DNN-Based Prediction Model for Spatial-Temporal Data

Junbo Zhang[1], Yu Zheng[1,2,3], Dekang Qi[4], Ruiyuan Li[2], Xiuwen Yi[4]
[1]Microsoft Research, Beijing, China
[2]School of Computer Science and Technology, Xidian University, China
[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[4]School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China
{junbo.zhang, yuzheng, v-deq, v-ruiyli, v-xiuyi}@microsoft.com

ABSTRACT

The advances in location-acquisition and wireless communication technologies have led to massive spatio-temporal (ST) data, which has unique spatial properties (i.e. the geographical hierarchy and distance) and temporal properties (i.e. sequential, periodic and seasonal trend patterns). In this paper, we propose a Deep learning-based prediction model for Spatial-Temporal data (DeepST). We leverage ST domain knowledge to design the architecture of DeepST, which is composed of three components: 1) temporal dependent instances: describing temporal closeness, period and seasonal trend; 2) convolutional neural networks: capturing near and far spatial dependencies; 3) early and late fusions: fusing similar and different domains' data. Using DeepST, we build a real-time crowd flow forecasting system UrbanFlow[1]. The experimental results on diverse ST datasets verify DeepST's capability of capturing ST data's spatio-temporal properties, showing the advantages of DeepST beyond four baseline methods.

Keywords

Deep Learning; Spatio-Temporal Data; Prediction

1. INTRODUCTION

The advances in location-acquisition and wireless communication technologies have resulted in massive data with spatial coordinates and timestamps, entitled ST data, in a diversity of domains, ranging from transportation to environment science, from communication systems to social networking services. Being different from text and image data, ST data has unique 1) spatial properties, which consists of a geographical hierarchy and distance, and 2) temporal properties, which is consisted of a sequential, periodic and seasonal trend pattern.

1) Spatial properties: first, locations at a higher level of a geographical hierarchy have a coarser granularity, and the territory of a parent node is composed of those of its children. For example, a tourist attraction is located in a district, which further belongs to a city. Second, there is a geographical distance between two locations, which can measure the correlation between the two locations. For instance, near locations are more similar than distant ones, according to the first law of geography.

2) Temporal properties: The timestamp of each instance in an ST dataset allows us to order instances chronologically, generating sequential properties where adjacent timestamps usually have a higher similarity than distant ones. On the other hand, ST data usually has a certain periodic pattern, which repeats with a certain frequency. For instance, traffic conditions in morning rush hours may be similar in consecutive workdays, repeating every 24 hours.

Learning an effective prediction for ST data will significantly contribute to a variety of urban applications, such as air quality forecasting [5], crowd flows prediction, bike rent/return estimation in bike-sharing systems [3]. However, it is very challenging to capture all spatial and temporal properties simultaneously. To address these challenges, we propose a deep neural network (DNN)-based prediction model (entitled DeepST) which includes three key components: temporal dependent instances, convolutional neural networks, early and late fusions. The contributions of our work are two-fold:

- We design a novel deep learning architecture for spatial-temporal data using the domain knowledge and propose to employ 1) temporal closeness, period, and seasonal trend to generate input instance, 2) multiple convolutions to describe spatial near and far dependencies, 3) early and late fusions to fuse similar and different domains' ST data.
- We apply the proposed deep learning model to predict citywide crowd flows, and develop a real-time flow forecasting system (called as UrbanFlow), which can effectively monitor the fine-grained crowd flows and provide the future ones in cities.

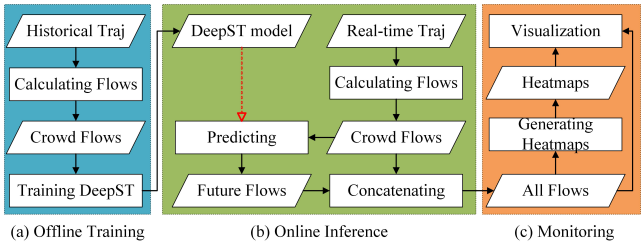


Figure 1: System Framework. Traj: trajectories.

[1]http://urbanflow.chinacloudsites.cn/

ACM ISBN .
DOI:

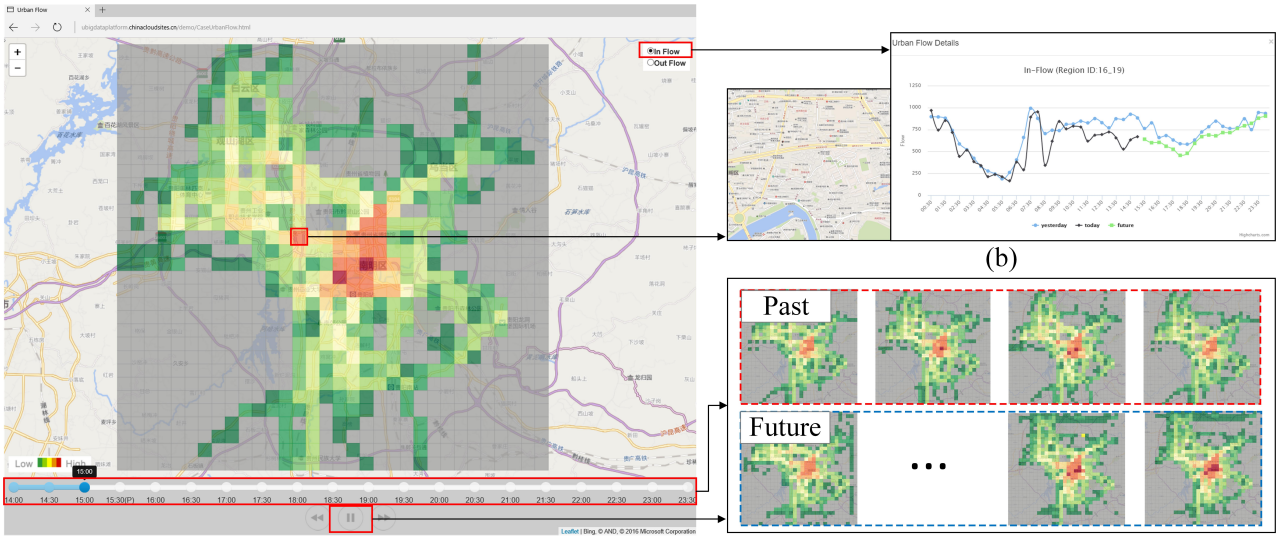


Figure 2: UrbanFlow: A real-time crowd flow forecasting system

1.1 System Framework

The framework of the system is shown in Figure 1. A case study about "crowd flows prediction" is used through the demo paper. There are three major components in our framework: offline training, online inference, and monitoring website. In the offline training, the collected trajectories (e.g. taxi) from a city are fed in "Calculating Flows" module that output two types of flows (see Definition 2). Then these historical flows are used to learn the DeepST model, which will be introduced in Section 2.2. In the online inference, starting with calculating crowd flows from real-time trajectories, the learned DeepST model is used to predict future flows that are concatenated with the real-time flows later. In the last component, in order to monitor intuitively, we generate heatmaps from the real-time and predicted crowd flows that can show the global status in the city. At the same time, curves in a single region show the more detailed flows. Section 2 will introduce the details of monitoring website.

1.2 Demonstration of the System

The system is built as a website, named as UrbanFlow. Users could view the real-time and forecasting crowd flows in the cities. The user interface of the system is shown in Figure 2a. Here we apply UrbanFlow to the area of the Guiyang city, China. The top right corner of the website shows the buttons which can switch between different types of flows. A user could select any grid (represents a region) on the website and click it to see the region's detailed flows, as shown in Figure 2b where blue, black and green curves indicate flows of yesterday, past & future time at today, respectively. The bottom of the website shows a few sequential timestamps. The heatmap at a certain timestamp will be shown in the website when a user clicks the associated timestamp. Intuitively, the user can watch the movie-style heatmaps by clicking "play button" (Figure 2c).

2. MODELS

In this section, we first formulate the ST prediction problem and then introduce DNN-based prediction model (DeepST).

2.1 Formulation of ST Prediction Problem

DEFINITION 1 (REGION). *There are many definitions of a location in terms of different granularities and semantic meanings. In this study, we partition a city into an $M \times N$ grid map based on the longitude and latitude. A grid denotes a region.*

DEFINITION 2 (MEASUREMENTS). *There are different types of measurements in a region for different ST applications, such as crowd flows, air quality [5], bike rent/return [3]. In this study, we use crowd flows as measurements for the case. Typically, the movement of crowds can be represented by a collection of trajectories $\mathbb{P}$. For a grid (m,n) that lies at the m^{th} row and the n^{th} column, two types of crowd flows at the k^{th} timestamp, namely in-flow, out-flow, are defined respectively as*

$$x_k^{in,m,n} = \sum_{Tr_k \in \mathbb{P}} |\{i > 1 | g_{i-1} \notin (m,n), g_i \in (m,n)\}|$$

$$x_k^{out,m,n} = \sum_{Tr_k \in \mathbb{P}} |\{i \geq 1 | g_i \in (m,n), g_{i+1} \notin (m,n)\}|$$

where $Tr_k : g_1 \to g_2 \to \cdots \to g_{|Tr_k|}$ means the trajectory at the k^{th} timestamp; g_i means the geospatial coordinate; $g_i \in (m,n)$ means the point g_i lies within grid (m,n); $|\cdot|$ means the cardinality of a set.

At the k^{th} timestamp, in-flow and out-flow in all $M \times N$ regions can be denoted as $X_k \in \mathbb{R}^{2 \times M \times N}$ where $(X_k)_{0,m,n} = x_k^{in,m,n}$, $(X_k)_{1,m,n} = x_k^{out,m,n}$.

Formally, for a dynamical system over a spatial region represented by a $M \times N$ grid map, there are Q varying measurements in each grid over time. Thus, the observation at any time can be represented by a tensor $X \in \mathbb{R}^{Q \times M \times N}$.

PROBLEM 1. *Given the historical observations X_k for $k = 0, \cdots, t-1$, predict X_t.*

2.2 DeepST

Figure 3 shows the architecture of DeepST. Input includes two parts: previous sequences and global features. Previous sequences consist of three kinds of sequences (closeness, period, and seasonal trend), each of which is fed into a convolutional layer. Then these three layers are mixed (early fusion) followed by three sequential convolutional layers. The global features are fed into a fully-connected layer and then combined with convolutional layers (late fusion). The target is the 3D-tensor at time t.

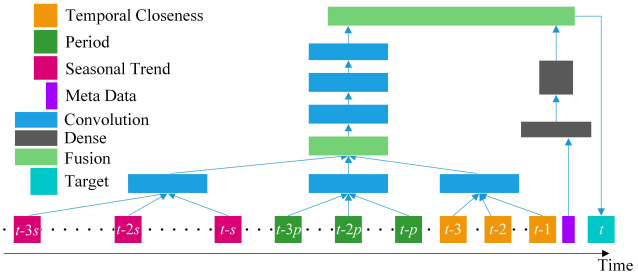


Figure 3: DeepST Architecture.

1) **Temporal-Dependent Instances.** For an ST prediction problem, the input may be a very long sequence of observations, which become very challenging to learn temporal and spatial properties in a single model. On the other side, some timestamps in the sequence own higher correlation than others for prediction. With domain knowledge, we can effectively select these higher-dependent timestamp to reduce the size of the input. To our best knowledge, a time serials always has one, or two, or all following temporal properties: 1) temporal closeness; 2) period; 3) seasonal trend. Based on these insights, the first stage of DeepST is to generate the input from all given historical observations which preserves temporal dependencies.

2) **Convolutions over Sequences.** We here leverage the CNN module to capture spatial closeness dependency. The input of the classical convolution is a tensor (e.g. RBG image), therefore it can be written as $f(W * X + b)$ where $*$ denotes the convolution operator followed by an activation f, W and b are the parameters. Figure 4 shows the convolutions that naturally provide the capacity of capturing spatial dependencies. We found that one convolutional layer can commendably describe near dependency in spatial regions, and two convolutional layers can further depict far dependency. It means more convolutions can capture much farther dependency, and even city-wide dependency.

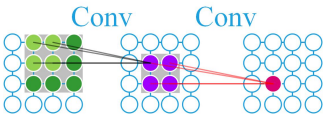


Figure 4: Convolutions for capturing near and far dependencies. A node represents a spatial region.

In our case, the input are three sequences of tensors (see Figure 3): 1) the temporal closeness part $[X_{t-l_c}, \cdots, X_{t-1}]$; 2) the period part $[X_{t-l_p \cdot p}, X_{t-(l_p-1)\cdot p}, \cdots, X_{t-p}]$; 3) the seasonal trend part $[X_{t-l_s \cdot s}, X_{t-(l_s-1)\cdot s}, \cdots, X_{t-s}]$, where l_c, l_p, l_s denote lengths of closeness, period, and trend sequences, respectively, and p and s are a fixed period (e.g., one-day) and seasonal trend span (e.g. one-week), respectively. With these notation, the convolution over sequences of tensors can be written as

$$H_c^{(1)} = f\left(\sum_{j=1}^{l_c} W_{cj}^{(1)} * X_{t-j} + b_c^{(1)}\right)$$

$$H_p^{(1)} = f\left(\sum_{j=1}^{l_p} W_{pj}^{(1)} * X_{t-j\cdot p} + b_p^{(1)}\right)$$

$$H_s^{(1)} = f\left(\sum_{j=1}^{l_s} W_{sj}^{(1)} * X_{t-j\cdot s} + b_s^{(1)}\right)$$

where $*$ denotes the convolution operator; f is an activation function, *e.g.* the rectifier $f(z) := \max(0, z)$ [2] in the paper; $W_\cdot^{(1)}, b_\cdot^{(1)}$ are the parameters in the first layer. $H_c^{(1)}$, $H_p^{(1)}$, $H_s^{(1)}$ are the outputs of the first convolutional layer over close, periodic, trend sequences, respectively.

3) **Fusions.** Fusion, also knows as feature-fusion in the deep learning, can combine multiple datasets within a model using feature-level-based methods [4]. According to fusing time, there are two common types in DNN: early and late fusions [1], which have different functions and will be used to fuse different types of ST data in our model.

(a) **Early Fusion**

To capture sequential, periodic and seasonal trend patterns together, we employ *early fusion* followed by a convolution layer which is good at fusing the similar domains' data [1]. The early fusion based convolution can be written as

$$H^{(2)} = f\left(W_c^{(2)} * H_c^{(1)} + W_p^{(2)} * H_p^{(1)} + W_s^{(2)} * H_s^{(1)} + b^{(2)}\right)$$

Afterwards, one can stack more convolutional layers upon it. In our architecture, we continue to stack two convolutional layers. Therefore, there are totally 4 convolutional layers in the current setting.

(b) **Late Fusion**

Being different from early fusion, *late fusion* is more adept at fusing different domains' data. Meta feature can provide some global information such as dayofweek, meteorological condition, which are always beneficial to predict the crowd flows, air quality. In our implementation, we use external factors (i.e. dayofweek, weekday/weekend) as the global features. Let G_t be the global feature vector, the late fusion can be written as

$$\widehat{X}_t = \tanh\left(W^{(5)} \cdot H^{(4)} + W_G^{(5)} \cdot G_t + b^{(5)}\right)$$

where $\widehat{X}_t$ is the predicted tensor. tanh is a hyperbolic tangent that ensures the output values are between -1 and 1.

The loss function used is mean squared error: $\|\widehat{X}_t - X_t\|_2^2$.

Table 1: Description on Models

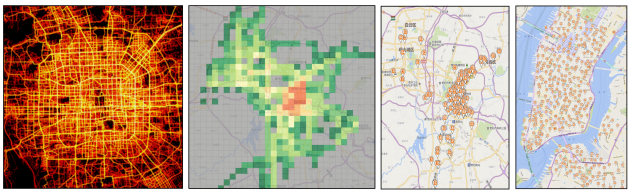
Models	Description
	Baselines
ARIMA	autoregressive integrated moving average
SARIMA	seasonal ARIMA
VAR	vector autoregressive model
CNN	convolutional neural networks, the input is X_{t-1}
	DeepST
C	temporal closeness sequence
CP	C + periodic sequence
CPT	CP + seasonal trend sequence
CPTM	CPT + meta data

Note: Convolutions in DeepST and CNN have the similar setting. There are totally 4 convolutional layers, each of which has 64 feature maps with 3×3 kernels.

3. EVALUATION

Models: According to different temporal-dependent instances generating processings, DeepST has 4 variants (i.e. C, CP, CPT, CPTM). Table 1 shows the detail of these 4 variants as well as 4 baselines.

Datasets (Figure 5): (a) **TaxiBJ15**: In-/Out- flows are calculated from taxi trajectories in Beijing from 3/1/2015 to 6/30/2015 (time interval = 30 minute); (b) **TaxiGY16**: In-/Out- flows are calculated from taxi trajectories in Guiyang from 3/18/2016 to 5/4/2016 (time interval = 30 minute); (c) **LoopGY16**: Two types of traffic flows are collected from loop detectors in Guiyang from 10/1/2015 to 4/1/2016 (time interval = 30 minute); (d) **BikeNYC14**: Bike rent/return numbers are collected from bike stations in NYC from 4/1/2014 to 9/30/2014 (time interval = 1 hour). Each of them are divided into two parts: the last week's data is used as test set, the rest is used as training set.



(a) Beijing > 30,000 Taxicabs TS: (2, 32, 32) | (b) Guiyang > 6,000 Taxicabs TS: (2, 32, 32) | (c) Guiyang 75 Loop Detectors TS: (2, 32, 32) | (d) New York City 344 Stations TS: (2, 16, 8)

Figure 5: Datasets. TS: tensor shape, i.e., the size of the instance at certain timestamp.

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

$$RMSE = \sqrt{\frac{1}{z}\sum_{i}(v_i - \hat{v}_i)^2}$$

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

3.1 Diverse ST Applications

We evaluate the proposed deep learning-based prediction models on different ST datasets, as shown in Table 2. It is easy to see that our DeepST models outperform 4 baselines and CPTM is almost the best among them. It means meta data is beneficial.

Table 2: RMSE. The smaller, the better.

Models	TaxiBJ15	TaxiGY16	LoopGY16	BikeNYC14
ARIMA	25.58	23.31	137.83	10.56
SARIMA	29.11	26.51	135.25	10.07
VAR	25.59	22.70	146.16	9.92
CNN	26.08	22.92	183.51	8.55
C	23.63	22.09	132.26	8.39
CP	23.84	21.51	**129.13**	7.64
CPT	23.33	20.98	130.53	7.56
CPTM	**22.59**	**19.97**	130.25	**7.43**

3.2 Multi-step Ahead Prediction

One can use historical and near predicted future values to predict farther values in the future, which is called as multi-step ahead prediction in this paper. Figure 6 shows the related results on the dataset TaxiBJ15. DeepST here is the best of 4 DeepST variants. It demonstrates that DeepST perform best and can also effectively predict a sequence of values in future.

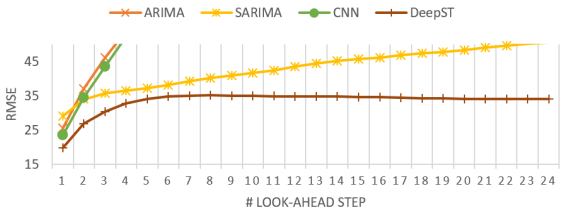


Figure 6: Multi-step Ahead Prediction

3.3 More Data Much Better?

We here collect more taxi trajectories (more than 400 days from 2013 to 2016) in Beijing. The data in the last week is used as testing data. We use previous 7 days, 14 days, ..., 427 days as training data to learn 61 DeepST models with the same network setting, and evaluate them on testing data, as shown in Figure 7. We found that the more data make the model more robust.

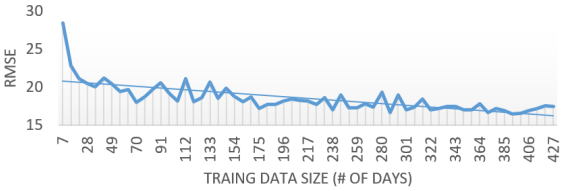


Figure 7: More training data

4. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a DNN-based prediction model for ST data that can capture both temporal and spatial properties at once. We applied it to build a real-time crowd flows forecasting system UrbanFlow which help users monitor the past crowd flows and tell them the future ones. We evaluated DeepST on a variety of ST prediction tasks, including crowd flows, rent/return of bikes, traffic flows. We will continue to improve DeepST, polish UrbanFlow, and do more experiments & evaluation.

Acknowledgments

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

5. REFERENCES

[1] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Y. Li, Y. Zheng, H. Zhang, and L. Chen. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 33. ACM, 2015.

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

[5] Y. Zheng, F. Liu, and H.-P. Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.