Flow Prediction in Spatio-Temporal Networks Based on Multitask Deep Learning

Junbo Zhang, *Member, IEEE*, Yu Zheng, *Senior Member, IEEE*, Junkai Sun, and Dekang Qi

Abstract—Predicting flows (e.g., the traffic of vehicles, crowds, and bikes), consisting of the in-out traffic at a node and transitions between different nodes, in a spatio-temporal network plays an important role in transportation systems. However, this is a very challenging problem, affected by multiple complex factors, such as the spatial correlation between different locations, temporal correlation among different time intervals, and external factors (like events and weather). In addition, the flow at a node (called node flow) and transitions between nodes (edge flow) mutually influence each other. To address these issues, we propose a multitask deep-learning framework that *simultaneously* predicts the node flow and edge flow throughout a spatio-temporal network. Based on fully convolutional networks, our approach designs two sophisticated models for predicting node flow and edge flow, respectively. These two models are connected by coupling their latent representations of middle layers, and trained together. The external factor is also integrated into the framework through a gating fusion mechanism. In the edge flow prediction model, we employ an embedding component to deal with the sparse transitions between nodes. We evaluate our method based on the taxicab data in Beijing and New York City. Experimental results show the advantages of our method beyond 11 baselines, such as ConvLSTM, CNN, and Markov Random Field.

Index Terms—Deep learning, spatio-temporal data, urban computing

1 Introduction

Spatio-temporal networks (ST-networks), like transportation networks and sensor networks, are widely available in the real world, with each node incorporating a spatial coordinate and each edge being associated with dynamic properties. Flows in such ST-networks have two representations (see Fig. 1): 1) node flow, i.e., the in—and out-flows at a node, and 2) edge flow, namely, the transitions between nodes. In a transportation system, these two types of flows can be measured by ① the number of cars driven nearby roads, ② the number of people traveling by metro/bus, ③ the number of pedestrians, or ④ *all of them together* if data is available. Fig. 1b presents an illustration. Taking node r_1 as an example, we can calculate the inflow as 3, and outflow as 3 according to the mobile phone signals and the GPS trajectories of vehicles, respectively. In detail, we can see the transition from r_3 to r_1 is 3, and the transition from r_1 to r_2 and r_4 are 2 and 1, respectively. Therefore, we can get two levels of flows: node-level and edge-level, as shown in Fig. 1c, of which the *inflow* and *outflow* of four nodes (r_1, r_2, r_3, r_4) are $(3, 3, 0, 5)$ and $(3, 2, 5, 1)$, respectively, with *transitions* over all edges being viewed as a directed graph.

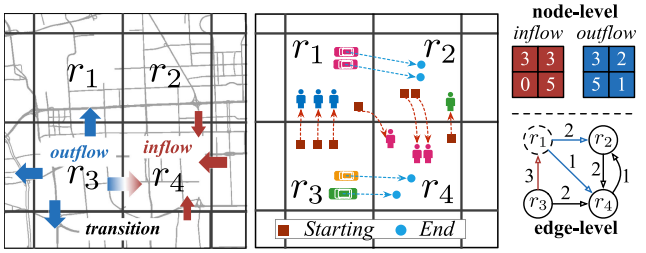
Predicting these types of flows in a ST-network is of great importance to public safety, traffic management and network optimization [34]. Taking the crowd flow [33] as an example, amounts of people streamed into a strip region at the 2015 New Year's Eve celebrations in Shanghai, resulting in a catastrophic stampede that killed 36 people. If one can predict the transitions between regions and the crowd flow in each region, such tragedies can be prevented or mitigated by utilizing emergency mechanisms (e.g., sending out warnings, evacuating people, or conducting traffic control).

However, *simultaneously* predicting in/out flows at all nodes and transitions over edges of a ST-network is very challenging because of the following aspects:

1) *Scale and complexity*: The in/out flow of a location depend on that of its near neighbors as well as distant neighbors in geographical spaces, as people can transit between any locations, particularly when some events take place in a city. Given a big city with a large number (N) of locations, there are N^2 possibility of transitions, though these transitions may not occur simultaneously at a time interval. Thus, to predict the flow of location, either the in/out flow or transition flow, we need to consider the

- *J. Zhang is with the JD Intelligent City Research & Urban Computing Business Unit, JD Digits, Beijing 100176, China, and the Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China. E-mail: msjunbozhang@outlook.com.*
- *Y. Zheng is with the JD Intelligent City Research & Urban Computing Business Unit, JD Digits, Beijing 100176, China, the School of Computer Science and Technology, Xidian University, Xi'an 710071, China, and the Shenzhen Institues of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: msyuzheng@outlook.com.*
- *J.K. Sun is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China. E-mail: junkaisun@outlook.com.*
- *D. Qi is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China. E-mail: dekangqi@outlook.com.*

Manuscript received 8 Dec. 2017; revised 28 Oct. 2018; accepted 31 Dec. 2018. Date of publication 9 Jan. 2019; date of current version 4 Feb. 2020. (Corresponding author: Junbo Zhang.) Recommended for acceptance by K. Selcuk Candan. Digital Object Identifier no. 10.1109/TKDE.2019.2891537



(a) In-out/transition (b) Measurement (c)

Fig. 1. Flows in a simple spatio-temporal network.

dependence between the location and others throughout a city. In addition, the prediction is also concerned with the flow at past time intervals. Moreover, we cannot predict the flow of each location individually and respectively, because locations in a city are connected, correlated, and mutually influence each other. The complexity and scale have posed huge challenges to traditional machine learning models like probabilistic graphical models.

2) *Model multiple correlations and external factors*: There are three types of correlations we need to model when dealing with such a prediction problem. The first one is the spatial correlation between flows of different locations, including the correlation between near locations and that between distant locations. The second one is the temporal correlation between flows of a location at different time intervals, consisting of the temporal closeness, periodic and trend properties. Third, the in/out flows and transition flow are highly correlated and mutually reinforced. The sum of transitions streaming into a location is the in-flow of the location. Likewise, an accurate prediction of the total out-flow in a location can help predict the transition flows from the location to other places more accurately, vice versa. Additionally, these flows are affected by external factors, such as events, weather, and accidents. How to integrate them into the predictive model is non-trivial.

3) *Dynamics and sparsity*: Because of the N^2 possibility, the flow of transitions between locations changes over time much more tremendously than the in/out flow. The transitions (between a location and the rest of places) that will really occur at the next time interval may be a very small portion of the N^2 possibilities (i.e., very sparse). Predicting such a sparse transition in such a high dimensional space is a very challenging task.

To tackle the aforementioned challenges, we propose a Multitask Deep-Learning (MDL, see Fig. 4) framework to predict the flows at nodes and on edges collectively and simultaneously. The contributions of the research are threefold:

- The MDL devises a deep neural network for predicting the flow at nodes (entitled NodeNet) and that on edges (entitled EdgeNet) respectively. These two deep neural networks are coupled through a concatenation of their latent layers, and trained together. In addition, the correlation between these two types of flows are modeled by a regularization in the loss function. The deep learning-based model can handle the complexity and scale problem in the prediction, while the multitask framework mutually reinforces the prediction of each type of flows.
- Both NodeNet and EdgeNet are three-stream fully convolutional networks (3S-FCNs), where closeness-stream, period-stream, and trend-stream capture three different temporal correlations. Each stream FCN also captures spatial correlations between both near and distant locations. A gating component is employed to fuse the external factors with the spatio-temporal correlations. To deal with the transition sparsity problem, in the EdgeNet we design an embedding component, which encodes the sparse (and high dimensional) input with a latent and low-dimensional representation.
- We evaluated our approach using the taxicab data in Beijing and the New York City. The results demonstrate advantages of our MDL beyond 11 baselines, such as CNN/RNN/LSTM and Markov Random Field, and the improvement beyond individual predictions.

Table 1 lists the mathematical notation used in this paper.

TABLE 1
Description of Notation

Symbol	Description
$V = \{r_{ij}\}$	spatial node set, $1 \le i \le I, 1 \le j \le J$
N	number of nodes, i.e., $I \times J$
$\mathcal{T}$	available time interval set
$\mathcal{X}_t \in \mathbb{R}^{2\times I\times J}$	tensor of inflow/outflow at nodes at time t
$\mathbf{S}_t \in \mathbb{R}^{N\times N}$	matrix of transition over edges at time t
$\mathcal{M}_t \in \mathbb{R}^{2N\times I\times J}$	tensor of transition converted from $\mathbf{S}_t$
$\mathcal{E}_t \in \mathbb{R}^{l_e}$	external features at time t
$\mathcal{X}_t(:,i,j), \mathcal{M}_t(:,i,j)$	vector of node r_{ij}
$\mathcal{X}_t(c,:,:), \mathcal{M}_t(c,:,:)$	matrix of cth channel
X_t^{dep}	dependent set of $\mathcal{X}_t$
M_t^{dep}	dependent set of $\mathcal{M}_t$
2	# channels of node flow $\mathcal{X}_t$
$2N$	# channels of edge flow $\mathcal{M}_t$

2 Problem Formulation

Definition 1 (Node). *A spatial map is divided into $I \times J$ grids based on the longitude and latitude, denoted by $V = \{r_1, r_2, \ldots, r_{I\times J}\}$, each of which represents a spatial node, as shown in Fig. 2a.*

Let (τ, x, y) be a temporal geospatial coordinate, of which τ denotes timestamp, and (x, y) denotes geospatial point. The movement of an object can be recorded as a time-ordered spatial trajectory, among which the start point and end point (i.e., start-end pair), denoted by $s = (\tau_s, x_s, y_s)$ and $e = (\tau_e, x_e, y_e)$, represent the source and destination, respectively. Let $\mathbb{P}$ be all start-end (i.e., (s, e)) pairs.

Definition 2 (In/out flows). *Given a set of start-end pairs $\mathbb{P}$. Let $\mathcal{T} = \{t_1, \ldots, t_T\}$ be a sequence of time intervals. For a node r_{ij} that lies at the ith row and the jth column of the map, the outflow and inflow during the interval t are defined respectively as*

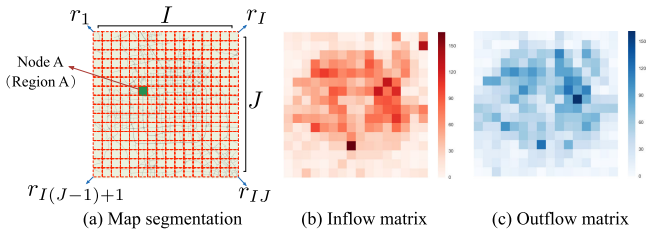


Fig. 2. Spatial nodes (regions) and flow matrices.

$$\mathcal{X}_t(0,i,j) = |\{(s,e) \in \mathbb{P} : (x_s, y_s) \in r_{ij} \wedge \tau_s \in t\}| \quad (1)$$

$$\mathcal{X}_t(1,i,j) = |\{(s,e) \in \mathbb{P} : (x_e, y_e) \in r_{ij} \wedge \tau_e \in t\}|, \quad (2)$$

where $\mathcal{X}_t(0,:,:)$ and $\mathcal{X}_t(1,:,:)$ mean outflow and inflow matrices, respectively. $(x,y) \in r_{ij}$ means the point (x,y) lies within the node r_{ij}, and $\tau_e \in t$ means the timestamp τ_e is in the time interval t. The inflow and outflow matrices at a certain time are shown in Fig. 2.

Considering two types of flows (i.e., *inflow* and *outflow*), a time-varying spatial map is conventionally represented as a time-ordered sequence of tensors, with each tensor corresponding to a snapshot of the map during a certain time interval. In detail, each tensor consists of two matrices: inflow matrix and outflow matrix, as shown in Fig. 2.

Let V denote the set of all nodes in a ST-network under study, and $N \triangleq |V| = I \times J$ be the number of nodes. A temporal graph consisting of T discrete non-overlapping time intervals is represented by the time-ordered sequence of directed graphs $G_{t_1}, \ldots, G_{t_T}$. A particular graph $G_t = (V, E_t)$ captures the topological state of the spatio-temporal system during the tth time interval. For each graph G_t (where $t = t_1, \ldots, t_T$), there exists a counterpart weight matrix $\mathbf{S}_t \in \mathbb{R}^{N \times N}$ that represents the weighted directed edges between nodes during the tth time interval. In our study, the weight of the edge from node r_s to node r_e at time t is a non-negative scalar representing the *transition* from r_s to r_e in the corresponding time interval. In a case where there is no connection between two nodes at time t, the corresponding element in $\mathbf{S}_t$ should be 0.

Definition 3 (Transition). *Given a set of start-end pairs $\mathbb{P}$. Let $\mathcal{T} = \{t_1, \ldots, t_T\}$ be a sequence of time intervals. Let $\mathbf{S}_t$ be the transition matrix during the interval t. The transition from node r_s to r_e, denoted $\mathbf{S}_t(r_s, r_e)$, is defined as*

$$\mathbf{S}_t(r_s, r_e) = |\{(s,e) \in \mathbb{P} : (x_s, y_s) \in r_s \wedge (x_e, y_e) \in r_e \wedge \tau_s \in t \wedge \tau_e \in t\}|, \quad (3)$$

where $r_s, r_e \in V$ are the start and end nodes, respectively. $(x,y) \in r$ means the point (x,y) lies within the grid r. $\tau_s \in t$ and $\tau_e \in t$ mean that the timestamp τ_s and τ_e are both in the time interval t. Here we consider the transitions that only happen at a certain time interval. Therefore, for a real-world application, we can predict a real transition whose start and end points are both in future.

2.1 Converting Time-Varying Graphs into Tensors

To apply deep neural networks to time-varying graphs, we propose converting each graph at time t into a tensor first. Given a directed graph $G_t = (V, E_t)$ at time t, we unroll it first, then compute the directed weight matrix (e.g., transition matrix $\mathbf{S}_t$), and finally get a tensor $\mathcal{M}_t \in \mathbb{R}^{2N \times I \times J}$. Fig. 3 presents an illustration. (a) Given a graph consisting of 4 nodes and 6 edges at time t. (b) We first unroll it that is a directed graph. (c) For each node, there are incoming and outgoing transitions, represented by a vector (dimension $= 8$). Taking Node r_1 for example, its outgoing and incoming transition vectors are respectively $[0,2,0,1]$ and $[0,0,3,0]$, which are further concatenated into one vector $[0,2,0,1,0,0,3,0]$, containing both outgoing and incoming information. (d) Finally, we can reshape the matrix into a tensor, among which each node has a fixed spatial position according to the original map segmentation, protecting the spatial correlations.

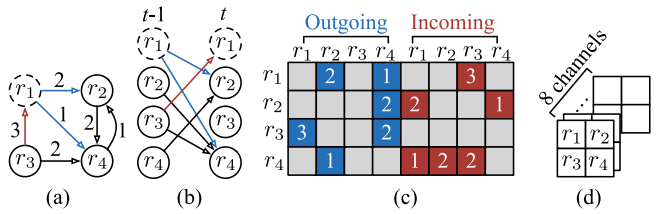


Fig. 3. Converting a time-varying graph into a tensor.

2.2 Flow Prediction Problem

Flow prediction, generally speaking, is a time series problem, which aims to predict the citywide flows in each region at time interval T+1 given the historical observations until time T. But the flows in our paper contain two perspectives, which are inflow,outflow in regions and transition flows between regions, as defined above. Our goal in this paper is to predict all these flows at the same time. In addition, we also integrate some external factors such as holidays information, weather conditions, temperature and so on. These external features can be collected and provide some extra useful information. The related notations are listed in Table 1.

Problem 1. *We here define the goal of our paper. Given the historical flow observations $\{\mathcal{X}_t, \mathcal{M}_t | t = t_1, \ldots, t_T\}$ and external features $\mathcal{E}_T$, we propose a model to* collectively *predict $\mathcal{X}_{t_{T+1}}$ and $\mathcal{M}_{t_{T+1}}$ in the future.*

3 Multitask Deep Learning

Fig. 4 presents our MDL framework, consisting of three components, which are used for data converting, node flow modeling, and edge flow modeling, respectively. As illustrated in the left part of Fig. 4, we first convert the trajectory (or trip) data over a map along time into two types of flows: i) node flow that is a time-ordered sequence of tensors $\{\mathcal{X}_t | t = t_1, \ldots, t_T\}$ (Step (1a)); ii) edge flow that is a time-ordered sequence of graphs (transition matrices) $\{\mathbf{S}_t | t = t_1, \ldots, t_T\}$ (Step (2a)), which is further converted into a sequence of tensors $\{\mathcal{M}_t | t = t_1, \ldots, t_T\}$ (Step (2b)) according to the method introduced in Section 2.1. These two types of video-like data are then fed into NodeNet and EdgeNet, respectively. Taking NodeNet as an example, it selects three different types of fragments, and feed them into a 3S-FCN, which can model the temporal correlations, including closeness, period, and trend. Among them, each steam FCN can capture spatial near and distant correlations

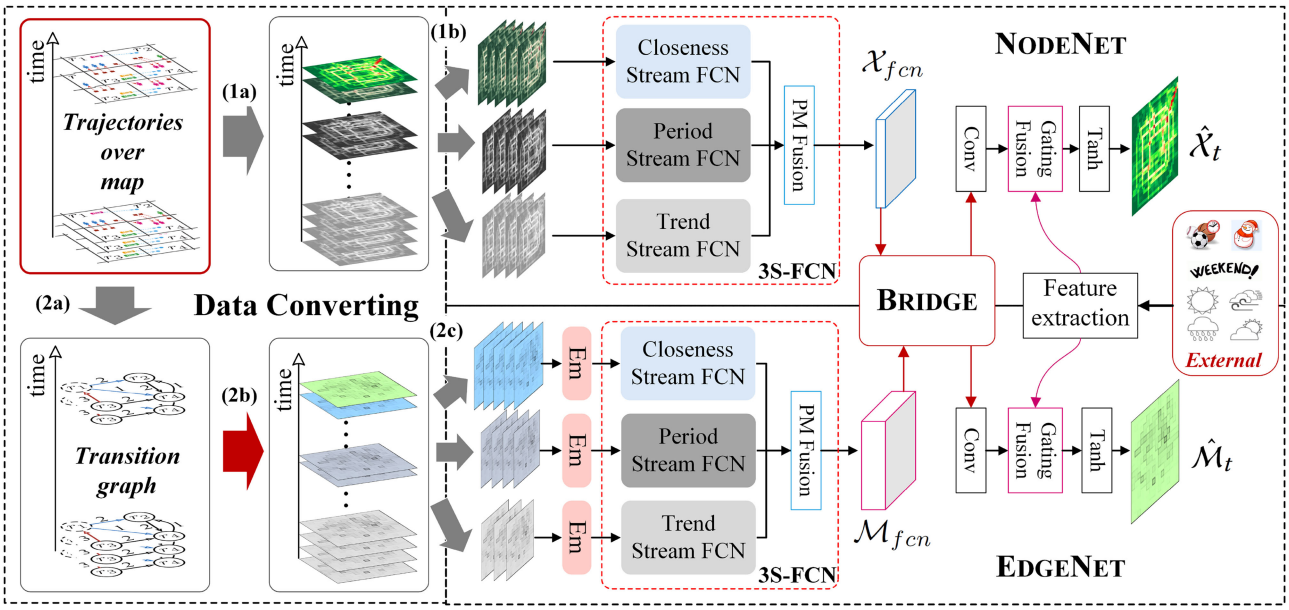


Fig. 4. MDL framework. Em: Embedding; Conv: Convolution; FCN: Fully convolutional network.

via multiple convolutions. The latent representations of middle layers of NODENET and EDGENET are coupled by a BRIDGE component, and trained together. We employ a embedding layer (called Em) to handle transition sparsity problem. A gating fusion component is used to integrate the external factors. In addition, the correlation between the node flow and edge flow are modeled by a regularization between $\hat{\mathcal{X}}_t$ and $\hat{\mathcal{M}}_t$.

3.1 EDGENET

According to the aforementioned converting method, the transition graph at each time interval can be converted into a tensor $\mathcal{M}_t \in \mathbb{R}^{2N \times I \times J}$. For each node r_{ij}, it has up to $2N$ transition possibility, including N incomings and N outgoings. However, for a certain time interval, the transition between nodes may be very sparse. Inspired by the embedding method of natural language processing [23], we propose employing a spatial embedding method, to tackle such sparse and high-dimensional ($2N$, depending on the number of nodes in the ST-network) problem. In detail, the spatial embedding tends to learn a function that maps a $2N$-dimension vector of node r_{ij} into a k-dimension space as follows:

$$\mathcal{Z}_t(:,i,j) = \mathbf{W}_m \mathcal{M}_t(:,i,j) + \mathbf{b}_m, 1 \le i \le I, 1 \le j \le J, \quad (4)$$

where $\mathbf{W}_m \in \mathbb{R}^{k \times 2N}$ and $\mathbf{b}_m \in \mathbb{R}^k$ are the learnable parameter matrix and vector, respectively. All $I \times J$ nodes share these parameters. $\mathcal{M}_t(:,i,j) \in \mathbb{R}^{2N}$ means the vector located at (i,j).

The flows, like the traffic of crowds in a city [33], are always affected by spatio-temporal dependencies. To capture different temporal dependencies (closeness, period, and trend), Zhang et al. proposed a deep spatio-temporal residual network that selects different key frames along the time. Inspired by this, we here selects recent, near, and distant key frames to predict the time interval t, respectively denoted $M_t^{dep} = \{M_t^{close}, M_t^{period}, M_t^{trend}\}$, as follows:

- *Closeness* dependents:
  $M_t^{close} = \{\mathcal{Z}_{t-l_c}, \ldots, \mathcal{Z}_{t-1}\}$.
- *Period* dependents:
  $M_t^{period} = \{\mathcal{Z}_{t-l_p \cdot p}, \mathcal{Z}_{t-(l_p-1)\cdot p}, \ldots \mathcal{Z}_{t-p}\}$.
- *Trend* dependents:
  $M_t^{trend} = \{\mathcal{Z}_{t-l_q \cdot q}, \mathcal{Z}_{t-(l_q-1)\cdot q}, \ldots \mathcal{Z}_{t-q}\}$.

where p and q are the period and trend span, respectively. l_c, l_p, and l_q are the lengths of these three parts of sequences.

The output (i.e., the prediction at next time interval) has the same resolution as the inputs. Such task is very similar to the well-known image segmentation problem, which can be handled by a fully convolutional network (FCN) [22]. Inspired by this, we here propose a three-stream FCN (3S-FCN, see Fig. 4) to capture temporal closeness, period, and trend dependencies. Among that, each stream is a FCN, consisting many convolutions (see Fig. 5). According to the property of convolution, one convolutional layer can capture spatial near dependencies. As the number of convolutional layers increases, FCN can capture farther and farther dependencies, even citywide spatial dependencies. However, such deep convolution network become very hard to train. Therefore, we employ residual connections [12] to help the training. Similar to the residual block used in the residual network [13], we use a block that consists of *Batch Normalization* (BN, [16]), *Rectified Linear Unit* (ReLU, [19]), and *Convolution* (Conv). Let the outputs of closeness-, period-, and trend-stream FCNs be $\mathcal{M}_c, \mathcal{M}_p, \mathcal{M}_q$, respectively. Different nodes may have different properties of closeness, period, and trend. To address this issue,

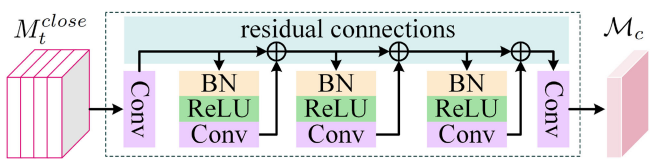


Fig. 5. FCN with residual connections.

we propose using a parametric-matrix-based fusion [33] (PM fusion in Fig. 4), to merge them

$$\mathcal{M}_{fcn} = \mathbf{W}_c \odot \mathcal{M}_c + \mathbf{W}_p \odot \mathcal{M}_p + \mathbf{W}_q \odot \mathcal{M}_q, \quad (5)$$

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

3.2 NodeNet and Bridge

Similar to EdgeNet, NodeNet is also a 3S-FCN, we select recent, near, and distant key frames as the closeness, period, and trend dependents. The difference is that NodeNet does-not have the embedding layer because the number of channels of inputs is only 2. These three different sets of dependents are fed into three different stream FCNs, whose outputs are further merged by a PM fusion component (see Fig. 4), too. Then, we can get the output of 3S-FCN, denoted $\mathcal{X}_{fcn} \in \mathbb{R}^{C_x \times I \times J}$.

Considering that node flow is correlated with edge flow, so the representations learned from NodeNet and EdgeNet should be connected. To connect NodeNet and EdgeNet, assuming two latent representations of NodeNet and EdgeNet are $\mathcal{X}_{fcn}$ and $\mathcal{M}_{fcn}$ respectively. We here propose two fusion methods.

Sum *Fusion*. The sum fusion method directly sum up these two representations, the output map at the same spatial node r_{ij} across channel c is as follows:

$$\mathcal{H}(c,:,:) = \mathcal{X}_{fcn}(c,:,:) + \mathcal{M}_{fcn}(c,:,:), \quad c = 0, \ldots, C-1, \quad (6)$$

where C is the number of channels of $\mathcal{X}_{fcn}$ and $\mathcal{M}_{fcn}$, and $\mathcal{H} \in \mathbb{R}^{C \times I \times J}$. It's obvious that this fusion method is subjected to the fact that both representations of two tasks should have a same shape, i.e., $\mathcal{X}_{fcn}$ and $\mathcal{M}_{fcn}$ have a same size at channel dimension.

Concat *Fusion*. In order to be free from the restraint. We propose an another fusion method called Concat.Formally, the concatenation of two latent representation maps $\mathcal{X}_{fcn}$ and $\mathcal{M}_{fcn}$ at the same spatial node r_{ij} across channel c as follows:

$$\mathcal{H}(c,:,:) = \mathcal{X}_{fcn}(c,:,:), c = 0, \ldots, C_x - 1 \quad (7)$$

$$\mathcal{H}(C_x + c,:,:) = \mathcal{M}_{fcn}(c,:,:), c = 0, \ldots, C_m - 1, \quad (8)$$

where C_x and C_m are the numbers of channels of $\mathcal{X}_{fcn}$ and $\mathcal{M}_{fcn}$, respectively, and $\mathcal{H} \in \mathbb{R}^{(C_x + C_m) \times I \times J}$. Concat fusion actually can better integrates two levels of node and edge flows by mutually reinforcing. We also discuss another fusion method as Bridge (see Section 4.3).

After Concat fusion, we append a convolutional layer into NodeNet and EdgeNet, respectively. The convolution is used to map combined latent feature maps $\mathcal{H}$ into different-size-channel outputs, i.e., $\mathcal{X}_{res} \in \mathbb{R}^{2 \times I \times J}$ and $\mathcal{M}_{res} \in \mathbb{R}^{2N \times I \times J}$, see Fig. 6.

3.3 Fusing External Factors Using a Gating Mechanism

External factors, such as events and weather, that can affect the flows in the different parts of a ST-network. For example, an accident may block the traffic of a certain area locally, and a rainstorm may reduce the citywide flows globally. Such an external factor just like a switch, the flows would be tremendously changed if it happen. Based on this insight, we here develop a *gating-mechanism-based* fusion, as shown in Fig. 6. At time t, one can obtain the corresponding external features in the ST-network, denoted $\mathcal{E}_t \in \mathbb{R}^{l_e \times I \times J}$, of which $\mathcal{E}_t(:,i,j) \in \mathbb{R}^{l_e}$ represents the feature vector of a particular node. Formally, we can obtain the following gating values for EdgeNet as follows:

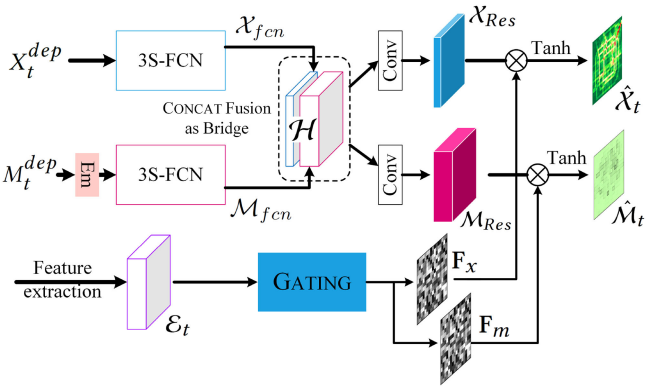


Fig. 6. MDL using Concat fusion.

$$\mathbf{F}_m(i,j) = \sigma(\mathbf{W}_e(:,i,j) \cdot \mathcal{E}_t(:,i,j) + \mathbf{b}_e(i,j)), \quad 1 \le i \le I, 1 \le j \le J, \quad (9)$$

where $\mathbf{W}_e \in \mathbb{R}^{l_e \times I \times J}$ and $\mathbf{b}_e \in \mathbb{R}^{I \times J}$ are learnable parameters. $\mathbf{F}_m \in \mathbb{R}^{I \times J}$ is the output of Gating, of which $\mathbf{F}_m(i,j)$ is the gating value of the corresponding node r_{ij} in the ST-network. $\sigma(\cdot)$ is the sigmoid function, "$\cdot$" is the dot product (inner product) of two vectors.

Then we employ a Product fusion based on the gating mechanism as follows:

$$\hat{\mathrm{M}}_t(c,:,:) = \tanh(\mathbf{F}_m \odot \mathcal{M}_{Res}(c,:,:)), c = 0, \ldots, 2N-1, \quad (10)$$

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

Similarly, the final prediction of NodeNet at time t is

$$\hat{\mathrm{X}}_t(c,:,:) = \tanh(\mathbf{F}_x \odot \mathcal{X}_{Res}(c,:,:)), c = 0, 1, \quad (11)$$

where $\mathbf{F}_x \in \mathbb{R}^{I \times J}$ is another output of Gating. One reason of using different Gating values (i.e., $\mathbf{F}$) for node and edge flows is that the external factors can affect the in/out flows and transitions of different locations differently.

3.4 Losses

Let ϕ be all the learnable parameters in EdgeNet, we intend to learn them by minimizing the following objective function between predicted transitions $\hat{\mathrm{M}}$ and true transitions $\mathcal{M}$

$$\arg\min_{\phi} \mathcal{J}_{edge} = \sum_{t \in \mathcal{T}} \sum_{c=0}^{2N-1} \left\| Q_t^c \odot \left(\hat{\mathrm{M}}_t(c,:,:) - \mathcal{M}_t(c,:,:) \right) \right\|_F^2, \quad (12)$$

where Q_t^c is an indication matrix for all the non-zero entries in $\mathcal{M}_t(c,:,:)$, i.e., $Q_t^c(i,j) = 1$ if and only if $\mathcal{M}_t(c,i,j) > 0$. $\mathcal{T}$ is a set of available time intervals. $\|\cdot\|_F$ is the Frobenius Norm of a matrix.

TABLE 2
Data Statistics

Dataset	TaxiBJ	TaxiNYC
# time intervals	35064	11472
Shape of $\mathcal{X}_t$	16×16	16×16
Shape of $\mathcal{M}_t$	$512 \times 16 \times 16$	$512 \times 16 \times 16$

Similarly, let θ be all the learnable parameters in NodeNet. For the square loss it yields the following optimization problem

$$\arg\min_{\theta} \mathcal{J}_{node} = \sum_{t \in \mathcal{T}} \sum_{c=0}^{1} \| P_t^c \odot \big(\hat{\mathcal{X}}_t(c,:,:) - \mathcal{X}_t(c,:,:)\big) \|_F^2, \tag{13}$$

where P_t^c is an indication matrix for all the non-zero entries in $\mathcal{X}_t(c,:,:)$, i.e., $P_t^c(i,j) = 1$ if and only if $\mathcal{X}_t(c,i,j) > 0$.

We know that the sum of transitions streaming into node $r_i j$ is the inflow of the node, and the sum of transitions streaming out is the outflow. From Definition 2, $\hat{\mathcal{X}}_t(0,:,:)$ and $\hat{\mathcal{X}}_t(1,:,:)$ are outflow and inflow matrices, respectively. According to the transition tensor constructed method introduced in Section 2.1, we know that the first N channels represent outgoing transitions, and the last N channels represent incoming transitions. Therefore, it yields the following loss function:

$$\arg\min_{\theta,\phi} \sum_{t \in \mathcal{T}} \sum_{i} \sum_{j} \Bigg(\|\hat{\mathcal{X}}_t(0,i,j) - \sum_{c=0}^{N-1} \hat{\mathrm{M}}_t(c,i,j)\|^2 + \|\hat{\mathcal{X}}_t(1,i,j) - \sum_{c=N}^{2N-1} \hat{\mathrm{M}}_t(c,i,j)\|^2 \Bigg). \tag{14}$$

Or, equivalently, it can be written as

$$\arg\min_{\theta,\phi} \mathcal{J}_{mdl} = \sum_{t \in \mathcal{T}} \Bigg(\| \underbrace{\hat{\mathcal{X}}_t(0,:,:)}_{\text{outflow}} - \underbrace{\sum_{c=0}^{N-1} \hat{\mathrm{M}}_t(c,:,:)}_{\text{outgoing transitions}} \|_F^2 + \| \underbrace{\hat{\mathcal{X}}_t(1,:,:)}_{\text{inflow}} - \underbrace{\sum_{c=N}^{2N-1} \hat{\mathrm{M}}_t(c,:,:)}_{\text{incoming transitions}} \|_F^2 \Bigg). \tag{15}$$

Finally, we obtain the combined loss as follows:

$$\arg\min_{\theta,\phi} \lambda_{node}\mathcal{J}_{node} + \lambda_{edge}\mathcal{J}_{edge} + \lambda_{mdl}\mathcal{J}_{mdl}, \tag{16}$$

where λ_{node}, λ_{edge}, and λ_{mdl} are adjustable hyper-parameters.

3.4.1 Optimization Algorithm

Algorithm 1 outlines the MDL training process. We first construct training instances from the original sequence of observations (lines 1-4). During each iteration, we optimize the objective (16) on the selected batch of training instances $\mathcal{D}_{batch}$ (lines 7-8).

Algorithm 1. Training of MDL Algorithm

Input: Historical observations: $\{\mathcal{X}_t, \mathcal{M}_t | t = t_1, \ldots, t_T\}$;
external features: $\{\mathcal{E}_{t_1}, \ldots, \mathcal{E}_{t_T}\}$;
lengths of *closeness*, *period*, *trend* sequences: l_c, l_p, l_q;
period: p; trend span: q.
Output: MDL model.
// construct training instances
1 $\mathcal{D}_{train} \longleftarrow \emptyset$
2 **for** $t \in \mathcal{T}$ **do** // $\mathcal{T}$ is available time available set
3 put an training instance $(\{X_t^{dep}, M_t^{dep}, \mathcal{E}_t\}, \mathcal{X}_t)$ into $\mathcal{D}_{train}$
4 **end**
// train the model
5 initialize the parameters θ, ϕ
6 **repeat**
7 randomly select a batch of instances $\mathcal{D}_{batch}$ from $\mathcal{D}_{train}$
8 find θ, ϕ by minimizing the objective (16) with $\mathcal{D}_{batch}$
9 **until** stopping criteria is met
10 output the learned MDL model

4 Experiments

We consider two kinds of datasets: *TaxiBJ* and *TaxiNYC*, see Table 2. To evaluate the prediction performance, we consider the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

4.1 Settings

4.1.1 Datasets

We use two different sets of data as shown in Table 3. Each dataset contains two sub-datasets: trajectories/trips, and external factors, detailed as follows.

- *TaxiBJ*: Trajectory data is the taxicab GPS data and meteorology data in Beijing from four time intervals: 1st Jul. 2013 - 30th Oct. 2013, 1st Mar. 2014 - 30th Jun. 2014, 1st Mar. 2015 - 30th Jun. 2015, 1st Nov. 2015 - 10th Apr. 2016. We choose data from the last four weeks as the test set, and all data before that as the training set.
- *TaxiNYC*: Taxi trip records are taken from the NYC from 2011 to 2014. Trip data includes: pick-up and drop-off dates/times, pick-up and drop-off locations. Among the data, the last four weeks are chosen as the test set, and the others as the training set.

4.1.2 Baselines

Table 4 lists the baselines compared

- *HA:* Historical Average model that uses the average of historical values in corresponding periods.
- *ARIMA:* Auto-Regressive Integrated Moving Average model.
- *SARIMA:* Seasonal ARIMA model.
- *VAR:* Vector Auto-Regressive that can capture the pairwise relationships among all flows.
- *RNN:* Recurrent Neural Network [10]. We selected previous L frames to predict the next frame. Hyper-parameters: L is set as one of $\{3, 6, 12\}$, the hidden

TABLE 3
Datasets (Holidays Include Adjacent Weekends)

Dataset	TaxiBJ	TaxiNYC
Data type	Taxi GPS	Taxi Trip
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	1/1/2011 - 12/30/2014
Time interval	1 hour	1 hour
Gird map size	(16, 16)	(16, 16)
Trajectory data		
Average sampling rate (s)	~ 60	\
# taxis	34,000+	\
# available time interval	11,472	35,064
External factors (holidays and meteorology)		
# holidays	106	451
Weather conditions	16 types (e.g., Sunny, Rainy)	\
Temperature / °C	$[-24.6, 41.0]$	\
Wind speed / mph	$[0, 48.6]$	\

TABLE 4
Baselines

Model	Temporal	Spatial	External	Transition
HA	✓			
ARIMA	✓			
SARIMA	✓			
VAR	✓			
RNN	✓			
LSTM	✓			
GRU	✓			
ST-ANN	✓	✓		
ConvLSTM	✓	✓		
ST-ResNet	✓	✓	✓	
MRF	✓	✓	✓	✓
MDL [ours]	✓	✓	✓	✓

units is set as one of $\{32, 64\}$, learning rate set as one of $\{0.1, 0.01, 0.001, 0.0001\}$,.

- *LSTM:* Long-Short-Term-Memory network [15]. The setting is same to RNN.
- *GRU:* Gated-Recurrent-Unit network [6]. The setting is same as RNN.
- *ST-ANN*: Spatio-Temporal Artificial Neural Network, which takes spatial (nearby 8 regions) and temporal (8 previous time intervals) values as input features.
- *ConvLSTM*: Convolutional LSTM [29], a state-of-the-art model for precipitation nowcasting using the radar echo dataset (image sequence). The crowd flow data used in this paper can be viewed as a sequence of images, each of which is crowd flows at a time interval. Previous 3 frames are used to predict the next frame. The model consists of two ConvLSTM layers and a convolutional layer, in which the kernel size is $(3, 3)$ and the filter number is 32. Other hyperparameters are same to RNN.
- *ST-ResNet*: Spatio-Temporal Residual Convolutional Network [33], showing state-of-the-art performance on node flow prediction.
- *MRF*: Markov-Random-Field-based citywide flow prediction model [14], that leverages flows in all individual regions and transitions between regions as well as external factors (e.g., weather).

For both datasets, we select last four weeks (i.e., 672 time intervals) as the test set, and the others as the training set. MDL is implemented using TensorFlow [2] and Keras [7], and trained via backpropagation and the Adam [18] optimization.

4.1.3 Preprocessing

In the output of the MDL, we use $\tanh$ as our final activation, 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 ground truth. For external factors, we use one-hot encoding 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 $[0, 1]$.

4.1.4 Hyperparameters

We here introduce the hyperparameter settings of our MDL. By default, we set $\lambda_{node} = 1$ and $\lambda_{edge} = 1$, which means two tasks are equally important, and λ_{mdl} as 0.0005. p and q are empirically fixed to one-day and one-week, respectively. For lengths of the three dependent sequences, we set them as: $l_c \in \{1, 2, 3\}, l_p \in \{1, 2, 3\}, l_q \in \{1, 2, 3\}$. We set the number of convolutions of FCN as 5 by default. We select 90 percent of the training data for training each model, and the remaining 10 percent 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 epochs). Network parameters are trained from a random start,[1] using the Adam [18] optimization to perform all weight updated with a fixed learning rate. The batch size is 32. The learning rate is set as one of $\{0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005\}$.

4.1.5 Evaluation Metrics

We measure the accuracy of our methods and baselines by Root Mean Square Error (RMSE) and Mean Absolute Error (MAE)[2] for both node-level (i.e., inflow/outflow) and edge-level (i.e., *transition*) prediction as

$$RMSE = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2}, \quad MAE = \frac{1}{n}\sum_i |y_i - \hat{y}_i|,$$

where y and $\hat{y}$ are the available ground truth and the corresponding predicted value, respectively; n is the number of all available ground truths.

1. The learnable parameters are initialized using a uniform distribution with the default parameter in Keras [7].
2. The smaller the better for RMSE and MAE.

TABLE 5
Comparisons with Baselines on TaxiBJ and TaxiNYC in Node Flow Prediction

Model	RMSE				MAE			
	TaxiBJ		TaxiNYC		TaxiBJ		TaxiNYC	
	inflow	outflow	inflow	outflow	inflow	outflow	inflow	outflow
HA	21.23	22.49	417.49	401.33	13.49	13.98	85.51	86.68
ARIMA	10.83	11.41	108.83	100.42	7.03	7.28	25.14	26.46
SARIMA	11.00	11.14	96.07	85.95	6.98	7.12	21.70	23.09
VAR	10.05	10.38	104.29	93.84	6.74	6.86	24.19	22.53
RNN	8.68	8.48	118.61	108.06	5.39	5.24	29.37	30.24
LSTM	9.39	9.06	121.01	110.16	5.64	5.44	28.28	29.12
GRU	9.37	9.30	124.12	106.89	5.66	5.55	28.95	27.51
ST-ANN	8.71	8.59	73.50	68.20	5.46	5.45	19.69	20.26
ConvLSTM	8.95	8.55	66.57	55.70	5.73	5.47	18.56	19.91
ST-ResNet	8.21	7.89	69.00	55.50	5.18	5.15	19.28	18.28
MRF	**7.35**	**7.08**	87.86	76.98	**4.57**	**4.50**	18.30	18.35
MDL [ours]	7.71	7.15	**53.68**	**47.44**	4.95	4.75	**13.98**	**14.63**

4.2 Results

Node Flow Prediction. We first compare various methods on the task of predicting in/out flows in the test test, given observed trained data. Table 5 shows the RMSE and MAE of node flow prediction on TaxiBJ and TaxiNYC. We can observe that, MDL and MRF consistently outperform all other baselines. In detail, our MDL performs apparently better than MRF on TaxiNYC. On the dataset TaxiBJ, MDL has a competitive result against MRF. The reason may be that TaxiNYC is 3 times bigger (T in Table 2) than TaxiBJ. In other words, our MDL has better performance on larger data than MRF. We also notice that it is time-consuming to train MRF, which takes about one week to finish the whole training process on TaxiBJ using the code provided in [14]. In detail, taking the inflow prediction of TaxiNYC as an example, the results of RMSE demonstrate that MDL is relatively 87 percent better than HA, 50 percent better than ARIMA, 44 percent better than SARIMA, 48 percent better than VAR, 26 percent better than ST-ANN, 54 percent better than RNN, 22 percent better than ST-ResNet, and 19 percent better than ConvLSTM.

Results of Edge Flow Prediction. We next compare the methods on the task of forecasting transitions. Table 6 presents the RMSE and MAE of edge flow prediction on TaxiBJ and TaxiNYC. The experiments on the transition prediction task is very time-consuming. We mainly run the experiments on MDL and HA, ARIMA, SARIMA, ST-ANN, and ST-ResNet, demonstrating that MDL outperforms others. The results show that our MDL significantly outperforms 5 baselines.

4.3 Evaluation on Fusing Mechanisms

In this section, we present the empirical experiments on different fusing mechanisms. To couple NodeNet and EdgeNet, we introduce the Concat fusion in Section 3.2. A straight-forward fusion method is to use the Sum fusion by $\mathcal{H} = \mathcal{X}_{fcn} + \mathcal{M}_{fcn}$. Note that Sum requires two latent feature maps have the same shape. For fusing external factors, one can choose one of the following ways: the Gated fusion introduced in Section 3.3, Simple fusion (the sum fusion in [33]), or not use (i.e., w/o). Therefore, there are a total of 6 variants of MDL, as shown in Table 7. The same hyper-parameter setting (e.g., number of training iterations) is used for all variants. We can observe that the Concat + Gating method outperforms other methods based on RMSE and MAE. In detail, Fig. 7 shows the converge curve of different fusion methods on the dataset TaxiNYC. Taking Fig. 7(a) as an example, we fix all components and hyper-parameters except Bridge.

4.4 Evaluation on Model Hyper-Parameters

4.4.1 Effect of Training Data Size

To demonstrate the effectiveness of training data size for deep learning model, here we select 3-month, 6-month, 1-year and 3-year data from TaxiNYC. Experiments are run on the same MDL model with $l_c = 3, l_p = 1, l_q = 1$. Fig. 8

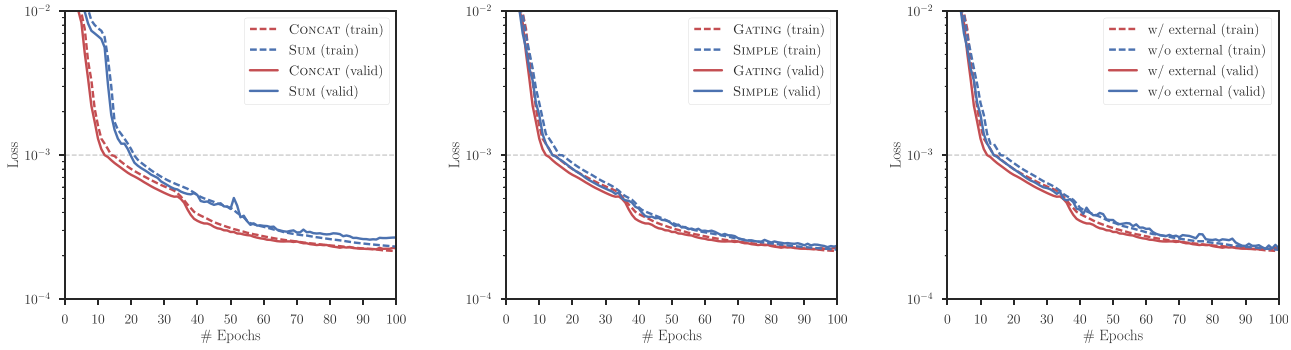
TABLE 6
Transition Prediction Results

Model	TaxiBJ	TaxiNYC
HA	1.05/ 0.68	45.03/ 10.14
ARIMA	0.98/ 0.69	16.06/ 4.89
SARIMA	1.26/ 0.77	16.21/ 5.06
ST-ANN	0.92/ 0.63	12.87/ 4.18
ST-ResNet	0.72/ 0.37	14.75/ 4.82
MDL [ours]	**0.65/ 0.32**	**9.89 / 3.48**

RMSE/ MAE for each method.

TABLE 7
RMSE and MAE on the TaxiNYC Test Set Using MDL with Different Types of Fusions

Fusing type		RMSE/ MAE		
Bridge	External	inflow	outflow	transition
Concat	Gating	**53.68/ 13.98**	**47.44/ 14.63**	**9.89/ 3.48**
Concat	Simple	55.68/ 14.48	49.03/ 15.00	10.12/ 3.55
Concat	w/o	55.70/ 14.64	47.81/ 14.82	10.10/ 3.57
Sum	Gating	55.77/ 14.24	48.32/ 14.88	10.10/ 3.54
Sum	Simple	55.81/ 14.50	49.53/ 15.17	10.29/ 3.62
Sum	w/o	54.85/ 14.14	49.32/ 15.12	10.11/ 3.57



(a) Concat vs Sum fusion (b) Gating vs Simple fusion (c) With vs without external factors

Fig. 7. Training curves on *TaxiNYC* of various fusions. The vertical axis corresponds to training and validation (valid) losses, and the horizontal axis corresponds to the number of epochs.

presents the results. We can observe that more data always has better results on both node flow and edge flow prediction.

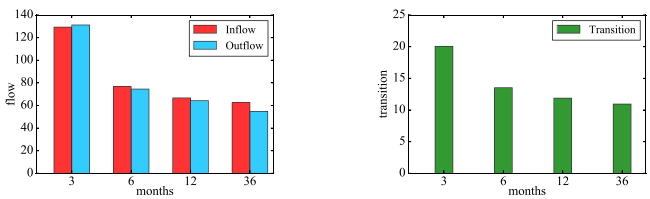
4.4.2 Effect of Network Depth

Fig. 9 presents the effect of network depth on TaxiNYC (3-month data). As the network goes deeper (i.e., the number of convolutions increases), the RMSE of the model first decreases, demonstrating that the deeper network often has a better result because it can capture not only spatial near dependencies but also distant ones. However, the RMSE increases when the network becomes much deeper, showing that the training process becomes much more difficult.

4.4.3 Effect of Multi-Task Component

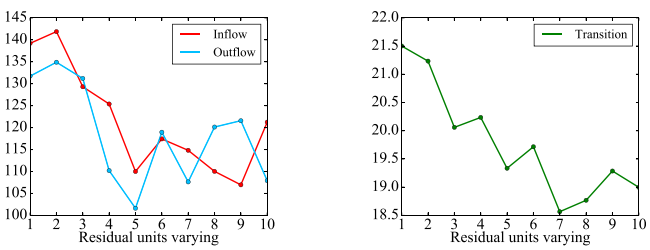
Table 8 and Fig. 10 demonstrate the influence our multi-task component on the final experiments performance.

From the table and figure, we can find that transition flow prediction task can be improved in most cases, and when the $\lambda_{node} = \lambda_{edge} = 1$ and λ_{mdl} =0.1, our multi-task model achieves best performance against others, under this circumstance, both tasks get better results compared with two single tasks, which proves the effectiveness and reliability of the fact that our multi-task part can mutually promote the performance of each task.



(a) In- and out-flows (b) Transition

Fig. 8. Effect of training data size.



(a) In- and out-flows (b) Transition

Fig. 9. Effect of network depth.

4.5 Flow Predictions

Fig. 11 depicts two nodes' predictive results of our MDL over the next one hour against the ground truth in New York City (NYC) in the last 4 weeks of 2014. In detail, Node $(10, 1)$ always have higher flow than Node $(8, 3)$. We can observe that our model is very accurate in tracing the ground truth curves (including sudden changes) of traffic flow on both nodes in NYC, which demonstrates the effectiveness of our proposed model.

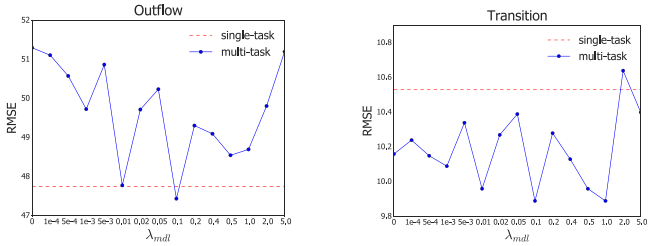
5 Related Work

5.1 Spatio-Temporal Prediction

Many works are trying to find some patterns and correlations from spatio-temporal datasets [17], [31], [32]. There are some previously published works on predicting an individual's movement based on their location history [9], [24], [27]. 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 public safety situations. Some other researchers aim to predict travel speed and traffic volume on the road [1], [25]. Most of them are predicting single or multiple road segments, rather than citywide ones [5], [30]. Recently, researchers have started to focus on city-scale traffic flow prediction [14], [21]. 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 requires a more complex method to find irregular regions first. Deng et al.

TABLE 8
Single-Task versus Multi-Task

Hyper-Parameters			RMSE / MAE		
λ_{node}	λ_{edge}	λ_{mdl}	inflow	outflow	transition
0	1	0	/	/	10.53/3.63
1	0	0	56.66/14.60	51.30/15.34	/
1	1	0.1	**53.68/13.98**	**47.44/14.63**	**9.89/3.48**



(a) outflow (b) Transition

Fig. 10. Effect of multi-task component when $\lambda_{node} = \lambda_{edge} = 1$.

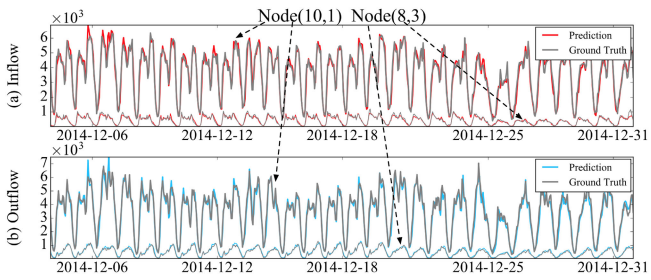


Fig. 11. Predictions of MDL against the ground truths.

proposed a latent space model for predicting time-varying traffic [8] on the fixed graph (i.e., road network), which is different from ours.

5.2 Classical Models for Time Series Prediction

Forecasting flow in a spatio-temporal network can be viewed as a time series prediction problem. Existing time-series models, like the auto-regressive integrated moving average model (ARIMA, [3]), seasonal ARIMA [26], and the vector autoregressive model [4] can capture the temporal dependencies very well, yet it fails to handle spatial correlations.

5.3 Neural Networks for Sequence Prediction

Neural networks and deep learning [10] have gained numerous success in the fields such as compute vision [19], speech recognition [11], and natural language understanding [20]. Recurrent neural networks (RNNs) have been used successfully for sequence learning tasks [28]. The incorporation of long short-term memory (LSTM) [15] or gated recurrent unit (GRU) [6] enables RNNs to learn long-term temporal dependency. However, these neural network models can only capture spatial or temporal dependencies. Recently, researchers have combined the above networks and proposed a convolutional LSTM network [29] 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. Zhang et al. proposed a spatio-temporal residual network [33], capable of capturing spatio-temporal dependencies as well as external factors, yet it may be not suited to deal with transitions over large dynamic graphs.

6 Conclusions

We proposed a novel multitask deep learning (MDL) framework for simultaneously predicting in/out flows (node flow) and transitions (edge flow) in a spatio-temporal network. MDL can not only handle the complexity and scale problem in the prediction, but also mutually reinforce the prediction of each type of flow. In addition, MDL is capable of capturing the spatial correlations (near and distant), temporal correlations (closeness, period, trend), and external factors (like events and weather). We evaluate our MDL on two real-world datasets in Beijing and NYC, achieving performances which are significantly better than 11 baseline methods.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (Grant No. 61672399, No. U1401258, and No. 61773324), and the China National Basic Research Program (973 Program, No. 2015CB352400).

References

[1] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 653–662, Apr. 2015.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," in *Proc. 12th OSDI*, 2016, pp. 265–283.

[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.

[4] S. R. Chandra and H. Al-Deek, "Predictions of freeway traffic speeds and volumes using vector autoregressive models," *J. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 53–72, 2009.

[5] P.-T. Chen, F. Chen, and Z. Qian, "Road traffic congestion monitoring in social media with hinge-loss Markov random fields," in *Proc. IEEE Int. Conf. Data Mining*, 2014, pp. 80–89.

[6] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2014, pp. 1724–1734. [Online]. Available: http://aclweb.org/anthology/D/D14/D14–1179.pdf

[7] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[8] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1525–1534.

[9] Z. Fan, X. Song, R. Shibasaki, and R. Adachi, "CityMomentum: An online approach for crowd behavior prediction at a citywide level," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 559–569.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[11] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 6645–6649.

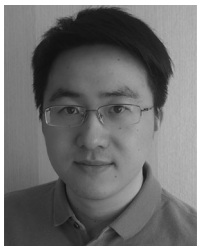
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.90

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. 14th Eur. Conf. Comput. Vis.*, 2016, pp. 630–645. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46493-0_38

[14] M. X. Hoang, Y. Zheng, and A. K. Singh, "FCCF: Forecasting citywide crowd flows based on big data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Advances Geographic Inf. Syst.*, 2016, pp. 6:1–6:10.

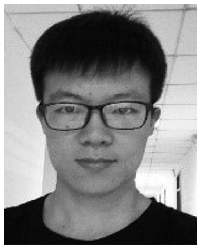
[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[17] T. Jindal, P. Giridhar, L.-A. Tang, J. Li, and J. Han, "Spatiotemporal periodical pattern mining in traffic data," in *Proc. 2nd ACM SIGKDD Int. Workshop Urban Comput.*, 2013, pp. 11:1–11:8. [Online]. Available: http://doi.acm.org/10.1145/2505821.2505837
[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *In Proc. 3rd Int. Conf. Learn. Representations.*, 2014.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[20] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31th Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196. [Online]. Available: http://jmlr.org/proceedings/papers/v32/le14.html
[21] Y. Li, Y. Zheng, H. Zhang, and L. Chen, "Traffic prediction in a bike-sharing system," in *Proc. 23rd SIGSPATIAL Int. Conf. Advances Geographic Inf. Syst.*, 2015, pp. 33:1–33:10.
[22] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[24] B. Prabhala and T. L. Porta, "Next place predictions based on user mobility traces," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Apr. 2015, pp. 93–94.
[25] R. Silva, S. M. Kang, and E. M. Airoldi, "Predicting traffic volumes and estimating the effects of shocks in massive transportation systems," *Proc. Nat. Academy Sci. United States America*, vol. 112, no. 18, pp. 5643–5648, 2015.
[26] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transp. Res. Part C: Emerging Technol.*, vol. 10, no. 4, pp. 303–321, 2002.
[27] X. Song, Q. Zhang, Y. Sekimoto, and R. Shibasaki, "Prediction of human emergency behavior and their mobility following large-scale disaster," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 5–14.
[28] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[29] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. WOO, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 802–810.
[30] Y. Xu, Q.-J. Kong, R. Klette, and Y. Liu, "Accurate and interpretable Bayesian MARS for traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2457–2469, Dec. 2014.
[31] Q. Yuan, W. Zhang, C. Zhang, X. Geng, G. Cong, and J. Han, "PRED: Periodic region detection for mobility modeling of social media users," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 263–272.
[32] C. Zhang, K. Zhang, Q. Yuan, H. Peng, Y. Zheng, T. Hanratty, S. Wang, and J. Han, "Regions, periods, activities: Uncovering Urban dynamics via cross-modal representation learning," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 361–370. [Online]. Available: https://doi.org/10.1145/3038912.3052601
[33] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1655–1661. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14501
[34] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, 2014, Art. no. 38.



Junbo Zhang is currently a data scientist with JD Digits, leading the AI Platform Division of Urban Computing Business Unit, JD Digits. His research interests include urban computing, machine learning, data mining, and big data analytics. Before joining JD Digits, he was a researcher with Microsoft Research Asia. He has published more than 30 research papers (e.g., *Artificial Intelligence*, the *IEEE Transactions on Knowledge and Data Engineering*, KDD, AAAI, IJCAI) in refereed journals and conferences, among which one paper was selected as the ESI Hot Paper, two as the ESI Highly Cited Paper, and two as the Best Paper Award. He received the ACM Chengdu Doctoral Dissertation Award in 2016, the Chinese Association for Artificial Intelligence (CAAI) Excellent Doctoral Dissertation Nomination Award in 2016, the Si Shi Yang Hua Medal (Top $1/1000$) of SWJTU in 2012, and the Outstanding PhD Graduate of Sichuan Province in 2013. He is a member of the IEEE, ACM, CAAI, and China Computer Federation.

Yu Zheng is a vice president and chief data scientist with JD Digits, leading the Urban Computing Business Unit and serving as the director of JD Intelligent City Research. He is also a chair professor with Shanghai Jiao Tong University and an adjunct professor with the Hong Kong University of Science and Technology. Before Joining JD Digits, he was a senior research manager with Microsoft Research. He currently serves as the editor-in-chief of the *ACM Transactions on Intelligent Systems and Technology*. He has served as chair on more than 10 prestigious international conferences, e.g., as the program co-chair of ICDE 2014 (Industrial Track) and CIKM 2017 (Industrial Track). In 2013, he was named one of the Top Innovators under 35 by *MIT Technology Review* (TR35) and featured by *Time Magazine* for his research on urban computing. In 2014, he was named one of the Top 40 Business Elites under 40 in China by *Fortune Magazine*. In 2017, he was named an ACM distinguished scientist. He is a senior member of the IEEE.



Junkai Sun is working toward the master's degree at Xidian University, majoring in computer science and technology. His research interests mainly include spatio-temporal data mining with deep learning, urban computing. He is also an intern student with the Urban Computing Business Unit, JD Digits.

Dekang Qi is working toward the PhD degree in the Cloud Computing & Intelligent Technology Lab, Southwest Jiaotong University. His research focuses on spatio-temporal data mining and urban computing.