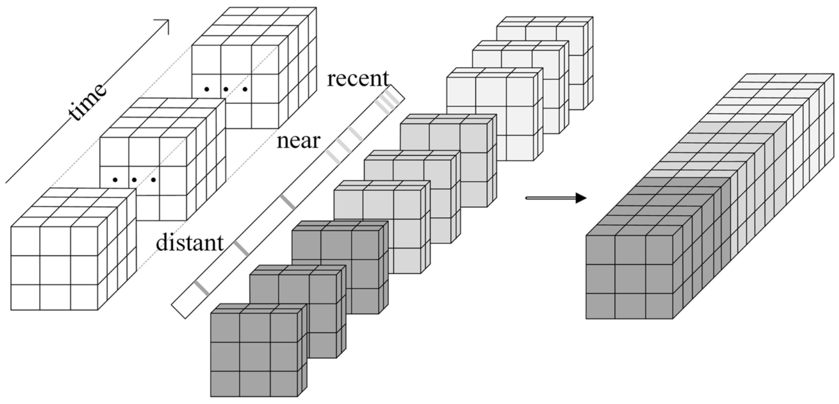

Fig. 4 Modeling process

$I_{input} \in \mathbb{R}^{(2l_c+2l_p+2l_q)\times N\times M}$. For simplicity, we set $(2l_c + 2l_p + 2l_q)$ as L. Figure 4 represents the modeling process. Then, the input cube $I_{input} \in \mathbb{R}^{L\times N\times M}$ is fed into the multi-perspective CNN component.

4.3 Multi-perspective CNN component

Inspired by the advantage of the 2D CNN in capturing the correlations in a plain, we design a multi-perspective CNN component to explicitly capture the rich spatial-temporal correlations. As shown in Fig. 2, from the front perspective, the new-flows of r_1, r_2 and r_3 are on one plain. Thus, the correlations among the new-flows of r_1, r_2 and r_3 (i.e., the red lines) can be captured by the CNN from the front perspective. From the side perspective, the new-flow of r_1 at t_1 and t_2 and the end-flow and new-flow of r_3 at t_2 are on the same plain. Thus, the correlations between r_1 at different times (i.e., the green line) and the correlations between r_1 and r_3 can be explicitly captured by the CNN from the side perspective. Specifically, the correlations between r_1 and r_3 include the correlations of the new-flows of r_1 at t_1 and the end-flows of r_3 at t_2 (i.e., the blue line), the correlations of the new-flows of r_1 at t_1 and the new-flows of r_3 at t_2 (i.e., the yellow line). Similarly, from the top perspective, the new-flow of r_1 at t_1 and t_2 and the end-flow and new-flow of r_2 at t_2 are on the same plain. Thus, the complex spatial-temporal correlations between r_1 and r_2 can be explicitly captured by the CNN from the top perspective. Note that, from the side and the top perspective, because the two flows are stacked together, the new-flow and end-flow are associated with the same plain. Thus, the new-flow and end-flow are computed simultaneously, and the correlations between them can be captured by the CNNs.

The architecture of the multi-perspective CNN component is given in Fig. 5. The component consists of three perspective CNNs, i.e., the front CNN, the side CNN and the top CNN, a fusion layer, and a convolution layer. First, for the input cube I_{input}, we first obtain its three perspectives: the front perspective of slices $X^{(0)}_{front} \in \mathbb{R}^{L\times N\times M}$, the side perspective of slices $X^{(0)}_{side} \in \mathbb{R}^{M\times N\times L}$, and the top perspective of slices $X^{(0)}_{top} \in \mathbb{R}^{N\times L\times M}$. Then, we feed the perspectives into the front CNN, the side CNN, and the top CNN.

The three perspective CNNs share a similar network structure, which consists of a convolution *Conv 1* and several residual units [37]. Considering the front CNN as an example, the transformation at *Conv 1* is defined as follows:

$$X^{(1)}_{front} = f(W^{(1)}_{front} * X^{(0)}_{front} + b^{(1)}_{front}) \tag{3}$$

where $*$ denotes the convolution operation, and f is the ReLU activation function. $W^{(1)}_{front}$ and $b^{(1)}_{front}$ are all learnable parameters in the first convolutional layer.

The crowd flows of a region are affected by nearby regions and by long-distance regions, such as the regions connected by subways or highways. Thus, we use multi-layer CNNs to capture the correlations of the long-distance regions. However, too many convolutional layers make model training difficult. Therefore, we introduce residual units [37] into the proposed method. The residual unit consists of two combinations of "ReLU and convolution", as shown in Fig. 5. The transformation at the i-th residual unit is defined as follows:

$$X^{(i+1)}_{front} = X^{(i)}_{front} + \mathcal{F}(\omega^{(i)}_{front}, X^{(i)}_{front}), i = 1, 2, \ldots, K \tag{4}$$

where the function $\mathcal{F}$ represents the residual function and $\omega^{(i)}_{front}$ represents learnable parameters in the i-th residual unit of the front CNN. In addition, the proposed method uses zero-padding in the convolutional layers of the residual unit, and the number of convolution kernels used in the residual unit is equal to the depth of its input. For example, in the

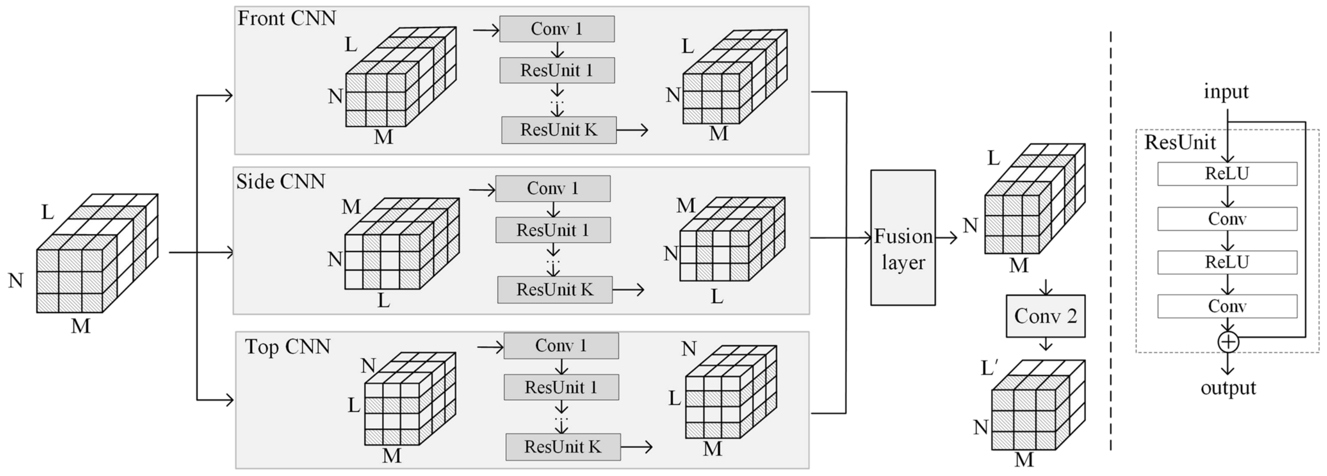

Fig. 5 The left figure shows the architecture of the multi-perspective CNN component, and the right figure shows the details of the residual unit in each perspective CNN

front CNN, the number of convolution kernels used in the residual unit is L, and in the side and top CNNs, they are M and N, respectively. Thus, the output and input of each perspective CNN have the same size (i.e., $L \times N \times M$).

Next, we introduce the fusion layer that can fuse the results of three perspective CNNs. In particular, the influence on crowd flows among regions changes over time. For example, we consider the example of Fig. 1 again. When the football game ends at 9:00 P.M., the crowd flows in region r_1 will become large soon. Then, the crowd flows in the nearby regions r_2 and r_3 are strongly influenced by r_1. However, after a long time, the crow flows in r_1 may decrease, and their influence will also gradually decrease. To learn such a correlation between different regions, we propose a weight-matrix-based fusion as follows:

$$X_{fusion} = W_f \circ X_{front} + W_s \circ X_{side} + W_t \circ X_{top} \tag{5}$$

where $\circ$ denotes elementwise multiplication, W_f, W_s, and W_t are learnable parameters, and X_{front}, X_{side}, and X_{top} are outputs of the three perspective CNNs.

Next, we apply a convolutional layer (i.e., *Conv 2* shown in Fig. 5) to the fusion cube X_{fusion}. In particular, we use two convolutional kernels in the *Conv 2* layer; thus, the final output of the multi-perspective CNN component is $X_{output} \in \mathbb{R}^{2 \times N \times M}$.

4.4 External component

Crowd flows can be affected by many external factors, such as metadata (e.g., weekday/weekend), holiday information and meteorological features. Specifically, the crowd flows during weekdays can be different from those on weekends. The flows during the holidays are different from those during normal days. Compared to the same day in the recent week, heavy rains often cause a sharp decrease in crowd flows. Therefore, we consider metadata (e.g., weekday/weekend), holiday information and meteorological features (i.e., weather condition, temperature and wind speed) as the external factors.

Similar to the method of [37], we turn the metadata, holiday information and weather conditions into vectors by one-hot coding. In addition, we normalized the values of wind speed and temperature into [0,1] by min-max normalization. Then, the external features are fed into two fully connected layers. The first layer is regarded as an embedding layer, and the second layer maps low to high dimensions that have the same shape as X_{output}. The output of the external component is X_{ext}.

4.5 Prediction and training

To obtain the predicted value, we merge the output of the external component X_{ext} with the output X_{output} of the multi-perspective CNN component. Then, the aggregation is mapped into [-1,1] by the *tanh* function as the predicted value.

To learn the model parameters, we optimize the mean-square error (MSE) as follows:

$$L(\theta) = \| X_t - \widehat{X}_t \|_2^2 \tag{6}$$

where θ means all learnable parameters in the method, and X_t and $\widehat{X}_t$ are the ground truth and predicted value, respectively. Algorithm 1 describes the training scheme of the MPCNN, including the construction of training instance sets (lines 2-8) and backpropagation training (lines 10-14).

Algorithm 1 Training scheme of MPCNN.

```
Input: history crowd flows: ${X_0, X_1, ..., X_{n-1}}$, external features: ${E_0, ..., E_{n-1}}$, lengths of closeness, period and trend:${l_c, l_p, l_q}$, closeness span: $c$, period span: $p$, trend span: $q$
Output: the learned model
1: initialization: training set $\Phi \leftarrow \emptyset$
2: for all available time interval $t \in [1, n-1]$ do
3:     $I_c = [X_{t-l_c}, X_{t-(l_c-1)}, ..., X_{t-1}]$;
4:     $I_p = [X_{t-l_p \cdot p}, X_{t-(l_p-1)\cdot p}, ..., X_{t-p}]$
5:     $I_q = [X_{t-l_q \cdot q}, X_{t-(l_q-1)\cdot q}, ..., X_{t-q}]$
6:     combine $I_c$, $I_p$, and $I_q$ as $I_{input}$
7:     add a training instance (${I_{input}, E_t}, X_t$) to train set $\Phi$; // $X_t$ is the target at time interval $t$
8: end for
9: initialize all learnable parameters $\theta$ in the proposed model
10: repeat
11:     randomly select a batch of training instances $\Phi_b$ from $\Phi$
12:     put $\Phi_b$ into the multi-perspective CNN component and external component
13:     optimize $\theta$ by minimizing the loss function (6)
14: until stopping criteria is met
15: output the learned model
```

4.6 Discussion and analysis

First, we compare the proposed model with the existing methods. Then, we present the complexity analysis of the proposed model.

4.6.1 Comparison with existing 2D CNN-based methods

The challenge of crowd flow prediction is how to capture the complex spatial-temporal correlations between crowd flows, i.e., the correlations between (x_1, y_1, t_1) and (x_2, y_2, t_2), where (x_i, y_i, t_i) denotes the crowd flows of region $r(x_i, y_i)$ at time t_i. In fact, existing 2D CNN-based methods, such as [14, 37], are essentially equivalent to the proposed front CNN, which often captures the spatial correlations, i.e., the correlation between $r(x_1, y_1)$ and $r(x_2, y_2)$. As shown in Fig. 2, the front CNN works on slices with the same temporal information (e.g., t_1 or t_2). Thus, the front CNN often captures the correlations between two regions with the same temporal information. In MPCNN, we add two additional perspectives. Because the cubes with the side and top perspectives contain spatial and temporal information about the new and end flows, the spatial and temporal information can be learned simultaneously by the CNNs. Specifically, the side CNN can explicitly capture the correlation between (y_1, t_1) and (y_2, t_2), and the top CNN can explicitly capture the correlation between (x_1, t_1) and (x_2, t_2). Then, the fusion layer can capture the spatial-temporal correlations between (x_1, y_1, t_1) and (x_2, y_2, t_2) by fusing the results of three perspective CNNs.

The side and top CNN capture the correlations between regions of the same latitude and longitude at different times. The front CNN in MPCNN can adequately capture the spatial relationship between different regions to complement the side and top CNNs. The fusion layer can regulate the influence of different spatial-temporal correlations in different regions. Thus, compared with existing 2D CNN-based methods, the proposed method can capture more complex spatial-temporal correlations.

4.6.2 Comparison with existing 3D CNN-based methods

In 3D CNN, the convolution operation is performed in the spatial dimension and temporal dimension. However, the different types of channels are separately handled in 3D CNN. Thus, the correlations between the different channels cannot be handled. In other words, the correlations among the regions with different types of flows cannot be captured by 3D CNN.

4.6.3 Comparison with existing multi-view methods

There are some multi-view CNN-based methods, such as [21, 29], that have been proposed for 3D shape recognition. In general, these methods use multiple data sources, such as multiple photos taken from cameras at multiple angles. Thus, multi-view methods often use multiple data inputs. However, these methods still use traditional 2D CNN convolution with only one single perspective, while multi-perspective CNNs are used on the crowd flow data in the proposed model.

4.6.4 Complexity analysis

We now describe the complexity analysis of the proposed model from the following two perspectives.

- Model size: In the MPCNN, there are three major components that require the parameters. In the multi-perspective CNN component, each perspective CNN contains qhg parameters, where q is the number of CNN layers, h is the number of convolution kernels used in each CNN layer and g is the number of parameters in one convolution kernel. In the MPCNN, h is set to L, M and N in the front CNN, the side CNN and the top CNN, respectively. Then, the fusion layer contains $3LNM$ parameters, and the external factor component contains $f_1 f_2 NM$ parameters, where f_1 and f_2 denote the output dimensions in two fully connected layers. Thus, the model size of the MPCNN is equal to $qg(L + N + M) + 3LNM + f_1 f_2 NM$.
- Time complexity: The primary time cost of the model contains the time for both the multiple perspective CNN component and the fusion layer. The multi-perspective CNN component contains the stacking of multiple CNN layers. Thus, the time cost of the component is $O(qc)$, where q is the number of CNN layers, and c is the time cost of each CNN layer. The time cost of the fusion layer is $O(LNM)$. Thus, the time cost of the MPCNN is $O(qc + LNM)$.

5 Experiments

In this section, we describe empirical studies of the proposed model, including the experimental setup, the experimental results on the effectiveness and efficiency, and the case study.

5.1 Experimental setup

The experimental environment is an Intel Xeon CPU E5-2620 2.10GHz*8 CPU and 64 GB memory, and one GeForce RTX 2080Ti GPU. The proposed model is implemented with Keras and TensorFlow.

5.1.1 Datasets

In these experiments, we use four real-world datasets: BikeNYC01, BikeNYC02, TaxiNYC and TaxiBJ. Each dataset contains crowd flows data and external factor data. The details are as follows.

- BikeNYC01.[1] BikeNYC01 is taken from Citi Bike, New York's bike sharing system, which includes bike rent records and metadata (i.e., weekdays or weekends). The time span of BikeNYC01 is from 4/1/2014 to 9/30/2014, and the time interval is one hour. The map is divided into 20×20 regions. The size of BikeNYC01 is 27 MB.

[1] https://www.citibikenyc.com/system-data

- BikeNYC02. These trajectory data are also collected from New York's bike share system, spanning from 4/1/2014 to 9/30/2014, and the time interval is one hour. Different from BikeNYC01, the map of BikeNYC02 is divided into 16×8 regions. The size of BikeNYC02 is 9 MB.
- TaxiNYC[2]. These trajectory data are generated by taxicabs in New York. The time span of TaxiNYC is from 4/1/2015 to 9/30/2015, and the time interval is an hour. The external factor data are metadata. The map of TaxiNYC is divided into 20×20 regions. The size of TaxiNYC is 27 MB.
- TaxiBJ [37]. This dataset contains taxicab trajectory data, meteorology data and metadata in Beijing from four time intervals: 7/1/2013 - 10/30/2013, 3/1/2014 - 6/30/2014, 3/1/2015 - 6/30/2015 and 11/1/2015 - 4/10/2016. The time interval is half an hour. The external data include metadata, holiday data and meteorology data. The map of TaxiBJ is divided into 32×32 regions. The size of TaxiBJ is 1069 MB.

For BikeNYC01, BikeNYC02 and TaxiNYC, we choose the last 10 days as testing data. For TaxiBJ, we choose the last four weeks as testing data and the other data as training data. Ten percent of the training data are chosen to be validation data. In addition, the data are scaled to the range [-1, 1] by the min-max normalization method, re-scaled to normal values and compared with the ground truth in the evaluation.

5.1.2 Comparison methods

To demonstrate the effectiveness of the proposed model, we compare the proposed method with the following methods.

- HA: We predict new-flow and end-flow of crowds using the average value of historical new-flow and end-flow in the corresponding periods.
- VAR: Vector auto-regression is an advanced spatial-temporal model that can capture pairwise relationships among all flows.
- Deepst [36]: This DNN-based prediction model uses four CNN layers for crowd flows prediction.
- ST-ResNet [37]: This deep learning model uses ResNet to model spatial-temporal correlations in grid-based spatial-temporal prediction.
- MST3D [2]: This method uses 3D CNNs to learn the spatial-temporal features jointly for prediction.
- DeepLGR [14]: This method combines CNN and spatial pyramid pooling for crowd flows prediction.
- GCN+LSTM. This method stacks a graph convolution layer [9] with an LSTM layer. In the method, the graph $G = (V, E)$ is constructed, where V and E represent the sets of vertices and edges, respectively. Each vertex in graph G indicates an individual region, and the edge between two vertices denotes the two regions geographically adjacent to each other. Besides, the external components are not considered in the method.

[2]https://www1.nyc.gov/site/tlc/about/tlc-triprecord-data.page

5.1.3 Evaluation metrics

We use two common criteria to evaluate the proposed model, including root mean square error (RMSE) and mean absolute error (MAE), which are are defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n}(x_i - \widehat{x_i})^2} \tag{7}$$

$$MAE = \frac{1}{n}\sum_{i}^{n}|x_i - \widehat{x_i}| \tag{8}$$

where n is the number of all predicted values, and x_i and $\widehat{x_i}$ are the ground truth and predicted values, respectively. RMSE is used to measure the standard deviation of differences between the ground truth and predicted value, and MAE is used to compute the average of the absolute errors between the ground truth and predicted value.

5.1.4 Parameters

For all baselines, we use the implementations provided by either their authors or open source libraries. All deep models are trained end-to-end by the Adam optimizer. For the MPCNN, the convolution kernel size is 3 × 3 for the three datasets. The number of residual units in the front CNNs, side CNNs, and top CNNs are empirically set to 4, 2, and 3 for BikeNYC01; 4, 3, and 2 for BikeNYC02; 4, 3, and 3 for TaxiNYC; and 12, 12 and 12 for TaxiBJ, respectively. The last convolutional layer (i.e., *Conv 2*) uses 2 convolution kernels. We tune the batch size in {16, 32, 64, 128} and the learning rate in {0.0001, 0.0002, 0.0005, 0.001, 0.005, 0.01}. We set p and q to one day and one week, and $l_c = 3$, $l_p = 4$, $l_q = 4$. We also use early stopping in the training process and then continue to train the model on the full training data for a fixed number of epochs (e.g., 100 epochs).

5.2 Performance comparison

Performance comparisons include single-step prediction and multistep prediction. The single-step head prediction indicates that we make predictions of a single time step ahead. The multistep prediction denotes that we make predictions of multiple time steps ahead. Table 1 reports the performance comparison of single-step prediction with

Table 1 Comparison with different baselines

Methods	BikeNYC01		BikeNYC02		TaxiNYC		TaxiBJ	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
HA	6.93	4.21	11.33	6.56	7.53	3.72	45.77	24.50
VAR	5.58	3.78	9.5	6.12	6.87	2.98	20.55	12.22
Deepst	4.17	1.99	7.29	3.95	4.26	1.59	20.96	13.45
ST-ResNet	3.91	1.65	6.33	3.03	3.65	1.24	16.76	9.58
MST3D	3.83	1.67	6.32	3.07	3.65	1.23	16.63	9.79
DeepLGR	3.88	2.22	6.21	3.39	5.05	2.43	19.07	12.23
GCN+LSTM	3.70	1.81	6.68	3.37	3.93	1.49	16.51	9.84
MPCNN	3.61	1.59	6.11	2.99	3.50	1.15	16.44	9.66

all baselines on four datasets. We make the following observations:

- The conventional methods (i.e., HA and VAR) cannot accurately predict crowd flows because they cannot capture complex spatial-temporal correlations.
- Deepst does not perform better than the other deep learning methods because its framework is too simple to sufficiently capture spatial-temporal correlations.
- MST3D generally achieves better performance than Deepst, ST-ResNet and DeepLGR. Such improvement is attributed to the 3D CNN, which can capture the spatial-temporal correlations, while Deepst, ST-ResNet and DeepLGR adopt the 2D CNN, which focuses on capturing the spatial correlations. Thus, MST3D achieves better performance than these methods.
- GCN+LSTM does not perform better than MPCNN because GCN+LSTM addresses spatial and temporal correlations separately and neglects the correlations between different types of flows.
- The proposed MPCNN outperforms other baselines. Using multiple perspective CNNs, MPCNN can efficiently capture rich spatial-temporal correlations explicitly and uses a fusion layer to efficiently combine the results of the three perspective CNNs.

To evaluate the performance of the MPCNN in more detail, Table 2 reports the results of multistep prediction with other deep learning methods on BikeNYC02. As shown in Table 2, the MPCNN outperforms the other methods as the step number varies from 2 to 4. The RMSE and MAE of all methods also increase as the predicted time step increases. This is because, as the predicted time step increases, simulating the propagation of crowd flows becomes more difficult. Deepst performs the worst because it only uses a few layers, making it difficult to capture long-range dependencies. Thus, the MPCNN achieves better performance compared with existing methods of crowd flow prediction, which demonstrates its superiority in capturing rich spatial-temporal correlations.

5.3 Ablation study

To verify the effectiveness of different components of the MPCNN, we perform ablation experiments on three datasets. In particular, we design four variants of the MPCNN as follows: M-withF denotes that the MPCNN model only uses the front CNN; M-withST denotes that the MPCNN model only uses the side and top CNNs; M-withExt denotes that the MPCNN model only uses the front

Table 2 Performance of multistep prediction

Methods	step=2		step=3		step=4	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Deepst	8.62	4.47	9.40	4.98	10.53	5.29
ST-ResNet	6.72	3.20	7.24	3.37	7.53	3.55
MST3D	7.01	3.29	7.62	3.47	8.05	3.67
DeepLGR	7.57	3.47	8.17	3.93	8.58	4.21
GCN+LSTM	6.72	3.57	7.20	3.80	7.68	4.06
MPCNN	6.61	3.20	7.02	3.32	7.49	3.51

Table 3 Effect of different components

Methods	BikeNYC01		BikeNYC02		TaxiNYC		TaxiBJ	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
M-withF	3.69	1.60	6.35	3.03	3.68	1.29	17.06	9.88
M-withST	3.72	1.63	6.14	2.99	3.56	1.18	16.87	9.80
M-withExt	3.64	1.59	6.24	3.00	3.60	1.21	16.77	9.85
M-Sum	3.65	1.63	6.15	2.99	3.76	1.31	17.75	10.34
MPCNN	3.61	1.59	6.11	2.99	3.50	1.15	16.44	9.66

CNN and considers external factors; and M-Sum denotes that the MPCNN model directly sums the results of the three perspective CNNs.

Table 3 compares these variants of the MPCNN. The MPCNN outperforms M-withF and M-withST, which verifies that the three perspective CNNs are beneficial for crowd flow prediction. The importance of each of the three perspective CNNs is also different. Thus, the performance of M-Sum, which uses equal weight addition, is worse than that of MPCNN with a fusion layer. M-withExt outperforms M-withF, which shows that external factors are beneficial for prediction. We also note that the contributions of different components are different in different datasets. Specifically, in BikeNYC01, M-withF outperforms M-withST, which indicates that the front CNN component contributes more than the side and top CNNs. In BikeNYC02, M-withST outperforms M-withF, which indicates that the side and top CNNs contribute more than the front CNN. In TaxiNYC and TaxiBJ, M-Sum performs the worst among all variants, which indicates that the direct summation fusion of different perspective CNNs yields poor performance and highlights the importance of the fusion mechanism. In a word, the MPCNN performs better than the other investigated variants. The components of the MPCNN are also shown to be effective.

5.4 Evaluation of parameter settings

Next, we study the influence of the number of residual units in the three perspective CNNs and the impact of the temporal closeness, period and trend length. We also study the impact of region partition.

5.4.1 Impact of the number of residual units

We conduct experiments on the impact of the number of residual units in the three perspective CNNs (i.e., the number of residual units n_f, n_s and n_t in the front CNN, the side CNN and the top CNN, respectively). We report the results on the BikeNYC01 dataset. We set n_f, n_s and n_t as 4, 2 and 3, respectively, by default and obtain the performance of the MPCNN with $n_f, n_s, n_t = \{1, 2, 3, 4, 5\}$, as shown in Fig. 6. Figure 6 shows that as the number of residual units increases, the model can capture more information; thus, the performance improves. However, too many layers degrade the performance because trivial information may be captured and the possibility of overfitting also increases. Jointly analyzing Table 1 and Fig. 6, MPCNN is shown to be consistently superior to the other methods when varying n_f, n_s and n_t. These results also demonstrate the robustness and effectiveness of MPCNN.

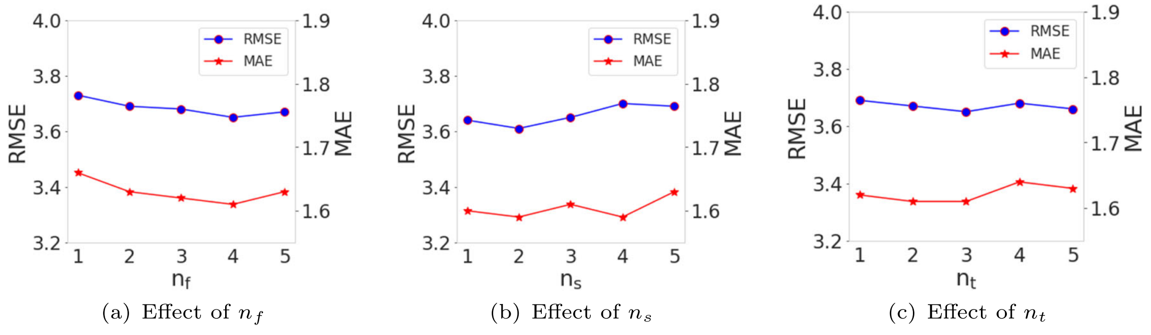

Fig. 6 Impact of the number of residual units in the three perspective CNNs

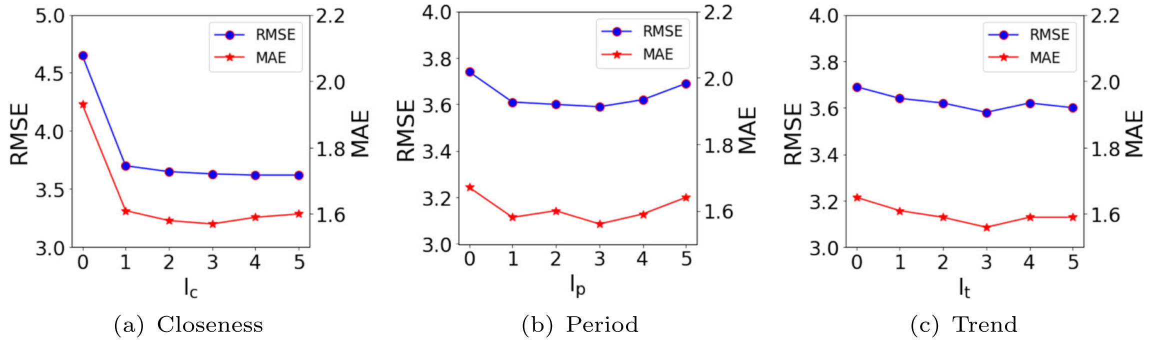

Fig. 7 Impact of temporal closeness, period and trend

5.4.2 Impact of temporal closeness, period and trend

We conduct experiments on the impact of temporal closeness, period and trend on BikeNYC01, as shown in Fig. 7. We set l_c, l_p and l_t as 3, 4 and 4, respectively, by default and obtain the performance of the MPCNN with $l_c, l_p, l_t = \{0, 1, 2, 3, 4, 5\}$. Fig. 7(a) shows the impact of the temporal closeness parameter l_c, where we fix l_p and l_t but change l_c. Specifically, l_c=0 indicates that the model removes the temporal closeness cube, resulting in poor performance. This result verifies the effectiveness of the temporal closeness cube. Figure 7(b) shows the impact of the temporal period parameter l_p. The model without the temporal period cube (l_p=0) performs the worst, which verifies the effectiveness of the temporal period cube. As l_p increases, the RMSE and MAE first decrease and then increase, and l_p=3 has the best performance because a long-range period may be useless and harm the model performance. Figure 7(c) shows the impact of the temporal trend parameter l_t, which has a curve similar to that of l_p. Thus, the temporal closeness, period and trend are demonstrated to be effective, and an excessively long period and trend may not be helpful for prediction.

5.4.3 Impact of region partition

To evaluate the impact of the region partition, we vary the number of regions (i.e., grid cells) divided on BikeNYC01. We divide the city into $N \times M$ disjoint grid cells of equal sizes, such as 5×5, 10×10, 15×15 and 20×20. N and M control the number of regions. The larger N and M are, the greater the number of regions divided and the smaller the size of each region. As shown in Table 4, as the number of regions decreases, the RMSE and MAE increase because when the number of regions is too small, the region partition is too coarse to capture the feature of the region. Also, when the number of regions is too small, the size of the region is large, and the crowd flow is large. Then, when we re-scale the predicted values to normal values and compare them with the ground truth, the error also increases. With more regions, the training time and test time of the model increase because more grids mean a larger resolution, and therefore, more convolutions are required.

5.5 Efficiency comparison

We show the efficiency of the MPCNN from two perspectives: model size and time consumption. For the model size, the number of parameters for the deep models on TaxiBJ are shown in Table 5. The number of parameters of the MPCNN is the smallest compared to other methods except for Deepst. Because the MPCNN can explicitly capture richer spatial-temporal correlations from multiple perspectives, each perspective CNN only requires a smaller number of convolution kernels. Thus, the MPCNN requires fewer parameters and achieves competitive prediction performance compared with other methods. The MPCNN is also a lightweight and effective framework. In addition, the reason why the number of parameters of Deepst is the smallest of the methods investigated in this study is that its model is too simple. GCN+LSTM has the highest number of parameters due to the complex gating mechanism of LSTM.

Table 4 Impact of region partition

The number of regions	RMSE	MAE	Training time(s)	Test time(ms)
5×5	15.14	7.89	264	3
10×10	7.13	3.48	430	9
15×15	4.53	2.12	449	12
20×20	3.61	1.59	524	33

Table 5 Efficiency comparison on TaxiBJ dataset

Methods	Training time(s)	Test time(ms)	Paramters
Deepst	1487	260	300k
ST-ResNet	4145	856	2697k
DeepLGR	3335	777	833k
MST3D	4618	1770	4345k
GCN+LSTM	11784	3250	9442k
MPCNN	2951	233	536k

Regarding time consumption, we compare the training time and test time of MPCNN with other deep models on TaxiBJ. As shown in Table 5, the proposed model is faster than the other models except for Deepst primarily because the fusion layer of the MPCNN model requires more time. Although Deepst takes the least time, its prediction performance is unsatisfactory. Thus, the MPCNN is the most efficient among all methods with high-quality performance.

5.6 Visualization of the fusion layer

To study the fusion layer in the MPCNN in more detail, we visualize the learned weights from (5) in the fusion layer on BikeNYC02 to analyze the contribution of each perspective CNN. Figure 8 shows a portion of the learning weights for fusing the results of three perspective CNNs. Figure 8(a), 8(b) and 8(c) are three 16×8 matrices because the city map is divided into 16×8 grids. Each element in each grid denotes a learned weight of a certain region that reflects the influence degree by the three perspective CNNs. The darker color represents the larger weight. In different regions, the weights of the three-perspective CNNs are different (i.e., the contributions of the three-perspective CNNs are different). For example, the regions r_1 marked by the red circle are more affected by the front and side CNNs than the top CNN, while the regions r_2 marked by the yellow circle are more affected by the top CNN than the front and side CNNs. These results show that different regions are affected differently by the three perspective CNNs, which also confirms that the model with the fusion layer performs better than the direct equal weight summation in ablation experiments.

5.7 Case study

To verify the effectiveness of the MPCNN, we present a real case study of the proposed model on the BikeNYC01 dataset. To show that the MPCNN can capture the correlations between different types of flows, we draw six heat maps of the prediction and ground truth of new-flow and end-flow at 9:00 A.M and 10:00 A.M. Each grid denotes a region. The darker color represents the larger value. As shown in Fig. 9, by comparing Fig. 9(a) with Fig. 9(d), 9(b) with Fig. 9(e) ,9(c) with 9(f) , we find that the predictions are basically consistent with the ground truth. Then, we select three representative regions: $r1$ marked with a red frame, and its nearby regions $r2$ and $r3$ marked with yellow frames. As shown in Fig. 9(d), 9(e) and 9(f), the new-flow of $r1$ at 9:00 A.M is large, which indicates that the crowd flows quickly leave $r1$ at 9:00 A.M. Compared with the

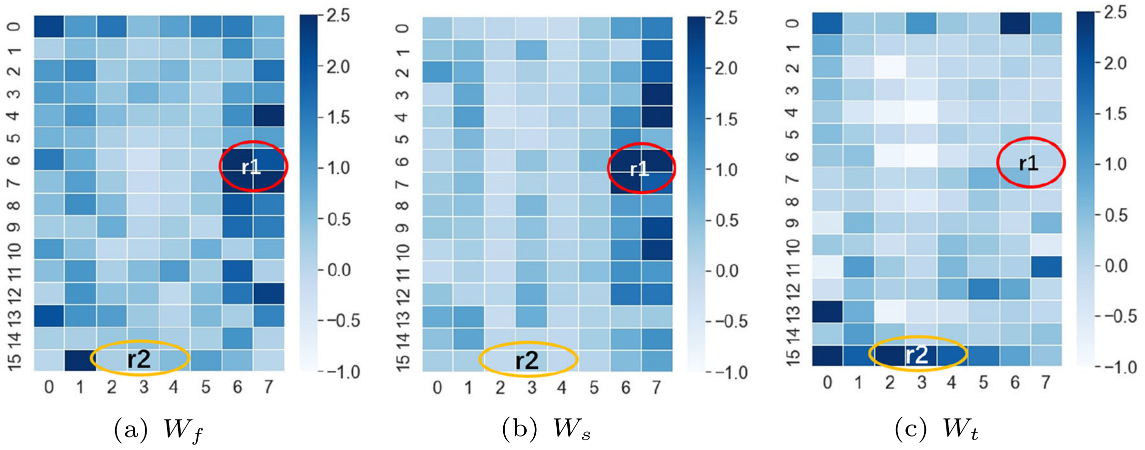

Fig. 8 Visualization of learned weights in the fusion layer on BikeNYC02. (a)(b)(c) denote the learned weights of the front, side and top CNNs, respectively

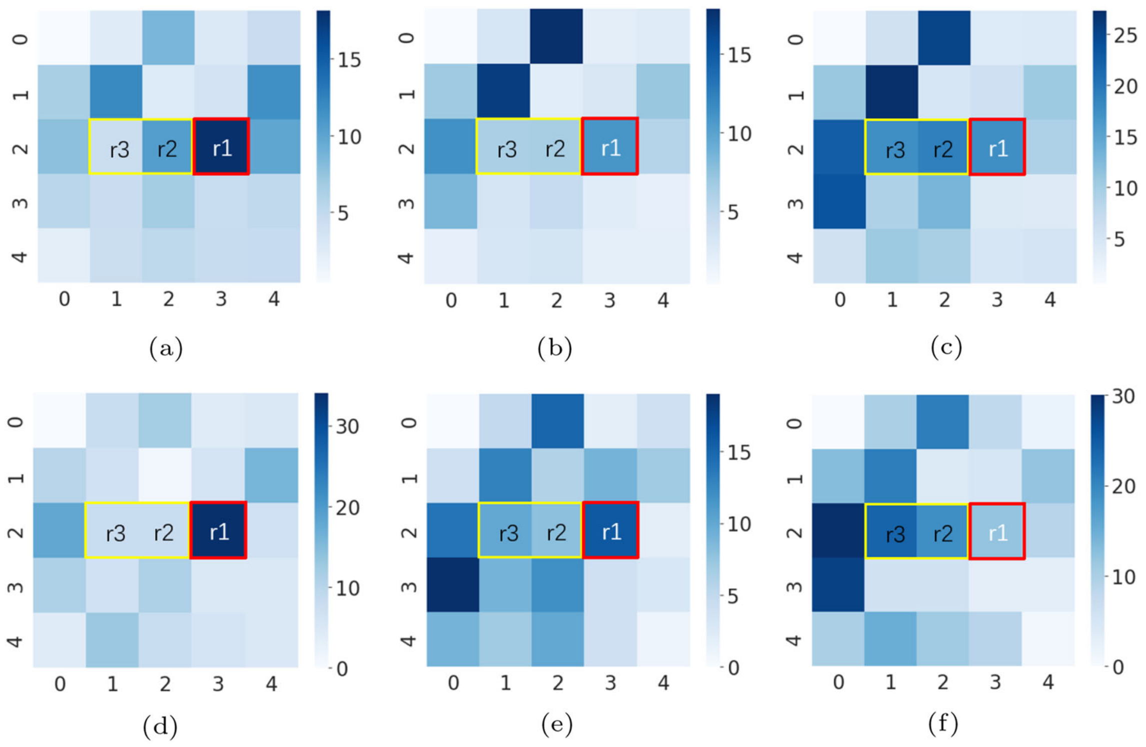

Fig. 9 Case study on the BikeNYC01 dataset. (a)(b)(c) denote the prediction of new-flows at 9:00 A.M, end-flows at 9:00 A.M and 10:00 A.M, respectively; (d)(e)(f) denote the ground truth on new-flows at 9:00 A.M, end-flows at 9:00 A.M and 10:00 A.M., respectively

end-flows of $r2$ and $r3$ at 9:00 A.M, the end-flows of $r2$ and $r3$ at 10:00 A.M. increase markedly. The people flow from $r1$ to the nearby $r2$ and $r3$ at 10:00 A.M. As shown in Figs. 9 (b) and 9(c), the MPCNN can efficiently capture the changes in the end-flows of $r2$ and $r3$ between 9:00 A.M. and 10:00 A.M., which verifies that the MPCNN can capture the correlations of the new-flow of $r1$ at 9:00 A.M on the end-flow of $r2$ and $r3$ at 10:00 A.M. Thus, this case study shows that the MPCNN efficiently captures the correlations among regions with different types of flows for crowd flow prediction.

In addition, the predicted crowd distribution is beneficial for risk assessment and traffic management. For example, the heat map in Fig. 9 shows the crowd distribution, where each grid stands for a region and the color associated with it denotes its new-flow and end-flow. When the color of a certain region in the predicted crowd distribution is dark, the flows in this region are high. Some safety measures can be taken in advance, such as sending out warnings, conducting traffic control, or evacuating people.

6 Conclusion

In this paper, we propose a novel 2D CNN-based model called MPCNN to predict crowd flows. In MPCNN, we propose three perspective CNNs to effectively capture the complex spatial-temporal correlations hidden in crowd flows data. We also propose a fusion layer to fuse the results of the three perspective CNNs. In addition, we combine the external factors to enhance prediction performance. Experimental results on real-world datasets verify the effectiveness and efficiency of the proposed model.

References

1. Belhadi A, Djenouri Y, Djenouri D et al (2020) A recurrent neural network for urban long-term traffic flow forecasting. Appl Intell 50(10):3252–3265
2. Chen C, Li K, Teo SG et al (2018) Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction. In: IEEE international conference on data mining, ICDM, pp 893–898
3. Dai G, Hu X, Ge Y et al (2021) Attention based simplified deep residual network for citywide crowd flows prediction. Front Comput Sci 15(2):152,317
4. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems, pp 3844–3852
5. Feng J, Lin Z, Xia T et al (2020) A sequential convolution network for population flow prediction with explicitly correlation modelling. In: Bessiere C (ed) Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI, pp 1331-1337
6. Guo S, Lin Y, Feng N et al (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting

7. Hoang MX, Zheng Y, Singh AK (2016) Fccf: forecasting citywide crowd flows based on big data. In: Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems, pp 6:1–6:10
8. Huang F, Yi P, Wang J et al (2022) A dynamical spatial-temporal graph neural network for traffic demand prediction. Inf Sci 594:286–304
9. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations
10. Kumar SV (2017) Traffic flow prediction using kalman filtering technique. Procedia Eng 187:582–587
11. LeCun Y, Bottou L, Bengio Y et al (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
12. Li Y, Yu R, Shahabi C et al (2018) Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In: 6th international conference on learning representations, ICLR
13. Liang Y, Ouyang K, Jing L et al (2019) Urbanfm: inferring fine-grained urban flows. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, KDD, pp 3132–3142
14. Liang Y, Ouyang K, Wang Y et al (2020) Revisiting convolutional neural networks for citywide crowd flow analytics. In: Machine learning and knowledge discovery in databases - European conference, pp 578–594
15. Lin Z, Feng J, Lu Z et al (2019) Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In: The thirty-third AAAI conference on artificial intelligence, AAAI, pp 1020–1027
16. Pan Z, Liang Y, Wang W et al (2019a) Urban traffic prediction from spatio-temporal data using deep meta learning. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp 1720–1730
17. Pan Z, Wang Z, Wang W et al (2019b) Matrix factorization for spatio-temporal neural networks with applications to urban flow prediction. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp 2683–2691
18. Poon KH, Wong PK, Cheng JCP (2022) Long-time gap crowd prediction using time series deep learning models with two-dimensional single attribute inputs. Adv Eng Inform 51:101,482
19. Singh U, Determe J, Horlin F et al (2020) Crowd forecasting based on wifi sensors and LSTM neural networks. IEEE Trans Instrum Meas 69(9):6121–6131
20. Song C, Lin Y, Guo S et al (2020) Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting. In: The thirty-fourth AAAI conference on artificial intelligence, AAAI, pp 914–921
21. Su H, Maji S, Kalogerakis E et al (2015) Multi-view convolutional neural networks for 3d shape recognition. In: 2015 IEEE international conference on computer vision, pp 945–953
22. Sun J, Zhang J, Li Q et al (2022) Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks. IEEE Trans Knowl Data Eng 34(5):2348–2359
23. Tian C, Zhu X, Hu Z et al (2020) Deep spatial-temporal networks for crowd flows prediction by dilated convolutions and region-shifting attention mechanism. Appl Intell 50(10):3057–3070
24. Tu Y, Lin S, Qiao J et al (2021) Deep traffic congestion prediction model based on road segment grouping. Appl Intell 51(11):8519–8541
25. Wang J, Zhu W, Sun Y et al (2021) An effective dynamic spatiotemporal framework with external features information for traffic prediction. Appl Intell 51(6):3159–3173
26. Wang S, Miao H, Chen H et al (2020) Multi-task adversarial spatial-temporal networks for crowd flow prediction. In: The 29th ACM international conference on information and knowledge management, pp 1555–1564
27. Wu C, Yin T, Ge S et al (2017) Ensemble learning for crowd flows prediction on campus. In: Proceedings of international conference on smart computing and communication, pp 103–113
28. Xia T, Lin J, Li Y et al (2021) dgcn: 3-dimensional dynamic graph convolutional network for citywide crowd flow prediction. ACM Trans Knowl Discov Data (TKDD) 15(6):110:1–110:21
29. Xu J, Zhang X, Li W et al (2020) Joint multi-view 2d convolutional neural networks for 3d object classification. In: Proceedings of the twenty-ninth international joint conference on artificial intelligence, pp 3202–3208
30. Yabe T, Tsubouchi K, Sudo A et al (2016) Predicting irregular individual movement following frequent mid-level disasters using location data from smartphones. In: Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems, GIS. ACM, pp 54:1–54:4
31. Yang B, Sun S, Li J et al (2019) Traffic flow prediction using LSTM with feature enhancement. Neurocomputing 332:320–327
32. Yao H, Wu F, Ke J, et al (2018) Deep multi-view spatial-temporal network for taxi demand prediction. In: Proceedings of the thirty-second AAAI conference on artificial intelligence, pp 2588–2595
33. Yao H, Tang X, Wei H et al (2019) Revisiting spatial-temporal similarity: a deep learning framework for traffic prediction. In: The Thirty-third AAAI conference on artificial intelligence, pp 5668–5675
34. Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: Proceedings of international joint conferences on artificial intelligence, pp 3634–3640
35. Yuan H, Zhu X, Hu Z et al (2020) Deep multi-view residual attention network for crowd flows prediction. Neurocomputing 404:198–212
36. Zhang J, Zheng Y, Qi D et al (2016) Dnn-based prediction model for spatio-temporal data. In: Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems, pp 92:1–92:4
37. Zhang J, Zheng Y, Qi D (2017) Deep spatio-temporal residual networks for citywide crowd flows prediction. In: Proceedings of AAAI conference on artificial intelligence, pp 1655–1661
38. Zhang X, Huang C, Xu Y et al (2020) Spatial-temporal convolutional graph attention networks for citywide traffic flow forecasting. In: The 29th ACM international conference on information and knowledge management, pp 1853–1862
39. Zhang Y, Yang Y, Zhou W et al (2021) Multi-city traffic flow forecasting via multi-task learning. Appl Intell 51(10):6895–6913
40. Zhou Q, Gu J, Ling C et al (2020) Exploiting multiple correlations among urban regions for crowd flow prediction. J Comput Sci Technol 35(2):338–352
41. Zhou X, Shen Y, Zhu Y et al (2018) Predicting multi-step citywide passenger demands using attention-based neural networks. In: Proceedings of the eleventh acm international conference on web search and data mining, WSDM, pp 736–744