International Journal of Machine Learning and Cybernetics (2021) 12:1879–1891
https://doi.org/10.1007/s13042-021-01282-z

ORIGINAL ARTICLE

An efficiency-enhanced deep learning model for citywide crowd flows prediction

Zhongyi Zhai[1] · Peipei Liu[1] · Lingzhong Zhao[1] · Junyan Qian[1] · Bo Cheng[2]

Received: 8 October 2019 / Accepted: 3 February 2021 / Published online: 7 April 2021

Abstract
The crowd flows prediction plays an important role in urban planning management and urban public safety. Accuracy is a challenge for predicting the flow of crowds in a region. On the one hand, crowd flow is influenced by many factors such as holidays and weather. On the other hand, sample data about crowd flows are generally high-dimensional, which not only has a negative impact on the prediction accuracy but also increases computational complexity. In this paper, an efficiency-enhanced model is constructed for predicting citywide crowd flows based on multi-source data using deep learning techniques. Specifically, a data reconstruction mechanism is built with Bernoulli restricted Boltzmann machine (BRBM), for the purpose of reducing the dimension of sample data. A collaborative prediction mechanism is introduced to improve the prediction accuracy of crowd flows, in which a spatio-temporal data oriented prediction model is constructed based on bottleneck residual network that can reduce the effectively computational complexity of model training, and an auxiliary prediction to further optimize the prediction accuracy based on the fully-connected network. The proposed method is evaluated by using two open datasets. The experimental results show that our method can significantly improve the prediction accuracy and reduce the training time of the prediction model, compared with other methods.

Keywords Deep learning · Bernoulli-RBM · Data reconstruction · Bottleneck residual network · Crowd flows prediction

1 Introduction

With the rapid growth of population and economic development in a city, crowds sometimes surge around in a prosperous region rapidly. Consequently, a series of safety accidents, e.g., crowd stampedes, may be caused by such emergency. For instance, 36 persons were killed in a stampede accident at Shanghai's Bund area when huge crowds of revelers gathered to celebrate the 2014 New Year. If crowd flows of every region in a city can be predict accurately, some safety accidents will be avoided effectively by taking emergency plans and measures in advance. Therefore, researches related to smart cities [1] and public safety have paid great attention to the problem of crowd flows forecasting.

Deep learning (DL) has become a promising approach for crowd flows prediction with the abundance of city data in recent years. Compared to traditional machine learning-based prediction models, DL-based models can explore nonlinear features from crowd flows data, which enhances prediction accuracy and applicability. Presently, some research works have utilized DL techniques to predict crowd flows [2–4]. These works usually use single traffic data to predict crowd flows. Nevertheless, the flow of crowds is influenced by many factors. For example, people may prefer to take part in big celebrations on holiday, reduce the number of going out on rainy days. Thus, multi-source data (eg., traffic data, weather data, holiday data) based prediction is a more competitive method for crowd flows in the cities.

Zhang et al. [5] proposed a deep ST-ResNet, which uses traffic and weather data to predict crowd flows. Although the ST-ResNet model improves the prediction accuracy compared to other existing models, there are still some problems. Firstly, source data related to crowd flows are high dimensional generally, which may reduce the prediction accuracy

✉ Lingzhong Zhao
zhaolingzhong163@163.com

✉ Junyan Qian
qjy2000@gmail.com

[1] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

of crowd flows and increase computational complexity. Then, the ST-ResNet model is complex, it not only requires more computational cost but also spends more time training the network.

To address the above problems, a multi-source data-based deep spatio-temporal bottleneck residual network (ST-B-ResNet) prediction method is introduced. Firstly, BRBM-based data reconstruction mechanism is used to reconstruct high-dimensional crowd flows data of a city. Then, a collaborative prediction mechanism is introduced to predict crowd flows. Specifically, a convolution-based neural network is employed to construct spatial dependencies between any two regions, and three bottleneck residual networks are used to model three temporal properties of crowd flows, such as *closeness*, *day*, *week*. Also, an auxiliary prediction method is introduced based on external factors to optimize the prediction accuracy of crowds flows. Overall, number of parameters and computational complexity of prediction method is greatly reduced by establishing the bottleneck remaining cells. By calculation, the number of connections in ST-B-ResNet is about a quarter of ST-ResNet. The experiment result shows that the proposed model can improve effectively the prediction accuracy, meanwhile reduce the training time.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes some and basic theories. Section 4 presents introduces the process of data reconstruction and the proposed crowd flows prediction model. Section 5 presents the training algorithms or data reconstruction model and prediction model. Section 6 shows the experiment and evaluation results. Finally, we conclude this paper in Sect. 7.

2 Related work

This section introduces the related work from two aspects: single data source-based crowd flows prediction, and multi-source data-based crowd flows prediction.

2.1 Single data source-based crowd flows prediction

Wang et al. studied the Gaussian regularization residual learning and proposed a noise-residual network (noise-ResNet) to predict crowd flows. Compared with the ST-ResNet algorithm, this method uses traffic flow data, which greatly reduces the training time and slightly improves prediction accuracy [2]. Nicholas G. Polson et al. proposed an architecture to predict traffic flows. In the traffic prediction architecture, a regularized vector autoregressive model is used to perform variable selection, and a sequence of *tanh* activation functions is used to address the issue of non-linear and non-stationary relations between variables (speed measurements) [3]. Daisuke Sato et al. proposed a multi-agent simulator to predict human traffic and proposed a data assimilation method that can real-time acquire data to improve simulation accuracy [6]. Daisuke Sato et al. developed a human traffic prediction system, which combines the multi-agent simulator to predict the occurrence of congestion. Even if the past information is not available, the system can use the collected data on the day to predict the flow of crowds.

2.2 The multi-source data-based crowd flows prediction

Minh X. Hoang et al. proposed a method to predict two types of crowds flow for every region of a city based on big data, including human mobility data, weather conditions, and road network data [7]. In this method, seasonal and trend models are constructed as intrinsic Gaussian Markov random fields. It can cope with noisy and missing data. LiuYang et al. proposed an end-to-end deep learning architecture to predict passenger inflow and outflow of the subway. The architecture integrates external environmental factors, temporal dependencies, spatial features, and metro operational properties to predict short-term metro passenger flow [8]. Wenzhe et al. proposed a mixed model based on Gaussian mixture clustering model to predict the number of bicycles rented/returned by each station group in a period of time in the future, in order to carry out redistribution in advance [9]. The model considers the impact of meteorological factors on the prediction results. Zhang J et al. proposed a deep learning architecture combining the ResNet, GCN, and LSTM to forecast short-term passenger flow in urban rail transit on a network scale [10]. The proposed approach combines external factors eg., weather, holidays, and workdays or weekends, to collaboratively forecast the crowd flows. Lin Z et al. proposed the DeepSTN+ model based on space-time convolution to predict traffic flow [11]. The model considers the impact of external factors on crowd flows prediction. Although this model has improved the prediction accuracy, it does not consider the problem of gradient vanishing as the depth of the network increases. Zheng et al. proposed deep ST-ResNet for predicting citywide crowd flow with external factors [5]. Deep ST-ResNet solves the problem of network performance degradation with the increase of network's depth. However, a large number of high-dimensional crowd flows data still reduce prediction accuracy and increase computational complexity of deep ST-ResNet.

3 Basic models

This section introduces some basic models used in crowd flows prediction, including crowds flow model, Bernoulli RBM and Bottleneck Residual Network.

3.1 Modeling crowd flows problem

Definition 1 (*Region*) Formally, the domain of a city is divided into several $P \times Q$ grid maps based on longitude and latitude, where each grid represents a region. As shown in Fig. 1, $r_{i,j}$ is one of $P \times Q$ grids. For a grid $(p_1, q_1) \in (P, Q)$, it lies at the p_1-th row and the q_1-th column, and $(p_1, q_1) \Leftrightarrow r_{1,1}$. Each gird has two types of crowd flows: inflow and outflow.

Inflow denotes the total crowd flows entering a region from other areas within a given time interval. The outflow is the total crowd flows leaving a region for other regions within a certain time interval. Inflow/outflow can be measured by the number of cars driving on nearby roads, the number of pedestrians, and the number of people on public transportation systems (such as subways, buses).

Definition 2 (*Inflow/outflow*) Let M be a set of crowd flow trajectories at the n-th time interval. For a grid $(p, q) \in (P, Q)$, it lies at the p-th row and the q-th column. The following two formulas indicate the inflow and outflow of the crowd at the time interval n, respectively.

$$x_n^{in,p,q} = \sum_{T_r \in M} |\{z > 1 | g_{z-1} \notin (p,q) \wedge g_z \in (p,q)\}| \quad (1)$$

$$x_n^{out,p,q} = \sum_{T_r \in M} |\{z \geq 1 | g_z \in (p,q) \wedge g_{z+1} \notin (p,q)\}| \quad (2)$$

Where T_r: $g_1 \rightarrow g_2 \rightarrow g_3 \cdots \rightarrow g_{T_r}$ represents a trajectory in the set M, and g_z is the geospatial coordinate, $g_z \in (p, q)$ means g_z lies within the grid (p, q), and vice versa, $|\cdot|$ represents the cardinality of a set. Inflow and outflow in all $P \times Q$ regions can be represented as a tensor $X_n \in R^{2 \times p \times q}$, where $(X_n)_{0,p,q} = x_n^{in,p,q}$, $(X_n)_{1,p,q} = x_n^{out,p,q}$. A spatial region is represented by a $P \times Q$ grid map. There are two types of flows in the (p, q), i.e., the crowd inflow and crowd outflow. Thus, the historical observation at any time can be represented by a tensor $X_n \in R^{2 \times p \times q}$.

Crowd flow prediction can be abstracted as follow: Predict X_n based on the historical observations $\{X_n | n = 0, 1, 2, 3, \cdots, t-1\}$.

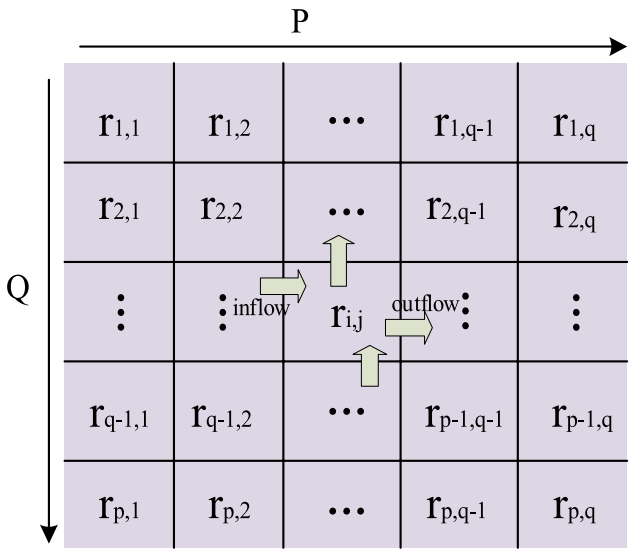

Fig. 1 The sketch map of crowd inflow/outflow meta-model in a city

3.2 Bernoulli restricted Boltzmann machine

Restricted Boltzmann Machine (RBM) is a variant of Boltzmann Machine. It is an energy-based model for unsupervised learning [12]. It has been used in data dimension reduction [13], collaborative filtering [14] and feature learning. RBM consists of a hidden layer with some random hidden units, which obey the Bernoulli distribution, and a visible layer with some random visible units, which obey the usually Bernoulli distribution or Gauss distribution.

If visible units in the RBM obey Bernoulli distribution, it is called Bernoulli-RBM (BRBM). All units are binary random units, that is, input data are either binary or real-valued between 0 and 1.

The structure of the Bernoulli-RBM is shown in Fig. 2. Where,

(1) i, j: represent the number of neurons contained in the visible layer and the hidden layer respectively.
(2) $v = (v_1, v_2, v_3, \cdots, v_i)^T$ represents the state vector of the visible layer, in which v_i represents the state of the i-th neuron.
(3) $h = (h_1, h_2, h_3, \cdots, h_j)^T$ represents the state vector of the hidden layer, in which h_j represents the state of the j-th neuron.

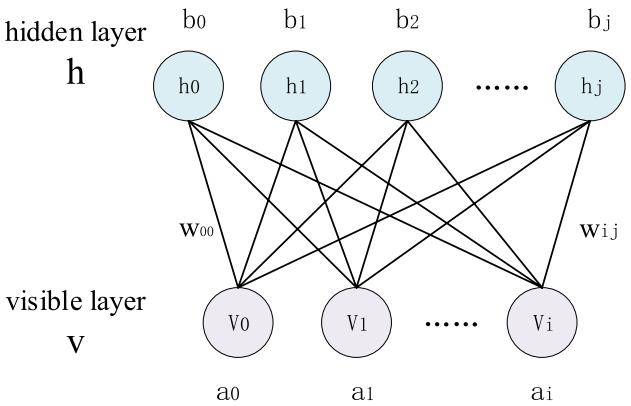

Fig. 2 The structure of Bernoulli RBM model

(4) $a = (a_1, a_2, a_3, \cdots, a_i)^T$ represents the bias value of the visible layer, in which a_i represents the bias value of the i-th neuron.
(5) $b = (b_1, b_2, b_3, \cdots, b_j)^T$ represents the bias value of the hidden layer, in which b_j represents the bias value of the j-th neuron.
(6) $W = (w_{i,j})$ represents the weight matrix connecting the hidden layer and visible layer, in which $w_{i,j}$ represents the weight between the i-th neuron in the visible layer and the j-th neuron in hidden layer.

3.3 Bottleneck residual network

Bottleneck ResNet(B-ResNet) [15] is a deep convolutional neural network. It can protect the integrity of information by bypassing input information to the output. The whole network only needs to learn the different parts between input and output. B-ResNet solves the problem of model performance degradation with the increase of network layers. It can also reduce the number of network parameters greatly and has achieved good effect in many challenging areas such as object detection, and image classification, segmentation and localization [15].

Formally, the desired underlying mapping is denoted as $H(x)$, and the stacked bottleneck residual blocks satisfied the mapping of $F(x) := H(x) - x$. The original mapping is reconverted into $F(x) + x$ [16]. Figure 3(a) shows a building block of B-ResNet, where the upper 1×1 layer is used to reduce dimensions, the 3×3 layer extracts the features with smaller input/output dimensions, the lower 1×1 layer recovers the data style as upper layer's, and *Relu* is an activation function. Figure 3(b) shows the ordinary convolution unit adopted by ST-ResNet, and two 3×3 convolution kernels are used to calculate the convolution. X_l and X_{l+1} are the input and output of the bottleneck residual block respectively. $F(x)$ is a residual function.

4 Multi-source data-based deep spatio-temporal bottleneck ResNet prediction model

Figure 4 shows the architecture of the deep ST-B-ResNet prediction model. It consists of two major components: data reconstruction mechanism, multi-source data fusion prediction mechanism. The fusion prediction mechanism is composed of spatio-temporal data oriented prediction module, and auxiliary prediction module based on external factors. Here, spatio-temporal data refer to traffic flows data of a city, such as taxi trajectory data and bicycle trajectory data, and these data include spatial property and temporal property. External data mainly include weather data, holiday data, and metadata (i.e., Weekend/Weekday). Traffic flows data and external data are used to predict collaboratively the flow of crowds in each region of a city.

As shown in left side of Fig. 4, the reconstruction module firstly convert data of crowds flow into a multi-channel image format, where the inflow and outflow are mapping into a channel respectively. Then, the reshaped multi-channel image data are fed into BRBM for reconstructing into single-channel images, i.e.,two-dimensional flow matrix.

Spatio-temporal data oriented prediction module consists of three time dimensions: *closeness*, *day*, and *week*. According to temporal properties of traffic flows data, different timestamps are selected and concatenated together to model three temporal properties respectively, that is, *closeness*, *day*, and *week*. For every time dimension, a sequence

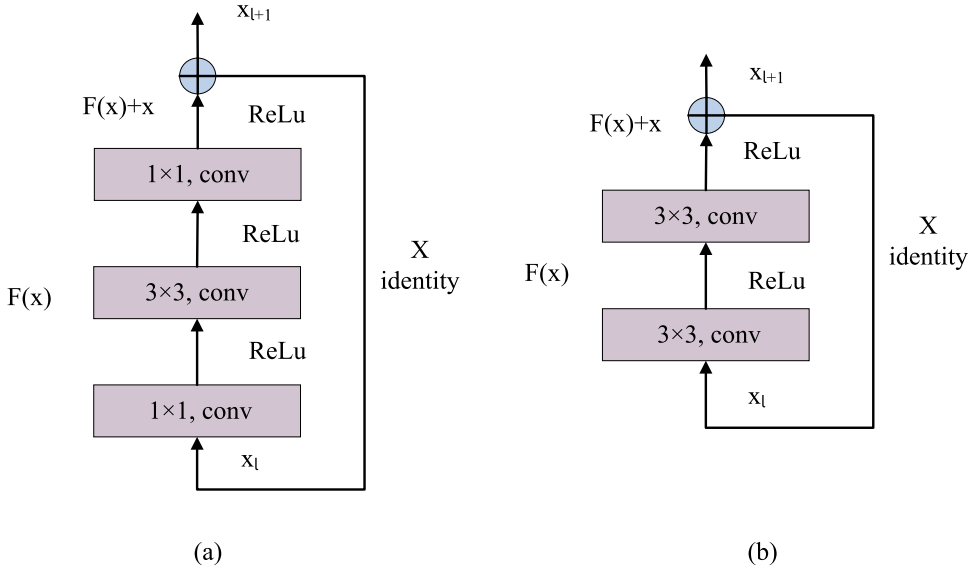

Fig. 3 Bottleneck convolution unit and traditional convolution unit

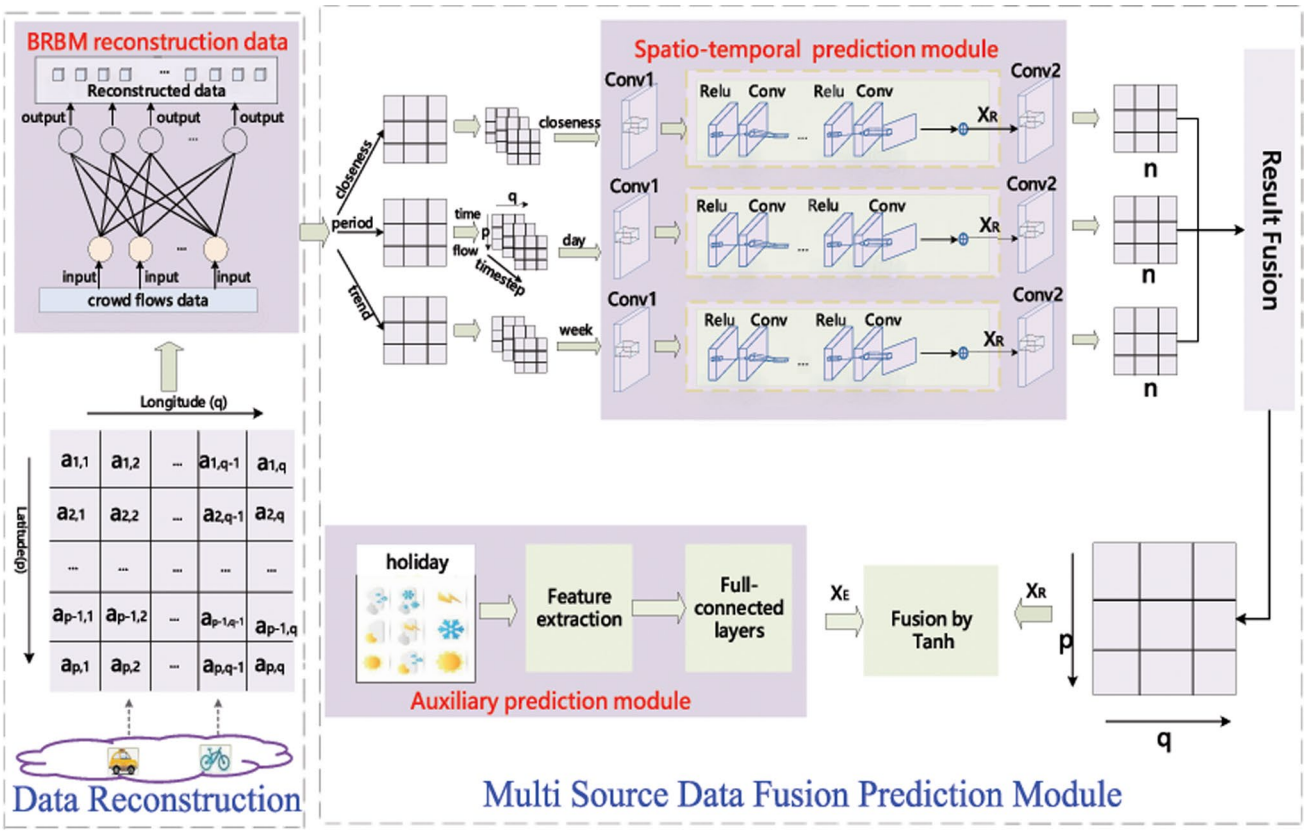

Fig. 4 The architecture of multi-source data-based deep spatio-temporal bottleneck ResNet prediction model

of two-dimensional flow matrices is constructed. These three sequences separately trains the prediction model according to their own temporal data. In this way, a three-dimension time-awareness learning architecture is built. The outputs of three-dimension architecture are fused based on the parameter matrix [17] as X_R. The network structures of every time dimension is composed of two CNNs and a sequence of bottleneck residual blocks. This network structure can deal with spatial dependencies between nearby and remote areas. In the auxiliary prediction module, external factors data are converted into binary vectors. The temperature and wind speed data are normalized to map the data between [0, 1]. Then, the data are fed into the two-layer fully-connected networks. The outputs of the full-connected network are defined as X_E. X_E and X_R are intergrated together to further improve the prediction accuracy. Finally, the fusion results are mapped to $[-1, 1]$ by Tanh activation function.

4.1 High-dimensional data reconstruction

In the multi-source data-based crowds flow prediction, the reference data are from many fields, and have different forms. That is, the source data for crowds flow prediction may be high-dimensional. Generally, traffic flows data of a city are high-dimensional, which has a negative impact on the prediction accuracy of crowd flows. In addition, due to the environmental specificity of regions, the crowds flow of two regions may have a huge difference. For example, the crowd flows of one region may be 0 sometimes, while crowd flow of nearby region may be 10000. The difference between these data values may reduce the prediction accuracy of crowds flow. Therefore, a data reconstruction mechanism based on BRBM network is built to reconstruct traffic flows data and extract important features for improving the performance and effectiveness of prediction.

In the first step of data reconstruction, the source data are reshaped into multi-channel image format, as shown in Fig. 5. Two single-channel images at adjacent time points are combined into a multi-channel image. The inflow and outflow are mapping into a channel respectively. In this process, multiple channels of images are divided according to important factors such as the input and output of traffic. In the next step, the reshaped multi-channel image should be encoded by BRBM to obtain the single-channel image, so as to achieve dimension reduction. In BRBM, the value of each pixel point is between 0 and 1. It indicates the probability of the flow in this area.

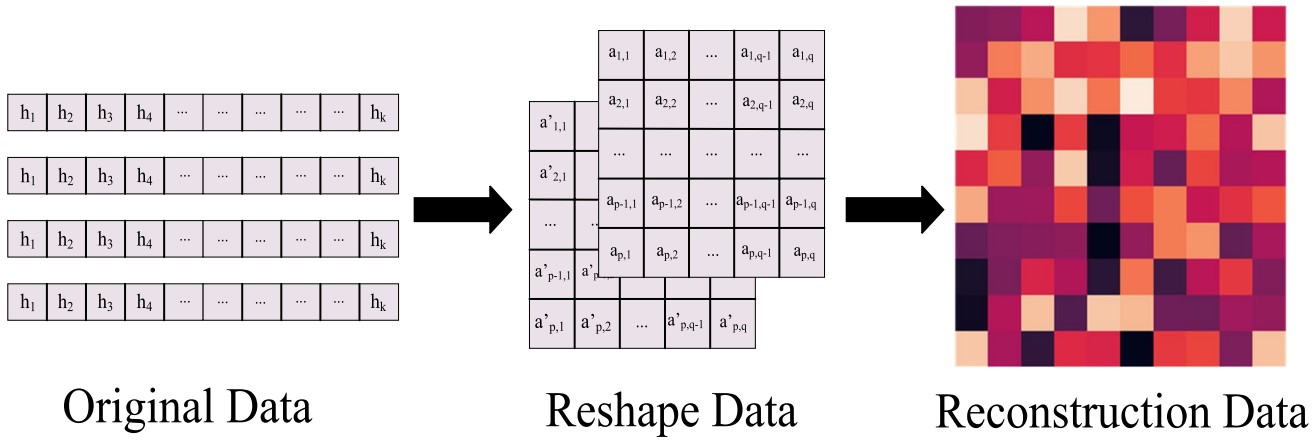

Fig. 5 Reconstruction process

The BRBM network includes two layers: visible layer and hidden layer. The visible layer includes varity of visible units, and each visible unit corresponds to a image pixel. The hidden layers also include a lot of hidden units which are capable of modeling dependencies between input data, i.e. dependencies between pixels in an image. Generally ,the hidden units is regarded as non-linear feature detectors [18]. Through the training of BRBM, the probability distribution of reconstructed data can be approximately equal to the input data, so as to achieve the purpose of data dimension reduction.

Data reconstruction based on BRBM. What the energy model needs to do is to define an appropriate energy function firstly. Then the probability distribution of traffic flows data can be calculated with energy function. According to the probability distribution, an objective function, i.e., maximum likelihood can be solved. The joint configuration energy equation $E(v, h|\theta)$ of BRBM can be defined as:

$$E(v, h|\theta) = -\sum_i \sum_j w_{i,j} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j \tag{3}$$

Where $\theta = \{a, b, w\}$, the joint probability distribution of the (v, h) can be got as follows:

$$p(v, h|\theta) = \frac{1}{z_\theta} \exp(-E(v, h|\theta)) \tag{4}$$

Where z_θ is normalized factor, and the expression is:

$$z_\theta = \sum_v \sum_h \exp(-E(v, h|\theta)) \tag{5}$$

Based on eq. (3), $p(v|\theta)$, which represent the probability distribution function of crowd flows data of input layer, can be computed with the following:

$$p(v|\theta) = \frac{1}{z_\theta} \sum_h \exp(-E(v, h|\theta)) \tag{6}$$

The process of training BRBM is to find a set of θ so that the outputs of visible layer are as same as the distribution of traffic flows data. To achieve this goal, a likelihood function is introduced as follow:

$$\ln L_{\theta,s} = \sum_{i \in s} p(v_i) \tag{7}$$

Via maximizing the likelihood function, an optimal set of parameters θ can be obtained. In eq.(7), v_i is traffic flows data of the i-th region in a city, and s is the number of regions. The edge probability distribution of traffic flows data can be also calculated with following two formulas:

$$p(h_j|v, \theta) = \sigma\left(b_j + \sum_i v_i w_{i,j}\right) \tag{8}$$

$$p(v_i|h, \theta) = \sigma\left(a_j + \sum_j h_j w_{i,j}\right) \tag{9}$$

In eq. (8) and (9), "σ" is sigmoid activation function, as follows:

$$\sigma = \frac{1}{1 + \exp(-x)} \tag{10}$$

Equations (8) and (9) are used repeatedly k times, which can represent the k-th sampling result. The probability distribution approximate to the input data can be obtained, i.e., $\tilde{p}(h|v)$. After m iterations, BRBM training can be completed. The processing of data reconstruction is shown in Fig. 6.

4.2 Multi source data fusion prediction

The fusion prediction mechanism consists of two major components: spatio-temporal prediction module, auxiliary prediction module. Spatio-temporal prediction module adopts a three-dimension time awarness learning structure. The three-dimensions has same structure.

4.2.1 Spatial feature extraction with CNN

Intuitively, the crowds flows in nearby regions may affect each other. Moreover, with the existence of various types of transportation such as subway and highway, two remote locations can create spatial dependencies between each other. CNN has a powerful ability to capture spatial structure information [19]. More convolutional layers can capture a wider range of spatial dependencies. Therefore, a multi-layers CNN is used to capture the spatial dependencies of any region. Being different from classical CNN, the convolutions only used without subsampling.

In "closeness" dimension, the prediction model adopts a few 2-channel flows matrices in the recent time intervals to build temporal closeness dependence. Let historical data of the recent time fragment of entire region be $[X_{n-l_c}, X_{n-(l_c-1)}, \cdots, X_{n-1}]$. This formula is regarded as the closeness dependent sequence. The dependent sequence is concatenated along with the time axis (i.e. time interval) as one tensor $X^{(0)} \in R^{2l_c \times p \times q}$. Then, the tensor is fed into the Conv1, and the size of convolution kernel is 3×3.

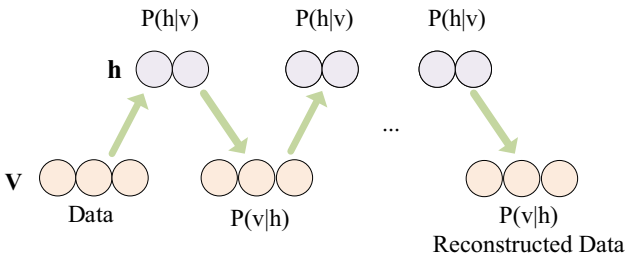

Fig. 6 The processing of BRBM-based data reconstruction

4.2.2 Bottleneck residual block

In the traditional residual block, the accuracy of prediction model is generally improved by increasing network layers. However, with the increase of network layers, it increases the training parameters of the network and computational cost. It needs to take more time to train the prediction model. As a result, the prediction accuracy of the model is reduced. In recent years, bottleneck ResNet [15] has achieved state-of-the-art results for training super deep neural networks. To solve above issues, the bottleneck ResNet is employed to deal with the prediction task. It consists of stacked sequentially bottleneck residual blocks, as follows:

$$X^{l+1} = X^l + F(X^l, W^l), l = 1, 2, 3, \cdots, L \tag{11}$$

Where X^{l+1} and X^l are output and input of the l-th bottleneck residual block in deep bottleneck ResNet. F is a residual function and W^l is the learnable parameter of the l-th bottleneck residual block.

As shown in Fig. 7, a 3×3 convolution is surrounded by two 1×1 convolutions, batch normalization [20] and Relu function. Two 1×1 convolutions are used to reduce the dimension and then restore the data style respectively. The 3×3 convolution layer is as a bottleneck with smaller input/output dimensions. The structure can greatly reduce the number of training parameters and computational complexity. Specifically, a residual block is composed of three BN+Relu+Conv. The Conv2 is added to follow the L-th bottleneck residual block.

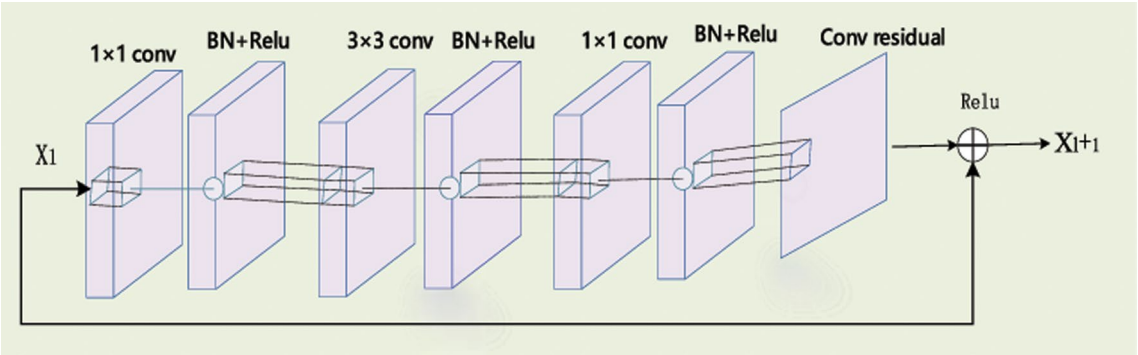

Fig. 7 The bottleneck residual block

Following the three-dimension structure, the spatial tempural prediction model can be bulid. The *day* dependent sequence is $[X_{n-l_d \cdot d}, X_{n-(l_d-1)\cdot d}, \cdots, X_{n-d}]$, l_d is time intervals from the day fragment, the trend is d, and d represents one-day that describes daily periodicity. The *week* dependent sequence is $[X_{n-l_w \cdot w}, X_{n-(l_w-1)\cdot w}, \cdots, X_{n-w}]$, l_w represents the length of week dependent sequence, the trend is w, and w represents one-week that shows the weekly trend.

Beside, the crowd flows can also be affected by many complex external factors such as weather, holidays and activity events. Among these factors, holidays and other active events are certain, while the weather is uncertain in the future time interval of n. The forecasting weather at a time interval n or the approximate weather at a time interval of $n-1$ can be adopted instead of the weather at a future time interval of n. Then, a two fully-connected layers are adopted to deal with external factors data and metadata. The output shape are similar to X_n. X_E is used to represent the output of the auxiliary prediction module.

4.2.3 Prediction results fusion

In the fusion prediction mechanism, a parametric-matrix-based fusion method [17] is used to fuse the outputs of the spatio-temporal prediction model and auxiliary prediction module.

In order to get more accurate prediction result, the trainable parameter w is used to represent the degree to which crowd flows in different regions are affected in different periods. By adjusting w, the impact degree of the different dimensions are affected by *closeness*, *day*, and *week* can be set. Furthermore, the Hadamard product is made with the outputs of the three dimensions (i.e., *closeness*, *day*, and *week*) and their respective influence factors w, to get more accurate prediction results. The results of the three dimensions are as follows:

$$X_R = W_c \circ X_c^{(L+2)} + W_d \circ X_d^{(L+2)} + W_w \circ X_w^{(L+2)} \tag{12}$$

Where, w_c, w_d, w_w denote the weight matrix of *closeness*, *day* and *week*, respectively. The three weights are used to adjust the degree affected by *closeness*, *day* and *week*. "$\circ$" represents Hadamard product. $X_c^{(L+2)}$ is the output of the *closeness* dimension. $X_d^{(L+2)}$ is the output of the *day* dimension, and $X_w^{(L+2)}$ is the output of the *week* dimension. X_R is the fusion result of three dimensions. The output of auxiliary prediction model is defined as X_E, and X_E is merged with X_R. The predicted value at the t-th time interval is defined as X_n:

$$X_n = Tanh(X_R + X_E) \tag{13}$$

Where, *Tanh* is a hyperbolic tangent, and it maps the output results into [0,1].

5 Algorithm

The Persistent Contrastive Divergence (PCD) algorithm [21] is used to train Bernoulli-RBM. Algorithm 1 outlines training process. First, the state of hidden layer nodes is initialized randomly. Then a single Markov chain sampling is used to get the state of hidden layer nodes after k-step (lines 7-9)and calculate gradient(lines 10-13). In the next iteration, the state of the last hidden layer nodes is used to initialize the value of hidden layer nodes sampled by Markov chain (lines 14). The above operation is executed circularly to represent approximately the joint probability distribution of (v, h). Training process of deep ST-B-ResNet is shown in Algorithm 2. A training instances from the original sequence data is constructed. Then deep ST-B-ResNet is trained via back-propagation [22] and Adam (lines 7-11) [23].

Algorithm 1 Persistent Contrastive Divergence (PCD)

1: **Input:** BRBM($V_1, V_2, \cdots, V_m, H_1, H_2, \cdots, H_n$), train batch
2: **Output:** gradient approximation,$\Delta w_{i,j}$,Δb_j and Δa_i for i = $1, \cdots, n, j = 1, \cdots, m$
3: init $\Delta w_{i,j}$=$\Delta a_i = \Delta b_j = 0$ for $i = 1, \cdots, n, j = 1, \cdots, m$
4: $v_{temp} = v$
5: **forall the** $v \in S$ **do**
6: $v^{(0)} \leftarrow v_{temp}$
7: **for** $t = 0, \cdots, k-1$ **do**
8: **for** $i = 1, \cdots, n$ **do** $sample{h_i}^{(t)} \sim p(h_i|v^{(t)})$
9: **for** j = 1,$\cdots$,m **do** sample $v_j^{(t+1)} \sim p(v_j|h^{(t)})$
10: **for** $i = 1, \cdots, n$,$j = 1, \cdots, m$ **do**
11: $\Delta w_{i,j} \leftarrow \Delta w_{i,j} + p(H_i|v^{(0)}) \bullet v_j^{(0)}$- $p(H_i = 1|v^{(k)}) \bullet v_j^{(k)}$
12: $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
13: $\Delta a_i \leftarrow \Delta a_i$+$p(H_i = 1|v^{(0)})$- $p(H_i = 1|v^{(k)})$
14: $v_{temp} = h^{(k)}$

Algorithm 2 ST-B-ResNet Training Algorithm

Input: Historical data:$\{E_0, ..., E_{n-1}\}$;
external factors:$\{x_0, ..., x_{n-1}\}$;
lengths of closeness,day,week sequences:l_c,l_d,l_w;
Day peroid:d; Week peroid:w.
Output: Learned Bottleneck-ResNet model;
//construct training instances
$M \leftarrow \emptyset$
for all available time interval$t(1 \leq n \leq t-1)$ **do**
  $S_c = \left[X_{n-l_c}, X_{n-(l_c-1)}, ..., X_{n-1}\right]$;
  $S_d = \left[X_{n-l_d \cdot d}, X_{n-(l_d-1)\cdot d}, ..., X_{n-d}\right]$;
  $S_w = \left[X_{n-l_w \cdot w}, X_{n-(l_w-1)\cdot w}, ..., X_{n-w}\right]$;
// X_n is the target at time n
put an training instance$(\{S_c, S_d, S_w, E_n\}, X_n)$ into M
// train the model
initialize all learnable parameters θ in Bottleneck-ResNet
repeat
  randomly select a batch of instances M_n from M
  find θ by minimizing the objective (6) with M_n
until stopping criteria is met

6 Experiment and evaluation

6.1 Experiment settings

Platform: The experiment is set on windows Server 2016 with a 2.5GHz 8-Core CPU node. Pycharm intelligent integrated development tool is used as the development environment. The python libraries and Keras [24] with the support of Theano [25] library are used to complete the experiment.

Data preparation: In the experiment, two different datasets are used to verify the deep ST-B-ResNet model. The detailed description of two datasets is as follows:

- **TaxiBJ**: Trajectory data are taxi GPS data and meteorology data in Beijing from four-time intervals : 1st Jul.2013 - 30th Otc.2013, 1st Mar.2014 - 30th Jun.2014, 1st Mar.2015 - 30th Jun.2015, 1st Nov.2015 - 10th Apr.2016. According to Definition 2, two types of crowd flows are obtained. The data are selected from the past four weeks as testing data, and all data before as training data.
- **BikeNYC**: Trajectory data are taken from the New York bicycle system from April 1 to September 30, 2014. The trajectory data include the duration of trajectory, the start and end station IDs, and the start and end times. Among the data, the past 10 days were selected as testing data and the others as training data.

Preprocessing: The traffic flows data contain temporal and spatial properties. Therefore, before the data are reconstructed, the time variables need to be removed, and the Min-Mix-Scaler normalization method is used to map data into the range [0, 1]. The external factors data, such as holiday data, weather data, and metadata, are converted into binary vectors. The temperature and wind speed data are normalized to [0,1].

Table 1 shows 7 baseline models, including 4 traditional machine learning models and 3 deep learning models.

Hyperparameters settings: In the architecture of deep ST-B-ResNet, the convolutions of Conv1 adopt 32 filters of size 3×3 and the convolutions of Conv2 adopt two filters of size 3×3. The convolutions of all bottleneck residual blocks use 16 filters of size 1×1, 16 filters of size 3×3 and 64 filters of size 1×1. The batch size is 32. There are 5 extra hyper-parameters in the deep ST-B-ResNet the p and q are empirically fixed to one-day and one-week, respectively. Lengths of the three dependent sequences are set as: $l_c \in \{3,4,5\}, l_p \in \{1,2,3,4\}, l_q \in \{1,2,3,4\}$.

ST-ResNet adopts the common residual units [5]. The first convolutional layer adopts 32 filters of size 3×3 , and the second convolutional layer adopts 64 filters of size 3×3 .

ST-ResNet and ST-B-ResNet adopt similar structure and the input data. In this way, the operation times of these two models can be illustrated by calculating the number of connections of residual elements. The number of partial connections of residual element in ST-B-ResNet

Table 1 Baselines models

Model	Describtion
HA	The average value of historical crowd flows data for a certain time period is used to predict the flow of crowds for that time period.
ARIMA	The Auto-regressive Integrated Moving Average(ARIMA) is primarily used to understand and predict future values in time series.
SARIMA	Seasonal ARIMA
VAR	Vector auto-regressive(VAR) is a spatio-temporal model that captures the pairwiserelationship between all streams, and it has greatly computational. cost with a large number of parameters.
ST-ANN	It extracts spatial and temporal features firstly, then these features are fed into artificial neural networks.
DeepST	It is a deep neural network (DNN)-based spatio-temporal data prediction model that shows good results on crowd flows prediction.
ST-ResNet	It is a deep spatio-temporal residual networks prediction model. Compared with the above several models, the ST-ResNet model improves the prediction accuracy.

is:$1 \times 1 \times 16 + 3 \times 3 \times 16 + 1 \times 1 \times 64 = 224$. The number of partial connections of residual element in ST-ResNet is : $3 \times 3 \times 32 + 3 \times 3 \times 64 = 864$. The connections of ST-B-ResNet is about a quarter of ST-ResNet's. Overall, the computational complexity of ST-B-ResNet is less than ST-ResNet's. Generally, the lower computational complexity learning model has, the less training time the learning architecture needs to consume.

Evaluation metrics: The mean squared error (MSE) is used as metrics to evaluate the accuracy of Bernoulli-RBM on data reconstruction, as follows:

$$MSE = \frac{1}{z}(v_i - \tilde{v})^2 \tag{14}$$

Where, $\tilde{v}$ and v_i represent the reconstructed data and the original input data respectively, and z represents the number of train data.

Taking the data provided by the New York bicycle system as an example, the relationship between the errors of data reconstruction and the number of training batch is shown in Figs. 8 and 9 respectively. The ordinate of the image represents the reconstruction error and the abscisic coordinate represents the batch size. With the increase of training batch times, reconstruction error becomes smaller and smaller and tends to be stable gradually. The reconstruction error of crowd inflow data is reduced to 0.0017 and the reconstruction error of crowd outflow data is reduced to 0.023. Finally, the experiment results show that it is very effective to reconstruct the data using BRBM.

The Root mean squared error (RMSE) is used to evaluate the prediction accuracy of the deep ST-B-ResNet model, as follows:

$$RMSE = \sqrt{\frac{1}{z}(y_i - \tilde{y})^2} \tag{15}$$

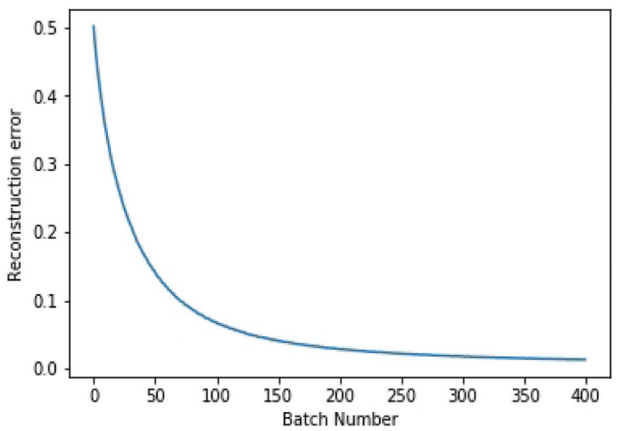

Fig. 8 Reconstruction error of NYC bicycle inflow data

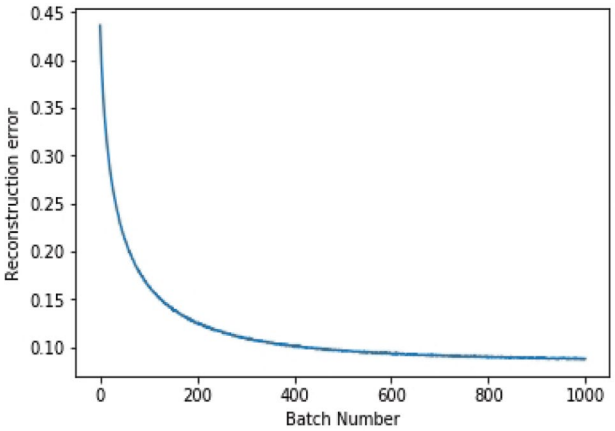

Fig. 9 Reconstruction error of NYC bicycle outflow data

where, $\tilde{y}$ and y_i are the predicted value and ground truth respectively, z is the number of all predicted values.

6.2 Results on TaxiBJ

Several variants of deep ST-B-ResNet with different factors are designed, such as L12-E. L12-E considers all external factors and has 12 bottleneck residual blocks. The details are as follows:

- **The internal structure of bottleneck residual blocks.** In the experiment, two different types of bottleneck residual blocks are used. L12-E adopts 12 bottleneck residual blocks. Compared with L12-E, the bottleneck residual block of L12-E-BN adds two batch normalization(BN) layers, and each of the two BN layers is added before ReLU.
- **External factors.** L12-E considers the external factors, including holiday, weather, and metadata. If not, the model is denoted as L12.
- **Results fusion.** Compared with L12-E, L12-E-noFusion adopt parametric matrix fusion.

Table 2 shows the RMSE of the ST-B-ResNet model and the other seven models on the TaxiBJ dataset. It can be seen that there is a big gap in performance between the traditional model and the deep learning model. In particular, The RMSE of ST-B-ResNet is smaller than ST-ResNet's under the same structure. That is to say, the performance of ST-B-ResNet has been improved. In the comparison experiment, the ST-B-ResNet has a minimum RMSE, i.e., 14.94. In addition, the RMSE of L12-E-noFUSION is 14.98, which increase about 0.04 compared to L12-E-BN. That is, the fusion prediction can improve the accuracy to some extent.

Table 2 Comparison among different models on TaxiBJ

Model		RMSE
HA		57.69
ARIMA		22.78
SARIMA		26.88
VAR		22.88
ST-ANN		19.57
DeepST		18.18
ST-RESNET/L12-E	12 residual blocks+E	16.89
ST-B-RESNET/L12-E(ours)	12 residual blocks+E	14.95
ST-RESNET/L12-E-BN	L12-E with BN	16.69
ST-B-RESNET/L12-E-BN(ours)	L12-E with BN	14.94
ST-RESNET/L12	12 residual blocks	17.00
ST-B-RESNET/L12(ours)	12 residual blocks	14.96
ST-RESNET/L12-E-noFusion	12 residual blocks + E without fusion	17.96
ST-B-RESNET/L12-E-noFusion(ours)	12 residual blocks + E without fusion	14.98

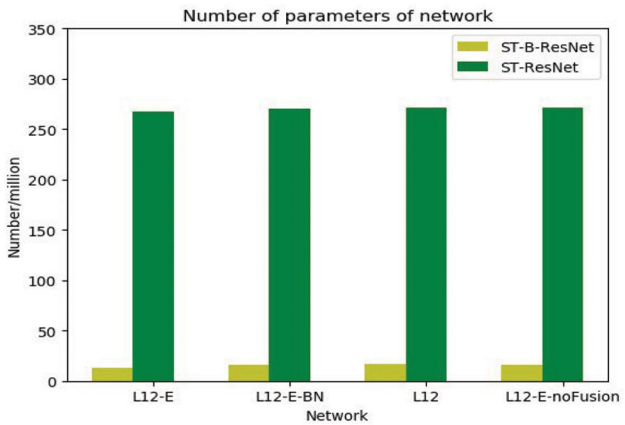

Fig. 10 Number of parameters on TaxiBJ of the network

The deep ST-B-ResNet model also reduces greatly the number of trainable parameters, saves a lot of computational costs, and shortens the training time of the model. Figures 10 and 11 compare the structural parameters and training time of four different ST-ResNet models and ST-B-ResNet models. The ST-B-ResNet reduces the number of trainable parameters by 16.5 to 20.1 times, compared to ST-ResNet on TaxiBJ, and it also reduces the running time of network models by 2 to 3 times.

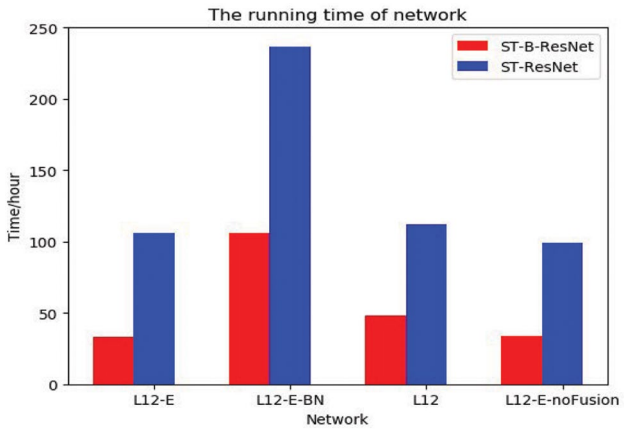

Fig. 11 The running time on TaxiBJ

Table 3 Comparison among different models on BikeNYC

Model	RMSE
ARIMA	10.07
SARIMA	10.56
VAR	9.92
DeepST-C	8.39
DeepST-CP	7.64
DeepST-CPT	7.56
DeepST-CPTM	7.43
ST-ResNet[4 residual blocks]	6.33
ST-B-ResNet [ours, 4 residual blocks]	**3.58**

6.3 Results on BikeNYC

In the following experiments, four bottleneck residual blocks are used in the ST-B-ResNet model, and the metadata is considered as auxiliary features like DeepST [11]. Table 3 shows the results based on BikeNYC. The RMSE of ST-B-ResNet is 3.58, which have a huge improvement compared to other models. Figure 12 shows the number of parameter of ST-B-ResNet and ST-ResNet. As can be seen, the scale of parameters of ST-B-ResNet is a ninth of ST-ResNet's. Figure 13 shows the running time of ST-B-ResNet and ST-ResNet. The ST-B-ResNet just need about 1/6 training time of ST-ResNet. The experiment results show that the ST-B-ResNet model saves computational cost and effectively improves the prediction accuracy.

7 Conclusion

In this paper, a multi-source data-based deep ST-B-ResNet prediction model has been proposed to predict the flow of crowds in every region of a city, and a Bernoulli-RBM-based

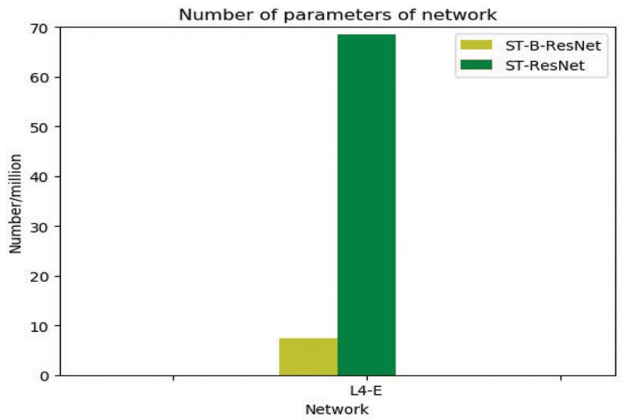

Fig. 12 Number of parameters on BikeNYC of the network

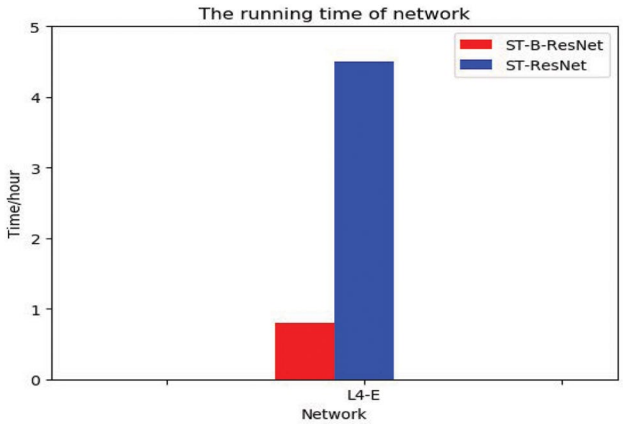

Fig. 13 The running time on BikeNYC

data reconstruction mechanism is employed to reconstruct crowd flows data for optimizing the prediction model. The deep ST-B-ResNet model is evaluated based on two types of crowd flows data in Beijing and New York. The results show that the deep ST-B-ResNet model achieves better prediction results than other baseline models. The prediction accuracy of the proposed model is beyond other baseline models. Deep ST-B-ResNet can handle effectively the tasks of crowd flows prediction. In the future, we will consider adopting the DenseNet model to improve the prediction accuracy of crowds flow. DenseNet establishes a dense connection between all the previous layers and the later layers. In this way, it implements feature reuse. In theory, DenseNet can achieve better prediction performance with fewer parameters and computational costs than ResNet.

Acknowledgements This work was supported by the National Nature Science Foundation of China (Nos. 61862014, 61902086, 61966009), Guangxi Natural Science Foundation of China (2018GXNSFBA281142), Innovation Project of Guangxi Department of Science and Technology (AD18281054), Open Foundation of State key Laboratory of Networking and Switching Technology in China (SKLNST-2018-1-04), Guangxi Key Laboratory of Trusted Software (kx201718).

References

1. Chithaluru P, Al-Turjman F, Kumar M, Stephan T (2020) I-areor: an energy-balanced clustering protocol for implementing green iot in smart cities. Sustain Cities Soc 61:102254
2. Wang B, Yan Z, Lu J, Zhang G, Li T (2018) Explore uncertainty in residual networks for crowds flow prediction. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, pp 1–7
3. Polson NG, Sokolov VO (2017) Deep learning for short-term traffic flow prediction. Transp Res Part C 79:1–17
4. Zhang Y, Zheng D, Qi R, Li X, Yi, Li T (2018) Predicting citywide crowd flows using deep spatio-temporal residual networks. Artif Intell 259:147–166
5. Zhang Y, Zheng Qi D (2016) Deep spatio-temporal residual networks for citywide crowd flows prediction. Proc AAI Conf Artif Intell 61:1655–1661
6. Sato D, Matsubayashi T, Nagano S, Toda H (2019) People flow prediction by multi-agent simulator. In: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, pp 1–4
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 1–10
8. Liu Y, Liu Z, Jia R (2019) Deeppf: a deep learning based architecture for metro passenger flow prediction. Transp Res Part C 101:18–34
9. Jia W, Tan Y, Liu L, Li J, Zhang H, Zhao K (2019) Hierarchical prediction based on two-level gaussian mixture model clustering for bike-sharing system. Knowl-Based Syst 178:84–97
10. Zhang F, Chen Z, Cui Y, Guo ZhuY (2020) Deep learning architecture for short-term passenger flow forecasting in urban rail transit. IEEE TranS Intell Transp Syst. https://doi.org/10.1109/TITS.2020.3000761
11. Lin Z, Feng J, Lu Z, Li Y, Jin D (2019) Deepstn+: context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 1020–1027
12. Han T, Nijkamp E, Fang X, Hill M, Zhu S-C, Wu YN (2019) Divergence triangle for joint training of generator model, energy-based model, and inferential model. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 8670–8679
13. Vrábel Pořízka P, Kaiser J (2020) Restricted Boltzmann machine method for dimensionality reduction of large spectroscopic data. Spectrochim Acta Part B 167:105849
14. Salakhutdinov R, Mnih A, Hinton G (2007) Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on Machine learning, pp 791–798
15. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
16. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. European conference on computer vision. Springer, Berlin, pp 630–645
17. Zheng Y (2015) Methodologies for cross-domain data fusion: an overview. IEEE Trans Big Data 1(1):16–34
18. Torres Joaquin J (2015) Boltzmann machine. Springer, Berlin, pp 417–421

19. Zhang W, Tang P, Zhao L (2019) Remote sensing image scene classification using cnn-capsnet. Remote Sens 11(5):494
20. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. Proc Int Conf Mach Learn 37:448–456
21. Tieleman T (2008) Training restricted boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th international conference on Machine learning, pp 1064–1071
22. Banner R, Hubara I, Hoffer E, Soudry D (2018) Scalable methods for 8-bit training of neural networks. Advances in neural information processing systems. Springer, Berlin, pp 5145–5153
23. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization, the 3rd International Conference on Learning Representations, pp 1–15
24. Chollet et al (2015) keras.GitHub repository. https://github.com/fchollet/keras. Accessed on 2015
25. Theano Development Team (2016) Theano: a python framework for fast computation of mathematical expressions. Available: http://arxiv.org/abs/1605.02688