Exploiting Spatio-Temporal Correlations with Multiple 3D Convolutional Neural Networks for Citywide Vehicle Flow Prediction

***Abstract*—Predicting vehicle flows is of great importance to traffic management and public safety in smart cities, and very challenging as it is affected by many complex factors, such as spatio-temporal dependencies with external factors (e.g., holidays, events and weather). Recently, deep learning has shown remarkable performance on traditional challenging tasks, such as image classification, due to its powerful feature learning capabilities. Some researchers have applied deep learning on traffic prediction problems considering either spatial relations with 2D convolutional neural networks (CNNs) or temporal relations with long short-term memory networks (LSTMs) independently. These two approaches have shown better performance as compared to many classical methods in traffic prediction. These works are hard to model spatial and temporal relations simultaneously. Some works have utilized LSTMs to connect the high-level layers of CNNs to address this limitation. However, these works only build temporal connections on the high-level features at the top layer while leaving the spatio-temporal correlations in the low-level layers not fully exploited. In this paper, we propose to apply 3D CNNs to learn the spatio-temporal correlation features jointly from low-level to high-level layers for traffic data. We also design an end-to-end structure, named as MST3D, especially for vehicle flow prediction. MST3D can learn spatial and multiple temporal dependencies jointly by multiple 3D CNNs, combine the learned features with external factors and assign different weights to different branches dynamically. To the best of our knowledge, it is the first framework that utilizes 3D CNNs for traffic prediction. Experiments on two vehicle flow datasets Beijing and New York City have demonstrated that the proposed framework, MST3D, outperforms the state-of-the-art methods.**

***Index Terms*—3D CNNs, spatio-temporal dependencies, traffic prediction, vehicle flow prediction.**

I. Introduction

Traffic prediction is of great importance to the traffic management, public safety, and environmental pollution [1]. Vehicle flow prediction is one of vital activities in traffic prediction [2]. As more traffic data of vehicles are collected from GPS devices [3], traffic cameras [4], mobile devices [5], and traditional road sensors [6], the problem is being more complex and voluminous. Therefore, vehicle flow prediction on such traffic data is a challenging task. Robust and scalable traffic prediction models are demanded to handle computational complexity of massive traffic data, extract spatio-temporal correlations of road traffic information, and predict road traffic conditions for the near future.

The latest report from the United Nations [7] states that more than 55% of the world's population now lives in city areas in 2017. As the urban population grows, this figure is expected to grow to 68% by 2050 [7]. Therefore, many cities, such as New York and Beijing, are facing many different pressures and challenges, among which traffic congestion is one of serious problems, resulting in low car speeds, long traveling, waiting time and many more. Many researchers attempt to use machine learning techniques to predict traffic flows so as to avoid traffic congestion situations in cities. A city first can be divided into many small regions. The inflow and outflow of a region are the number of vehicles that have entered the region and the number of vehicles that have left the region, respectively. The traffic condition of each region then is predicted based on the inflow and outflow of the region. Predicting the traffic condition in every region of a city can be basically affected by three categories of important factors as follows.

Factor 1: Spatial dependencies. The inflow of one region (i.e., r_i) of a city can be affected by outflows of its nearby and distant regions. The nearby regions are the neighbors which are either adjacent or near to the border of the region r_i and the distant regions otherwise. In a similar way, the outflow of that region r_i can affect other regions of the city. Besides, it is affected by its own inflow of the region r_i.

Factor 2: Multiple temporal dependencies. The inflow and outflow of the region r_i of the city are affected by short, middle, and long term intervals. For example, the traffic congestion of the region r_i occurring at 6 pm will affect the region's traffic condition at the following hour, i.e., 7 pm. The rush hour of workdays (i.e., from Monday until Friday) is between 8am and 9am in most of cities. It is easily observed that the rush hour pattern repeats every 24 hours on the workdays.

Factor 3: External factors. The inflow and outflow of the region r_i can be directly affected by external factors such as vehicle accidents, road maintenance, weather conditions, and other special events. These factors may also significantly change the inflows and outflows of the regions in a city.

Many existing machine learning techniques have been used in traffic prediction, e.g., k-nearest neighbours (KNN) [8], [9] is used to predict traffic speeds and volumes, and support vector machines (SVM) [10], [11] is used to predict traffic flow. However, all the existing machine learning techniques cannot capture spatio-temporal features of traffic network and cannot be applied to perform traffic prediction on massive traffic data neither. The fast development of neural networks in recent years, especially in deep learning techniques, provides flexibility and generalizability to perform prediction on large multi-dimensional data (e.g, image and video recognition, and bioinformatics). One of examples is that many researchers have successfully applied deep learning techniques to predict future traffic conditions using a sequence of historical sensor data with traffic conditions, and the physical roadway linkages and networks [5], [12], [13].

CNN is one of the most commonly-used neural networks for traffic prediction problems. Many studies use two dimensional CNN (2D CNN) and LSTM [5], [12], [14], [15] to capture spatio-temporal feature of traffic data. The models of the methods only connect temporal features at the high level of spatial features, while not connecting temporal features at low-level of spatial features. In this regards, the temporal correlations of the low-level spatial features cannot be fully exploited. Therefore, some discriminative features cannot be extracted so as to improve the traffic prediction accuracy.

To overcome the limitations, we propose a novel spatio-temporal correlation based multiple 3D CNNs architecture (MST3D) in this work. The proposed MST3D builds a traffic prediction model considering all the three categories of factors as discussed above. To the best of our knowledge, our proposed MST3D is the first time that 3D CNNs are applied in the traffic prediction problem and our proposed MST3D can capture both low-level and high-level layers of spatio-temporal features jointly.

In this paper, our contributions are summarized in the following.

- We propose to utilize novel spatio-temporal correlation based 3D convolutional neural networks (3D CNNs) to learn the spatio-temporal features jointly for traffic prediction. The spatio-temporal correlation features can be extracted and learned simultaneously for traffic data from low-level to high-level layers.
- We design a neural network framework, named as MST3D, based on multiple 3D CNNs for vehicle flow prediction, considering spatial and multiple temporal dependencies with external factors. The MST3D can combine the output features of the multiple 3D CNNs with external factors, assigning different weights to different branches dynamically. The inflow and outflow of vehicles can be jointly predicted in our framework
- We evaluate our approach using Beijing taxi and NYC bike datasets. The results demonstrate the advantages of our proposed MST3D over other state-of-the-art methods in the literature.

We organize the remaining of our work as follows. In the next section, we discuss the related work of the traffic prediction. Section III describes our problem definition and analyzes the effectiveness of learning spatio-temporal features using deep 3D CNNs compared with other models. Section IV presents our proposed framework, MST3D. The experimental results are discussed in Sections V. Section VI concludes the work.

II. Related Work

In recent years, many researchers have used different traffic prediction approaches to solve traffic prediction problems occurred in many cities. The proposed traffic approaches can be grouped into two categories, (i) Classical statistical methods and (ii) Artificial neural networks (ANNs), including deep learning methods.

Classical statistical methods can construct different linear or non-linear models to predict the traffic flow. KNN uses the periodicity of the traffic evolution for short-term traffic speed and volume forecasting to construct a traffic prediction model [8], [9]. SVM model [10] and the extensions [11] are selected due to its distinct advantage to input reasonable a larger data source than other similar methods [16]. Additionally, other classical traffic predictive models, such as Kalman filter [17], Markov chain [18], Bayesian networks [19], and Auto-regressive Integrated Moving Average (ARIMA) [20]–[22], are constructed to solve time-series traffic prediction problems. These models can give better correlations on the successive time sequences of traffic variables. However, it is hard for the classical statistical methods to discover the non-linear spatial and temporal relations of traffic networks.

Unlike the classical statistical methods, ANNs can easily capture the non-linear spatial and temporal relations among the spatiotemporal data. Hence, ANNs are widely applied in different fields such as speech recognition [23], recommendation system [24], computer vision [25], [26], and many more. Recently, ANNs, especially deep learning models, have been applied in the traffic prediction problems [5], [6], [12]–[14], [27]. Some of the studies, such as recurrent neural network (RNN) and its variants, such as long short-term memory (LSTM) network and gated recurrent unit (GRU)), have shown promising performance in traffic prediction compared to the above-mentioned classical statistical methods [12], [13]. One of the main reasons is that RNN and its variants can effectively extract the characteristics from temporal dependencies [12], [13].

Another neural network, CNN or its related convolutional-based residual network can extract the spatial dependencies of the traffic networks by converting the dynamic traffic data into images [6]. But, these neural networks can only capture spatial or temporal information or correlation of the traffic flow data respectively. To overcome the limitation, some combinations of both RNN and CNN networks [5], [15], [27]–[29] have been proposed to learn spatial and temporal dependencies. Yu et al. [12] proposed a deep LSTM model and mixture deep LSTM model using the normal traffic hours and the incident traffic period, respectively. Yao et al. [15] proposed a multi-view spatio-temporal network that combines local CNN, LSTM and semantic network to predict short-time traffic condition. Du et al. [14] proposed a hybrid multi-modal deep learning framework based on multiple CNN-GRU algorithms, which can effectively extract local spatial features and long dependency features together with spatio-temporal correlations from the multi-modal traffic data. However, these

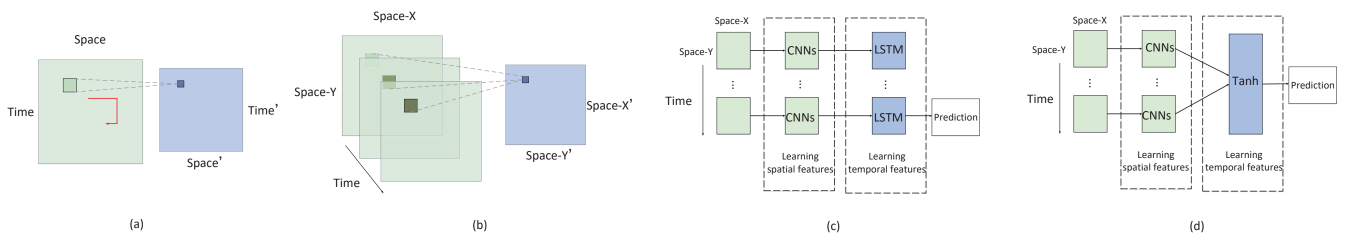


Fig. 1: Limitations of 2D CNN based methods; (a) Model spatio-temporal dependencies in a 2D image; (b) Utilize RGB channels to construct the time dimension; (c) Combine 2D CNN with LSTM or RNN; (d) Utilize 2D CNN for a slice of image along the time dimension and then aggregate them together by a Tanh function.

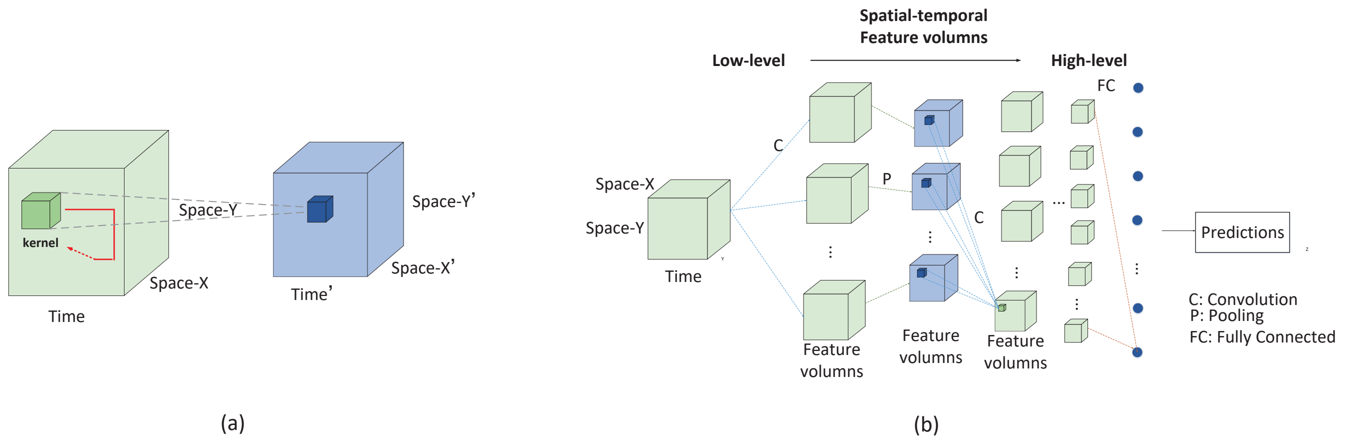


Fig. 2: Extracting spatio-temporal features with 3D CNN; (a) 3D convolution operations, the inputs are 3D volumes, and each generated feature map is also a 3D volume; (b) The hierarchical architecture of 3D CNN.

approaches consider one aspect at one time, either temporal or spatial dependency, while constructing traffic prediction models. We propose multiple 3D Convolutional Neural Networks (MST3D) to address the limitations of the methods discussed previously.

III. Problem Definition and Analysis

A. Citywide Traffic Prediction Problem

In this section, we shall first define the citywide traffic prediction problem and the corresponding notations as follows.

Definition 1 (Citywide Region): Following the previous studies [5], [6], [29], [30], we divide a city into an $I \times J$ grid map based on the longitude and latitude where a grid denotes a region. The regions in the grid map could be defined as non-overlapping pairs (i, j) where i and j mean the region is in the i^{th} row and the j^{th} column of the grid map.

Definition 2 (Historical Traffic Status): The whole time span T of historical traffic status can be spitted as non-overlapping time intervals $T = 1, 2, ..., t-1$.

Definition 3 (Citywide Vehicle Flow): Following the previous studies [5], [29], let P be a collection of trajectories at t time interval. For a grid (i, j) that lies at the i row and the j column, the inflow and outflow of the vehicles at the time interval t are defined as Equ. 1 and Equ. 2, respectively.

$$x_t^{in,i,j} = \sum_{Tr \in \mathrm{P}} |\{\lambda > 1 | g_{\lambda-1} \notin (i,j) \wedge g_\lambda \in (i,j)\}| \quad (1)$$

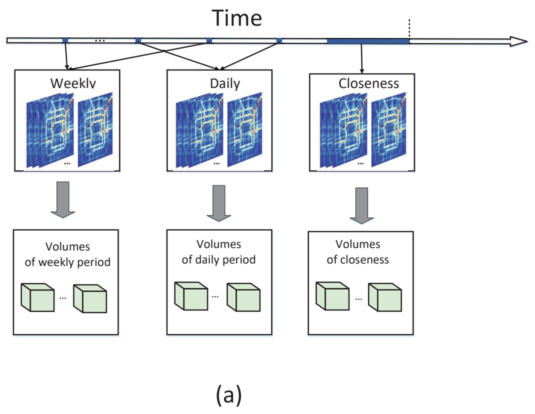
$$x_t^{out,i,j} = \sum_{Tr \in \mathrm{P}} |\{\lambda \geq 1 | g_\lambda \in (i,j) \wedge g_{\lambda+1} \notin (i,j)\}| \quad (2)$$

where $Tr : g_1 \to g_2 \to, ..., \to g_{|Tr|}$ is a trajectory in P, and g_λ is the geospatial coordination; $g_\lambda \in (i,j)$ means the point g_λ lies within grid (i,j), and vice versa.

Problem 1 (Citywide Vehicle Flow Prediction): Given a set of observed historical citywide vehicle flow data with time span $T = 1, 2, ..., t-1$. The problem of vehicle flow prediction aims to predict the inflow and outflow at the next time interval t of the whole grid map of the city.

B. Limitations of 2D CNN based Methods for Traffic Prediction

As discussed in Section I, we need to consider the spatial and temporal dependencies jointly in traffic prediction. We demonstrate that 3D CNN is well suited for spatio-temporal correlation feature learning compared with the 2D CNN and other 2D CNN based methods (e.g., 2D CNN plus LSTM).



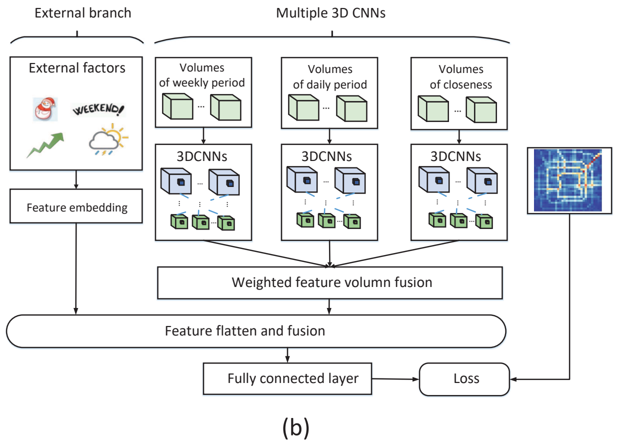


Fig. 3: (a) Modeling spatio-temporal dependencies; (b) Architecture of MST3D.

In 2D CNN, two dimensions of features can be learned owing to 2D convolution and 2D pooling operations. convolutional layers are applied on the 2D local neighborhood feature map to extract spatial features with a 2D convolutional kernel. 2D CNN performs well in learning the features of images which only contains 2 dimensions (longitude and dimensionality) [25], [31]. Fully learning spatial and temporal features is vitally important to the traffic problem. However, it is not suited to learn the features when an extra temporal dimension is included.

Many researchers have made efforts to utilize 2D CNN to learn the spatio-temporal correlation features. Generally, these works can be divided into four catalogs. (1) Treat a dimension of a 2D image as space and another dimension as time [6] as shown in Figure 1(a). However, the two-dimension of spatial dependency are flattened into one dimension and some actual information in the spatial dimension is lost. (2) Another way to adapt 2D CNN to support spatio-temporal feature learning is to replace the RGB channels with slices of the time. However, 2D convolution applied on multiple images (treating them as different channels) also results in an image. Hence, it also loses temporal information of the input signal right after every convolution operation. (3) Some works utilize 2D CNN with LSTM or RNN [15], [28], [32] as shown in Figure 1(c). 2D CNN captures the near- and far-side spatial dependencies and subsequently the long-term temporal feature is learned by LSTMs. Therefore, the integrated networks inherit the advantages of 2D CNNs and LSTMs neural networks. This category of approaches, however, only build temporal connections on the high-level features at the top layer while the correlations in the low-level spatial features cannot be fully exploited. (4) Some works utilize 2D CNN to learn the spatial feature for a slice of image in time dimension and then aggregate them together by a Tanh function [5]. However, similar to the third method, the temporal dependency of the low-level spatial features cannot be learned.

C. *Learning Spatio-temporal Features with 3D CNN*

Compared to 2D CNN, 3D CNN has the ability to model 3-dimension information owing to 3D convolution and 3D pooling operations. If we model the traffic data into 3D volumes with spatial and temporal dimensions, 3D CNN could preserve the temporal dependencies of the volumetric data resulting in an output volume [33], [34], as shown in Figure 2(a). Moreover, adopting the same kernel sharing across space and time dimensions (highlighted in dark blue), the model could take full advantage of spatial and temporal dependencies and potential hidden correlations of the traffic data. Due to the construction of 3D CNN, if we apply 3D convolutions with a 3D kernel, it sweeps over the whole 3D topology.

In view of the definite advantage of 3D CNN to learn spatial and temporal features, an architecture of 3D CNNs is proposed by stacking the convolution layers, pooling layers and fully connection layers. As shown in Figure 2, 3D feature volumes are learned from low-level to high-level by stacking the convolution layer, and it employs different spatio-temporal kernels followed by the non-linear activation functions. In the layer of pooling, the produced feature volumes can be subsampled with max-pooling operation in order to reduce variance and computation complexity and extract low level features from the cubic neighborhood [33], [34]. In the final stage, all the extracted 3D features are flattened into a vector and subsequently an activation function is employed to predict the ultimate output.

IV. Proposed MST3D Framework

This section describes the details of our proposed MST3D framework to predict citywide vehicle flow. Utilizing MST3D consists of 3 steps where each step is described in the following:

Step 1. The citywide vehicle flow data is first converted into multiple 3D volumes with spatial and temporal information.

Step 2. A training dataset of 3D volumes is used to train a model using our proposed framework MST3D.

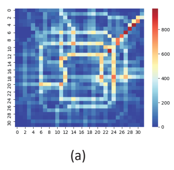
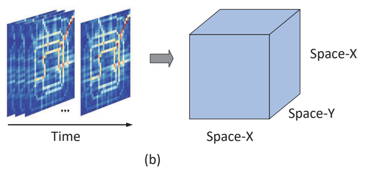



Fig. 4: Modeling spatio-temporal correlation dependency. (a) A multi-channel image that denotes the inflow/outflow of a city; (b) A multi-channel volume that denotes a slice of multi-channel images along the time dimension.

Step 3. Last step, the trained model is to predict vehicle flow in citywide areas.

The process of modeling vehicle flow data with spatio-temporal correlation based 3D volumes is depicted in Figure 3(a). The framework of MST3D is depicted in Figure 3(b). There are two components in MST3D, multiple 3D CNNs and external branch. The multiple 3D CNN components also includes multiple branches based on 3D CNN.

The spatial and multiple temporal correlation features of traffic data can be learned in MST3D using 3D convolution layers and pooling layers. Multiple temporal dependencies contain the *closeness* dependency and different *periodic* (e.g., *daily*, *weekly* and *monthly*) dependencies. A single 3D CNN branch is normally to learn the spatial information and a single type of temporal dependency together. In the external branch, we need to manually extract some features from external datasets that provide weather conditions, event information and other external information. Then, we feed the extracted features into a two-layer fully-connected neural network. As a result, spatio-temporal features and external features are fused together. Lastly, we apply a fully-connected neural network that calculates the cross-entropy loss in traffic prediction of the vehicle flow.

We describe the way of data modeling, architecture of MST3D and correspond training process in the following.

A. Modeling

The process of modeling the citywide vehicle flow data into spatio-temporal correlation based 3D volumes as shown in Figure 4. We first describe how to model citywide inflow/outflow traffic situation with spatio-temporal correlation based 3D volumes. Then we analyse the influence of multiple temporal dependencies in traffic prediction and present the process of modeling multiple temporal dependencies in our methodology.

1) Modeling Spatio-temporal Correlation with 3D Volumes: Given a city that is partitioned into $I \times J$ grid map, at the t time interval, the traffic status can be represented by a tensor $X_t \in R^{i \times j \times k}$, where k denotes the number of traffic variables. The generated tensor can be regarded as a special multi-channel image with i pixels height, j pixels width and k channels of each pixel as shown in Figure 4(a). This multi-channel image captures the spatial information of citywide traffic conditions.

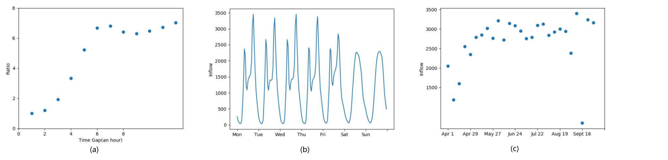


Fig. 5: Multiple temporal dependencies for NYCBike. (a) closeness; (b) daily; (c) weekly.

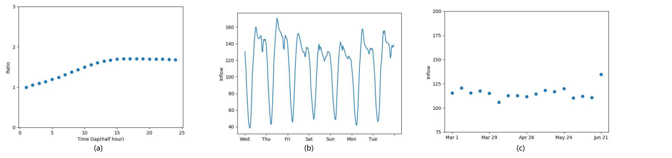


Fig. 6: Multiple temporal dependencies for BJTaxi. (a) closeness; (b) daily; (c) weekly.

Specifically, the multi-channel image can capture the spatio-temporal correlation information of citywide vehicle inflow and outflow by setting $k = 2$. In the next contents, we set $k = 2$ for convenient.

In case of given h time segments, the inflow/outflow of these time segments can be denotes as a tensor $V \in R^{h \times i \times j \times 2}$ This tensor can be regarded as a 3D volume as shown in Figure 4(b). with as a size of $h \times i \times j \times 2$, where h is the number of frames (images). This 3D volume captures the spatio-temporal information of citywide traffic conditions for a slice of time segments.

Formally, the multi-channel image for citywide vehicle flow and the 3D volume for a certain time of citywide vehicle flow are defined in Definition 4 and 5 separately

Definition 4 (Multi-channel Images for Citywide Vehicle Flow): At the time segment t, inflow and outflow in all $I \times J$ regions can be denoted as a multi-channel image $X_t \in R^{i \times j \times 2}$ where $(X_t)_0$ and $(X_t)_1$ denotes the inflow matrix and outflow matrix, respectively.

Definition 5 (3D Volumes for a Certain Time of Citywide Vehicle Flow): Given h time segments, all the multi-channel images in these time segments can be denoted as a 3D volume $V \in R^{h \times i \times j \times 2}$.

2) Modeling Multiple Temporal Dependencies: Figure 5(a) and Figure 6(a) show inflow value in the recent time at a given time interval t plotting by NYCBike and BJTaxi dataset, respectively. The two curves show empirical temporal correlation: the inflow value is affected by the close time intervals. Figure 5(b) and Figure 6(b) describe the inflow value at each time intervals in 7 days. From these two figures, the plotting curves show a certain repeatability pattern. Therefore, we can identify the daily periodic activity clearly on the two datasets. Moreover, Figure 5(c) and Figure 6(c) are plotted to explore the weekly periodicity of the traffic data, where y-axis is the vehicle flow variables at a fixed time interval (e.g., from 9:00 am to 9:30 am) on every Tuesday, and x-axis is the observed time span (e.g., from March 2015 to June 2015).

From the above cases, it is clearly shown that the temporal dependencies and correlation have significant impacts on the traffic state, such as close time, daily periodicity and weekly periodicity regardless of the degrees of influences which are not completely the same. Inspired by above the observations, 3D CNN can be separately constructed for the temporal properties, based on the images with local spatial dependencies.

The steps of modeling are presented in Figure 4(a). We consider the temporal dependency of *closeness* and *periods*. For the *closeness* 3D volumes, a few 2 channel images of intervals in the recent time are used to model temporal *closeness* dependency. Let a recent fragment be $[X_{t-l_c}, X_{t-(l_c-1)}, ..., X_{t-1}]$. This *closeness* is dependent sequence that can be constructed as a 3D volume, $V_c \in R^{l_c \times i \times j \times 2}$.

In a similar way, we also can construct the *periods* volumes. We take the *daily* period as an example. Suppose that l_d is time intervals from the period fragment, and d is the period span. Therefore, the *daily* period of a dependent sequence is $[X_{t-l_d \times d}, X_{t-(l_d-1)\times d}, ..., X_{t-1}]$. This sequence can be constructed as a 3D volume $V_d \in R^{l_d \times i \times j \times 2}$. We only use *daily* and *weekly* periods in our implementation. Generally, our proposed framework can support other user-defined periods (e.g., monthly, seasonally). Other 3D volumes for *periods* dependencies can also be constructed in a similar way.

B. Multiple 3D CNNs

The spatio-temporal correlation based 3D volumes constructed in the modeling phase is then fed into our proposed MST3D as shown in Figure 3(b). Each branch of 3D CNNs targets for a type of temporal dependency. For example, the *closeness* 3D volumes are fed into the *closeness* branch, and the *daily* branch takes the *daily* 3D volumes as inputs.

We take the *closeness* branch to describe how to learn the spatio-temporal features simultaneously. The equation of 3D convolutional operation is as follows:

$$u_{ij}^{\beta}(x,y,z) = \sum_{m,n,l} V_i^{\beta-1}(x-m, y-n, z-l) W_{ij}^{\beta}(m,n,l), \tag{3}$$

where W_{ij}^{β} is the 3D kernel in the β^{th} layer convolving over the 3D feature volume $h_i^{\beta-1}$, $W_{ij}^{\beta}(m,n,l)$ is the element-wise weight in the 3D convolution kernel.

Thus, the equation of the 3D feature volume for the *closeness* branch in β^{th} layer is:

$$V_j^{\beta} = f(\sum_i u_{ij}^{\beta} + b_j^{\beta}), \tag{4}$$

where f is an activation function, b is a bias term connecting the feature maps of adjacent layers.

We also apply max-pooling operation so as to reduce variance and computation complexity and to extract low-level features from the cubic neighborhood.

As discussed in Section IV-A, all the regions are affected by multiple temporal dependencies. The degrees of influence on the regions may be different. Inspired by the observations, we propose a novel parametric-tensor-based fusion method that can fuse *closeness*, *daily* and *weekly* branches, similar to the method in [5]. The equation of fusion method is as follows,

$$V_{fusion} = W_c \otimes V_c + W_d \otimes V_d + W_w \otimes V_w \tag{5}$$

where V_{fusion} denotes the fused features; $\otimes$ is Hadamard product (i.e., element-wise multiplication for tensors); V_c, V_d, V_w are the feature volumes extracted by *closeness*, *daily* and *weekly* branches respectively; W_c, W_d, W_w are the learnable parameters that adjust the degrees affected by different branches.

Then, the fused features V_{fusion} is flattened into a vector named as V_{mc}. V_{mc} is the output of the multiple 3D CNNs.

C. External Branch

Many complex external factors, such as weather conditions and special events, have great influence on the citywide traffic situation. For example, holiday like Chinese New Year and Christmas can have a heavy traffic flow compared to non-holiday. Another example is that heavy rain can also sharply slow down the vehicle speed due to slippery road.

In this paper, we mainly focus on the weather condition, holiday event, and metadata (i.e., day of the week, weekday and weekend). To predict traffic flow at time interval t, the holiday event and metadata can be easily calculated. However, the weather condition at future time interval t is hard to calculate. Instead, to solve the issue, we can use the forecasting weather condition at time interval t or the approximately weather condition from historical weather data at time interval $t-1$. We stack two fully-connected layers upon E_t, i.e., the first layer can be viewed as an embedding layer for each subfactor followed by an activation, and the second layer is to map low to high dimensions as V_{ext} that have the same shape with V_{mc} which is generated by multiple CNNs.

We then directly merge the output of the multiple CNNs with that of the external components. The fused output $\widehat{V}$ of the multiple CNNs and the external components is defined in Equ. 6:

$$\widehat{V} = V_{mc} + V_{ext} \tag{6}$$

.

Finally, the fused output $\widehat{V}$ is connected with a fully-connected layer using Tanh function, which yields a faster convergence than the standard logistic function in the process of back-propagation learning [25]. The output of that value is utilized to predict inflow/outflow of the city.

D. Training Process of MST3D

The training process of MST3D is outlined in Algorithm 1. At the time interval t, the 2 channels image which denote the infow/outflow values of the city is regarded as the ground-truth. The 3D volumes generated by the historical traffic status is regarded as inputs of MST3D. We adopt *daily* and *weekly* periods in our implementation. Generally, our proposed framework can support other user-defined periods. All the trainable parameters in the proposed MST3D can be initialized randomly and optimized by the back propagation. The back propagation adopts stochastic gradient descent to minimize the cross entropy loss function of the MST3D. We also apply dropout strategy to improve the capability of model generalization of our proposed MST3D.

Algorithm 1 MST3D Algorithm

```
Require: Historical observations: $X_{0,...,n-1}$;
    External features: $E_{1,...,n-1}$;
    Lengths of closeness, daily and weekly: $l_c, l_d, l_w$;
    Daily span: $d$, Weekly span: $w$.
Ensure: Learned MST3D model.
 1: $D \leftarrow \emptyset$;
 2: //Modeling traffic values into multiple temporal and spatial volumes;
 3: for all available time interval $t (1 \leq t \leq n-1)$ do
 4:     $V_c = [X_{t-l_c}, X_{t-(l_c-1)}, ..., X_{t-1}]$;
 5:     $V_d = [X_{t-l_d \times d}, X_{t-(l_d-1) \times d}, ..., X_{t-1}]$;
 6:     $V_w = [X_{t-l_w \times w}, X_{t-(l_w-1) \times w}, ..., X_{t-1}]$ ;
 7:     //$X_t$ is the target at time $t$;
 8:     put a training instance $(V_c, V_d, V_w, E_t)$ into $D$;
 9: end for
10: //Training the model;
11: Initialize all learnable parameters $\theta$ in MST3D;
12: while stopping criteria is not met do
13:     Randomly select a batch of instances $D_b$ from $D$;
14:     Feed each $V_c, V_d, V_w, E_t$ of an instance in $D_b$ into the
        corresponding branch respectively;
15:     Find $\theta$ by minimizing the objective with $D_b$;
16: end while
17: return
```

V. Experiment

In this section, we evaluate the performance of our proposed MST3D. To show a comprehensive quantitative evaluation, we also compare other existing methods with our proposed MST3D.

A. Experiment Settings

We configure a Linux server and the other configurations to run the experiment as follows:

- 8 Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHZ
- 256GB RAM
- 4 NVIDIA P100 GPUs

Two large real-world datasets, i.e., New York City (NYC) and BeiJing (BJ) datasets, are used in the experiment. Each dataset that contains trip records and external contexts is detailed in the following.

- **NYCBike:** The bike trip data is collected from NYC Bike system in 2014 from Apr. 1st to Sept. 30th. We choose the data in the last 10 days as the testing data, while others as the training data. For the external information, holidays are provided in the dataset.
- **BJTaxi:** This is the taxi trajectoriy data in Beijing that has 4 time periods: 1st Jul. 2013 - 30th Otc. 2013, 1st Mar. 2014 - 30th Jun. 2014, 1st Mar. 2015 - 30th Jun. 2015, 1st Nov. 2015 - 10th Apr. 2016. The last four weeks are selected as the testing data, and others as the training data. External information includes holidays, weather conditions and temperature.

B. Implementation Details

Data Preprocessing. For the NYCBike dataset, we split the whole city into 8×16 regions. The length of each time segment is set to 1 hour. For the BJTaxi dataset, we split the whole city into 32×32 regions. The length of each time segment is set to 30 minutes. Using Definition 3, we get two types of vehicle flows of NYCBike and BJTaxi. For external influence, day-of-week, weekend/weekday, holidays and weather conditions are transformed into binary vectors. The method of transformation we used is similar to the method proposed in [5]. We apply Min-Max normalization to convert traffic values by $[0, 1]$ scale. After the prediction step, we de-normalize the prediction values and then use them for traffic evaluation. Similarly, we also apply Min-Max normalization to the existing methods before compared them with our proposed MST3D[1].

Parameters. The python libraries, the Tensorflow (version 1.2.1) and the Keras (version 2.1.6) are used to build our models. The inflow and outflow are fixed as 2 channels in the generated volumes. Thus, the inflow and outflow of vehicles can be predicted jointly. In our implementation, the lengths of *closeness*, *daily* and *weekly* on NYCBike are set to 4, 4, and 4, respectively. As the time segment in BJTaxi is set to half an hour, the lengths of *closeness*, *daily* and *weekly* are set to 6, 4, and 4, respectively. Batch normalization is used and the batch size is set to 64 in the experiment.

While designing the structure of the deep 3D CNNs, two important factors need to be considered: (i) depth and (ii) hyper parameters of convolutional layer and pooling layer (e.g., convolutional filter size and polling size). For the NYCBike dataset, we apply two 3D convolutional layers as the sizes of our 3D volumes are small (i.e., $4 \times 8 \times 16$ in all the branches). The kernel sizes in all the branches are set to $(2, 3, 3)$. The number of 3D convolutional filters of the first layer is 32, and of that the second layer is 64. A max pooling layer with the small size of $(1, 2, 2)$ is applied with the following two 3D convolutional layers. An extra dropout layer is set to 0.25 dropout rate so as to reduce the over-fitting issue.

For the BJTaxi dataset, three 3D convolutional layers are applied on all the multiple CNNs. One of the main reasons is that the spatial dimensions are 32×32 which is bigger than that of the NYCBike dataset. The number of 3D convolutional filters on three 3D convolutional layers are set to 32, 64 and 64, respectively. In the *closeness* branch, the kernel size is set to $(2, 3, 3)$ for these three layers. In the *dail* and *weekly* branches, the kernel size of the first layer is $(2, 3, 3)$, while the other 2 layers are $(1, 3, 3)$. The above settings can align the

[1]The source code of MST3D will be provided upon request. It will be released to the public after this paper review.

TABLE I: Baseline comparison on NYCBike and BJTaxi.

Method	NYCBike		BJTaxi	
	RMSE	MAPE	RMSE	MAPE
HA	14.35	39.22%	57.69	38.34%
ARIMA	10.07	29.23%	22.78	22.13%
LinUOTD	9.76	28.12%	21.23	20.22%
XGBoost	6.93	23.12%	17.84	17.62%
MLP	7.68	24.93%	18.25	17.83%
ConvLSTM	7.98	25.62%	19.54	18.63%
ST-ResNet	6.33	21.81%	16.89	15.48%
STDN	6.20	21.57%	16.65	15.27%
MST3D	**5.81**	**20.68%**	**15.99**	**14.78%**

output sizes of the *closeness*, *dail* and *weekly* branches. The parameters of the max pooling layer and the dropout layer are the same as those set for the NYCTaxi dataset.

Evaluation Metrics. In our experiment, we choose Rooted Mean Square Error (RMSE) and Mean Average Percentage Error (MAPE) as the evaluation metrics, which are the same metrics used in [5], [15], [29], [30]. In the calculation of MAPE value, the samples with flow values that are less than 10 are ignored, which is a common practice used in the traffic prediction [15], [29]. The two evaluation metrics are defined as follows:

$$RMSE = \sqrt{1/N \sum_{\alpha=1}^{N} (\hat{y_t} - y_t)} \tag{7}$$

$$MAPE = 1/N \sum_{\alpha=1}^{N} \frac{|\hat{y_t} - y_t|}{y_t} \tag{8}$$

where $\hat{y_t}$ and y_t denote the prediction value and the real value for time interval t, and N is the total number of samples.

C. Methods in Traffic Prediction

We compare our model with the following three categories of spatio-temporal prediction methods.

Classical Time-Series Prediction Methods.

- **HA:** Historical average predicts traffic (HA) for a given region basing on the average values of the previous relative time interval in the same region.
- **ARIMA:** Auto-regressive integrated moving average (ARIMA) is a well-known model for understanding and predicting future values in a time series.

Classical Statistical Prediction Methods.

- **LinUOTD [35]:** LinUOTD is a linear regression model with a spatio-temporal regularization.
- **XGBoost [36]:** XGBoost is a widely used boosting tree method.

Deep Learning Methods.

- **MultiLayer Perceptron (MLP):** We compare our proposed MST3D with a neural network which contains four fully connected layers.

TABLE II: Inflow and outflow results on NYCBike

Methods	Inflow		Outflow	
	RMSE	MAPE	RMSE	MAPE
HA	14.02	38.75%	14.91	39.68%
ARIMA	9.97	28.97%	10.49	29.49%
LinUOTD	9.56	27.59%	10.11	28.74%
XGBoost	6.89	22.93%	7.06	23.43%
MLP	7.25	24.63%	8.07	25.29%
ConvLSTM	7.74	25.57%	8.32	25.72%
ST-ResNet	6.08	21.23%	6.63	22.17%
STDN	5.98	21.01%	6.51	21.96%
MST3D	**5.66**	**20.21%**	**5.96**	**21.14%**

TABLE III: Inflow and outflow results on BJTaxi

Methods	Inflow		Outflow	
	RMSE	MAPE	RMSE	MAPE
HA	57.57	37.76%	57.89	39.68%
ARIMA	22.58	22.12%	22.96	22.19%
LinUOTD	21.19	20.02%	21.44	20.33%
XGBoost	17.61	17.42%	18.23	17.69%
MLP	18.23	17.54%	18.30	18.21%
ConvLSTM	19.29	18.55%	19.98	18.72%
ST-ResNet	16.74	15.01%	17.01	15.78%
STDN	16.43	15.12%	16.78	15.44%
MST3D	**15.98**	**14.71%**	**16.11**	**14.85%**

- **ConvLSTM [28]:** ConvLSTM adds convolutional layers to LSTM.
- **ST-ResNet [27]:** ST-ResNet is a CNN-based deep learning framework for traffic prediction. The model uses CNNs to capture trend, period, and closeness information. ST-ResNet models a city's traffic at different times as images. CNNs are used to extract features from historical images. Then a Tanh function is utilized to aggregate the relevant features together. We set the length trend, period and closeness as 4, 4, and 4, respectively.
- **STDN [29]:** STDN uses local CNNs, LSTM and attention mechanism to model the spatial, temporal and dynamics dependencies. We modify some codes of STDN to make STDN predict the inflow/outflow of the city maintaining the network structure of STDN. The input size of local CNNs is set as 7×7, 4 for long-term periodic information. For the length of short-term LSTM, 4 is set for NYCBike and 6 is set for BJTaxi.

D. Performance Evaluation

Table I shows the RMSE and MAPE results of our proposed MST3D as compared to existing methods for the inflow and outflow together on the NYCBike and BJTaxi datasets. Table II and III present the detailed results of inflow and outflow. We run each baseline 10 times and report the average results of each baseline.

From Table I, we can see that our proposed MST3D with the *closeness*, *daily*, *weekly* and *external* branches outperforms above all the baselines by achieving the lowest RMSE and MAPE on both the datasets, i.e., the RMSE and the MAPE of our MST3D in the NYCBike dataset are 5.81 and 20.68%, respectivley. the RMSE and the MAPE of our MST3D in the BJTaxi dataset are 15.98 and 14.78%, respectivley. From Table II and III, we can clearly see that our proposed MST3D framework can give the best accuracy of both the inflow and outflow predictions.

In contrast, the traditional time-series prediction methods (i.e., HA and ARIMA) cannot get good prediction results as they rely on historical records to predict the future values and overlook spatial and other related external features. Even the regression-based methods (e.g., LinUOTD, XGBoost) take spatial correlations as their features. As a result, they can achieve better performances if compared with other conventional time-series approaches. However, they still failed to capture the complex non-linear temporal dependencies and the dynamic spatial relationships. Therefore, our proposed MST3D significantly outperforms above all the existing methods. The results prove the effectiveness of our schemes adopting multiple 3D CNN and extracting external features.

Our proposed model also achieves better performance than MLP and ST-ResNet. One of main reasons is that MLP cannot explicitly model spatial dependency and temporal sequential dependency. Also, ST-ResNet only uses CNN to capture spatial information without considering the temporal sequential dependency.

MST3D also outperforms the category of methods (e.g., ConvLSTM and STDN) which use 2D CNNs and LSTM together for traffic prediction. This category of methods only captures the temporal dependencies for the high-level spatial features, but not considers the temporal correlations with low-level spatial features.

E. Performance of Multiple 3D CNNs Architecture and External Factors

We also studied the performance of multiple 3D CNNs and external factors by applying different branches as proposed in our MST3D. Figures 7 and 8 show that the results of our proposed MST3D and its variants on the NYCBike and BJTaxi datasets, separately. The methods used in this study are listed as follows.

- **MST3D:** Our proposed framework, which combines *closeness*, *daily*, *weekly* and *external* branches.
- **MST3D-C:** This method only captures spatial and temporal dependency of *closeness*.
- **MST3D-CD:** This method uses *closeness* and *daily* branches only.
- **MST3D-CDW:** This method uses *closeness*, *daily* and *weekly* branches.

Figures 7 and 8, we can see that MST3D-C which uses the *closeness* branch only performs better than the other baselines. It demonstrates the effectiveness of applying 3D CNNs to learn spatio-temporal features in traffic prediction. The performance is further improved by adding the *daily* and *weekly* branches. Again, it proves that considering the multiple temporal dependencies can help to increase the accuracy of vehicle flow prediction. An interesting observation is that, for BJTaxi dataset, the results of MST3D-C and MST3D-CD are similar (the RMSE of MST3D-CD is a bit bigger than that of MST3D-C, while the MAPE of MST3D-CD is a bit smaller than that of MST3D-C). It illustrates that adding *daily* branch has a little effect on BJTaxi dataset. The RMSE and MAPE values further decrease by adding the *external* branch. It shows that external factors can significantly affect the performance of our method. Lastly, we can see that the method that combines the *closeness*, *daily*, *weekly* and the *external* branches together can achieve the lowest RMSE and MAPE.

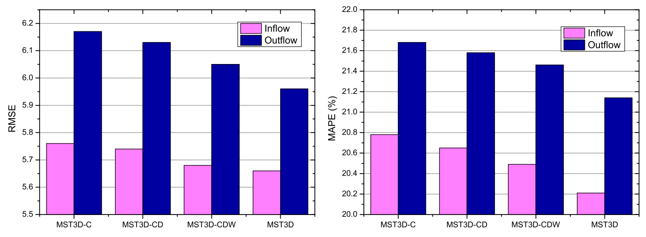


Fig. 7: Results of MST3D and its variants on NYCBike. (a) RMSE results. (b) MAPE results.

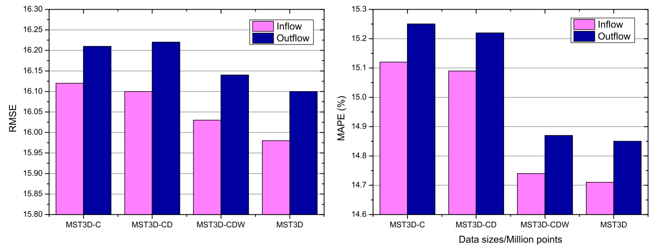


Fig. 8: Results of MST3D and its variants on BJTaxi. (a) RMSE results. (b) MAPE results.

F. Time Complexity of Different Methods

Table IV lists the running time of different existing methods in the training and testing. We utilized a single P100 GPU in all the methods. We only compare our MST3D with the existing methods which have achieved relatively good results. We can see that the running time of STDN are the longest for both of training and testing. The scheme of STDN uses a sliding window over the whole city to train and then predict every region. For example, in NYCTaxi, the size of input is almost 8×16 times of that of ST-ResNet and our proposed MST3D.

In NYCTaxi, our proposed MST3D performs slightly better than ST-ResNet. One of the main reasons is that MST3D uses two 3D conventional layers, while ST-ResNet only uses two conventional layers and 12 residual units. In BJTaxi, the training time of the MST3D is similar to that of the ST-ResNet. However, the testing time of MST3D that uses three 3D layers is slightly longer than that of the 2D ST-ResNet. Obviously, the three 3D layers are more complex than that the 2D layers in neural networks.

TABLE IV: Running time of different methods

Methods	NYCBike		BJTaxi	
	Training time (s)	Testing time (s)	Training time (s)	Testing time (s)
ST-ResNet	150	0.75	4994	1.92
STDN	18980	89.8	379600	207.4
MST3D	126	0.23	5902	2.74

VI. Conclusions

Traffic prediction is a vital part and challenging task in the domain of intelligent transportation system (ITS) as it is affected by many complex factors, such as spatio-temporal dependencies with external influences. In this paper, we propose to utilized 3D CNNs to learn the spatio-temporal features jointly for traffic prediction. A framework, named as MST3D, that based on multiple 3D CNNs is proposed especially for citywide flow prediction considering the correlation of spatial and multiple temporal dependencies, as well as external influences. The inflow and outflow of vehicles can be jointly predicted in our framework. Experiments on two different real-world datasets have been implemented and the results demonstrated that the proposed MST3D outperforms the state-of-the-art baselines. We plan to further investigate the spatio-temporal features learned by 3D CNNs for better interpretability in our future work. In addition, we will also blend road intersection network into our proposed framework.

References

[1] M. R. Jabbarpour, H. Zarrabi, R. H. Khokhar, S. Shamshirband, and K.-K. R. Choo, "Applications of computational intelligence in vehicle traffic congestion problem: a survey," in *Soft Computing*, 2018, pp. 2299–2320.

[2] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," in *TIST*, 2014, p. 38.

[3] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *KDD*, 2014, pp. 25–34.

[4] S. H. Kim, J. Shi, A. Alfarrarjeh, D. Xu, Y. Tan, and C. Shahabi, "Real-time traffic video analysis using intel viewmont coprocessor," in *DNIS*, 2013, pp. 150–160.

[5] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction." in *AAAI*, 2017, pp. 1655–1661.

[6] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," in *Sensors*, 2017, p. 818.

[7] D. Un, "World urbanization prospects: The 2017 revision," in *United Nations Department of Economics and Social Affairs, Population Division: New York, NY, USA*, 2018.

[8] G. A. Davis and N. L. Nihan, "Nonparametric regression and short-term freeway traffic forecasting," in *Journal of Transportation Engineering*, 1991, pp. 178–188.

[9] D. Xia, B. Wang, H. Li, Y. Li, and Z. Zhang, "A distributed spatial–temporal weighted model on mapreduce for short-term traffic flow forecasting," in *Neurocomputing*, 2016, pp. 246–263.

[10] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," in *IEEE Trans. Intelligent Transportation Systems*, 2004, pp. 276–281.

[11] W.-C. Hong, "Traffic flow forecasting by seasonal svr with chaotic simulated annealing algorithm," in *Neurocomputing*, 2011, pp. 2096–2107.

[12] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *SDM*, 2017.

[13] F. Altché and A. De La Fortelle, "An lstm network for highway trajectory prediction," in *ITSC*, 2017, pp. 353–359.

[14] S. Du, T. Li, X. Gong, Z. Yu, and S.-J. Horng, "A hybrid method for traffic flow forecasting using multimodal deep learning," in *arXiv preprint arXiv:1803.02099*, 2018.

[15] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, and J. Ye, "Deep multi-view spatial-temporal network for taxi demand prediction," in *AAAI*, 2018.

[16] M. T. Asif, J. Dauwels, C. Y. Goh, A. Oran, E. Fathi, M. Xu, M. M. Dhanya, N. Mitrovic, and P. Jaillet, "Spatiotemporal patterns in large-scale traffic speed prediction," in *IEEE Trans. Intelligent Transportation Systems*, 2014, pp. 794–804.

[17] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," in *IEEE Trans. Intelligent Transportation Systems*, 2013, pp. 871–882.

[18] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," in *IEEE Trans. Intelligent Transportation Systems*, 2015, pp. 653–662.

[19] S. Sun, C. Zhang, and G. Yu, "A bayesian network approach to traffic flow forecasting," in *IEEE Trans. Intelligent Transportation Systems*, 2006, pp. 124–132.

[20] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," in *Journal of transportation engineering*, 2003, pp. 664–672.

[21] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining kohonen maps with arima time series models to forecast traffic flow," in *Transportation Research Part C: Emerging Technologies*, 1996, pp. 307–318.

[22] Q. T. Tran, Z. Ma, H. Li, L. Hao, and Q. K. Trinh, "A multiplicative seasonal arima/garch model in evn traffic prediction," in *IJCNS*, 2015, p. 43.

[23] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2013, pp. 6645–6649.

[24] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *KDD*, 2015, pp. 1235–1244.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," in *Nature*, 2015, p. 436.

[26] H. Wei, Y. Wang, T. Wo, Y. Liu, and J. Xu, "Zest: a hybrid model on predicting passenger demand for chauffeured car service," in *CIKM*, 2016, pp. 2203–2208.

[27] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks," in *arXiv preprint arXiv:1701.02543*, 2017.

[28] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *NIPS*, 2015, pp. 802–810.

[29] H. Yao, X. Tang, H. Wei, G. Zheng, Y. Yu, and Z. Li, "Modeling spatial-temporal dynamics for traffic prediction," in *arXiv preprint arXiv:1803.01254*, 2018.

[30] D. Deng, C. Shahabi, U. Demiryurek, and L. Zhu, "Situation aware multi-task learning for traffic prediction," in *ICDM*, 2017, pp. 81–90.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[32] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," in *Sensors*, 2017, p. 1501.

[33] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015, pp. 4489–4497.

[34] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *ICCV*, 2017, pp. 5534–5542.

[35] Y. Tong, Y. Chen, Z. Zhou, L. Chen, J. Wang, Q. Yang, J. Ye, and W. Lv, "The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms," in *KDD*, 2017, pp. 1653–1662.

[36] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*, 2016, pp. 785–794.