Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction*

Junbo Zhang,[1] **Yu Zheng,**[1,2,3,4†] **Dekang Qi**[2,1]
[1]Microsoft Research, Beijing, China
[2]School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China
[3]School of Computer Science and Technology, Xidian University, China
[4]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
{junbo.zhang, yuzheng}@microsoft.com, dekangqi@outlook.com

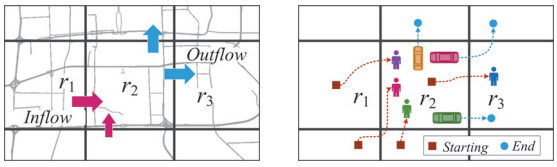
Abstract

Forecasting the flow of crowds is of great importance to traffic management and public safety, and very challenging as it is affected by many complex factors, such as inter-region traffic, events, and weather. We propose a deep-learning-based approach, called ST-ResNet, to *collectively* forecast the inflow and outflow of crowds in each and every region of a city. We design an end-to-end structure of ST-ResNet based on unique properties of spatio-temporal data. More specifically, we employ the residual neural network framework to model the temporal closeness, period, and trend properties of crowd traffic. For each property, we design a branch of residual convolutional units, each of which models the spatial properties of crowd traffic. ST-ResNet learns to dynamically aggregate the output of the three residual neural networks based on data, assigning different weights to different branches and regions. The aggregation is further combined with external factors, such as weather and day of the week, to predict the final traffic of crowds in each and every region. Experiments on two types of crowd flows in Beijing and New York City (NYC) demonstrate that the proposed ST-ResNet outperforms six well-known methods.

Introduction

Predicting crowd flows in a city is of great importance to traffic management and public safety (Zheng et al. 2014). For instance, massive crowds of people streamed into a strip region at the 2015 New Year's Eve celebrations in Shanghai, resulting in a catastrophic stampede that killed 36 people. In mid-July of 2016, hundreds of "Pokemon Go" players ran through New York City's Central Park in hopes of catching a particularly rare digital monster, leading to a dangerous stampede there. If one can predict the crowd flow in a region, such tragedies can be mitigated or prevented by utilizing emergency mechanisms, such as conducting traffic control, sending out warnings, or evacuating people, in advance.

In this paper, we predict two types of crowd flows (Zhang et al. 2016): inflow and outflow, as shown in Figure 1(a). Inflow is the total traffic of crowds entering a region from other places during a given time interval. Outflow denotes the total traffic of crowds leaving a region for other places during a given time interval. Both flows track the transition of crowds between regions. Knowing them is very beneficial for risk assessment and traffic management. Inflow/outflow can be measured by the number of pedestrians, the number of cars driven nearby roads, the number of people traveling on public transportation systems (*e.g.*, metro, bus), or *all of them together* if data is available. Figure 1(b) presents an example. We can use mobile phone signals to measure the number of pedestrians, showing that the inflow and outflow of r_2 are $(3, 1)$ respectively. Similarly, using the GPS trajectories of vehicles, two types of flows are $(0, 3)$ respectively.



(a) Inflow and outflow (b) Measurement of flows

Figure 1: Crowd flows in a region

Simultaneously forecasting the inflow and outflow of crowds in each region of a city, however, is very challenging, affected by the following three complex factors:

1. Spatial dependencies. The inflow of Region r_2 (shown in Figure 1(a)) is affected by outflows of nearby regions (like r_1) as well as distant regions. Likewise, the outflow of r_2 would affect inflows of other regions (*e.g.*, r_3). The inflow of region r_2 would affect its own outflow as well.
2. Temporal dependencies. The flow of crowds in a region is affected by recent time intervals, both near and far. For instance, a traffic congestion occurring at 8am will affect that of 9am. In addition, traffic conditions during morning rush hours may be similar on consecutive workdays, repeating every 24 hours. Furthermore, morning rush hours may gradually happen later as winter comes. When the temperature gradually drops and the sun rises later in the day, people get up later and later.
3. External influence. Some external factors, such as weather conditions and events may change the flow of crowds tremendously in different regions of a city.

*This research was supported by NSFC (Nos. 61672399, U1401258), and the 973 Program (No. 2015CB352400).

†Correspondence author. This work was done when the third author was an intern at Microsoft Research.

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

To tackle these challenges, we propose a deep spatio-temporal residual network (ST-ResNet) to *collectively* predict inflow and outflow of crowds in every region. Our contributions are four-fold:

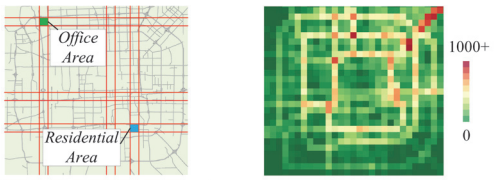
- ST-ResNet employs convolution-based residual networks to model nearby and distant spatial dependencies between any two regions in a city, while ensuring the model's prediction accuracy is not comprised by the deep structure of the neural network.
- We summarize the temporal properties of crowd flows into three categories, consisting of temporal closeness, period, and trend. ST-ResNet uses three residual networks to model these properties, respectively.
- ST-ResNet dynamically aggregates the output of the three aforementioned networks, assigning different weights to different branches and regions. The aggregation is further combined with external factors (*e.g.*, weather).
- We evaluate our approach using Beijing taxicabs' trajectories and meteorological data, and NYC bike trajectory data. The results demonstrate the advantages of our approach compared with 6 baselines.

Preliminaries

In this section, we briefly revisit the crowd flows prediction problem (Zhang et al. 2016; Hoang, Zheng, and Singh 2016) and introduce deep residual learning (He et al. 2016).

Formulation of Crowd Flows Problem

Definition 1 (Region (Zhang et al. 2016)) *There are many definitions of a location in terms of different granularities and semantic meanings. In this study, we partition a city into an $I \times J$ grid map based on the longitude and latitude where a grid denotes a region, as shown in Figure 2(a).*



(a) Grid-based map segmentation (b) Inflow matrix

Figure 2: Regions in Beijing: (a) Grid-based map segmentation; (b) inflows in every region of Beijing

Definition 2 (Inflow/outflow (Zhang et al. 2016)) *Let $\mathbb{P}$ be a collection of trajectories at the t^{th} time interval. For a grid (i,j) that lies at the i^{th} row and the j^{th} column, the inflow and outflow of the crowds at the time interval t are defined respectively as*

$$x_t^{in,i,j} = \sum_{Tr \in \mathbb{P}} |\{k > 1 | g_{k-1} \notin (i,j) \wedge g_k \in (i,j)\}|$$

$$x_t^{out,i,j} = \sum_{Tr \in \mathbb{P}} |\{k \geq 1 | g_k \in (i,j) \wedge g_{k+1} \notin (i,j)\}|$$

where $Tr : g_1 \rightarrow g_2 \rightarrow \cdots \rightarrow g_{|Tr|}$ is a trajectory in $\mathbb{P}$, and g_k is the geospatial coordinate; $g_k \in (i,j)$ means the point g_k lies within grid (i,j), and vice versa; $|\cdot|$ denotes the cardinality of a set.

At the t^{th} time interval, inflow and outflow in all $I \times J$ regions can be denoted as a tensor $\mathbf{X}_t \in \mathbb{R}^{2 \times I \times J}$ where $(\mathbf{X}_t)_{0,i,j} = x_t^{in,i,j}$, $(\mathbf{X}_t)_{1,i,j} = x_t^{out,i,j}$. The inflow matrix is shown in Figure 2(b).

Formally, for a dynamical system over a spatial region represented by a $I \times J$ grid map, there are 2 types of flows in each grid over time. Thus, the observation at any time can be represented by a tensor $\mathbf{X} \in \mathbb{R}^{2 \times I \times J}$.

Problem 1 *Given the historical observations $\{\mathbf{X}_t | t = 0, \cdots, n-1\}$, predict $\mathbf{X}_n$.*

Deep Residual Learning

Deep residual learning (He et al. 2015) allows convolution neural networks to have a super deep structure of 100 layers, even over-1000 layers. And this method has shown state-of-the-art results on multiple challenging recognition tasks, including image classification, object detection, segmentation and localization (He et al. 2015).

Formally, a residual unit with an identity mapping (He et al. 2016) is defined as:

$$\mathbf{X}^{(l+1)} = \mathbf{X}^{(l)} + \mathcal{F}(\mathbf{X}^{(l)}) \quad (1)$$

where $\mathbf{X}^{(l)}$ and $\mathbf{X}^{(l+1)}$ are the input and output of the l^{th} residual unit, respectively; $\mathcal{F}$ is a residual function, *e.g.*, a stack of two 3×3 convolution layers in (He et al. 2015). The central idea of the residual learning is to learn the additive residual function $\mathcal{F}$ with respect to $\mathbf{X}^{(l)}$ (He et al. 2016).

Deep Spatio-Temporal Residual Networks

Figure 3 presents the architecture of ST-ResNet, which is comprised of four major components modeling tempo-

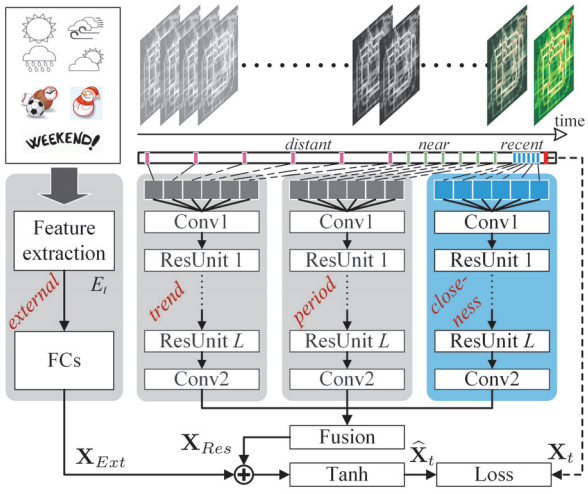
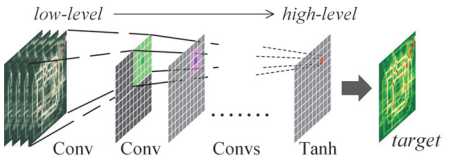


Figure 3: ST-ResNet architecture. Conv: Convolution; ResUnit: Residual Unit; FC: Fully-connected.

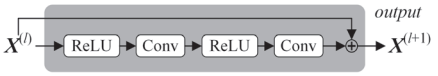
ral *closeness*, *period*, *trend*, and *external* influence, respectively. As illustrated in the top-right part of Figure 3, we first turn Inflow and outflow throughout a city at each time interval into a 2-channel image-like matrix respectively, using the approach introduced in Definitions 1 and 2. We then divide the time axis into three fragments, denoting recent time, near history and distant history. The 2-channel flow matrices of intervals in each time fragment are then fed into the first three components separately to model the aforementioned three temporal properties: *closeness*, *period* and *trend*, respectively. The first three components share the same network structure with a convolutional neural network followed by a Residual Unit sequence. Such structure captures the spatial dependency between nearby and distant regions. In the *external* component, we manually extract some features from external datasets, such as weather conditions and events, feeding them into a two-layer fully-connected neural network. The outputs of the first three components are fused as $\mathbf{X}_{Res}$ based on parameter matrices, which assign different weights to the results of different components in different regions. $\mathbf{X}_{Res}$ is further integrated with the output of the external component $\mathbf{X}_{Ext}$. Finally, the aggregation is mapped into $[-1, 1]$ by a Tanh function, which yields a faster convergence than the standard logistic function in the process of back-propagation learning (LeCun et al. 2012).

Structures of the First Three Components

The first three components (*i.e. closeness*, *period*, *trend*) share the same network structure, which is composed of two sub-components: convolution and residual unit, as shown in Figure 4.



(a) Convolutions



(b) Residual Unit

Figure 4: Convolution and residual unit

Convolution. A city usually has a very large size, containing many regions with different distances. Intuitively, the flow of crowds in nearby regions may affect each other, which can be effectively handled by the convolutional neural network (CNN) that has shown its powerful ability to hierarchically capture the spatial structural information (LeCun et al. 1998). In addition, subway systems and highways connect two locations with a far distance, leading to the dependency between distant regions. In order to capture the spatial dependency of any region, we need to design a CNN with many layers because one convolution only accounts for spatial near dependencies, limited by the size of their kernels. The same problem also has been found in the video sequence generating task where the input and output have the same resolution (Mathieu, Couprie, and LeCun 2015). Several methods have been introduced to avoid the loss of resolution brought about by subsampling while preserving distant dependencies (Long, Shelhamer, and Darrell 2015). Being different from the classical CNN, we do not use subsampling, but only convolutions (Jain et al. 2007). As shown in Figure 4(a), there are three multiple levels of feature maps that are connected with a few convolutions. We find that a node in the high-level feature map depends on nine nodes of the middle-level feature map, those of which depend on all nodes in the lower-level feature map (*i.e.* input). It means one convolution naturally captures spatial near dependencies, and a stack of convolutions can further capture distant even citywide dependencies.

The *closeness* component of Figure 3 adopts a few 2-channel flows matrices of intervals in the recent time to model temporal *closeness* dependence. Let the recent fragment be $[\mathbf{X}_{t-l_c}, \mathbf{X}_{t-(l_c-1)}, \cdots, \mathbf{X}_{t-1}]$, which is also known as the *closeness* dependent sequence. We first concatenate them along with the first axis (*i.e.* time interval) as one tensor $\mathbf{X}_c^{(0)} \in \mathbb{R}^{2l_c \times I \times J}$, which is followed by a convolution (*i.e.* `Conv1` shown in Figure 3) as:

$$\mathbf{X}_c^{(1)} = f\left(W_c^{(1)} * \mathbf{X}_c^{(0)} + b_c^{(1)}\right) \quad (2)$$

where $*$ denotes the convolution[1]; f is an activation function, *e.g.* the rectifier $f(z) := \max(0, z)$ (Krizhevsky, Sutskever, and Hinton 2012); $W_c^{(1)}, b_c^{(1)}$ are the learnable parameters in the first layer.

Residual Unit. It is a well-known fact that very deep convolutional networks compromise training effectiveness though the well-known activation function (*e.g.* ReLU) and regularization techniques are applied (Ioffe and Szegedy 2015; Krizhevsky, Sutskever, and Hinton 2012; Nair and Hinton 2010). On the other hand, we still need a very deep network to capture very large citywide dependencies. For a typical crowd flows data, assume that the input size is 32×32, and the kernel size of convolution is fixed to 3×3, if we want to model citywide dependencies (*i.e.*, each node in high-level layer depends on all nodes of the input), it needs more than 15 consecutive convolutional layers. To address this issue, we employ residual learning (He et al. 2015) in our model, which have been demonstrated to be very effective for training super deep neural networks of over-1000 layers.

In our ST-ResNet (see Figure 3), we stack L residual units upon `Conv1` as follows,

$$\mathbf{X}_c^{(l+1)} = \mathbf{X}_c^{(l)} + \mathcal{F}(\mathbf{X}_c^{(l)}; \theta_c^{(l)}), l = 1, \cdots, L \quad (3)$$

where $\mathcal{F}$ is the residual function (*i.e.* two combinations of "ReLU + Convolution", see Figure 4(b)), and $\theta^{(l)}$ includes all learnable parameters in the l^{th} residual unit. We also attempt *Batch Normalization* (BN) (Ioffe and Szegedy 2015)

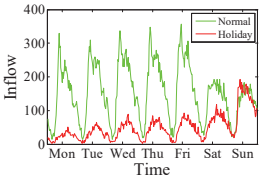
[1]To make the input and output have the same size (*i.e.* $I \times J$) in a convolutional operator, we employ a border-mode which allows a filter to go outside the border of an input, padding each area outside the border with a zero.

that is added before ReLU. On top of the L^{th} residual unit, we append a convolutional layer (*i.e.* `Conv2` shown in Figure 3). With 2 convolutions and L residual units, the output of the *closeness* component of Figure 3 is $\mathbf{X}_c^{(L+2)}$.

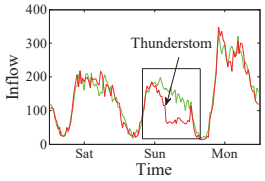
Likewise, using the above operations, we can construct the *period* and *trend* components of Figure 3. Assume that there are l_p time intervals from the period fragment and the period is p. Therefore, the *period* dependent sequence is $[\mathbf{X}_{t-l_p\cdot p}, \mathbf{X}_{t-(l_p-1)\cdot p}, \cdots, \mathbf{X}_{t-p}]$. With the convolutional operation and L residual units like in Eqs. 2 and 3, the output of the *period* component is $\mathbf{X}_p^{(L+2)}$. Meanwhile, the output of the *trend* component is $\mathbf{X}_q^{(L+2)}$ with the input $[\mathbf{X}_{t-l_q\cdot q}, \mathbf{X}_{t-(l_q-1)\cdot q}, \cdots, \mathbf{X}_{t-q}]$ where l_q is the length of the *trend* dependent sequence and q is the trend span. Note that p and q are actually two different types of periods. In the detailed implementation, p is equal to one-day that describes daily periodicity, and q is equal to one-week that reveals the weekly trend.

The Structure of the External Component

Traffic flows can be affected by many complex external factors, such as weather and event. Figure 5(a) shows that crowd flows during holidays (Chinese Spring Festival) can be significantly different from the flows during normal days. Figure 5(b) shows that heavy rain sharply reduces the crowd flows at Office Area compared to the same day of the latter week. Let E_t be the feature vector that represents these external factors at predicted time interval t. In our implementation, we mainly consider weather, holiday event, and metadata (*i.e.* DayOfWeek, Weekday/Weekend). The details are introduced in Table 1. To predict flows at time interval t, the holiday event and metadata can be directly obtained. However, the weather at future time interval t is unknown. Instead, one can use the forecasting weather at time interval t or the approximate weather at time interval $t-1$. Formally, we stack two fully-connected layers upon E_t, the first layer can be viewed as an embedding layer for each sub-factor followed by an activation. The second layer is used to map low to high dimensions that have the same shape as $\mathbf{X}_t$. The output of the *external* component of Figure 3 is denoted as $\mathbf{X}_{Ext}$ with the parameters θ_{Ext}.



(a) Feb 8-14 (red), Feb 15-21 (green), 2016



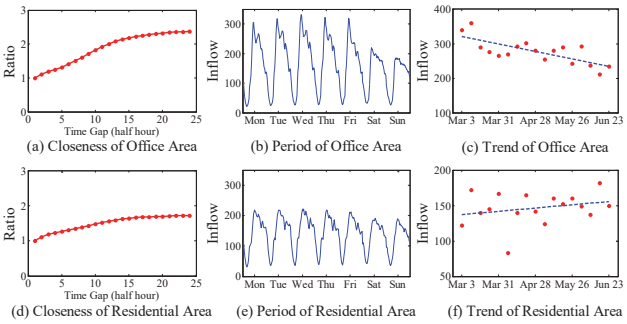
(b) Aug 10-12 (red), Aug 17-19 (green), 2013

Figure 5: Effects of holidays and weather in Office Area of Beijing (the region is shown in Figure 2(a)).

Fusion

In this section, we discuss how to fuse four components of Figure 3. We first fuse the first three components with a parametric-matrix-based fusion method, which is then further combined with the *external* component.

Figures 6(a) and (d) show the ratio curves using Beijing trajectory data presented in Table 1 where x-axis is time gap between two time intervals and y-axis is the average ratio value between arbitrary two inflows that have the same time gap. The curves from two different regions all show an empirical temporal correlation in time series, namely, inflows of recent time intervals are more relevant than ones of distant time intervals, which implies temporal *closeness*. The two curves have different shapes, which demonstrates that different regions may have different characteristics of closeness. Figures 6(b) and (e) depict inflows at all time intervals of 7 days. We can see the obvious *daily periodicity* in both regions. In Office Area, the peak values on weekdays are much higher than ones on weekends. Residential Area has similar peak values for both weekdays and weekends. Figures 6(c) and (f) describe inflows at a certain time interval (9:00pm-9:30pm) of Tuesday from March 2015 and June 2015. As time goes by, the inflow progressively decreases in Office Area, and increases in Residential Area. It shows the different trends in different regions. In summary, inflows of two regions are all affected by *closeness*, *period*, and *trend*, but the degrees of influence may be very different. We also find the same properties in other regions as well as their outflows.



(a) Closeness of Office Area (b) Period of Office Area (c) Trend of Office Area

(d) Closeness of Residential Area (e) Period of Residential Area (f) Trend of Residential Area

Figure 6: Temporal dependencies (Office Area and Residential Area are shown in Figure 2(a))

Above all, the different regions are all affected by *closeness*, *period* and *trend*, but the degrees of influence may be different. Inspired by these observations, we propose a parametric-matrix-based fusion method.

Parametric-matrix-based fusion. We fuse the first three components (*i.e. closeness*, *period*, *trend*) of Figure 3 as follows

$$\mathbf{X}_{Res} = \mathbf{W}_c \circ \mathbf{X}_c^{(L+2)} + \mathbf{W}_p \circ \mathbf{X}_p^{(L+2)} + \mathbf{W}_q \circ \mathbf{X}_q^{(L+2)} \quad (4)$$

where $\circ$ is Hadamard product (*i.e.*, element-wise multiplication), $\mathbf{W}_c$, $\mathbf{W}_p$ and $\mathbf{W}_q$ are the learnable parameters that adjust the degrees affected by closeness, period and trend, respectively.

Fusing the external component. We here directly merge the output of the first three components with that of the *external* component, as shown in Figure 3. Finally, the predicted value at the t^{th} time interval, denoted by $\widehat{\mathbf{X}}_t$, is defined as

$$\widehat{\mathbf{X}}_t = \tanh(\mathbf{X}_{Res} + \mathbf{X}_{Ext}) \tag{5}$$

where $\tanh$ is a hyperbolic tangent that ensures the output values are between -1 and 1.

Our ST-ResNet can be trained to predict $\mathbf{X}_t$ from three sequences of flow matrices and external factor features by minimizing mean squared error between the predicted flow matrix and the true flow matrix:

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

where θ are all learnable parameters in the ST-ResNet.

Algorithm and Optimization

Algorithm 1 outlines the ST-ResNet training process. We first construct the training instances from the original sequence data (lines 1-6). Then, ST-ResNet is trained via back-propagation and Adam (Kingma and Ba 2014) (lines 7-11).

Algorithm 1: ST-ResNet Training Algorithm

Input: Historical observations: $\{\mathbf{X}_0, \cdots, \mathbf{X}_{n-1}\}$;
external features: $\{E_0, \cdots, E_{n-1}\}$;
lengths of *closeness*, *period*, *trend* sequences: l_c, l_p, l_q;
peroid: p; trend span: q.
Output: Learned ST-ResNet model
// construct training instances
1 $\mathcal{D} \longleftarrow \emptyset$
2 **for** *all available time interval* $t (1 \leq t \leq n-1)$ **do**
3 $\quad \mathcal{S}_c = [\mathbf{X}_{t-l_c}, \mathbf{X}_{t-(l_c-1)}, \cdots, \mathbf{X}_{t-1}]$
4 $\quad \mathcal{S}_p = [\mathbf{X}_{t-l_p \cdot p}, \mathbf{X}_{t-(l_p-1)\cdot p}, \cdots, \mathbf{X}_{t-p}]$
5 $\quad \mathcal{S}_q = [\mathbf{X}_{t-l_q \cdot q}, \mathbf{X}_{t-(l_q-1)\cdot q}, \cdots, \mathbf{X}_{t-q}]$
$\quad$ *// $\mathbf{X}_t$ is the target at time t*
6 $\quad$ put an training instance $(\{\mathcal{S}_c, \mathcal{S}_p, \mathcal{S}_q, E_t\}, \mathbf{X}_t)$ into $\mathcal{D}$
// train the model
7 initialize all learnable parameters θ in ST-ResNet
8 **repeat**
9 $\quad$ randomly select a batch of instances $\mathcal{D}_b$ from $\mathcal{D}$
10 $\quad$ find θ by minimizing the objective (6) with $\mathcal{D}_b$
11 **until** *stopping criteria is met*

Experiments

Settings

Datasets. We use two different sets of data as shown in Table 1. Each dataset contains two sub-datasets: trajectories and weather, as detailed as follows.

- **TaxiBJ**: Trajectoriy data is the taxicab 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. Using Definition 2, we obtain two types of crowd flows. We choose data from the last four weeks as the testing data, and all data before that as training data.
- **BikeNYC**: Trajectory data is taken from the NYC Bike system in 2014, from Apr. 1st to Sept. 30th. Trip data includes: trip duration, starting and ending station IDs, and start and end times. Among the data, the last 10 days are chosen as testing data, and the others as training data.

Table 1: Datasets (holidays include adjacent weekends).

Dataset	TaxiBJ	BikeNYC
Data type	Taxi GPS	Bike rent
Location	Beijing	New York
Time Span	7/1/2013 - 10/30/2013 3/1/2014 - 6/30/2014 3/1/2015 - 6/30/2015 11/1/2015 - 4/10/2016	4/1/2014 - 9/30/2014
Time interval	30 minutes	1 hour
Gird map size	(32, 32)	(16, 8)
Trajectory data		
Average sampling rate (s)	~ 60	\
# taxis/bikes	34,000+	6,800+
# available time interval	22,459	4,392
External factors (holidays and meteorology)		
# holidays	41	20
Weather conditions	16 types (*e.g.*, Sunny, Rainy)	\
Temperature / °C	$[-24.6, 41.0]$	\
Wind speed / mph	$[0, 48.6]$	\

Baselines. We compare our ST-ResNet with the following 6 baselines:

- **HA**: We predict inflow and outflow of crowds by the average value of historical inflow and outflow in the corresponding periods, *e.g.*, 9:00am-9:30am on Tuesday, its corresponding periods are all historical time intervals from 9:00am to 9:30am on all historical Tuesdays.
- **ARIMA**: Auto-Regressive Integrated Moving Average (ARIMA) is a well-known model for understanding and predicting future values in a time series.
- **SARIMA**: Seasonal ARIMA.
- **VAR**: Vector Auto-Regressive (VAR) is a more advanced spatio-temporal model, which can capture the pairwise relationships among all flows, and has heavy computational costs due to the large number of parameters.
- **ST-ANN**: It first extracts spatial (nearby 8 regions' values) and temporal (8 previous time intervals) features, then fed into an artificial neural network.
- **DeepST** (Zhang et al. 2016): a deep neural network (DNN)-based prediction model for spatio-temporal data, which shows state-of-the-art results on crowd flows prediction. It has 4 variants, including DeepST-C, DeepST-CP, DeepST-CPT, and DeepST-CPTM, which focus on different temporal dependencies and external factors.

Preprocessing. In the output of the ST-ResNet, we use $\tanh$ as our final activation (see Eq. 5), whose range is between -1 and 1. Here, we use the Min-Max normalization method to scale the data into the range $[-1, 1]$. In the evaluation, we re-scale the predicted value back to the normal values, compared with the groundtruth. For external factors, we use one-hot coding to transform metadata (*i.e.*, DayOfWeek, Weekend/Weekday), holidays and weather conditions into binary

vectors, and use Min-Max normalization to scale the Temperature and Wind speed into the range $[0, 1]$.

Hyperparameters. The python libraries, including Theano (Theano Development Team 2016) and Keras (Chollet 2015), are used to build our models. The convolutions of `Conv1` and all residual units use 64 filters of size 3×3, and `Conv2` uses a convolution with 2 filters of size 3×3. The batch size is 32. We select 90% of the training data for training each model, and the remaining 10% is chosen as the validation set, which is used to early-stop our training algorithm for each model based on the best validation score. Afterwards, we continue to train the model on the full training data for a fixed number of epochs (*e.g.*, 10, 100 epochs). There are 5 extra hyperparamers in our ST-ResNet, of which p and q are empirically fixed to one-day and one-week, respectively. For lengths of the three dependent sequences, we set them as: $l_c \in \{3, 4, 5\}, l_p \in \{1, 2, 3, 4\}, l_q \in \{1, 2, 3, 4\}$.

Evaluation Metric: We measure our method by Root Mean Square Error (RMSE) as

$$RMSE = \sqrt{\frac{1}{z}\sum_{i}(x_i - \hat{x}_i)^2} \tag{7}$$

where $\hat{x}$ and x are the predicted value and ground thuth, respectively; z is the number of all predicted values.

Results on TaxiBJ

We first give the comparison with 6 other models on TaxiBJ, as shown in Table 2. We give 7 variants of ST-ResNet with different layers and different factors. Taking L12-E for example, it considers all available external factors and has 12 residual units, each of which is comprised of two convolutional layers. We observe that all of these 7 models are better than 6 baselines. Comparing with the previous state-of-the-art models, L12-E-BN reduces error to 16.69, which significantly improves accuracy.

Table 2: Comparison among different methods 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 [ours]	
L2-E	2 residual units + E	17.67
L4-E	4 residual units + E	17.51
L12-E	12 residual units + E	16.89
L12-E-BN	L12-E with BN	**16.69**
L12-single-E	12 residual units (1 conv) + E	17.40
L12	12 residual units	17.00
L12-E-noFusion	12 residual units + E without fusion	17.96

Effects of Different Components. Let **L12-E** be the compared model.

- *Number of residual units*: Results of L2-E, L4-E and L12-E show that RMSE decreases as the number of residual units increases. Using residual learning, the deeper the network is, the more accurate the results will be.
- *Internal structure of residual unit*: We attempt three different types of residual units. L12-E adopts the standard *Residual Unit* (see Figure 4(b)). Compared with L12-E, *Residual Unit* of L12-single-E only contains 1 ReLU followed by 1 convolution, and *Residual Unit* of L12-E-BN added two batch normalization layers, each of which is inserted before ReLU. We observe that L12-single-E is worse than L12-E, and L12-E-BN is the best, demonstrating the effectiveness of *batch normalization*.
- *External factors*: L12-E considers the external factors, including meteorology data, holiday events and metadata. If not, the model is degraded as L12. The results indicate that L12-E is better than L12, pointing out that external factors are always beneficial.
- *Parametric-matrix-based fusion*: Being different with L12-E, L12-E-noFusion donot use parametric-matrix-based fusion (see Eq. 4). Instead, L12-E-noFusion use a straightforward method for fusing, *i.e.*, $\mathbf{X}_c^{(L+2)} + \mathbf{X}_p^{(L+2)} + \mathbf{X}_q^{(L+2)}$. It shows the error greatly increases, which demonstrates the effectiveness of our proposed parametric-matrix-based fusion.

Results on BikeNYC

Table 3 shows the results of our model and other baselines on BikeNYC. Being different from TaxiBJ, BikeNYC consists of two different types of crowd flows, including new-flow and end-flow (Hoang, Zheng, and Singh 2016). Here, we adopt a total of 4-residual-unit ST-ResNet, and consider the metadata as external features like DeepST (Zhang et al. 2016). ST-ResNet has relatively from 14.8% up to 37.1% lower RMSE than these baselines, demonstrating that our proposed model has good generalization performance on other flow prediction tasks.

Table 3: Comparisons with baselines on BikeNYC. The results of ARIMA, SARIMA, VAR and 4 DeepST variants are taken from (Zhang et al. 2016).

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 [ours, 4 residual units]	**6.33**

Related Work

Crowd Flow Prediction. There are some previously published works on predicting an individual's movement based on their location history (Fan et al. 2015; Song et al. 2014). They mainly forecast millions, even billions, of individuals' mobility traces rather than the aggregated crowd flows in a region. Such a task may require huge computational resources, and it is not always necessary for the application

scenario of public safety. Some other researchers aim to predict travel speed and traffic volume on the road (Abadi, Rajabioun, and Ioannou 2015; Silva, Kang, and Airoldi 2015; Xu et al. 2014). Most of them are predicting single or multiple road segments, rather than citywide ones. Recently, researchers have started to focus on city-scale traffic flow prediction (Hoang, Zheng, and Singh 2016; Li et al. 2015). Both work are different from ours where the proposed methods naturally focus on the individual region not the city, and they do not partition the city using a grid-based method which needs a more complex method to find irregular regions first.

Deep Learning. CNNs have been successfully applied to various problems, especially in the field of computer vision (Krizhevsky, Sutskever, and Hinton 2012). Residual learning (He et al. 2015) allows such networks to have a very super deep structure. Recurrent neural networks (RNNs) have been used successfully for sequence learning tasks (Sutskever, Vinyals, and Le 2014). The incorporation of long short-term memory (LSTM) enables RNNs to learn long-term temporal dependency. However, both kinds of neural networks can only capture spatial or temporal dependencies. Recently, researchers combined above networks and proposed a convolutional LSTM network (Xingjian et al. 2015) that learns spatial and temporal dependencies simultaneously. Such a network cannot model very long-range temporal dependencies (*e.g.*, period and trend), and training becomes more difficult as depth increases.

In our previous work (Zhang et al. 2016), a general prediction model based on DNNs was proposed for spatio-temporal data. In this paper, to model a specific spatio-temporal prediction (*i.e.* citywide crowd flows) effectively, we mainly propose employing the residual learning and a parametric-matrix-based fusion mechanism. A survey on data fusion methodologies can be found at (Zheng 2015).

Conclusion and Future Work

We propose a novel deep-learning-based model for forecasting the flow of crowds in each and every region of a city, based on historical trajectory data, weather and events. We evaluate our model on two types of crowd flows in Beijing and NYC, achieving performances which are significantly beyond 6 baseline methods, confirming that our model is better and more applicable to the crowd flow prediction. The code and datasets have been released at: https://www.microsoft.com/en-us/research/publication/deep-spatio-temporal-residual-networks-for-citywide-crowd-flows-prediction.

In the future, we will consider other types of flows (*e.g.*, taxi/truck/bus trajectory data, phone signals data, metro card swiping data), and use all of them to generate more types of flow predictions, and *collectively* predict all of these flows with an appropriate fusion mechanism.

References

Abadi, A.; Rajabioun, T.; and Ioannou, P. A. 2015. Traffic flow prediction for road transportation networks with limited traffic data. *IEEE Transactions on Intelligent Transportation Systems* 16(2):653–662.

Chollet, F. 2015. Keras. https://github.com/fchollet/keras.

Fan, Z.; Song, X.; Shibasaki, R.; and Adachi, R. 2015. Citymomentum: an online approach for crowd behavior prediction at a citywide level. In *ACM UbiComp*, 559–569. ACM.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. In *IEEE CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Identity mappings in deep residual networks. In *ECCV*.

Hoang, M. X.; Zheng, Y.; and Singh, A. K. 2016. Forecasting citywide crowd flows based on big data. In *ACM SIGSPATIAL*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 448–456.

Jain, V.; Murray, J. F.; Roth, F.; Turaga, S.; Zhigulin, V.; Briggman, K. L.; Helmstaedter, M. N.; Denk, W.; and Seung, H. S. 2007. Supervised learning of image restoration with convolutional networks. In *ICCV*, 1–8. IEEE.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

LeCun, Y. A.; Bottou, L.; Orr, G. B.; and Müller, K.-R. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer.

Li, Y.; Zheng, Y.; Zhang, H.; and Chen, L. 2015. Traffic prediction in a bike-sharing system. In *ACM SIGSPATIAL*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, 3431–3440.

Mathieu, M.; Couprie, C.; and LeCun, Y. 2015. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Silva, R.; Kang, S. M.; and Airoldi, E. M. 2015. Predicting traffic volumes and estimating the effects of shocks in massive transportation systems. *Proceedings of the National Academy of Sciences* 112(18):5643–5648.

Song, X.; Zhang, Q.; Sekimoto, Y.; and Shibasaki, R. 2014. Prediction of human emergency behavior and their mobility following large-scale disaster. In *ACM SIGKDD*, 5–14. ACM.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3104–3112.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

Xingjian, S.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-k.; and WOO, W.-c. 2015. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 802–810.

Xu, Y.; Kong, Q.-J.; Klette, R.; and Liu, Y. 2014. Accurate and interpretable bayesian mars for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems* 15(6):2457–2469.

Zhang, J.; Zheng, Y.; Qi, D.; Li, R.; and Yi, X. 2016. DNN-based prediction model for spatial-temporal data. In *ACM SIGSPATIAL*.

Zheng, Y.; Capra, L.; Wolfson, O.; and Yang, H. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5(3):38.

Zheng, Y. 2015. Methodologies for cross-domain data fusion: An overview. *IEEE transactions on big data* 1(1):16–34.