DNN-Based Prediction Model for Spatio-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

Advances in location-acquisition and wireless communication technologies have led to wider availability of spatio-temporal (ST) data, which has unique *spatial* properties (*i.e.* geographical hierarchy and distance) and *temporal* properties (*i.e.* closeness, period and trend). In this paper, we propose a Deep-learning-based prediction model for Spatio-Temporal data (DeepST). We leverage ST domain knowledge to design the architecture of DeepST, which is comprised of two components: *spatio-temporal* and *global*. The *spatio-temporal* component employs the framework of convolutional neural networks to *simultaneously* model *spatial* near and distant dependencies, and *temporal* closeness, period and trend. The *global* component is used to capture global factors, such as day of the week, weekday or weekend. Using DeepST, we build a real-time crowd flow forecasting system called UrbanFlow[1]. Experiment results on diverse ST datasets verify DeepST's ability to capture ST data's spatio-temporal properties, showing the advantages of DeepST beyond four baseline methods.

Keywords

Deep Learning; Spatio-Temporal Data; Prediction

1. INTRODUCTION

Advances in location-acquisition and wireless communication technologies have resulted in massive amounts of data with spatial coordinates and timestamps, called ST data, in a diverse range of domains, including transportation, environmental science, communication systems, and social networking services [4]. Being different from text and image data, ST data has two unique attributes: 1) spatial properties, which consists of a geographical hierarchy and distance, and 2) temporal properties, which consists of closeness, period and trend.

*Yu Zheng is the correspondence author of this paper.
†The research was done when the third, fourth and fifth authors were interns in Microsoft Research.
[1]http://urbanflow.sigkdd.com.cn

SIGSPATIAL'16 October 31 - November 03, 2016, Burlingame, CA, USA

ACM ISBN 978-1-4503-4589-7/16/10.
DOI: http://dx.doi.org/10.1145/2996913.2997016

1) Spatial properties: First, locations at a higher level of 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 them. 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. Likewise, ST data usually has a certain periodic pattern, which repeats with a certain frequency. For instance, traffic conditions during morning rush hours may be similar on 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 [1], and 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 two key components: *spatio-temporal* and *global*. The contributions of our work are two-fold:

- We design a novel deep learning architecture for spatio-temporal data using domain knowledge and propose employing 1) temporal properties to select appropriate timestamps, for modeling temporal closeness, period, and trend; 2) convolutions to capture spatial near and distant dependencies; 3) early and late fusions to fuse similar ST data as well as the global information.
- We apply DeepST to predict citywide crowd flows, and develop a real-time flow forecasting system (called UrbanFlow), which can effectively monitor fine-grained crowd flows and provide the future ones in cities.

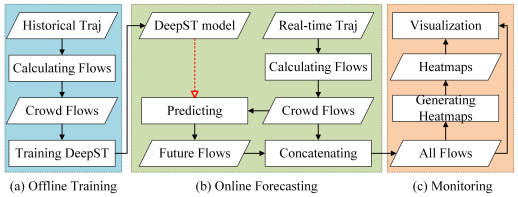
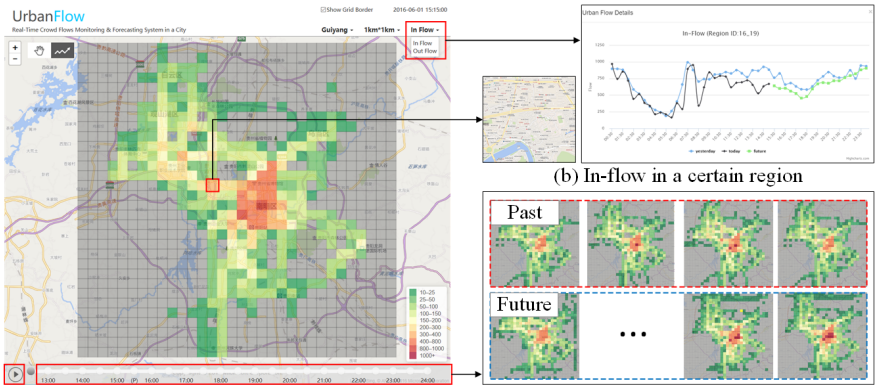


Figure 1: System Framework. Traj: trajectories.



(a) User interface of UrbanFlow (b) In-flow in a certain region (c) Heatmaps in past & future

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 paper. There are three major components in our framework: offline training, online forecasting, and website monitoring. In offline training, the collected trajectories (*e.g.*, taxi) from a city are fed into a "Calculating Flows" module that outputs two types of flows (see Definition 2). These historical flows are then used to learn the DeepST model, which will be introduced in Section 2.2. In online forecasting, starting with calculating crowd flows from real-time trajectories, the learned DeepST model is used to predict future flows that are concatenated with real-time flows later. In the last component, in order to monitor intuitively, we generate heatmaps from 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 more detailed flows. Section 2 will introduce the details of website monitoring.

1.2 Demonstration of the System

The system is built as a website, named UrbanFlow. Users can view real-time and forecasting crowd flows in cities. The user interface of the system is shown in Figure 2a. Here, we apply UrbanFlow to the area of Guiyang City, China. The top-right corner of the website shows the buttons which can switch between different types of flows. A user can select any grid (representing 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, and future times 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 (Figure 2c) by clicking "play button" at the bottom-left of Figure 2a.

2. MODELS

In this section, we first formulate the ST prediction problem, and then introduce our 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 where a grid represent a region.*

DEFINITION 2 (MEASUREMENTS). *There are many different types of measurements in a region for different ST applications, such as crowd flows [1], 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*

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

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

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, which is composed of two components: *spatio-temporal* and *global*. As illustrated in the bottom-left part of Figure 3, we feed all historical observations into the *spatio-temporal* component. According to temporal properties, different timestamps are selected and concatenated together to model *closeness*, *period* and *trend*, respectively. Taking *closeness* for example, we use a convolution to merge X_{t-3}, X_{t-2} and X_{t-1}. The outputs of *closeness*, *period* and *trend* are further combined via *early fusion*, feeding them into a number of convolutional layers. In the *global* component, we first get meta

data of the predicted time t, and transform it into a binary vector, which is then fed into a block that contains one (or a few) fully-connected layer(s). The outputs of the above components are merged via *late fusion* followed by the *Tanh* activation function. The predicted tensor at time t is $\widehat{X}_t$.

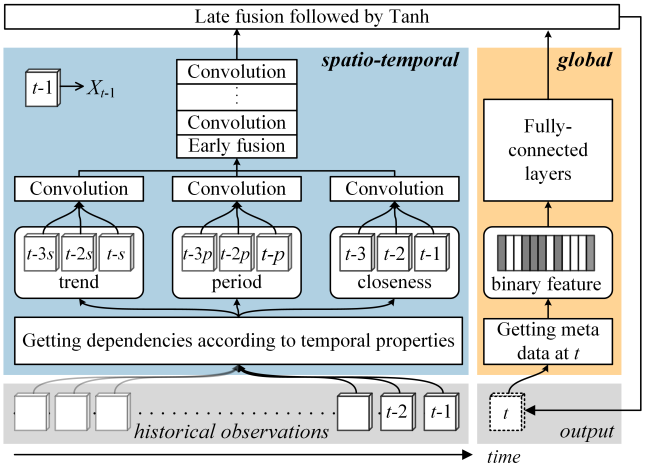


Figure 3: DeepST Architecture.

1) **Getting Temporal Dependencies.** For an ST prediction problem, the input may be a very long sequence of observations, for which it can be very challenging to learn temporal and spatial properties in a single model. Furthermore, some timestamps in the sequence own higher correlation than others for prediction. With ST domain knowledge, we can effectively select these higher-dependent timestamps to reduce input size. To the best of our knowledge, a time series always has one, or two, or all of the following temporal properties: 1) temporal closeness; 2) period; 3) trend. Based on these insights, we can get the recent, near, and distant timestamps from all given historical observations to model temporal closeness, period and trend, respectively. The closeness part is denoted as $[X_{t-l_c}, \cdots, X_{t-1}]$ where l_c is the number of dependent timestamps in the closeness part. Likewise, the period and trend parts are denoted as $[X_{t-l_p \cdot p}, X_{t-(l_p-1)\cdot p}, \cdots, X_{t-p}]$ where l_p, l_s are numbers of dependent timestamps in the period and trend parts, respectively, and p and s are a fixed period (*e.g.*, one-day) and seasonal trend span (*e.g.*, one-week), respectively.

2) **Convolutions.** Here, we leverage the convolutional operator to capture spatial 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, and W and b are the parameters. Figure 4 shows the convolutions that naturally provide the capacity of capturing spatial dependencies. We find that one convolutional layer can commendably describe near dependency in spatial regions, and two convolutional layers can further depict distant dependency. This means more convolutions can capture much farther dependency, and even city-wide dependency.

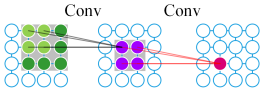


Figure 4: Convolutions for capturing near and distant dependencies.

The closeness, period and trend parts are all fed into the similar convolutional layer. With convolution, their outputs are respectively

$$H_c^{(1)} = f\left(\sum_{j=1}^{l_c} W_{cj}^{(1)} * X_{t-j} + b_c^{(1)}\right)$$
$$H_p^{(1)} = f\left(\sum_{j=1}^{l_p} W_{pj}^{(1)} * X_{t-j\cdot p} + b_p^{(1)}\right)$$
$$H_s^{(1)} = f\left(\sum_{j=1}^{l_s} W_{sj}^{(1)} * X_{t-j\cdot s} + b_s^{(1)}\right)$$

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.** According to fusing time, there are two common types in DNN: early and late fusions, which have different functions and will be used to fuse different types of ST data in our model.

(a) *Early Fusion.* To capture closeness, period and trend all together, we employ *early fusion* followed by a convolution layer which is good at fusing similar domains' data. The early fusion based convolution can be written as

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

where $W_\cdot^{(2)}$ and $b^{(2)}$ are the parameters.

Afterwards, one can stack more convolutional layers onto it. In our implementation, we continue to stack two convolutional layers. Therefore, there are a total of 4 convolutional layers in our current setting.

(b) *Getting Meta Data & Late Fusion.* Being different from early fusion, *late fusion* is more adept at fusing different domains' data. Meta features 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 get meta data (*i.e.* dayofweek, weekday/weekend) as the global features for the timestamp t. Let G_t be the meta feature vector at t, the late fusion can be written as

$$\widehat{X}_t = \tanh\left(W_{st} \cdot H_{st} + W_G \cdot G_t\right) \quad (1)$$

where H_{st} is the output of the *spatio-temporal* component of Figure 3, W_{st} and W_G are the parameters. tanh is a hyperbolic tangent that ensures the output values are between -1 and 1.

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

3. EVALUATION

Models: According to different temporal dependencies, our 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: Table 2 shows the datasets used in our experiment. a) **TaxiBJ15**: In-/out- flows are calculated from taxi trajectories according to Definitions 1 and 2; b) **TaxiGY16**: In-/out- flows are calculated from taxi trajectories in Guiyang; c) **LoopGY16**: Two types of traffic flows are collected from loop detectors in Guiyang; d) **BikeNYC14**: Bike rent/return numbers are collected from bike stations

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 a total of 4 convolutional layers, each of which has 64 feature maps with 3×3 kernels.

Table 2: Datasets

Dataset	**TaxiBJ15**	**TaxiGY16**	**LoopBJ15**	**BikeNYC14**
Data type	Taxi GPS	Taxi GPS	Loop detector	Bike rent
Location	Beijing	Guiyang	Guiyang	New York
Start time	3/1/2015	3/18/2016	10/1/2015	4/1/2014
End time	6/30/2015	5/4/2016	4/1/2016	9/30/2014
Test set	Last week	Last week	Last week	Last 10 days
Training set	others that are NOT in the test set			
Time interval	0.5 hour	0.5 hour	0.5 hour	1 hour
Gird map size	(32, 32)	(32, 32)	(32, 32)	(16, 8)
Trajectory data				
#taxis/bikes	30,000+	6,000+	\	6,800+
#time intervals	5,760	2,304	8,832	4,392

in NYC. Each of them are divided into two parts: training set and test set, shown in Table 2 in detail. We use a *Min-Max* normalization method to scale all data into the range $[-1, 1]$.

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 3. It is easy to see that our DeepST models outperform 4 baselines and CPTM is the best among them, thus demonstrating the benefits of metadata.

Table 3: 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
DeepST (ours)				
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 referred to multi-step ahead prediction in this paper. Figure 5 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.

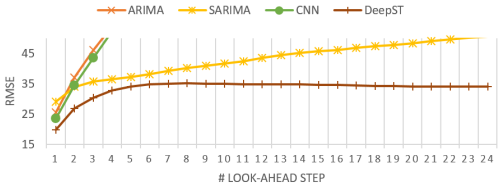


Figure 5: Multi-step Ahead Prediction

3.3 More Data Much Better?

Here, we 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 6. We can see that more data leads to a lower error, which means that the data amount is very important for deep neural networks.

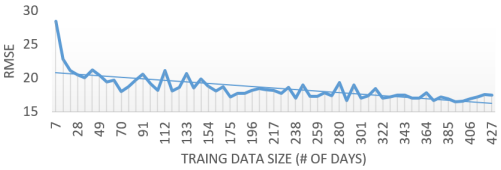


Figure 6: More training data

4. CONCLUSIONS

In this paper, we proposed a DNN-based prediction model for ST data that can capture both temporal and spatial properties at once. We apply it to building a real-time crowd flows forecasting system called UrbanFlow which help users monitor past crowd flows and predict future ones. We evaluate DeepST on a variety of ST prediction tasks, including crowd flows, rent/return of bikes, traffic flows, finding that its performance outperforms the 4 baselines.

5. ACKNOWLEDGMENTS

The work in this paper 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).

6. REFERENCES

[1] M. X. Hoang, Y. Zheng, and A. K. Singh. Forecasting citywide crowd flows based on big data. ACM SIGSPATIAL 2016, October 2016.

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

[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.