Transportation Research Part C 124 (2021) 102912

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

# Urban flow prediction with spatial–temporal neural ODEs ☆

Fan Zhou [a,*], Liang Li [a], Kunpeng Zhang [b], Goce Trajcevski [c]

[a] School of Information and Software Engineering, University of Electronic Science and Technology of China, China
[b] Department of Decision, Operations & Information Technologies, University of Maryland, College Park MD, United States of America
[c] Department of Electrical and Computer Engineering, Iowa State University, Ames IA, United States of America

ARTICLE INFO

*Keywords:*
Urban flow
Ordinary differential equations
Spatial–temporal learning
Time series prediction
Intelligent transportation system

ABSTRACT

With the recent advances in deep learning, data-driven methods have shown compelling performance in various application domains enabling the Smart Cities paradigm. Leveraging spatial–temporal data from multiple sources for (citywide) traffic forecasting is a key to strengthen the smart city management in areas such as urban traffic control, abnormal event detection, etc. Existing approaches of traffic flow prediction mainly rely on the development of various deep neural networks –e.g., Convolutional Neural Networks such as ResNet are used for modeling spatial dependencies among different regions, whereas recurrent neural networks are increasingly implemented for temporal dynamics modeling. Despite their advantages, the existing approaches suffer from limitations of intensive computations, lack of capabilities to properly deal with missing values, and simplistic integration of heterogeneous data. In this paper, we propose a novel urban flow prediction framework by generalizing the hidden states of the model with continuous-time dynamics of the latent states using neural ordinary differential equations (ODE). Specifically, we introduce a discretize-then-optimize approach to improve and balance the prediction accuracy and computational efficiency. It not only guarantees the prediction error but also provides high flexibility for decision-makers. Furthermore, we investigate the factors, both intrinsic and extrinsic, that affect the city traffic volume and use separate neural networks to extract and disentangle the influencing factors, which avoids the brute-force data fusion in previous works. Extensive experiments conducted on the real-world large-scale datasets demonstrate that our method outperforms the state-of-the-art baselines, while requiring significantly less memory cost and fewer model parameters.

## 1. Introduction

Citywide traffic flow prediction – a problem that leverages spatial–temporal data measured by various sensors to predict traffic patterns – has drawn increasing attention due to its great importance in many real-world applications, such as taxi/bicycle demand prediction (Wang et al., 2017; Yao et al., 2018; Rodrigues et al., 2019; Geng et al., 2019), urban traffic control and congestion avoidance (Qi et al., 2018), abnormal event detection (Huang et al., 2019; Jiang et al., 2019), travel time estimation (Ma et al., 2019) and route planning (Zhang et al., 2018; Li et al., 2018a). However, accurate and reliable prediction is not trivial as the traffic can be affected by multiple complex factors: (1) there exist strong *spatial correlations* among nearby regions — an accident on a particular road segment would not only increase the congestion on it, but would also propagate the congestion to its neighboring road segments. In addition, the in-flow of an area is strongly dependent on the out-flow of nearby regions. (2) the future traffic

☆ This article belongs to the Virtual Special Issue on IG005572 - VSI:Machine learning.

* Corresponding author.
*E-mail addresses:* fan.zhou@uestc.edu.cn (F. Zhou), liang.li@std.uestc.edu.cn (L. Li), kpzhang@umd.edu (K. Zhang), gocet25@iastate.edu (G. Trajcevski).

https://doi.org/10.1016/j.trc.2020.102912
Received 13 March 2020; Received in revised form 26 November 2020; Accepted 29 November 2020
Available online 16 December 2020
0968-090X/

flow, both citywide and local, is related to both the past and current flow in the proximal areas, i.e., exhibiting strong *temporal correlations*. An obvious example is that urban traffic flow is, to some extent, periodical, which repeats with a certain frequency, i.e., traffic conditions during rush hours may repeat every workday while there is (usually) relatively lower traffic flow in the weekend. (3) Certain *external factors*, such as weather conditions, as well as public events and holidays, have a significant impact on the accuracy of predicting the traffic flow.

There exist a number of prior studies on predicting urban traffic using spatial–temporal data obtained from different sensing systems. When it comes to machine learning based approaches, they can be roughly classified into two main categories, according to the way of modeling traffic data. The first category of models (Zhang et al., 2017; Liang et al., 2018; Rodrigues et al., 2019; Yao et al., 2019; Zhang et al., 2019a) apply deep learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for spatial–temporal data modeling, where CNNs (e.g., ResNet He et al., 2016) are usually used for non-linear spatial dependency learning while RNNs (i.e., LSTM Hochreiter and Schmidhuber, 1997 or GRU Chung et al., 2014) are *de facto* base learners for sequential and non-linear temporal dependency modeling. The second category of approaches use deep graph learning such as Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017) and Graph Attention Networks (GAT) (Velickovic et al., 2018; Zhou et al., 2020b), either modeling the spatial–temporal data that is captured by sensor networks (Li et al., 2018b) or building graphs for map regions (Guo et al., 2019a; Sun et al., 2019; Zhang et al., 2019a; Diao et al., 2019). These methods aim at learning the spatial–temporal correlations among different regions with graph-based neural networks. Additionally, other techniques (e.g., attention mechanism) and external factors (e.g., weather, holidays, periodical flow shift) are often taken into consideration for improving the performance of traffic flow prediction.

Despite the significant progress and promising results achieved by the aforementioned traffic flow prediction models, they still exhibit certain limitations due to inherent constraints that have not been adequately addressed yet:

- *High computational overhead*: Previous works (Zhang et al., 2017; Liang et al., 2018; Yao et al., 2019; Liang et al., 2019; Zhang et al., 2019a) pay no attention to the computational overheads and employ deeper ResNet (He et al., 2016) models to enhance the map feature extraction. However, as the networks grow more complex, they require a huge amount of training parameters, which hinders them from being generalizable to real-world applications. This is especially important in the IoT settings where memory limitations and computing demands are among the main bottlenecks for training large neural network models.
- *Prediction performance vs. Efficiency trade-offs*: To a certain extent, traffic prediction methods with deep learning models are inefficient in training large amounts of traffic data. In contrast, some downstream tasks and decision-making require flexible solutions that can take a tolerance of prediction errors in order to speed up the response time – e.g., in occurrences of abnormal events and extreme weather conditions. However, the existing methods fail to provide such a trade-off between prediction performance and decision response.
- *Undifferentiated use of external factors*: A key assumption underlying the existing urban traffic prediction methods is that external factors are important features and indicators. However, simply "concatenating" these factors in an end-to-end manner may result in entanglement of latent features and possibly lead to unsatisfactory prediction results. How to identify individual feature importance is issue that could help in more efficient data fusion and improvement of the prediction results.

To address the aforementioned challenges, we propose a novel approach, called **TODE** (**T**raffic flow prediction with neural **O**rdinary **D**ifferential **E**quations), which is among the first traffic flow models to deal with prediction in the context of continuous dynamical systems. Instead of deep but finite layers used in previous studies, TODE learns the dynamics via parameterizing the ODE with neural networks, which allows for more efficient memory and parameter use, as well as more flexible trade-offs between prediction accuracy and computational overheads. In summary, the main contributions of this work are:

- We present a novel traffic flow model that extends neural ODEs for more accurate and computationally efficient predictions. TODE trains the model with constant memory and therefore greatly alleviates the memory bottleneck in low-power scenarios. By incorporating a novel discretize-then-optimize mechanism into TODE, we address the problem of numerical instability and incorrect gradients in training ODE model.
- The proposed model allows flexible solutions for predicting traffic flow by conciliating prediction precision with computational cost. Solving the ODE guarantees a bound of the approximation error and can monitor the model error in real time, which acknowledges the adaptability of our model, i.e., its computational cost scales linearly with the requested training accuracy and task complexity. In addition, our model can be trained off-line with high-level data fitting and prediction accuracy, which can be reduced at test time for faster on-line flow prediction.
- We conduct an extensive analysis on the relationships between various factors and urban traffic flow, and distinguish these factors by grouping them into two types: internal features (e.g., flow periodicity) and external features (e.g., weather, wind speed, etc.). Through incorporating two identical networks to separately extract different types of features, TODE successfully disentangles the impact factors without affecting each other. Unlike the ambiguous data fusion in previous works, TODE is able to explicitly model the individual impact of different factors on the traffic flow prediction.
- We conduct extensive set of experiments to evaluate the proposed model on two large-scale real traffic datasets. The experimental results demonstrate that our TODE model significantly improves the flow prediction accuracy compared with the state-of-the-art baselines, while incurring less memory cost and fewer parameters.

The remainder of this paper is organized as follows. We discuss the related work in Section 2, and introduce the problem and provide the necessary background in Section 3. We provide model-free evidence by analyzing the impact factors in Section 4. The details of the proposed method are presented in Section 5, followed by the report on our extensive experimental evaluations in Section 6. We conclude the paper and point some future work remarks in Section 7.

## 2. Related work

We now position TODE with respect to the related literature, categorized in three main bodies of works.

### 2.1. Spatial–temporal pattern mining

Unveiling the underlying spatial–temporal patterns has been a trending research topic in AI (Zhou et al., 2018; Liu et al., 2016), GIS (Gao et al., 2018; Furtado et al., 2018) and intelligent transportation systems (Min and Wynter, 2011; Zhang et al., 2019c; Yu et al., 2020). Traditionally, spatial–temporal data are usually modeled via dynamic Bayesian networks such as hidden Markov models (Rabiner, 1986) and Kalman filters (Fildes, 1991), which can estimate the spatial and temporal auto-correlation in the data. Time series prediction methods, such as Autoregressive Integrated Moving Average (ARIMA) (Box and Jenkins, 1976) and exponential smoothing techniques (Gardner, 1985), are also frequently used for temporal data modeling and periodical pattern mining. All rely on stationary assumption of the time series and therefore have difficulty in dealing with abrupt changes in spatial–temporal data. Consequently, spatial correlations among the data have been incorporated into the modeling framework, where typical spatial statistics extraction methods are frequently used, including Kriging (Oliver and Webster, 1990), geographically weighted regression (GWR) (Brunsdon et al., 1996), spatial auto-regressive (SAR) (Kelejian and Prucha, 1999), and Markov random-field (Zhao et al., 2007).

Recently, deep learning models, especially the RNNs, have become popular paradigms for modeling spatial–temporal data, and have achieved performance improvements in many applications due to their ability of non-linear function approximation and flexibility on capturing longer-term dependency among spatial–temporal points. A number of works have been proposed to learn various spatial–temporal data, including prediction of next check-in location (Liu et al., 2016; Feng et al., 2018; Gao et al., 2019); venue recommendation (Zhou et al., 2019a, 2020a); identifying the users by whom the trajectories are generated (Zhou et al., 2018), and semantic understanding of movement (Zhou et al., 2019b). However, these works mainly focus on mining the individual mobility patterns rather than group behavior and citywide crowd variation that we study in this work. There exists another group of studies that estimate the travel time (Zhang et al., 2018; Li et al., 2019; Ma et al., 2019; Li et al., 2018a), predict the origin–destination demand of vehicles and/or persons Wang et al. (2019), Ke et al. (2017, 2019), Liu et al. (2019a) and Liang et al. (2020a) both of which, however, only learn the traffic conditions in a specific area or neighboring road networks.

### 2.2. Traffic flow prediction

Due to its importance in intelligent transportation systems and city safety monitoring, predicting the citywide traffic flow has attracted long-lasting attention from both industry and academics (Ryu et al., 2018; Wu et al., 2018; Do et al., 2019). Various typical machine learning techniques have been adopted and modified to model traffic flow, e.g., ARIMA (Van Der Voort et al., 1996), support vector regression (SVR) (Castro-Neto et al., 2009), and locally weighted learning (LWL) (Van Der Voort et al., 1996). These methods require expensive feature engineering which, however, cannot be generalized to different scenarios. More importantly, they typically suffer from the underfitting issue due to the lack of ability to learn meaningful patterns in a high-dimensional space.

To alleviate these limitations, deep learning methods have been introduced. Early efforts have developed stacked autoencoder (Huang et al., 2014; Polson and Sokolov, 2017) and deep belief networks (Huang et al., 2014) for traffic prediction — which are too shallow to fit the non-linear structure and to learn multi-modal patterns in the traffic data. More recently, researchers have combined deeper neural networks such as ResNet (He et al., 2016) and LSTM (Hochreiter and Schmidhuber, 1997) for traffic prediction (Zhang et al., 2017; Tian et al., 2018; Liang et al., 2018; Rodrigues et al., 2019; Yao et al., 2019; Jiang et al., 2019; Liang et al., 2019; Zhang et al., 2019a; Ryu et al., 2018; Wu et al., 2018; Do et al., 2019). For example, ST-ResNet (Zhang et al., 2017) aims to predict the in-flow and out-flow of people for each region by considering both temporal and spatial dependencies among flows, where the city heat-map is learned with stacking CNNs and ResNet deep neural networks. Multi-level attention-based LSTM model has been used for the geo-sensory traffic flow prediction (Liang et al., 2018). In two recent works, ResNet has been leveraged for traffic flow prediction with multi-task learning (Liang et al., 2019) and flow map super resolution (Zhang et al., 2019a). It is worthwhile to mention another group of works (Li et al., 2018b; Guo et al., 2019a; Pan et al., 2019; Sun et al., 2019; Zhang et al., 2019a; Diao et al., 2019; Zhang et al., 2019c; Gu et al., 2019; Yu et al., 2020) that predict the traffic conditions (e.g., speed and volume) with graph-based learning methods. Differing from the in-flow/out-flow traffic studied in this work, they either study the traffic data captured by sensor networks or model the road-networks as a graph, where graph neural networks such as GCN (Kipf and Welling, 2017; Zhou et al., 2020c) can be applied to learn and predict the traffic flow.

Despite achieving promising results on traffic flow prediction, these works are computationally expensive due to a large amount of parameters required as the networks go deeper. In addition, these models cannot tackle the missing data problem and unable to calculate precise hidden states due to the *discrete* network structure used. Furthermore, external factors are simply concatenated as feature vectors for training their prediction models, while the significant difference of individual factor remains elusive.

**Table 1**
Notations used in this study.

| Notation | Description |
|---|---|
| $\mathcal{M}$ | A map with a collection of regions |
| $r_{ij}$ | A spatial region (cell in a grid) |
| $\mathbf{T}$ | A collection of trajectories |
| $T$ | A particular trajectory in $\mathbf{T}$ |
| $\mathcal{X}$ | Traffic flow datasets |
| $\mathbf{X}_t$ | Traffic flow tensor at time instant $t$ |
| $F_t = \{F_t^{\text{int}}, F_t^{\text{ext}}\}$ | Intrinsic/extrinsic features at time instant $t$. |
| $\mathbf{F}_t$ | Output of the feature embedding |
| $\mathbf{I}$ | Input of TODE |
| $\mathbf{O}$ | Output of TODE |
| $H$ | Map height. |
| $W$ | Map width |
| $\Delta t$ | Step integration time |
| $\mathbf{h}_t$ | The hidden state at time step $t$. |
| $N_o$ | The number of time steps in one ODE block. |
| $N$ | The number of points in $\mathbf{T}$. |
| $f(\cdot)$ | Regular neural networks. |

### 2.3. Neural ordinary differential equations

Research works (Weinan, 2017; Lu et al., 2018) have established relationships between deep neural networks and stochastic dynamic systems, e.g., the ultra deep networks such as ResNets (He et al., 2016) can be interpreted as Euler solutions to ordinary differential equations (ODEs). NODE (Chen et al., 2018) directly models the dynamics of network hidden states with an ODE solver and uses the adjoint method to calculate the gradients in a memory efficient way. A neural stochastic ODE network has been proposed by (Liu et al., 2019b), which incorporates a variety of noise-injection based regularization mechanisms. Theoretical analysis (Avelin and Nyström, 2019) has proved that the stochastic gradient descent (SGD) as well as the corresponding loss function of the ResNet converges to the SGD for a neural ODE. Several most recent works have studied how to improve the neural ODEs from various aspects. For example, an augmented neural ODE method (Dupont et al., 2019) has been proposed for more expressive and stable function approximation, while ANODE (Gholami et al., 2019) and its extended version (Zhang et al., 2019b) address the training instability problem caused by incorrect gradient and the evolution of the neural network parameters, respectively. Despite its significant influence in the network structure design, existing works mainly focus on the application of ODE in computer vision (He et al., 2019; Heinonen and Lähdesmäki, 2019). Yang et al. (2013) used a flexible neural tree (FNT) model to predict small-time scale communication traffic while the system (communication network) is expressed by ODEs. In their work, FNT and ODE are completely different components, although they are combined in an ensemble learning manner. In contrast, the neural ODE, as well as our ST-ODE, extends the discrete neural networks to have continuous-time hidden dynamics defined by ODEs. In this spirit, this paper provides the first ODE-based crowd flow forecasting method.

## 3. Preliminaries

We now introduce the basic terminology (see Table 1) used in our crowd flow prediction task (Hoang et al., 2016), and review the ordinary differential equations (Chen et al., 2018).

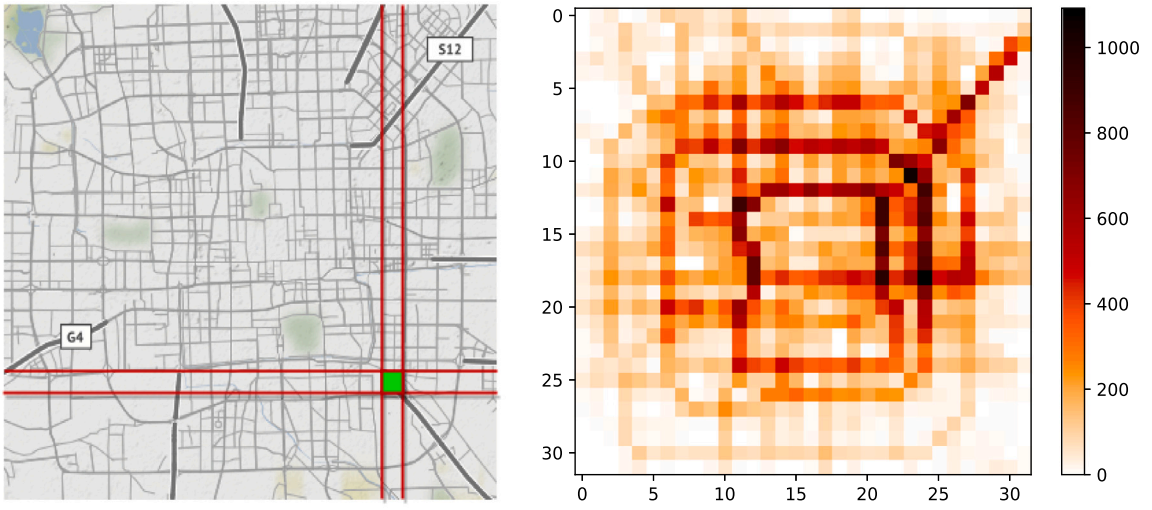
### 3.1. Problem definition

We consider a grid based segmentation (Zhang et al., 2017, 2019a), which divides a city map $\mathcal{M}$ into $H \times W$ spatial grid-cells based on their geographical locations, i.e., $\mathcal{M} = \{r_{ij}\}_{H \times M}$. Each cell $r_{ij}$ corresponds to a spatial region – the $i$th row and the $j$th column of $\mathcal{M}$. Figs. 1(a) and 1(b) illustrate this grid based segmentation for the city of Beijing, where there are 32 × 32 cells (20 km × 20 km) and the size of each cell is 625 m × 625 m.

Suppose there is a collection of trajectories $\mathbf{T}$ representing the movement of city crowds, e.g., pedestrians, bicycles, vehicles, etc. A particular trajectory $T^u \in \mathbf{T}$, records a sequence of time-series points, $T^u = s_1^u \rightarrow s_2^u \rightarrow \cdots s_k^u \rightarrow \cdots s_{N^u}^u$, where $N^u = |T^u|$ is the number of points in $T^u$, and $s_k^u$ denotes a spatio-temporal record $s_k^u \in \{r_{ij}, t_k\}$ – indicating that the location of the trajectory is in the region $r_{ij}$ at time $t_k$. In the sequel, without loss of generality and whenever there is no ambiguity, we will omit the superscript-index for trajectories.

With a note that there are several different definitions of traffic flow in the transportation literature (Bellomo et al., 2002; Zhou et al., 2019c), in this work we use the following:

**Definition 1** (*Crowd Inflow/Outflow*)**.** For a certain region $r_{ij}$, we consider two types of crowd flows – In-flow and Out-flow – at time instant $t$ (Zhang et al., 2017), which are respectively defined as:




(a) Region split. (b) Heatmap.

**Fig. 1.** Regions of a Beijing map based on geographical locations. In the left figure, the green area represents a cell corresponding to an individual region. The right portion of the figure shows a collection of regions corresponding to a heat map (of the traffic flow) of Beijing.

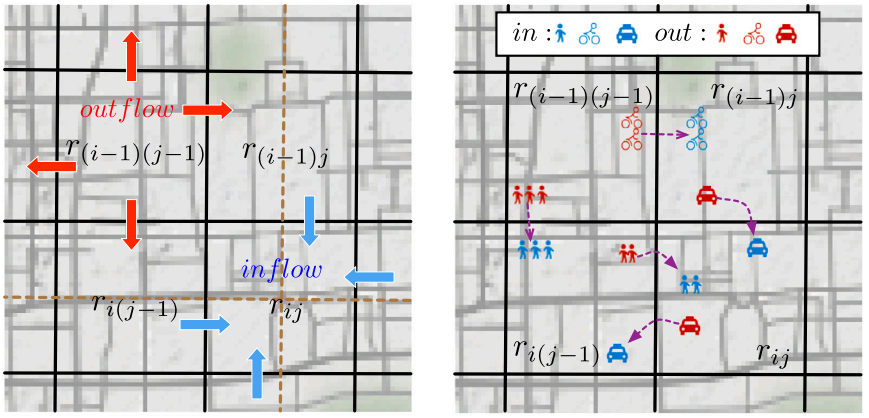



**Fig. 2.** In-flow and out-flow defined in traffic flow prediction task.

$$
\begin{aligned}
x^{\text{in}}_{r_{ij}}(t) &= \sum_{\forall T \in \mathbf{T}} \sum_{n=1}^{N} \left\{ s_{n-1} \notin r_{ij} \wedge s_n \in r_{ij} \right\}, \\
x^{\text{out}}_{r_{ij}}(t) &= \sum_{\forall T \in \mathbf{T}} \sum_{n=1}^{N} \left\{ s_n \in r_{ij} \wedge s_{n+1} \notin r_{ij} \right\},
\end{aligned} \tag{1}
$$

where $s_n \in r_{ij}$ or $s_n \notin r_{ij}$ indicates that the point $s_n$ lies or does not lie within region $r_{ij}$ at time $t$. For each region, we can compute its in-flow $x^{\text{in}}_{r_{ij}}(t)$ and out-flow $x^{\text{out}}_{r_{ij}}(t)$ by aggregating the value for a period of time. Therefore, the traffic flow of the whole city at any time can be represented by a tensor $\mathbf{X} \in \mathbb{R}^{2 \times H \times W}$, which consists of two matrices, i.e., in-flow/out-flow matrix (see Fig. 2).

**Definition 2** (*Intrinsic/Extrinsic Features*)**.** We divide all features at time $t$ into two disjoint categories: Intrinsic and Extrinsic features. Intrinsic features include HourOfDay, DayOfWeek, and Weekday/Weekend, which are direct attributes associated with the traffic characterization (e.g., average speed, traffic density, etc.) at a particular temporal value — which is obtained along sampling of the traffic data itself. The extrinsic features – e.g., temperature, wind speed, holidays and weather — in contrast, need to be obtained via sources that extrinsic to the road networks and traffic itself.

Essentially, flow prediction is a time series problem aiming at predicting the citywide flows at time $t$ given the historical observations up to time $t-1$. More specifically, we define the problem as:

**Problem 1** (*Flow prediction*). Given the historical observations $\mathcal{X}_{t-1} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{t-1}\}$ of previous $t-1$ time steps, and the intrinsic/extrinsic features $F_t^{\text{int}}/F_t^{\text{ext}}$, the task of flow prediction is to learn a model estimating the in-flow/out-flow at time $t$, i.e., predicting the flow $\mathbf{X}_t$ at time step $t$ as $\{\mathbf{X}_t|\mathcal{X}_{t-1}, F_t^{\text{int}}, F_t^{\text{ext}}\}$.

### 3.2. Neural ordinary differential equations

Residual neural networks (ResNet) (He et al., 2016) have skip connections to keep feature maps in different layers in the same scale, and have achieved promising results on many applications such as image recognition and language processing. ResNet is also widely employed as the main block in many traffic prediction models (Zhang et al., 2017, 2019a; Sun et al., 2019; Pan et al., 2019). It consists of multiple residual blocks, transforming a series of discrete hidden states according to:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f\left(\mathbf{h}_t, \theta\right), \tag{2}$$

where $f(\mathbf{h}_t, \theta)$ is some dense or convolutional non-linear function parameterized by $\theta$; $\mathbf{h}_t$ is the input to the $t$th layer.

Recently, neural ordinary differential equations (NODE) (Chen et al., 2018) were introduced, to express the time as a continuous variable, which can be considered as an approximation in the ResNet architecture:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t; \theta), \tag{3}$$

where the hidden state $\mathbf{h}(t)$ represents the continuous dynamics of hidden units, and function $f(\mathbf{h}(t), t; \theta)$ is the continuous form of the nonlinear function $f(\mathbf{h}_t, \theta)$ in Eq. (2). By changing the differential form of Eq. (3) to the integral form, we have:

$$\mathbf{h}(t+1) = \mathbf{h}(t) + \int_t^{t+1} f(\mathbf{h}(t), t; \theta)dt. \tag{4}$$

The continuous NODE described in Eq. (4) has several advantages over the discrete neural networks described in Eq. (2), in terms of less memory and fewer parameters, and faster model convergence (Weinan, 2017; Chen et al., 2018; Gholami et al., 2019; He et al., 2019).

## 4. Model-free evidence

We now explore the influence of different factors that could help in understanding the datasets and provide necessary evidence for our model.

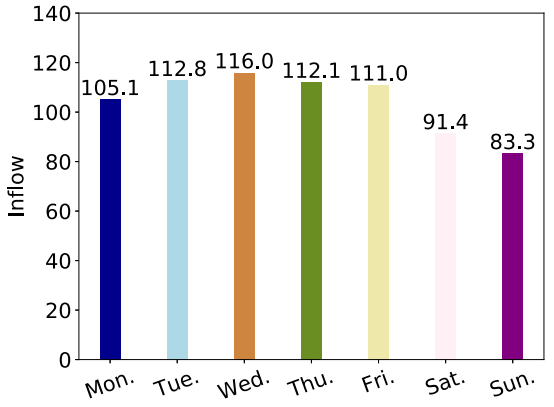
Different factors (e.g., weather, temperature, wind speed) can have different impacts on the traffic flow prediction and the corresponding degrees of influence vary greatly. Taking weather as an example: the crowd flow will decrease drastically in the situation of heavy rain. From a complementary perspective: the flow at noon is different from the flow at evening/night hours in any particular region. In this spirit, we specifically analyze the impact of each factor on the traffic flow by performing data analysis on TaxiBJ dataset (Zhang et al., 2017). According to the source of the factors obtained, we divide these factors into two groups: intrinsic features and extrinsic features. The intrinsic features have periodic characteristics on the temporal axis and can be obtained from the data itself, while extrinsic features have greater uncertainty and randomness, and can only be obtained from extrinsic sources.

**Intrinsic Features**: Fig. 3(a) depicts the averaged inflow over $32 \times 32$ cells from Monday to Sunday which shows how daily crowd flow changes. Among them, the flow reached its maximum value (116 taxis) on Wednesday and the lowest value (83.3) on Sunday. Fig. 3(b) shows that the traffic flow on weekdays is around 28% higher than in weekends which, intuitively could indicate that some people usually prefer to stay at home rather than go outside after a week of work. Fig. 3(c) depicts the changes of traffic flow within one day (averaged by all days). We can observe that the crowd flow reaches its lowest value at 4 am and highest value around 11 am.

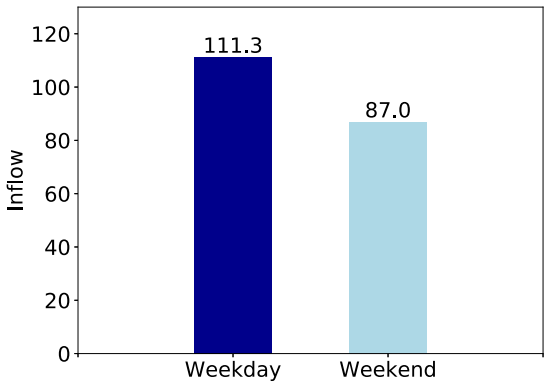
We collectively refer to the above three factors as intrinsic features because it can be collected from the data, which have the same characteristics in terms of time: periodicity – i.e., periodically affecting the traffic flow of an area without considering extra knowledge.

**Extrinsic Features**: Fig. 4(a) shows the statistics of the average traffic for all holidays and non-holidays, from which we can see that the work day traffic is 1.52 times over the holidays. Fig. 5(a) shows that during the holiday of Spring Festival in China (7 days), the traffic is significantly lower than normal days (the week before Spring Festival). Furthermore, Fig. 4(b) describes 15 types of weather conditions[1] that affect traffic flow differently. In order to highlight the impact of the weather on traffic flow prediction, we compare the traffic flow observed on a heavy rain weather with the traffic flow on the same day but a week prior, and without any rain. The results are shown in Fig. 5(b), which clearly demonstrates that bad weather (i.e., heavy rain) can significantly reduce the traffic flow.

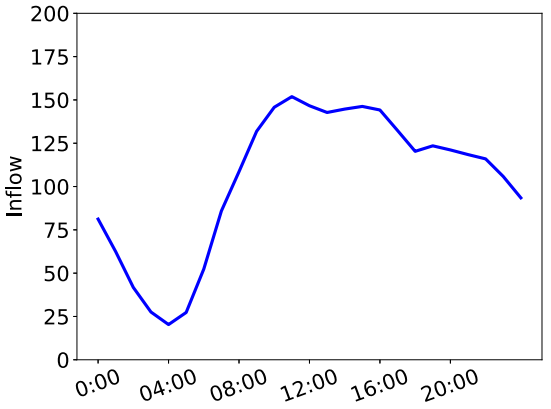
[1] The 15 weather conditions include: Sunny, Rainy, HeavyRain, Cloudy, Overcast, Sprinkle, Snowy, Rainstorm, Thunderstorm, FreezingRain, LightSnow, Dusty, Sandstorm, Foggy, and HeavySnow.




(a) DayOfWeek




(b) Weekday/Weekend




(c) HourOfDay

**Fig. 3.** The effect of intrinsic features on the traffic flow. We plot the average results of regions in TaxiBJ data.

Likewise, we also consider factors such as temperature and wind speed. Fig. 4(c) depicts how the traffic slowly increases as the temperature rises. Fig. 4(d) shows that the traffic increases steadily until the wind speed goes up to 20 mph. However, the flow suddenly decreases as the wind speed goes beyond it.

We categorize the above four factors as extrinsic factors because accessing these features relies on external sensing devices (temperature and wind speed) or domain knowledge.

In summary, the above evidence indicates that these two different types of features (intrinsic/extrinsic) may play an important role in traffic flow prediction. Hence, we explicitly incorporate these features into our proposed traffic prediction model and in Section 6 we demonstrate that these two types of features greatly improve our prediction results.

## 5. TODE fundamentals

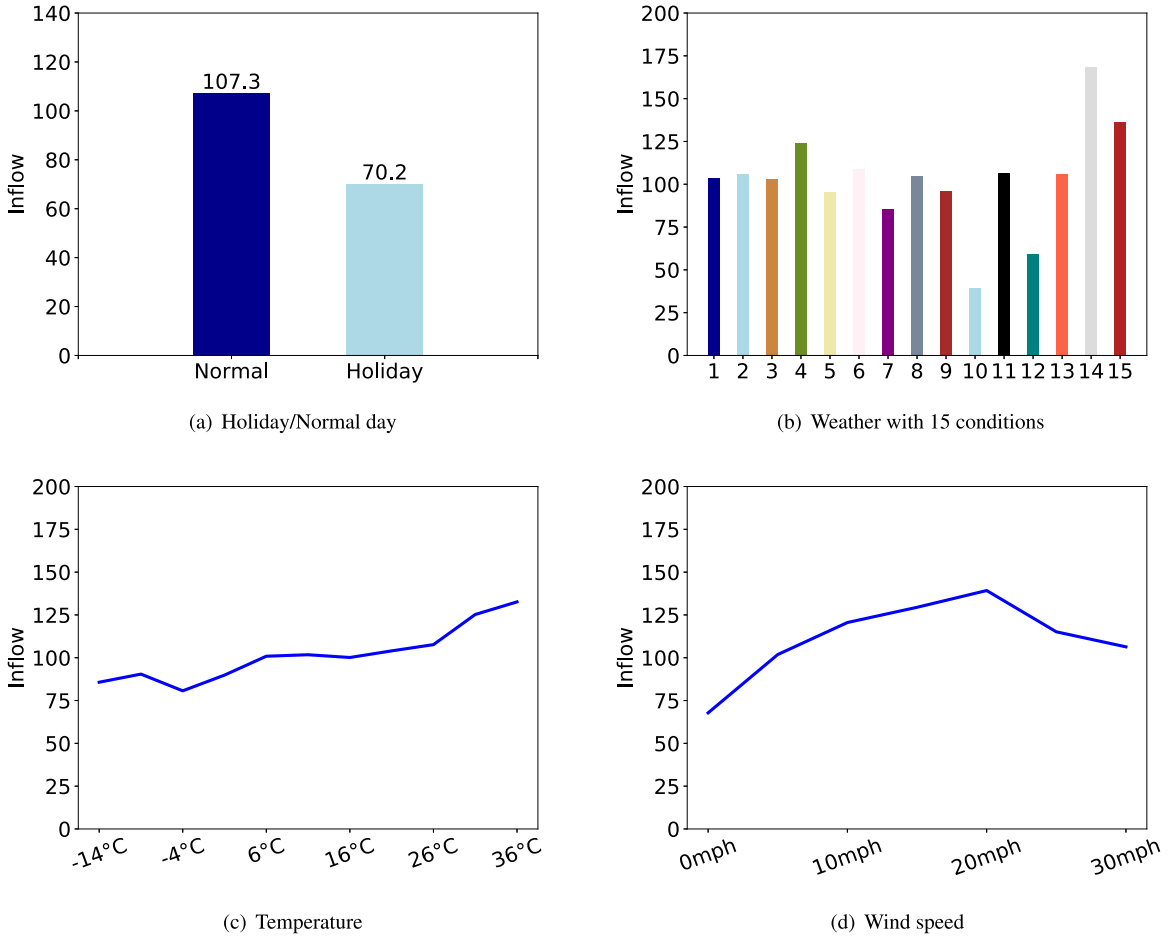
In this section, we describe the details of our proposed model TODE, which consists of four main components.

**(A) Time step selection:** We divide all observed data into three levels of sequences: recent, daily and weekly following (Zhang et al., 2017). For each training instance, it consists of the above three sequences and associated factors $F_t$.

**(B) Feature embedding:** All features are selectively fed into different embedding layers according to its feature category (continuous or categorical).

**(C) Spatial–Temporal ODE:** ST-ODE models the spatial and temporal dependencies of traffic flow with the stacking neural ODE blocks. It also plays a role of correlating the traffic flow (A) and the multi-factors (B) to learn the influence of factors on traffic prediction, which can discriminate the importance of different factors through learning from historical traffic data.

**(D) Multi-information fusion:** A parametric-matrix-based model is employed to learn the different weights of three components (i.e. closeness, period and trend) – the closeness component is designed to handle the short-term neighboring dependencies, while period and trend components aim to deal with the long-term periodical dependencies.



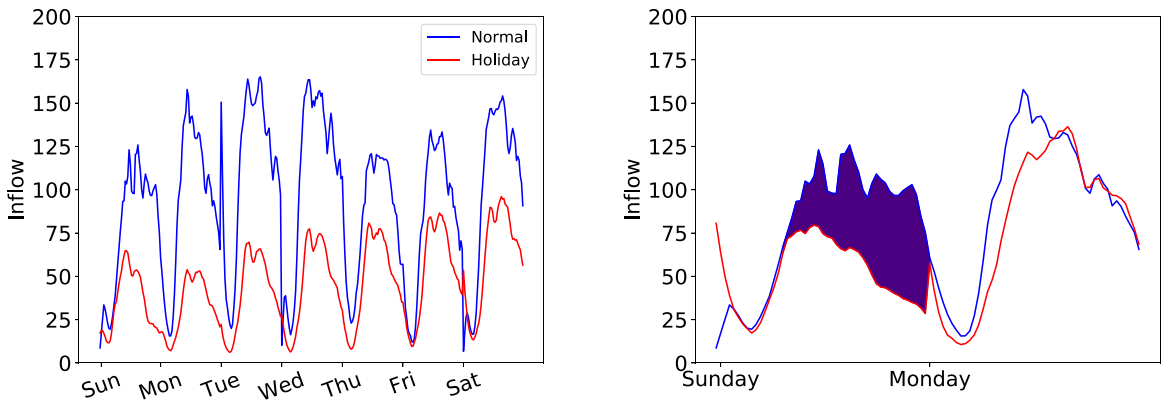

(a) Holiday/Normal day

(b) Weather with 15 conditions

(c) Temperature

(d) Wind speed

**Fig. 4.** The effect of extrinsic features on the traffic flow. We plot the average results of regions in TaxiBJ data.




(a) Holiday

(b) Weather

**Fig. 5.** Holiday and Weather influence on the traffic flow. (a) Feb 7–13 (red), Jan 31–Feb 6 (blue). (b) Nov 22–23 (red), Nov 29–30 (blue), 2015. The shaded area indicates that heavy rain greatly reduce the traffic flow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

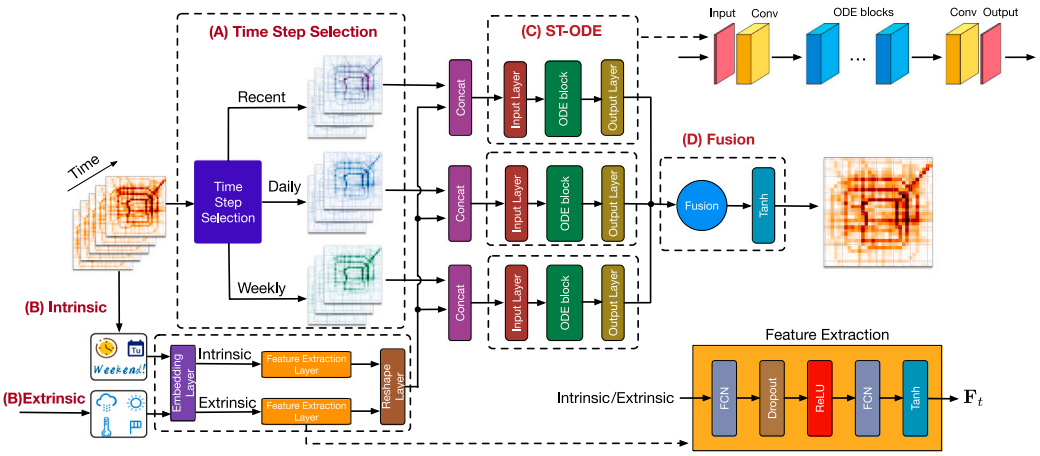



**Fig. 6.** TODE architecture. FCN denotes a fully-connected neural network; $\mathbf{F}_t$ denotes the output of feature extraction layer at time $t$.

### 5.1. Data partition

We consider three types of dependent sequences as key time steps on historical data $\mathcal{X}_{t-1}$, including closeness (recent), period (daily) and trend (weekly). Closeness refers that we sample data at short-time interval (e.g., half an hour) from the recent traffic flow. Similarly, the period and trend indicate that sampling is performed at a day and week level, respectively:

$$\begin{aligned}\mathbf{I}_c &= \text{concat}[\mathbf{X}_{t-l_c}, \dots, \mathbf{X}_{t-2}, \mathbf{X}_{t-1}],\\ \mathbf{I}_p &= \text{concat}[\mathbf{X}_{t-l_p*p}, \dots, \mathbf{X}_{t-2*p}, \mathbf{X}_{t-p}],\\ \mathbf{I}_q &= \text{concat}[\mathbf{X}_{t-l_q*q}, \dots, \mathbf{X}_{t-2*q}, \mathbf{X}_{t-q}],\end{aligned} \tag{5}$$

where $l_c$, $l_p$, $l_q$ are input lengths of closeness, period, and trend sequences, respectively; $p$ represents the daily period and $q$ denotes the weekly span; $\mathbf{I}_*(*\in\{c,p,q\})$ are the input tensors, i.e., $\mathbf{I}_c \in \mathbb{R}^{H\times W\times 2l_c}$, $\mathbf{I}_p \in \mathbb{R}^{H\times W\times 2l_p}$ and $\mathbf{I}_q \in \mathbb{R}^{H\times W\times 2l_q}$. After data partition, each sample is composed by three types of tensors. The first input of the model is $\{\mathbf{X}_{t-\tau}, \mathbf{X}_{t-\tau-1}, \dots, \mathbf{X}_{t-1}, \mathbf{X}_{t-p}, \mathbf{X}_{t-q}\}$, which suggests that the observations are not regularly sampled. In this work, we set $\tau = 3$, i.e., each sample consists of three most recent previous samples, a sample of a day before and a sample of a week before. In this way, we can extract two types of temporal dependencies – short-term neighboring (recent) and long-term periodic (daily and weekly) dependencies (Zhang et al., 2017).

### 5.2. Feature extraction and embedding

As stated above, we first divide all factors into intrinsic features $F_t^{\text{int}}$ and extrinsic features $F_t^{\text{ext}}$ according to different data sources. Based on our analysis in Section 4, we demonstrate that different types of features have different effects on traffic flow, e.g., intrinsic features are temporally periodical and extrinsic features are usually uncertain. Inspired by these facts, we use two fully-connected networks to capture different feature information. As shown in the lower right corner of Fig. 6, we selectively embed the features — continuous features (e.g. temperature and wind speed) are fed directly into the feature extraction layer, while categorical features (e.g. DayOfWeek and HourOfDay) are embedded as the low-dimensional vectors before feeding into the extraction layer. During feature extraction, fully-connected network (FCN) is used for learning initialization, while leveraging ReLU (Hahnloser et al., 2000) activation function to introduce non-linearity and dropout (Srivastava et al., 2014) to avoid overfitting. Meanwhile, we use tanh (Malfliet and Hereman, 1996) as the activation function in the last layer of feature extraction network to bound the output range within $[-1, 1]$, which is the same as the range of the elements in $\mathbf{X}_t$. Afterwards, we concatenate $F_{\text{int}}^{\text{out}}$ and $F_{\text{ext}}^{\text{out}}$ as the final feature output $\mathbf{F}_t$. Moreover, to make $\mathbf{F}_t$ have the same size of $\mathbf{X}_t$, we use a reshape layer $\mathcal{R}$. Finally, the feature output $\mathbf{F}_t$ can be written as:

$$F_{t,\text{out}}^{\text{int}} = \tanh\left(\text{ReLU}\left(F_t^{\text{int}} W_0 + b_0\right) W_1 + b_1\right), \tag{6}$$

$$F_{t,\text{out}}^{\text{ext}} = \tanh\left(\text{ReLU}\left(F_t^{\text{ext}} W_2 + b_2\right) W_3 + b_3\right), \tag{7}$$

$$\mathbf{F}_t = \mathcal{R}[F_{t,\text{out}}^{\text{int}}, F_{t,\text{out}}^{\text{ext}}], \tag{8}$$

where $F_{t,\text{out}}^{\text{int}}$ and $F_{t,\text{out}}^{\text{ext}}$ denote the output of the intrinsic features and extrinsic features at time $t$, respectively; $W_*$ and $b_*$ represent the learnable weight matrix and bias, respectively.

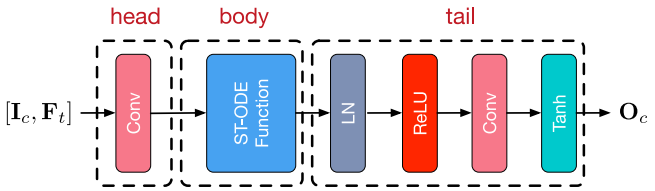



**Fig. 7.** The neural architecture of ST-ODE network. LN: layer normalization layer. Conv: convolution layer.

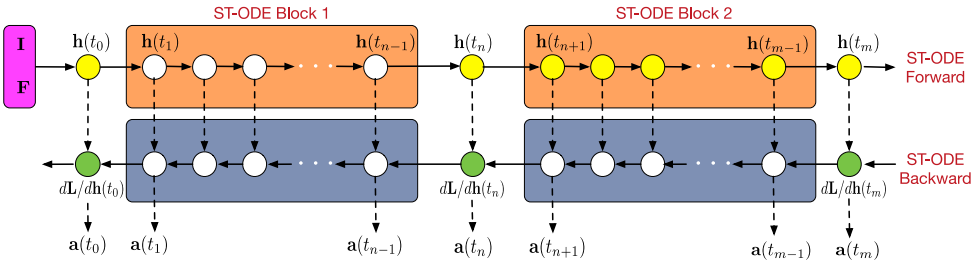



**Fig. 8.** Illustration of forward and backward in two ODE blocks with $n$ time steps in first ODE block and $m-n$ time steps in the second ODE block. During forward pass, the circles in the yellow area denote that we need to store these points in memory, which has the memory complexity of $\mathcal{O}(L_{\mathrm{O}} + N_{\mathrm{O}})$. Then, we used these points to solve the adjoint backwards using the DTO method. After we compute the gradients $d\mathbf{L}/dh_t$ and obtain the adjoint states $\mathbf{a}_t$, the memory allocated to the second ODE block will be released and reused for the first ODE block.

### 5.3. Spatial–temporal ODE (ST-ODE)

In previous studies, researchers often use ResNet (He et al., 2016) to build a deep neural network for spatial–temporal dependency learning (Zhang et al., 2017; Liang et al., 2019). However, as the number of residual blocks increases, memory costs and parameter size increase significantly which prevents these models from being deployed in IoT settings. To remedy this problem, we use the neural ODE to learn the spatial–temporal dependencies in our model. By adopting the so-called adjoint method, the memory cost reduces to $\mathcal{O}(L_{\mathrm{O}} + N_{\mathrm{O}})$ from $\mathcal{O}(L_{\mathrm{R}} N_{\mathrm{R}})$, where $L_{\mathrm{R}}$ denotes the number of layers in the ResNet, and $L_{\mathrm{O}}$ is the number of ODE blocks. $N_{\mathbf{O}}$ or $N_{\mathrm{R}}$ denotes the number of time steps in one ODE block or in one residual unit. Note that $N_{\mathrm{O}}$ depends on the methods to solve the differential equation, which can be adaptively specified according to the balancing between prediction precision and computational cost. In addition, the parameters in ST-ODE network are reusable.

The ST-ODE module contains three sub-components: input layer, ODE blocks and output layer, as illustrated in the upper right corner of Figs. 6 and 7. ST-ODE starts with a simply convolution layer (input layer), which turns the traffic flow data from 2-channels (image-like matrix for in-flow and out-flow) to 64-channels and can be seen as the head of ST-ODE network producing the initial state $\mathbf{h}_0$:

$$\begin{aligned}
\mathbf{h}_{c,0} &= W_c \otimes \mathrm{concat}[\mathbf{I}_c, \mathbf{F}_t] + b_c, \\
\mathbf{h}_{p,0} &= W_p \otimes \mathrm{concat}[\mathbf{I}_p, \mathbf{F}_t] + b_p, \\
\mathbf{h}_{q,0} &= W_q \otimes \mathrm{concat}[\mathbf{I}_q, \mathbf{F}_t] + b_q,
\end{aligned} \tag{9}$$

where $W_c$, $W_p$ and $W_q$ are the elements of the convolutional kernels and $\otimes$ denotes the convolution operation.

In each ODE block, as shown in Fig. 8, we consider two separate parts: forward pass and back propagation.

**Forward pass:** Because the integration is not easy to compute, we turn to some numerical methods to approximate the integral. Among them, the two most classic methods are the Euler method (Eq. (10)) and the Runge–Kutta method (RK4). In this paper, we use the Euler method, which approximates $\mathbf{h}(t)$ over the entire integration time and can be expressed as follows:

$$\mathbf{h}(t + \Delta t) = \mathbf{h}(t) + \Delta t f\left(t, \mathbf{h}(t); \theta\right), \tag{10}$$

where $\Delta t$ indicates the time interval in one time step. Then, if we take $n$ time steps as check points, we solve it through a one-step update:

$$\mathbf{h}_n = \mathbf{h}_{n-1} + \Delta t f\left(t_{n-1}, \mathbf{h}_{n-1}; \theta\right), \tag{11}$$

where $\mathbf{h}_n$ represents the hidden state at time point $t = n\Delta t$.

**Back propagation:** In order to train the ST-ODE model, we need to compute the gradients of the loss with respect to the parameters $\theta$. After solving Eq (4) using the Euler method, we can easily update parameters as follows:

$$\theta := \theta - \eta \frac{\partial \mathbf{L}(\mathbf{h}_n, \mathbf{h}_{\mathrm{targets}}; \theta)}{\partial \theta} = \theta - \eta \frac{\partial \mathbf{L}}{\partial \mathbf{h}_n} \frac{\partial \mathbf{h}_n}{\partial \mathbf{h}_{n-1}} \cdots \frac{\partial \mathbf{h}_1}{\partial \mathbf{h}_0} \frac{\partial \mathbf{h}_0}{\partial \theta}, \tag{12}$$

where $\mathbf{L}$ is the loss w.r.t. the output state, $\mathbf{h}_{\text{targets}}$ is the target hidden state at the end of ST-ODE, and $\eta$ is the learning rate. However, the memory cost can be huge since we need to store the intermediate activation at each checkpoint. To address this problem, an adjoint method is used to only store the activation at the end of the ODE layer:

$$\mathbf{a}(t_n) = \frac{\partial \mathbf{L}}{\partial \mathbf{h}(t_n)}, \tag{13}$$

$$\frac{d\mathbf{a}(t)}{d\mathbf{t}} = -\mathbf{a}(t)\frac{\partial f(\mathbf{h}(t), t; \theta)}{\partial \mathbf{h}(t)}dt, \tag{14}$$

$$\mathbf{a}(t_0) = \mathbf{a}(t_n) - \int_{t_n}^{t_0} \mathbf{a}(t)\frac{\partial f(\mathbf{h}(t), t; \theta)}{\partial \mathbf{h}(t)}dt, \tag{15}$$

where Eq. (13) is the adjoint equation and $\mathbf{a}(t_n)$ is the so-called adjoint state at time $t_n$. By Eq. (15), we can recalculate the adjoint state at start time $t_0$ by solving the ODE layers backwards from end time $t_n$. That is, we can compute the gradients w.r.t. the parameters $\theta$ (Eq. (16)) and update $\theta$ by Eq. (17) instead of the Eq. (12), as follows:

$$\frac{\partial \mathbf{L}}{\partial \theta} = \int_{t_n}^{t_0} \mathbf{a}(t)\frac{\partial f(\mathbf{h}(t), t; \theta)}{\partial \theta}dt, \tag{16}$$

$$\begin{aligned}\theta :&= \theta - \eta\frac{\partial \mathbf{L}}{\partial \theta} \\ &= \theta - \eta\int_{t_n}^{t_0} \mathbf{a}(t)\frac{\partial f(\mathbf{h}(t), t; \theta)}{\partial \theta}dt.\end{aligned} \tag{17}$$

With the adjoint method, we can obtain the adjoint state at *any* desirable checkpoints, and it only needs to store the output state at $t_n$. However, the gradient calculated in this way may be incorrect. Inspired by (Gholami et al., 2019; Zhang et al., 2019b), we use a discretize-then-optimize (DTO) method to effectively avoid numerical instability and divergence problems when training our model. To eliminate the ODE constraint and find the first-order optimality condition, we form the Lagrangian equation as:

$$\begin{aligned}&\mathcal{F}(\mathbf{h}, \theta, \mathbf{a}) = \mathbf{L}(\mathbf{h}_n, \mathbf{h}_{\text{targets}}; \theta) + \int_{\Psi} \mathbf{a}_0 \left(\mathbf{h}_0 - \mathbf{h}_0\right) d\Psi \\ &+ \int_{\Psi} \mathbf{a}_1 \left(\mathbf{h}_1 - \mathbf{h}_0 - \Delta t f\left(\mathbf{h}_0, \theta, t_0\right)\right) d\Psi \\ &\cdots \\ &+ \int_{\Psi} \mathbf{a}_n \left(\mathbf{h}_n - \mathbf{h}_{n-1} - \Delta t f\left(\mathbf{h}_{n-1}, \theta, t_{n-1}\right)\right) d\Psi \\ =&\mathbf{L}(\mathbf{h}_n, \mathbf{h}_{\text{targets}}; \theta) + \sum_{i=1}^{n}\int_{\Psi} \mathbf{a}_i \left(\mathbf{h}_i - \mathbf{h}_{i-1} - \Delta t f\left(\mathbf{h}_{i-1}, \theta, t_{i-1}\right)\right) d\Psi,\end{aligned} \tag{18}$$

where $\mathbf{a}_i$ is the adjoint state corresponding to the $i$th time step and $d\Psi$ represents the channels of activation map in ST-ODE. $\mathcal{F}(\mathbf{h}, \theta, \mathbf{a})$ is a new function consisting of an objective function, a constraint and a Lagrange multiplier. Then, by taking variations w.r.t. the hidden state $\mathbf{h}_i$ and parameters $\theta$, we have:

$$\frac{\partial \mathcal{F}}{\partial \mathbf{h}_i} = 0, \frac{\partial \mathcal{F}}{\partial \theta} = 0, \tag{19}$$

$$\mathbf{a}_n = \frac{\partial \mathbf{L}}{\partial \mathbf{h}_n}, \tag{20}$$

$$\mathbf{a}_{i-1} = \mathbf{a}_i\left(I + \Delta t\frac{\partial f(\mathbf{h}_{i-1}, t_{i-1}; \theta)}{\partial \mathbf{h}_{i-1}}\right), \tag{21}$$

$$\frac{\partial \mathbf{L}}{\partial \theta} = \sum_{i=1}^{n} \mathbf{a}_i \Delta t\frac{\partial f(\mathbf{h}_{i-1}, t_{i-1}; \theta)}{\partial \theta}. \tag{22}$$

Note that with the DTO method, as shown in Fig. 8, we need to store intermediate activation maps of $\mathbf{h}_i$ for $i = n, n+1, \cdots, m$. Afterwards, we use the DTO method to calculate the gradient of the current ODE block until we get the gradient $d\mathbf{L}/d\mathbf{h}_n$. Then, the memory allocated to the current ODE block will be released and reused for the previous ODE block — whose intermediate activation maps are therefore $\mathbf{h}_i$ for $i = 0, 1, \cdots, n$. Meanwhile, the parameters $\theta$ (c.f Eq. (17)), can be updated as:

$$\begin{aligned}\theta :&= \theta - \eta\frac{\partial \mathbf{L}}{\partial \theta} \\ &= \theta - \eta\sum_{i=1}^{n} \mathbf{a}_i \Delta t\frac{\partial f(\mathbf{h}_{i-1}, t_{i-1}; \theta)}{\partial \theta},\end{aligned} \tag{23}$$

which means that the memory complexity is now reduced to $\mathcal{O}(L_{\text{O}} + N_{\text{O}})$.

Now, we describe the neural networks used in ST-ODE. As shown in Fig. 9, we stack two identical sub-units, each including a Group Normalization (GN) (Wu and He, 2018) layer, a convolution layer (with 64 filters of kernel size of $3 \times 3$) and a ReLU activation function. With this ODE function structure, we can stack deep convolutional network, which aims to further capture spatial distant dependencies without suffering from the limits of memory cost and parameters. Particularly, because the parameters

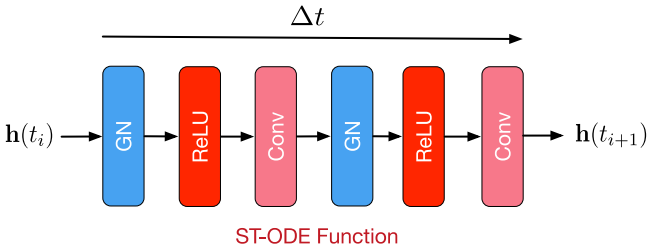



**Fig. 9.** The neural architecture of ST-ODE function. GN: group normalization layer. Conv: convolution layer.

in the ST-ODE function are reusable, we use *group normalization* instead of *batch normalization* (Ioffe and Szegedy, 2015) in the ST-ODE.

Once we have the ST-ODE function $f$, the initial condition $\mathbf{h}(t_0)$ and the integration time $[t_0, t_n]$, we can readily obtain the final hidden state and compute the correct gradient via the DTO approach for a Euler time stepping scheme (Eq. (10)) to solve Eq. (4).

At the tail of our ST-ODE network, we use a Layer Normalization (Ba et al., 2016) before the final prediction. After ReLU function, the convolution layer restores the tensors from 64 channels back to 2-channel (in-flow and out-flow) image-like matrix, while the tanh activation function maps the predictions to $[-1, 1]$. Finally, we can get the output of the three components represented by $\mathbf{O}_c$, $\mathbf{O}_p$, and $\mathbf{O}_q$.

### 5.4. Deep feature fusion

In previous studies (Zhang et al., 2017; Sun et al., 2019), FCN are used to model extrinsic factors which are then directly merged with the output of the spatial–temporal learning like:

$$\widehat{\mathbf{X}}_t = \tanh\left(\mathbf{O}_{ne} + \mathbf{F}_t\right), \tag{24}$$

where $\widehat{\mathbf{X}}_t$ represents the final traffic flow forecast and $\mathbf{O}_{ne}$ is the output of the three spatial–temporal learning components without combining extrinsic factors. However, there is only a slight improvement in directly adding the predicted traffic flow and factor influence, and sometimes even negatively affects the model prediction. For this phenomenon, we can explain it from three aspects: (a) The two fully-connected layers are always shallow neural networks, the factors that pass such a shallow network structure cannot impose real impacts on the traffic flow prediction. (b) It is unreasonable to pass historical traffic flow data and extrinsic factors through different networks, which greatly reduces the correlation and dependencies between the two. And it also misleads the model to ignore the factor information and hence fails in the traffic flow prediction task. (c) They did not distinguish the intrinsic and extrinsic features. Between the two, intrinsic features exhibit good characteristics regarding periodicity patterns. These intrinsic features combined with sequential traffic flow data allow TODE to more accurately predict traffic for the next-time point. However, if we separate the factors and traffic flow data and feed them into different networks, then the factors' influence to the traffic flow is likely to be ignored.

To address this problem, we make a slight modification that combines the factors $\mathbf{F}_t$ with the spatial–temporal learning component inputs, and feed them through the ST-ODE, which can be represented as:

$$\begin{aligned}
\mathbf{O}_c &= \tanh\left(\text{ST-ODE}([\mathbf{I}_c, \mathbf{F}_t])\right), \\
\mathbf{O}_p &= \tanh\left(\text{ST-ODE}([\mathbf{I}_p, \mathbf{F}_t])\right), \\
\mathbf{O}_q &= \tanh\left(\text{ST-ODE}([\mathbf{I}_q, \mathbf{F}_t])\right).
\end{aligned} \tag{25}$$

With this deep feature fusion component, the positive impact of factors on the traffic flow prediction can be significantly enhanced, which is consolidated by our experiments.

### 5.5. Multi-information fusion & optimization

We now discuss how to fuse the information learned from the components for final results prediction. For the human crowd forecasting, the general trend of traffic flow in different regions is cyclical. Thus, the effects of the period component and trend component in the model are more crucial than the closeness component. However, if there is a special event (e.g. car accident) in the most recent time period, the closeness component would play a significant role in the prediction task. In addition, in different areas, special events have different effects on the traffic flow. For example, if a car accident occurs in a traffic artery, it will have a very large impact on the traffic for a long while. If the car accidents occur in the suburb areas, then its impact is usually trivial. In the other words, the closeness, period and trend can affect the traffic flow in different regions in different degree. Inspired by this, we learn the different weights of the three components as below:

$$\mathbf{O}_t = \mathbf{W}_c \odot \mathbf{O}_c + \mathbf{W}_p \odot \mathbf{O}_p + \mathbf{W}_q \odot \mathbf{O}_q, \tag{26}$$

**Table 2**
Statistics of datasets.

| Dataset | TaxiBJ | BikeNYC |
|---|---|---|
| Time span | P1:7/1/2013-10/30/2013<br>P2:3/1/2014-6/30/2014<br>P3:3/1/2015-6/30/2015<br>P4:11/1/2015-4/10/2016 | 4/1/2014-9/30/2014 |
| Data type | Taxi GPS | Bike rent |
| Location | Beijing | New York |
| Time interval | 30 min | 1 h |
| Grid map size | 32 × 32 | 16 × 8 |
| **Intrinsic Features (day, hour, weekend)** | | |
| Day | Mon-Sun | Mon-Sun |
| Hour | 24 h | 24 h |
| weekend | 0, 1<br>(0 for weekday, 1 for weekend) | 0, 1 |
| **Extrinsic Features (meteorology, holidays)** | | |
| Temperature/°C | [−24.6, 41.0] | \ |
| Wind speed/mph | [0, 48.6] | \ |
| Weather conditions | 15 types (*e.g.*, Rainy, Sunny) | \ |
| Holidays | 41 | 20 |

where $\mathbf{W}_c$, $\mathbf{W}_p$, and $\mathbf{W}_q$ are learnable parameters that determine how much each component affects the final prediction, and $\odot$ is the Hadamard product between tensors. In the previous section, we have explained that if we directly merge the output of the three components with the factors, the impact of factors will be ignored in the end. Therefore, here, we only use tanh as the final activation function to map the predictions to [−1, 1] and get the final traffic flow forecast as follows:

$$\hat{\mathbf{X}}_{\mathbf{t}} = \tanh\left(\mathbf{O}_t\right), \tag{27}$$

where $\hat{\mathbf{X}}_t$ is the final traffic flow prediction at time $t$.

In this work, we aim to train the model before time $t$ and then predict the in-flow and out-flow at time $t$. By minimizing the mean squared error between the real traffic flow and the predicted value, the loss function is defined as:

$$\mathcal{L}(\theta) = \left\|\mathbf{X}_t - \hat{\mathbf{X}}_t\right\|_2^2, \tag{28}$$

where $\mathbf{X}_t$ is the real traffic flow and $\hat{\mathbf{X}}_t$ is the predicted value.

## 6. Experimental observations

We now present in detail the experimental observations demonstrating the advantages of the TODE model in three main aspects: parameters, memory cost and prediction performance.

### 6.1. Experimental settings

#### 6.1.1. Datasets

To compare TODE with different methods and verify its validity, we evaluate the TODE using two real-world traffic flow datasets. Each dataset contains two sub-datasets: traffic flow and extrinsic factors. A detailed description of the two datasets is shown in Table 2.

- **TaxiBJ**: This dataset is provided by Zhang et al. (2017). It is the taxicab GPS data in Beijing and is divided into four time intervals, e.g., July 1st 2013 to October 31st 2013, etc. More specifically, each trajectory is mapped into a 32 × 32 grids with two types of flows in each grid: in-flow and out-flow.
- **BikeNYC**[2]: This dataset is published by Zhang et al. (2017), and contains data from April 1st, 2014 to September 30th, 2014. Each trip data is mapped into a 16 × 8 grids containing start/stop time, start/stop station IDs and trip duration.

For fair comparison, we follow the pre-processing method in previous works (Zhang et al., 2017; Yao et al., 2019; Sun et al., 2019; Zhang et al., 2019a; Diao et al., 2019). First, we partition the traffic flow into three parts: recent, daily, and weekly of the crowd flow. We utilize the Min–Max normalization method for scaling the crowd volume and the extrinsic features (e.g. wind speed, weather, and temperature) into the range [-1,1]. Naturally, we re-scale the predicted flow and restore back to the real flow in the test. To map the output of the TODE model to [-1,1], we choose tanh as the final activation function that limits the predicted traffic flow in the range.

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

In addition to adding extrinsic factors, we extract some intrinsic features from the data itself, including DayOfWeek, HourOfDay, and Weekday/Weekend. In order to make the categorical features in the model well expressed, we embed them with different embedding layers to obtain low-dimensional vector representation. To ensure the integrity of the data during the day, we also delete the days without 48 timestamps in TaxiBJ and the days without 24 timestamps in BikeNYC.

#### 6.1.2. Baselines

We compare our TODE with the following 11 baselines:

- **Historical Average (HA)**: models historical average data of a certain time period to predict the traffic flow in the next time period. We average the crowd flow of last 5 days to estimate the values.
- **ARIMA** (Reis and Mandl, 2003): is a well-known time series model that combines autoregressive (AR) and moving average (MA) for prediction. For each grid region, we build one ARIMA model for the in/out crowd flow density prediction.
- **SARIMA**: A Seasonal ARIMA (SARIMA) model is formed by including additional seasonal terms in the ARIMA models. Here, we implemented it with the season length equals to a day or a week based on the prediction performance.
- **VAR**: Vector Auto-Regressive is usually used as spatial–temporal model which can capture the pairwise relationships among the in-flow and out-flow. Note that it is computationally costly due to involving a large number of parameters.
- **ST-ANN**: is based on artificial neural network which extracts spatial features from the values of the surrounding regions (nearby 8 regions) and temporal features using the previous eight time steps.
- **DeepST** (Zhang et al., 2016): is a deep neural network (DNN)-based prediction model for traffic flow data prediction, which is composed of three components: temporal closeness, period and seasonality trend capturing, convolutional neural networks for spatial closeness dependency learning, and extrinsic factor fusions.
- **ST-ResNet** (Zhang et al., 2017): builds a model by stacking residual networks and shows state-of-the-art performance on the crowd flow prediction. Specifically, ST-ResNet employs three residual neural networks to model the temporal closeness, period, and trend properties of the crowd flow. Likewise, some extrinsic factors, such as weather conditions and events, are also added into the fusion layer for the final prediction.
- **ST-3DNet** (Guo et al., 2019b): introduces 3D convolutions to capture the correlations of crowd flow data in both spatial and temporal dimensions. It presents a re-calibration block to quantify the difference of the contribution of the correlations in space.
- **DeepSTN** (Lin et al., 2019): is a deep learning-based convolutional model for predicting crowd flows in the metropolis. It employs a ConvPlus structure to model the long-range spatial dependence among crowd flows in different regions. Note that there are several variants of the original DeepSTN. They contain extra information such as POI distributions and time factors. For a fair comparison, we only compare to the vanilla model.
- **STDN** (Yao et al., 2019): utilizes a flow-gated local CNN to handle dynamic spatial similarity between regions based on the local flow information, and use a periodically shifted attention mechanism to handle long-term periodic temporal shifting of traffic flows, while LSTM is used to model the sequential dependency in a hierarchical way.
- **DeepLGR** (Liang et al., 2020b): extracts local region representations using a squeeze-and-excitation network, and aggregates the region representations using pooling operations. It introduces an upsampling process to generate global-aware features.
- **MVGCN** (Sun et al., 2019): models the spatial correlations and interactions between different regions with graph convolutional networks, in which each node represents a region with time-varying flows, and uses different views to capture different factors in irregular regions. ResNet is also employed to stack more GCN layers to train a deeper network. It builds several GCNs for different temporal periods (e.g. daily, weekly and yearly) that provide multi-view of traffic flow modeling.
- **MDL** (Zhang et al., 2019a): is a multitask deep learning framework for simultaneously predicting in/out flow (node flow) and transitions (edge flow) in a spatial–temporal network. The two networks are connected by coupling shared representation layers and are trained together. A gating component is employed to fuse the extrinsic factors with the spatial–temporal traffic flow correlations.
- **DGCNN** (Diao et al., 2019): is a dynamic spatial–temporal graph convolution neural network that models the evolution of spatial dependencies. It uses tensor decomposition operations to extract the global and local features from traffic samples. It learns the Laplacian matrix at each time step dynamically, which is sent to the graph convolutional layers for flow forecasting.

#### 6.1.3. Parameter settings

We implemented our model based on PyTorch and Python3.6 on a machine with two NVIDIA GeForce GTX 2080ti GPUs. The three dependent sequence components (closeness, period, trend) have the same structural networks: the ODE input layer, the ST-ODE block, and the ODE output layer. More specifically, the ODE input layer includes a convolution with 64 filters of size $3 \times 3$, its stride and padding are set to 1. The output layer contains a convolution with 2 filters of size $3 \times 3$. In the experiments, we only use one ST-ODE block because it outperforms all baselines — although it is of interest to further study the deeper stacking of ST-ODE, which is left for future work. Moreover, the batch size is selected from $\{64, 32, 16, 8\}$, and the corresponding epoch number is selected from $\{300, 250, 200, 150\}$, respectively. The learning rate is set to 0.0001 at the beginning. We utilize the ADAM optimizer (Kingma and Ba, 2014) to train our model with the setting of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = e^{-7}$. From the second epoch, we cut the learning rate by half every 30 epochs. For the lengths of three dependent sequences, we set them as: $l_c$, $l_p$, $l_q = 3, 1, 1$. Besides, in the TaxiBJ dataset, we select last four weeks (28 days) as testing data and the rest as training data. In the BikeNYC dataset, we choose last 10 days as testing data, and all data before that as training data.

**Table 3**
Comparison of prediction accuracy on TaxiBJ.

| Method | RMSE | MAE | MAPE |
|---|---|---|---|
| HA | 44.99 | 24.16 | 0.56 |
| ARIMA | 22.78 | 13.02 | 0.49 |
| SARIMA | 26.88 | 14.93 | 0.51 |
| VAR | 22.88 | 13.11 | 0.47 |
| ST-ANN | 19.57 | 11.51 | 0.45 |
| DeepST | 18.18 | 10.69 | 0.43 |
| DGCNN | 16.83 | 10.11 | 0.44 |
| MVGCN | 16.77 | 9.99 | 0.43 |
| MDL | 16.71 | 9.95 | 0.42 |
| ST-ResNet | 16.69 | 9.91 | 0.40 |
| ST-3DNet | 16.45 | 9.87 | 0.39 |
| STDN | 16.12 | 9.85 | 0.38 |
| DeepSTN | 16.05 | 9.73 | 0.36 |
| DeepLGR | 15.89 | 9.62 | 0.35 |
| **TODE (ours)** | | | |
| **TODE-GN** | **15.64** | **9.39** | **0.29** |
| **TODE-LN** | **15.53** | **9.35** | **0.29** |

**Table 4**
Comparison of prediction accuracy on BikeNYC.

| Method | RMSE | MAE | MAPE |
|---|---|---|---|
| HA | 19.21 | 9.50 | 1.36 |
| ARIMA | 10.07 | 4.86 | 0.71 |
| SARIMA | 10.56 | 5.22 | 0.74 |
| VAR | 9.92 | 4.83 | 0.70 |
| DeepST | 7.43 | 3.60 | 0.58 |
| DGCNN | 6.40 | 3.15 | 0.52 |
| MVGCN | 6.36 | 3.11 | 0.51 |
| MDL | 6.34 | 3.09 | 0.49 |
| ST-ResNet | 6.33 | 3.09 | 0.48 |
| ST-3DNet | 6.22 | 3.04 | 0.46 |
| STDN | 6.20 | 2.99 | 0.46 |
| DeepSTN | 6.18 | 2.98 | 0.45 |
| DeepLGR | 6.15 | 2.96 | 0.45 |
| **TODE (ours)** | | | |
| **TODE-LN** | **6.06** | **2.85** | **0.42** |
| **TODE-GN** | **6.05** | **2.84** | **0.41** |

#### 6.1.4. Evaluation metrics

We evaluate our model using three popular and widely used metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE):

$$RMSE = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\hat{\mathbf{X}}_t - \mathbf{X}_t)^2}, \tag{29}$$

$$MAE = \frac{1}{M}\sum_{i=1}^{M}|\hat{\mathbf{X}}_t - \mathbf{X}_t|, \tag{30}$$

$$MAPE = \frac{1}{M}\sum_{i=1}^{M}|\frac{\hat{\mathbf{X}}_t - \mathbf{X}_t}{\mathbf{X}_t}|, \tag{31}$$

where $\hat{\mathbf{X}}_t$ and $\mathbf{X}_t$ are the predicted and true traffic volumes at time $t$; $M$ is the collection of all samples. Note that we set $\mathbf{X}_t$ to 1 if $\mathbf{X}_t = 0$ to avoid an infinite MAPE value.

### 6.2. Performance comparison

Detailed experimental results on prediction accuracy are shown in Tables 3 and 4, where TODE-GN and TODE-LN are two variants that using Group Normalization and Layer Normalization in the ST-ODE block, respectively. From the experimental results, we can see that our TODE model outperforms the baselines in terms of all three metrics. In particular, TODE reduces the RMSE value down to 15.53, which is 2% to 73% better than 11 benchmarking models. Compared to the ST-ResNet, TODE reduces the MAE from 9.91 to 9.35 and reduces the MAPE from 0.40 to 0.29 compared with the ST-ResNet model. When comparing the two variants of TODE, we can see that layer normalization performs better than group normalization in our experiments.

**Table 5**
The details of TaxiBJ dataset partition.

| Dataset | Time Span | Size | | Max volume |
|---|---|---|---|---|
| | | Train | Test | |
| TaxiBJ | P1 | 2832 | 336 | 1161 |
| | P2 | 2160 | 336 | 1292 |
| | P3 | 4224 | 336 | 1274 |
| | P4 | 4512 | 336 | 1250 |

**Table 6**
Results on P1-P4 time spans (TaxiBJ).

| Method | | Inflow | | | Outflow | | |
|---|---|---|---|---|---|---|---|
| | | MAE | RMSE | MAPE | MAE | RMSE | MAPE |
| P1 | ST-ResNet-noF | 11.30 | 18.66 | 0.52 | 11.35 | 18.74 | 0.52 |
| | ST-ResNet | 11.39 | 18.66 | 0.53 | 11.44 | 18.74 | 0.54 |
| | TODE-noF | 10.31 | 17.20 | 0.42 | 10.41 | 17.33 | 0.43 |
| | **TODE** | **10.12** | **16.78** | **0.40** | **10.21** | **16.91** | **0.41** |
| P2 | ST-ResNet-noF | 12.25 | 19.69 | 0.51 | 12.28 | 19.73 | 0.51 |
| | ST-ResNet | 12.31 | 19.72 | 0.47 | 12.39 | 19.84 | 0.47 |
| | TODE-noF | 11.11 | 17.98 | 0.39 | 11.20 | 18.11 | 0.39 |
| | **TODE** | **10.92** | **17.67** | **0.38** | **11.00** | **17.78** | **0.38** |
| P3 | ST-ResNet-noF | 12.58 | 21.70 | 0.63 | 12.66 | 21.82 | 0.63 |
| | ST-ResNet | 12.56 | 21.81 | 0.62 | 12.64 | 21.84 | 0.62 |
| | TODE-noF | 11.32 | 19.59 | 0.47 | 11.47 | 19.71 | 0.48 |
| | **TODE** | **11.02** | **18.93** | **0.46** | **11.11** | **19.03** | **0.47** |
| P4 | ST-ResNet-noF | 10.11 | 15.84 | 0.46 | 10.20 | 15.98 | 0.46 |
| | ST-ResNet | 10.22 | 16.14 | 0.43 | 10.26 | 16.23 | 0.43 |
| | TODE-noF | 9.25 | 14.65 | 0.33 | 9.35 | 14.76 | 0.33 |
| | **TODE** | **9.06** | **14.23** | **0.32** | **9.12** | **14.34** | **0.32** |

**Table 7**
Comparison of parameters and memory cost (TaxiBJ).

| Method | #Parameters | Memory Cost |
|---|---|---|
| ST-ResNet | 2.71M | $O(L_R N_R)$ |
| **TODE-LN** | **1.55M** | $O(L_O + N_O)$ |
| **TODE-GN** | **0.76M** | $O(L_O + N_O)$ |

Among the baselines, traditional time-series prediction methods (HA, ARIMA, SARIMA and VAR) are incomparable with the deep learning based methods, since they only rely on historical records to model the linear spatial–temporal dependencies. For deep-learning-based methods, STDN and ResNet usually achieve the best performance among the baseline methods. This result proves the effectiveness of modeling spatial and temporal similarity by CNN and LSTM. On the other hand, we do not observe apparent advantages of modeling spatial dependency with GCNs. The GCN based models (i.e., DGCNN, MVGCN and MDL) are good at capturing irregular graph structures but may fail to model the temporal dependency among different time steps and the periodicity of time series data. We note, however, that these methods are usually computationally intensive, especially for large graph data, due to the high cost of training GCN.

**Results on Different Time Spans**: In particular, we extend our experiment with data from four different time periods on the TaxiBJ dataset. The detailed data split is referred to Tables 2 and 5. The results are illustrated in Table 6. In this experiment, we use the last week of data in TaxiBJ as test data, and the rest as training data. TODE-noF (ST-ResNet-noF) means that we train TODE (ST-ResNet) by feeding data without extrinsic factors. For P1 time span on in-flow and out-flow, TODE with extrinsic factors makes 10%, 11%, 24% improvement against ST-ResNet with extrinsic factors on RMSE, MAE, and MAPE, respectively. For P2–P4 time spans, TODE with extrinsic factors achieves about 10% to 30% improvement over ST-ResNet with extrinsic factors.

**Results on Multi-step Predictions**: In addition to next step (1 h) crowd flow prediction as in most previous works, we conduct an experiment to evaluate the performance of our model on multi-step prediction. Fig. 10 compares TODE with ST-ResNet on next 24 h crowd prediction. When the number of step is less than 12, the prediction error increases with the number of step. However, the error returns to a lower level when we further increase the time step. This result suggests that models can predict multi-step crowd and assure the multi-step errors in a lower bound, and also implies that the periodicity of the crowd flow in the city has been captured by the model.

### 6.3. Model efficiency

**Computational Cost**: As mentioned before, TODE is memory efficient. In particular, TODE costs $O(L_O + N_O)$ memory in total while ST-ResNet can takes memory up to $O(L_R N_R)$, as shown in Table 7. We compare parameter size between ST-ResNet and TODE,

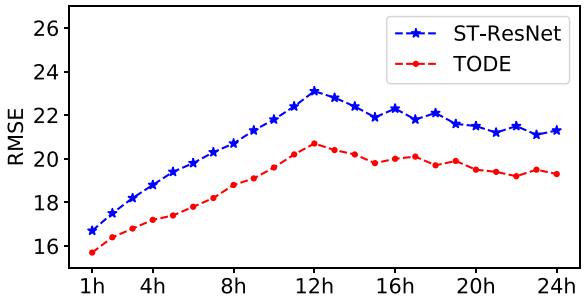



**Fig. 10.** Results on multi-step crowd flow prediction (TaxiBJ).

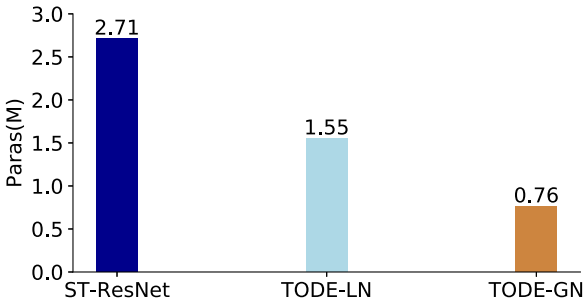



**Fig. 11.** Comparison of parameter usage with ST-ResNet (TaxiBJ).

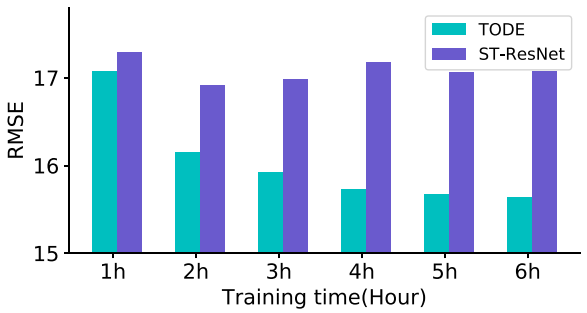



**Fig. 12.** Efficiency vs. Accuracy (TaxiBJ).

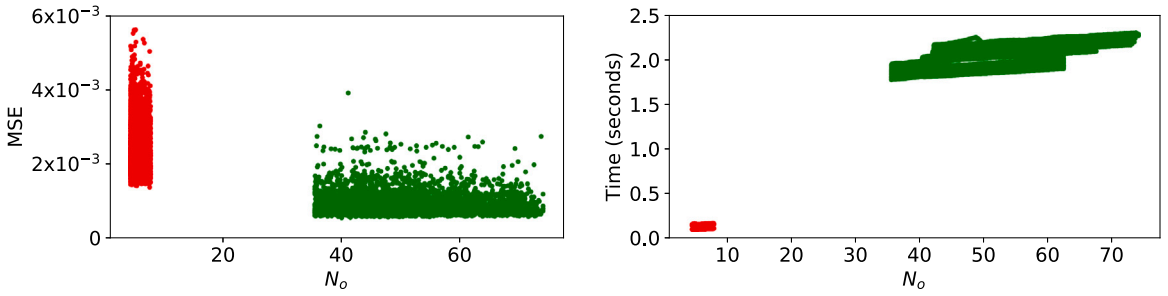
shown in Fig. 11. Our TODE with group normalization has only 0.76M parameters, which is almost 2.5 times less than what is used in ST-ResNet. While achieving the best performance, TODE with layer normalization requires more parameters than TODE-GN, i.e., 1.55M parameters, which is also 40% less than ST-ResNet.

In addition, Fig. 12 compares the model efficiency in terms of training time between ST-ResNet and TODE, which clearly shows that TODE achieves better performance given a specific time of training. Though requiring more time to obtain optimal performance, TODE provides a more flexible solution with higher performance — if training time is the main concern. This result also implies the adaptability of our model, i.e., the training time of TODE scales linearly with the requested training accuracy. Furthermore, we can see that TODE is more robust while ST-ResNet suffers obvious overfitting problem.

Thus, TODE achieves the state-of-the-art performance in urban flow prediction and is significantly more efficient than other deep learning networks.

**Prediction Accuracy vs. Computational Cost**: Another advantage of TODE is that we can dynamically balance the prediction accuracy of computational cost by varying the number of steps $N_O$ ($N \geq 0$) in TODE. This is achieved by the ODE solvers which provide approximate assurance that the output is within a given tolerance of the true solution. Changing the model tolerance would yield a different prediction performance of the model (Chen et al., 2018), while the error can be bounded.

In Fig. 13, we used two ranges of $N_t$ to solve the differential equations requested in TODE, where each point denotes a batch of data. Fig. 13(a) indicates that the more number of function evaluations in the ST-ODE block, the better prediction we can achieve during training. However, the higher the accuracy, the more time required to evaluate the functions, as illustrated in Fig. 13(b), from which we can also observe that the training time increase linearly with $N_o$ – the number of the time steps in TODE.




(a) Prediction error. (b) Computational time.

**Fig. 13.** Trade-offs between prediction accuracy and computational cost (TaxiBJ).

**Table 8**
Influence of external features (TaxiBJ).

| Factors | Without Factors | | With Factors | |
|---|---|---|---|---|
| Method | Metrics | | | |
| | RMSE | MAE | RMSE | MAE |
| ST-ResNet | 17.02 ± 0.18 | 10.14 ± 0.10 | 16.85 ± 0.17 | 9.95 ± 0.09 |
| **TODE** | **16.67 ± 0.13** | **9.66 ± 0.05** | **15.69 ± 0.11** | **9.38 ± 0.03** |

**Table 9**
Early fusion vs. Late fusion (TaxiBJ).

| Metric | RMSE | | | |
|---|---|---|---|---|
| Fusion strategy | Time span | | | |
| | P1 | P2 | P3 | P4 |
| Early fusion | 17.10 ± 0.18 | 18.07 ± 0.20 | 19.29 ± 0.22 | 14.44 ± 0.16 |
| Late fusion | **16.86 ± 0.15** | **17.75 ± 0.17** | **18.96 ± 0.20** | **14.30 ± 0.15** |

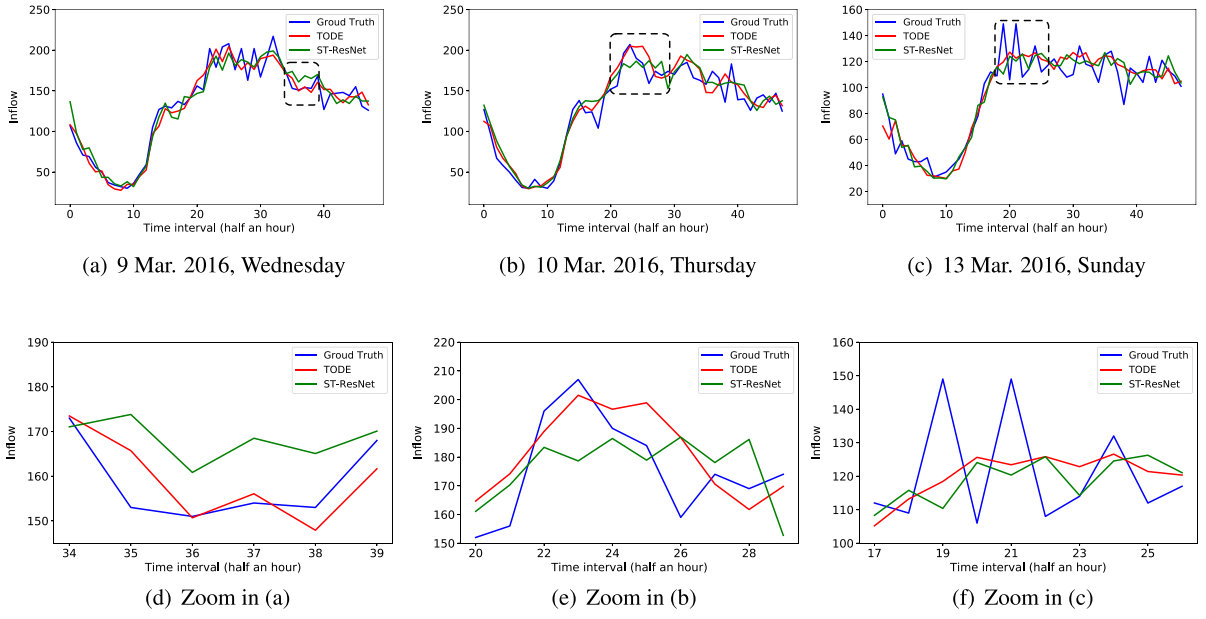
### 6.4. Influence of features

In previous studies (Zhang et al., 2017; Sun et al., 2019), researchers directly use an Add layer to merge the predicted traffic flow and feature output. However, this method may not be able to make extrinsic factors effective in traffic prediction tasks. From Table 6, we can observe that there is no obvious improvement in the ST-ResNet model after adding extrinsic factors, and sometimes it has a negative influence on the traffic flow prediction. However, TODE overcomes this problem with a new way of deep feature fusion. In fact, experiments in (Zhang et al., 2017) show that it slightly reduce RMSE to 16.89 from 17.00 when using extrinsic features. In our TODE model, rather than directly merging the predicted flow and feature output, we concatenate the feature output $\mathbf{F}_t$ with traffic flow data and pass it through a deep neural network (ST-ODE). This approach allows the TODE network to correctly understand the impact of factors on the predicted traffic flow. As shown in Table 8, we provide the mean and deviation of the RMSE and MAE. With our feature fusion method, we are able to reduce RMSE and MAE efficiently, as compared to the feature fusion method used in ST-ResNet.

Note that we use a late fusion strategy in our model, i.e., the external factors and three temporal dependencies are fused after the ST-ODE block. However, this fusion can be performed before the ST-ODE, which is also called early fusion. We compare the two fusion strategies and show the results in Table 9. According to the results, late fusion performs better than early fusion, because the latter needs to merge three different dependencies (i.e., recent, daily, and weekly) before the ST-ODE block, which may ignore important correlations between external factors and the periodical information of the flow. Although early fusion has been used in previous work (Lin et al., 2019), the performance gain lies in co-training of external factors and spatio-temporal correlations, rather than the so-called early fusion operation. In our work, we used the same co-training strategy as Lin et al. (2019), but the fusion happens after the neural ODE blocks, i.e., the late fusion strategy.

### 6.5. Qualitative analysis

To investigate the quality of prediction, we randomly select a week and plot the prediction of TODE and ST-ResNet against the ground truth traffic flow. Fig. 14(a), 14(b), 14(c) show the qualitative comparison of traffic flow prediction for a certain region in three days – i.e., Wednesday, Thursday and Sunday respectively. Among them, the blue line represents the ground truth in-flow, the red line denotes the prediction by TODE, and the green line points the prediction by ST-ResNet. To better visualize the quality of prediction, we select three temporal sub-intervals in Fig. 14(d), 14(e), 14(f), which correspond to traffic flow on Wednesday,




(a) 9 Mar. 2016, Wednesday (b) 10 Mar. 2016, Thursday (c) 13 Mar. 2016, Sunday

(d) Zoom in (a) (e) Zoom in (b) (f) Zoom in (c)

**Fig. 14.** Traffic forecasting results for three days in a week. (d), (e) and (f) are plots by zooming into the selected areas in (a), (b) and (c), respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

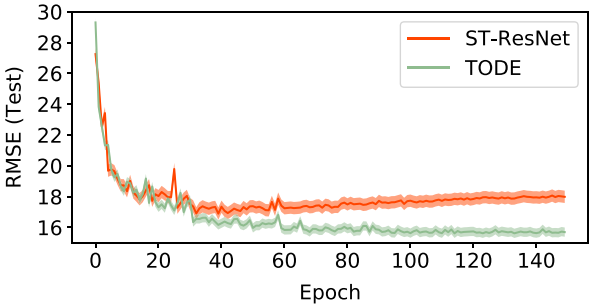



**Fig. 15.** RMSE (test mode) with TODE and ST-ResNet on TaxiBJ dataset.

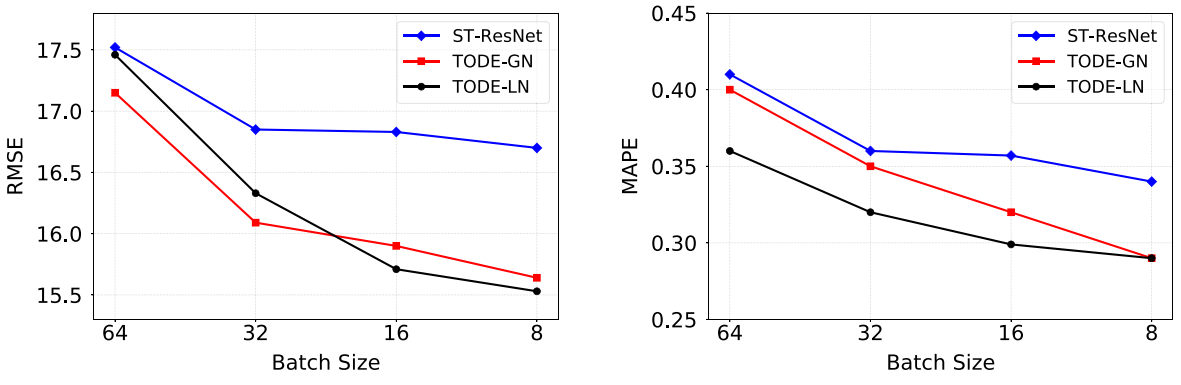
Thursday and Sunday, respectively. We can observe that the red curve (TODE) is closer to the blue curve (ground truth) than green curve (ST-ResNet) in most time, especially at the extreme point of the curve. Besides, the predicted flow line almost completely coincide with the ground truth in a smoother manner. However, in some continuous sudden changes area, as shown in Fig. 14(e), TODE and ST-ResNet are not performing well. We conjecture that this result may be caused by some special events that we have not considered, such as promotional activities in the mall and urban events, that typically may have happened in the weekends. Likewise, we observe that TODE has a delayed response to sudden changes of temporal features. One possibility for TODE to improve the traffic flow prediction may be to make a slight shift of the prediction curve to eliminate this hysteresis as suggested in (Yao et al., 2019), which we leave for the future works.

In a similar spirit, Fig. 15 depicts RMSE on the TaxiBJ test set as the number of training epoch increases, where we plot the averaged results on 10 runs. We note that ST-ResNet exhibits an obvious overfitting after 40 epochs, which can be successfully avoided in TODE, resulting in its lower loss value. Meanwhile, TODE is more stable (with less variance) during testing as compared to ST-ResNet.

### 6.6. Model robustness

We implemented ST-ResNet and TODE model using batch size values of $\{64, 32, 16, 8\}$, respectively. As shown in Fig. 16, we compare them with respect to RMSE and MAPE, from which we have the following observations:

(1) The TODE outperforms the ST-ResNet on all batch sizes. It shows that our deep feature fusion method and TODE model plays a significant role in improving the accuracy of predicting traffic flow.




(a) RMSE (b) MAPE

**Fig. 16.** Effect of batch size.

**Table 10**
Comparison of performance when there are missing data (TaxiBJ). $p$ denotes missing rate.

| Metric | RMSE | | | | |
|---|---|---|---|---|---|
| Method | $p$ | | | | |
| | $\lambda = 0$ | $\lambda = 0.05$ | $\lambda = 0.1$ | $\lambda = 0.15$ | $\lambda = 0.2$ |
| ST-ResNet | 16.85 ± 0.17 | 18.6 ± 0.19 | 21.1 ± 0.18 | 27.1 ± 0.19 | 38.8 ± 0.20 |
| **TODE** | **15.69 ± 0.11** | **16.8 ± 0.14** | **18.9 ± 0.15** | **23.2 ± 0.18** | **30.5 ± 0.18** |

(2) As the batch size decreases, the training errors also decrease accordingly, and the TODE reduces much more than the ST-ResNet. This result indicates that the TODE model is sensitive to the batch size and that TODE performs better for small batch sizes. We can explain this in two ways: on one hand, large batch sizes easily reach local minimum, while small batch sizes introduce more randomness, and it is likely to produce better results. On the other hand, the TODE network can be regarded as a continuous deep residual network, and the DTO method can effectively address the problem of gradient divergence and numeric instability. The randomness introduced by small batches makes the model not appear in the case of gradient divergence. The convergence is stably close to the global minimum, while also reducing the oscillation size.

(3) TODE with Layer Normalization is slightly better than TODE with Group Normalization on small batch size. This is because the parameters of TODE with Group Normalization are half of TODE with Layer Normalization.

**Influence of Missing Observations**: To evaluate our model performance with missing data, we prepare a dataset by randomly removing a few observations. That is, the observation (crowd flow) at a specific time is masked with a probability of $\lambda$. We run 10 experiments and report the averaged results. Table 10 shows the comparison between TODE and ST-ResNet, varying with the parameter $\lambda$. We can see that the discrepancy between two methods increases with the value of $\lambda$, indicating that our model is more robust and is capable of handling missing observations. TODE outperforms ST-ResNet due to its ability to generalize the discrete neural networks to continuous dynamics, allowing the model to integrate dynamics over whatever time interval is needed.

## 7. Conclusions and future works

In this paper, we proposed a novel deep architecture for predicting citywide mobility flow, alleviating the high computational overheads in previous traffic forecasting systems. TODE learns traffic dynamics through ODEs parameterized by neural networks, which allows for both memory and model parameter savings and enables computational efficiency, while guaranteeing prediction performance. In addition, it is capable of explicitly providing more flexible prediction performance by adaptively balancing prediction accuracy and computation overheads. We also provide a new approach for intrinsic and extrinsic feature fusion by disentangling the factors with separate neural networks, thus addressing the interwoven influence issue in previous works. Experimental results demonstrate the advantages in terms of prediction accuracy and computational efficiency over the baselines.

As for future work, we are interested in estimating the uncertainty of the current traffic prediction system which is crucial for determining if the predicted results should be considered reliable. In addition, we will incorporate data such as the changes in road segment conditions (e.g., due to accidents or road-work) and event locations, and investigate how beneficial they could be for learning and improving the estimate of the city crowd motion patterns. Moreover, it is worthwhile trying other map partition methods such as hexagon partition, which might better reflect the spatial dependencies among traffic flows.

## CRediT authorship contribution statement

**Fan Zhou:** Conceptualization, Methodology, Writing - original draft. **Liang Li:** Data curation, Experiments. **Kunpeng Zhang:** Visualization, Investigation, Validation. **Goce Trajcevski:** Supervision, Writing - review & editing.

## Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No. 62072077 and No. 61602097), the U.S. National Science Foundation (NSF) grants III 1213038 and CNS 1646107.

## References

Avelin, B., Nyström, K., 2019. Neural ODEs as the Deep Limit of ResNets with constant weights. arXiv preprint arXiv:1906.12183.

Ba, J.L., Kiros, J.R., Hinton, G.E., 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Bellomo, N., Delitala, M., Coscia, V., 2002. On the mathematical theory of vehicular traffic flow I: fluid dynamic and kinetic modelling. Math. Models Methods Appl. Sci. 12 (12), (Review Paper).

Box, G.E., Jenkins, G.M., 1976. Time series analysis: Forecasting and control. In: Holden-Day Series in Time Series Analysis, revised ed. Holden-Day, San Francisco.

Brunsdon, C., Fotheringham, A.S., Charlton, M.E., 1996. Geographically weighted regression: a method for exploring spatial nonstationarity. Geogr. Anal. 28 (4), 281–298.

Castro-Neto, M., Jeong, Y.S., Jeong, M.K., Han, L.D., 2009. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. Expert Syst. Appl. 36 (3), 6164–6173.

Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K., 2018. Neural ordinary differential equations. In: Advances in Neural Information Processing Systems. NeuIPS, pp. 6571–6583.

Chung, J., Gulcehre, C., Cho, K.H., Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Diao, Z., Wang, X., Zhang, D., Liu, Y., Xie, K., He, S., 2019. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 890–897.

Do, L.N., Vu, H.L., Vo, B.Q., Liu, Z., Phung, D., 2019. An effective spatial-temporal attention based neural network for traffic flow prediction. Transp. Res. C 108, 12–28.

Dupont, E., Doucet, A., Teh, Y.W., 2019. Augmented neural ODEs. In: Advances in Neural Information Processing Systems. NeurIPS.

Feng, J., Li, Y., Zhang, C., Sun, F., Meng, F., Guo, A., Jin, D., 2018. DeepMove: Predicting human mobility with attentional recurrent networks. In: International World Wide Web Conference. WWW, ACM, pp. 1459–1468.

Fildes, R., 1991. Forecasting, structural time series models and the Kalman Filter: Bayesian forecasting and dynamic models. J. Oper. Res. Soc. 42 (11), 1031–1033.

Furtado, A.S., Alvares, L.O.C., Pelekis, N., Theodoridis, Y., Bogorny, V., 2018. Unveiling movement uncertainty for robust trajectory similarity analysis. Int. J. Geogr. Inf. Sci. 32 (1), 140–168.

Gao, Q., Trajcevski, G., Zhou, F., Zhang, K., Zhong, T., Zhang, F., 2018. Trajectory-based social circle inference. In: Proceedings of the International Conference on Advances in Geographic Information Systems. SIGSPATIAL, ACM, pp. 369–378.

Gao, Q., Zhou, F., Trajcevski, G., Zhang, K., Zhong, T., Zhang, F., 2019. Predicting human mobility via variational attention. In: International World Wide Web Conference. WWW, ACM, pp. 2750–2756.

Gardner, Jr., E.S., 1985. Exponential smoothing: The state of the art. J. Forecast. 4 (1), 1–28.

Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., Liu, Y., 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 3656–3663.

Gholami, A., Keutzer, K., Biros, G., 2019. ANODE: Unconditionally accurate memory-efficient gradients for neural ODEs. In: International Joint Conference on Artificial Intelligence. IJCAI, pp. 730–736.

Gu, Y., Lu, W., Qin, L., Li, M., Shao, Z., 2019. Short-term prediction of lane-level traffic speeds: A fusion deep learning model. Transp. Res. C 106, 1–16.

Guo, S., Lin, Y., Feng, N., Song, C., Wan, H., 2019a. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 922–929.

Guo, S., Lin, Y., Li, S., Chen, Z., Wan, H., 2019b. Deep spatial–temporal 3D convolutional neural networks for traffic data forecasting. IEEE Trans. Intell. Transp. Syst. 20 (10), 3913–3926.

Hahnloser, R.H., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S., 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature 405 (6789), 947.

He, X., Mo, Z., Wang, P., Liu, Y., Yang, M., Cheng, J., 2019. ODE-inspired network design for single image super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE, pp. 770–778.

Heinonen, M., Lähdesmäki, H., 2019. ODE$^2$VAE: Deep generative second order ODEs with Bayesian neural networks. In: Advances in Neural Information Processing Systems. NeurIPS.

Hoang, M.X., Zheng, Y., Singh, A.K., 2016. FCCF: forecasting citywide crowd flows based on big data. In: Proceedings of the International Conference on Advances in Geographic Information Systems. SIGSPATIAL, ACM.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Huang, W., Song, G., Hong, H., Xie, K., 2014. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. IEEE Trans. Intell. Transp. Syst. 15 (5), 2191–2201.

Huang, C., Zhang, C., Zhao, J., Wu, X., Chawla, N., Yin, D., 2019. MiST: A multiview and multimodal spatial-temporal learning framework for citywide abnormal event forecasting. In: International World Wide Web Conference. WWW, ACM, pp. 717–728.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. ICML, pp. 448–456.

Jiang, R., Song, X., Huang, D., Song, X., Xia, T., Cai, Z., Wang, Z., Kim, K.S., Shibasaki, R., 2019. DeepUrbanEvent: A system for predicting citywide crowd dynamics at big events. In: Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD, ACM, pp. 2114–2122.

Ke, J., Qin, X., Yang, H., Zheng, Z., Zhu, Z., Ye, J., 2019. Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network. arXiv preprint arXiv:1910.09103.

Ke, J., Zheng, H., Yang, H., Chen, X.M., 2017. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. Transp. Res. C 85, 591–608.

Kelejian, H.H., Prucha, I.R., 1999. A generalized moments estimator for the autoregressive parameter in a spatial model. Internat. Econom. Rev. 40 (2), 509–533.

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

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations. ICLR.

Li, X., Cong, G., Sun, A., Cheng, Y., 2019. Learning travel time distributions with deep generative model. In: International World Wide Web Conference. WWW, ACM, pp. 1017–1027.

Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., Liu, Y., 2018a. Multi-task representation learning for travel time estimation. In: Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD, ACM Press, pp. 1695–1704.
Li, Y., Yu, R., Shahabi, C., Liu, Y., 2018b. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representations. ICLR.
Liang, X., de Almeida Correia, G.H., An, K., van Arem, B., 2020a. Automated taxis' dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times. Transp. Res. C 112, 260–281.
Liang, Y., Ke, S., Zhang, J., Yi, X., Zheng, Y., 2018. GeoMAN: Multi-level attention networks for geo-sensory time series prediction. In: International Joint Conference on Artificial Intelligence. IJCAI, pp. 3428–3434.
Liang, Y., Ouyang, K., Jing, L., Ruan, S., Liu, Y., Zhang, J., Rosenblum, D.S., Zheng, Y., 2019. UrbanFM: Inferring fine-grained urban flows. In: Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD, ACM, pp. 3132–3142.
Liang, Y., Ouyang, K., Wang, Y., Liu, Y., Zhang, J., Zheng, Y., Rosenblum, D.S., 2020b. Revisiting convolutional neural networks for citywide crowd flow analytics. In: Machine Learning and Knowledge Discovery in Databases - European Conference.
Lin, Z., Feng, J., Lu, Z., Li, Y., Jin, D., 2019. Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. pp. 1020–1027.
Liu, L., Qiu, Z., Li, G., Wang, Q., Ouyang, W., Lin, L., 2019a. Contextualized spatial–temporal network for taxi origin-destination demand prediction. IEEE Trans. Intell. Transp. Syst. 20 (10), 3875–3887.
Liu, Q., Wu, S., Wang, L., Tan, T., 2016. Predicting the next location: a recurrent model with spatial and temporal contexts. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 194–200.
Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., Hsieh, C.J., 2019b. Neural SDE: Stabilizing neural ODE networks with stochastic noise. arXiv preprint arXiv:1906.02355v1.
Lu, Y., Zhong, A., Li, Q., Dong, B., 2018. Beyond finite layer neural networks - bridging deep architectures and numerical differential equations. In: International Conference on Machine Learning. ICML, pp. 3282–3291.
Ma, J., Chan, J., Ristanoski, G., Rajasegarar, S., Leckie, C., 2019. Bus travel time prediction with real-time traffic information. Transp. Res. C 105, 536–549.
Malfliet, W., Hereman, W., 1996. The tanh method: I. Exact solutions of nonlinear evolution and wave equations. Phys. Scr. 54 (6), 563.
Min, W., Wynter, L., 2011. Real-time road traffic prediction with spatio-temporal correlations. Transp. Res. C 19 (4), 606–616.
Oliver, M.A., Webster, R., 1990. Kriging: a method of interpolation for geographical information systems. Int. J. Geogr. Inf. Syst. 4 (3), 313–332.
Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y., Zhang, J., 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In: Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD, ACM, pp. 1720–1730.
Polson, N.G., Sokolov, V.O., 2017. Deep learning for short-term traffic flow prediction. Transp. Res. C 79, 1–17.
Qi, L., Zhou, M., Luan, W., 2018. A two-level traffic light control strategy for preventing incident-based urban traffic congestion. IEEE Trans. Intell. Transp. Syst. 19 (1), 13–24.
Rabiner, L.R., 1986. An introduction to hidden Markov models. IEEE ASSP Mag. 3 (1), 4–16.
Reis, B.Y., Mandl, K.D., 2003. Time series modeling for syndromic surveillance. BMC Med. Inform. Decis. Mak. 3 (1), 2.
Rodrigues, F., Markou, I., Pereira, F.C., 2019. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. Inf. Fusion 49, 120–129.
Ryu, U., Wang, J., Kim, T., Kwak, S., Juhyok, U., 2018. Construction of traffic state vector using mutual information for short-term traffic flow prediction. Transp. Res. C 96, 55–71.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.
Sun, J., Zhang, J., Li, Q., Yi, X., Zheng, Y., 2019. Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks. arXiv preprint arXiv:1903.07789.
Tian, Y., Zhang, K., Li, J., Lin, X., Yang, B., 2018. LSTM-based traffic flow prediction with missing data. Neurocomputing 318, 297–305.
Van Der Voort, M., Dougherty, M., Watson, S., 1996. Combining kohonen maps with ARIMA time series models to forecast traffic flow. Transp. Res. C 4 (5), 307–318.
Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y., 2018. Graph attention networks. In: International Conference on Learning Representations. ICLR.
Wang, D., Cao, W., Li, J., Ye, J., 2017. DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks. In: International Conference on Data Engineering. ICDE, IEEE, pp. 243–254.
Wang, Y., Yin, H., Chen, H., Wo, T., Xu, J., Zheng, K., 2019. Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling. In: Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD, pp. 1227–1235.
Weinan, E., 2017. A proposal on machine learning via dynamical systems. Commun. Math. Stat. 5 (1), 1–11.
Wu, Y., He, K., 2018. Group normalization. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 3–19.
Wu, Y., Tan, H., Qin, L., Ran, B., Jiang, Z., 2018. A hybrid deep learning based traffic flow prediction method and its understanding. Transp. Res. C 90, 166–180.
Yang, B., Jiang, M., Chen, Y., Meng, Q., Abraham, A., 2013. Ensemble of flexible neural tree and ordinary differential equations for small-time scale network traffic prediction. J. Comput. 8 (12), 3039–3046.
Yao, H., Tang, X., Wei, H., Zheng, G., Li, Z., 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 5668–5675.
Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J., Li, Z., 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 2588–2595.
Yu, B., Lee, Y., Sohn, K., 2020. Forecasting road traffic speeds by considering area-wide spatio-temporal dependencies based on a graph convolutional neural network (GCN). Transp. Res. C 114, 189–204.
Zhang, Z., Li, M., Lin, X., Wang, Y., He, F., 2019c. Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies. Transp. Res. C 105, 297–322.
Zhang, H., Wu, H., Sun, W., Zheng, B., 2018. DeepTravel: a neural network based travel time estimation model with auxiliary supervision. In: International Joint Conference on Artificial Intelligence. IJCAI, pp. 3655–3661.
Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., Mahoney, M., 2019b. ANODEV2: A coupled neural ODE evolution framework. In: Advances in Neural Information Processing Systems. NeurIPS.
Zhang, J., Zheng, Y., Qi, D., 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1655–1661.
Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., 2016. DNN-based prediction model for spatio-temporal data. In: Proceedings of the International Conference on Advances in Geographic Information Systems. SIGSPATIAL, ACM, p. 92.
Zhang, J., Zheng, Y., Sun, J., Qi, D., 2019a. Flow prediction in spatio-temporal networks based on multitask deep learning. IEEE Trans. Knowl. Data Eng.
Zhao, Y., Zhang, L., Li, P., Huang, B., 2007. Classification of high spatial resolution imagery using improved Gaussian Markov random-field-based texture features. IEEE Trans. Geosci. Remote Sens. 45 (5), 1458–1468.

Zhou, F., Gao, Q., Trajcevski, G., Zhang, K., Zhong, T., Zhang, F., 2018. Trajectory-user linking via variational AutoEncoder. In: International Joint Conference on Artificial Intelligence. IJCAI, pp. 3212–3218.

Zhou, F., Mo, Y., Trajcevski, G., Zhang, K., Wu, J., Zhong, T., 2020a. Recommendation via collaborative autoregressive flows. Neural Netw.

Zhou, X., Shen, Y., Huang, L., 2019c. Revisiting flow information for traffic prediction. arXiv preprint arXiv:1906.00560v1.

Zhou, F., Yang, Q., Zhang, K., Trajcevski, G., Zhong, T., Khokhar, A., 2020b. Reinforced spatio-temporal attentive graph neural networks for traffic forecasting. IEEE Internet Things J.

Zhou, F., Yang, Q., Zhong, T., Chen, D., Zhang, N., 2020c. Variational graph neural networks for road traffic prediction in intelligent transportation systems. IEEE Trans. Ind. Inf.

Zhou, F., Yin, R., Zhang, K., Trajcevski, G., Zhong, T., 2019a. Adversarial point-of-interest recommendation. In: International World Wide Web Conference. WWW, ACM, pp. 3462–34618.

Zhou, F., Yue, X., Trajcevski, G., Zhong, T., Zhang, K., 2019b. Context-aware variational trajectory encoding and human mobility inference. In: International World Wide Web Conference. WWW, ACM, pp. 3469–3475.