2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)
October 8-12, 2022, Macau, China

A Fuzzy-based Convolutional LSTM Network Approach for Citywide Traffic Flow Prediction*

Huiyun Yu, Qi Zheng, Shuyun Qian, Yaying Zhang

***Abstract*—Citywide traffic flow prediction is of great importance to intelligent transportation systems and smart cities. Although many deep learning methods have been applied for citywide traffic flow prediction, deep learning is a deterministic representation and sheds little light on data uncertainty. In this paper, a fuzzy-based convolutional LSTM neural network (FConvLSTM) method is proposed to improve the accuracy of citywide traffic flow prediction by taking data uncertainty into consideration. FConvLSTM is a hybrid model which combines fuzzy learning with a convolutional LSTM neural network (ConvLSTM). The impact of data uncertainty is lessened with the help of fuzzy neural networks and ConvLSTM is adopted to explore the spatio-temporal characteristics of traffic data, which can learn spatial dependencies and temporal dependencies jointly. Experimental results on a real dataset verify the outperformance of the FConvLSTM method.**

I. Introduction

Predicting citywide traffic flow plays an essential role in traffic management, emergency plan, and public safety[1]. It will be helpful to optimize the traffic configuration, guide the public to make reasonable travel planning, and mitigate or prevent the occurrence of traffic stampede and traffic congestions. Citywide traffic flow forecast aims to predict the inflow (the number of crowds moving into the region) and outflow (the number of crowds going away from the region) in each region of a city in the given time interval collectively. It is a challenging task since its pattern is influenced by various factors involving spatial dependencies between divergent regions, temporal dependencies among different time intervals as well as external factors like weather, holidays and occasional incidents, etc. Massive observed traffic flow data may help to extract essential features.

Deep learning is a data-driven approach, which can learn essential features from large amounts of data. Some deep learning models have been already developed to deal with citywide traffic flow prediction. Zheng et al.[2] put forward the ST-ResNet framework. It uses three branches sharing the same architecture to model temporal dependencies and, in each branch, residual blocks based on convolutional neural network (CNN) are used to deal with spatial dependencies. A concise deep learning framework called STAR has been proposed in [3], which takes one fully-convolutional residual network to capture the spatio-temporal dependencies. Xu et al.[4] has proposed an entirely CNN-based framework PredCNN with cascade convolutions to deal with the spatio-temporal features of traffic data. In [5], a long short-term memory (LSTM) [5] framework is adopted for extreme condition traffic forecasting. Wu et al. [7] have proposed a hybrid model that integrates an attention mechanism, CNN and LSTM.

However, these deep learning models remain inaccurate due to little consideration on data uncertainty and failing to capture spatial and temporal dependencies jointly. Fuzzy learning[8], based on fuzzy logic, has its advantages in dealing with data uncertainty which is very difficult to be supplanted by other learning approaches. Lots of practical problems have been well solved using fuzzy learning in image processing[10], stock investment [11][12], and Intelligent Motor Control [13], etc. Compared with conventional deterministic knowledge representation, the fuzzy logic representations flexibly construct fuzzy rules to reduce the uncertainties in raw data[14].

Concerning the above issues, we propose a new model called FConvLSTM for improving the citywide traffic flow prediction accuracy, which integrates fuzzy learning with deep learning. In this model, we applied a fuzzy neural network module aiming to reduce data uncertainty. Regarding spatial and temporal dependencies, three convolutional LSTM (ConvLSTM) [15] modules are used to exploit the spatio-temporal features based on the temporal proximity to the predicted frame. Compared with CNNs or LSTMs, ConvLSTM can exploit spatial and temporal features jointly. The deep learning representation (the output of ConvLSTM module) and the fuzzy learning representation (the output of the fuzzy neural network module) are fused and sent to dense layers for general learning. In this way, the citywide traffic flow can be forecasted more accurately.

The main contributions of this paper are summarized as follows:

(1) Taking data uncertainty into consideration, a fuzzy deep learning method for citywide traffic flow prediction is proposed. The proposed FConvLSTM is a combination of fuzzy learning with deep learning which explores the fuzzy logic expression and deep learning expression of traffic data altogether. Moreover, the parameters of membership function are learned adaptively that do not need to rely on human experience and manual intervention.
(2) An end-to-end fuzzy convolutional LSTM network model is designed. The proposed FConvLSTM can capture the spatial and temporal features of traffic data jointly.

* This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB2100800.

Huiyun Yu, Qi Zheng, Shuyun Qian and Yaying Zhang are all with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, 200092, China.(Author email: {1730805, zhengqi97, shuyun_qian, yaying.zhang}@tongji.edu.cn).

2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC) | 978-1-6654-6880-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITSC55140.2022.9922491

(3) We validate our approach on the real taxi trajectory dataset for Beijing. And the experimental results demonstrate the outperformance of our approach.

II. Preliminaries

A. Preliminaries

Definition 1 Region: For convenience, a city is mapped into $I \times J$ grids, where each grid denotes a *region* as shown in Fig. 1(a).

Definition 2 Inflow/Outflow: For each region, we record two measurements on it, namely *inflow* and *outflow*. Inflow represents the total crowds (i.e. vehicles) entering a certain region during a given time interval, while outflow indicates the sum of crowds leaving from a certain region. The formulas are defined as follows:

$$x_t^{in,i,j} = \sum_{Tr \in \mathbb{P}} |\{k > 1 | g_{k-1} \notin (i,j) \wedge g_k \in (i,j)\}| \qquad (1)$$

$$x_t^{out,i,j} = \sum_{Tr \in \mathbb{P}} |\{k \geq 1 | g_k \in (i,j) \wedge g_{k+1} \notin (i,j)\}| \qquad (2)$$

$x_t^{in,i,j}$ and $x_t^{out,i,j}$ represent the inflow and outflow of the grid (i,j) during the t^{th} time interval respectively. $\mathbb{P}$ denotes a collection of trajectories during t^{th} time interval. g_k means the geographic location of the k^{th} point of trajectory Tr. $g_k \in (i,j)$ indicates that g_k lies in the grid (i,j). Hence, $\boldsymbol{X}_t^{in}$ and $\boldsymbol{X}_t^{out}$ denotes the inflow matrix and outflow matrix during the t^{th} time interval respectively, where $\left(\boldsymbol{X}_t^{in}\right)_{i,j} = x_t^{in,i,j}$, $(\boldsymbol{X}_t^{out})_{i,j} = x_t^{out,i,j}$.

With the definition of region, inflow, and outflow, the crowds of the t^{th} time interval of a city can be expressed by a tensor $\boldsymbol{X}_t \in R^{2 \times I \times J}$ where $(\boldsymbol{X}_t)_0 = \boldsymbol{X}_t^{in}, (\boldsymbol{X}_t)_1 = \boldsymbol{X}_t^{out}$, the outflow matrix $\boldsymbol{X}_t^{out} \in R^{I \times J}$ is illustrated in Fig 1(b). If we periodically record the crowds, we can get a sequence of $\boldsymbol{X}_t$. Thus, the citywide traffic flow prediction problem can be defined as follows:

Problem. Given a sequence of historical observations $\{\boldsymbol{X}_t | t = 0,1,\ldots,n-1\}$, predict$\{\boldsymbol{X}_t | t = n, n+1, \cdots, n+k-1\}$.

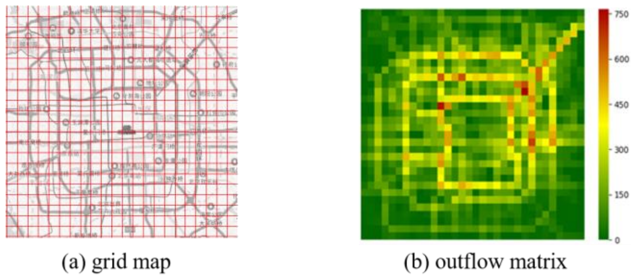

Fig. 1. regions of Beijing (a) grid map (b) outflows in every region of Beijing

B. Convolutional LSTM

Convolutional LSTM (ConvLSTM)[15] is an extension of LSTM, in which convolution is added. It addresses the issue that LSTM cannot pay attention to the spatial correlation between sequences. The essence of ConvLSTM is similar to LSTM, whose key is the cell memory that can alternatively remember the past information states. Whether cells are visited, recorded, or erased is carefully regulated by three self-parameterized gates. Each time a new input appears, the input gate i_t determines which new information is going to be stored in the cell. In addition, the information to be forgotten about the state of the past cell is decided by the forgetting gate f_t. And whether the current cell state C_t can be transmitted to the ultimate state h_t is farther manipulated by the output gate o_t. The current state of a certain ConvLSTM cell in the grid is determined by the inputs as well as past states of the surrounding grids. The key equations are shown in (3):

$$\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma\left(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f\right) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \qquad (3) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ tanh(\mathcal{C}_t)
\end{aligned}$$

where $'*'$ and $'\circ'$ stand for the convolution operation and Hadamard product respectively.

III. Fuzzy Convolutional LSTM network

Traffic flow prediction is influenced by multiple factors, including spatial dependencies, temporal dependencies and external factors, etc. For spatial correlation, a certain region's inflow is influenced by the outflow of its nearby regions. Similarly, the outflow of a region also affects the inflow of its surrounding regions. However, as traffic facilities become more and more convenient, crowds can also travel from one region to another distant region in a short time which indicates that the flows of a region can be affected by distant regions. Therefore, the flows of a region are not only affected by its nearby regions, but also by the farther regions.

Moreover, traffic flow also contains temporal dependencies. By dividing the city into $I \times J$ regions and recording the inflow with the outflow of each region, we can transform the trajectory data into a series of image-like observations $\{X_1, X_2, \cdots X_{t-1}\}$. However, if we put all these observations into the model, it will make the whole training process non-trivial. Fortunately, based on knowledge in the spatial and temporal domain, we know that only a few previous keyframes will affect the next keyframe. Therefore, we leverage temporal dependencies at three different scales, namely *close* trend, *daily* trend, and *weekly* trend to select the keyframes for modeling. The *close* trend means that the traffic flow in the nearby time intervals shares similar characteristics, *daily* trend (*weekly* trend) means that the tendency of traffic flow in the nearby time intervals of adjoining days (adjoining weeks) is similar. The corresponding frames of *close*, *daily,* and *weekly* trend are shown as follows:

$$S_c : [X_{t-l_c}, X_{t-(l_c-1)}, \cdots, X_{t-1}] \qquad (4)$$

$$S_d : [X_{t-l_d \times d-r}, \cdots, X_{t-l_d \times d}, \cdots, X_{t-d-r} \cdots, X_{t-d}] \qquad (5)$$

$$S_w : [X_{t-l_w \times w-r}, \cdots X_{t-l_w \times w}, \cdots, X_{t-w-r}, \cdots, X_{t-w}] \qquad (6)$$

where S_c, S_d, S_w denote *close*, *daily* and *weekly* trend sequence respectively. l_c, l_d and l_w stand for the length of *close*, *daily,* and *weekly* trend. d and w refer to day span and week span respectively. r represents the length of sub-fragment. For example, if we set the time interval to 30 minutes, correspondingly, d is 48 (one day has 48-time intervals) and w is 336 (one week has 336-time intervals), l_c, l_d, l_w and r are set to $3, 1, 1$ and 1 respectively. Then $[X_{t-3}, X_{t-2}, X_{t-1}]$, $[X_{t-49}, X_{t-48}]$and $[X_{t-337}, X_{t-336}]$ will be selected as keyframes. It is necessary to take into account the temporal connections between keyframes in each sequence, rather than treating them as separate data features only.

Attention should also be paid to the effects of external factors (weather, temperature, holidays, etc.). For example, when the weather is bad (rainy, snowy, thunderstorms), people are more inclined to stay at home instead of going out. During the holiday season, most people choose to travel, and the crowds of each scenic spot increase greatly.

In addition to the spatio-temporal correlations and external factors, the high uncertainty contained in traffic data also cannot be ignored. The proposed fuzzy convolutional LSTM (FConvLSTM) takes all of the spatial dependencies, temporal dependencies, external factors, and data uncertainty into account.

In a nutshell, we use a fuzzy neural network to reduce the uncertainty among historical traffic observations and adopt multiple layers of ConvLSTM to simultaneously explore the spatio-temporal characteristics of historical traffic observations. As for external factors, they are manually extracted from the external datasets and input into a two-layer fully connected neural network. Fig. 2 shows the overall framework of the proposed model FConvLSTM. We explore the *close* trend, *daily* trend, and *weekly* trend simultaneously.

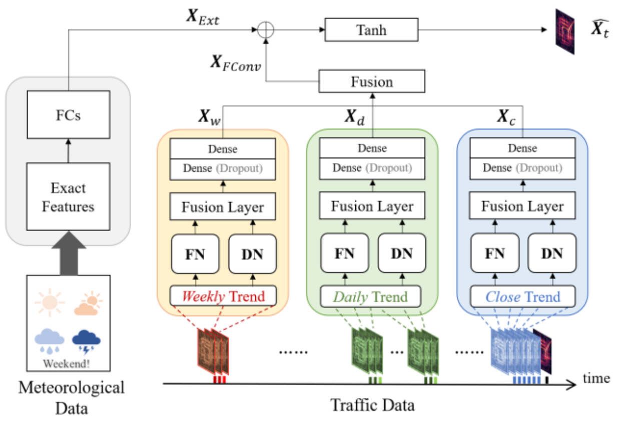

Fig. 2. FConvLSTM architecture
(FN: fuzzy neural network, DN: deep neural network FC: fully-connected)

A. *Structure of close trend*

The structures of the *close* trend, *daily* trend, and *weekly* trend are similar. Taking *close* trend as an example, its structure is composed of four components, namely fuzzy neural network (FN), deep neural network (DN), fusion part, and general learning part. The detailed structure is shown in Fig. 3. In short, the input data go to fuzzy neural network and deep neural network to make a fuzzy logic representation (black part) and neural representation (blue part) respectively. Then, the representations of these two views are combined at the fusion layer (green part). In addition, the fused information is further transformed sequentially inputting to the final task-driven layer which performs traffic flow prediction. In the following, we use l to denote the currently discussed layer, and note that we do not distinguish distinct layers by assigning distinct layer symbols l, $a_i^{(l)}$ refers to the input of i^{th} node in layer l, and $o_i^{(l)}$ represents the corresponding output.

Part 1: fuzzy neural network (FN). The fuzzy neural network is based on fuzzy logic representations using simple *if-then* rules, these rules can be flexibly constructed from the input data by supervised learning. Input data are flattened in the input layer of the fuzzy part which means the inflow and outflow data of all grids in the input sequence and their interaction are investigated simultaneously. The second layer is the fuzzification layer. Each node in the fuzzification layer represents a membership function that calculates the degrees to which an input node belongs to a certain fuzzy set. Since the Gaussian function is the most widely used, it is selected as the membership function.

$$o_i^{(l)} = u_i\left(a_k^{(l)}\right) = e^{-\left(a_k^{(l)}-\mu_i\right)^2/\sigma_i^2} \quad \forall i \tag{7}$$

where$\mu_i$and σ_i denote the center and width of the Gaussian function respectively. In the fuzzy rule layer, each node represents the "if part" of a fuzzy rule. The number of nodes in this layer stands for the rules' number, usually AND operation is performed in the rule layer.

$$o_i^{(l)} = \prod_{j=1}^{n} o_j^{(l-1)} \tag{8}$$

where n denotes the nodes' number on the $(l-1)^{th}$ layer that connected to node i, and $\prod$ represents the minimalization which acts the AND operation according to the fuzzy theory. And the outputs of this layer can be treated as fuzzy degrees. In general, a fuzzy neural network also contains a normalization layer and a defuzzification layer which perform the "then" part of a fuzzy rule. Here, the first three layers of a fuzzy neural network are enough for getting the fuzzy logic representation. Through the fuzzy neural network part, fuzzy degrees of the original input data are transformed and further integrated with the deep neural network part. The FN part provides a new perspective to deal with massive raw data. It extracts the universal information contained in the data through several concise fuzzy rules to reduce the uncertainty of the data.

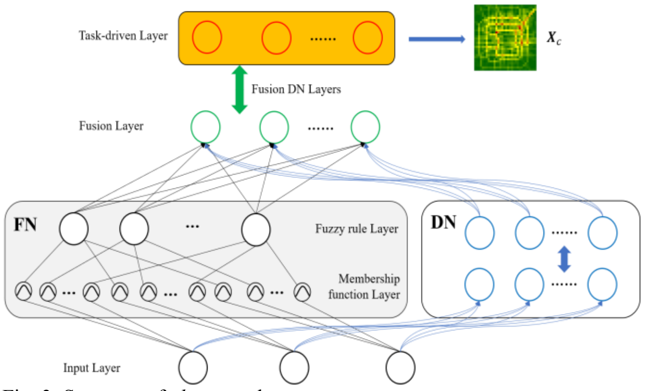

Fig. 3. Structure of *close* trend

Part 2: Deep neural network (DN). Deep neural network explores the concept of neural learning to express input as some high-level representation. ConvLSTM is chosen for the deep network because of its ability to exploit spatial correlation as well as temporal correlation of traffic data jointly. To acquire deeper features, multiple ConvLSTMs are used as illustrated in Fig. 4.

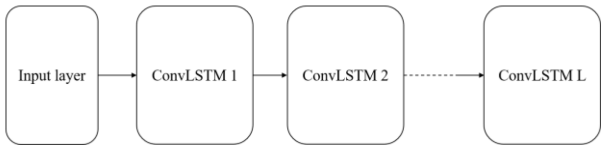

Fig. 4. Structure of DN

Part 3 Fusion layer. Inspired by recent successes of multi-modal learning[16], we adopt the fusion concept to integrate the fuzzy logic representation (the output of FN) and the deep neural representation (the output of DN). In multi-modal learning, it is believed that features extracted from a single aspect are not representative enough for high-content data. Therefore, these methods try to extract multiple features from multiple views consistently which are further synthesized into high-level representations for various learning tasks. In FConvLSTM, deep neural parts are employed to explore the spatio-temporal characteristics of traffic data while fuzzy parts are adopted to reduce data uncertainty, and then they are fused. To better understand the design of FConvLSTM, we can treat the outputs of fuzzy parts as features rather than their original fuzzy basis. Since the dimension of fuzzy logic representation is different from that of neural representation, according to [16], we combine them with one fully connected layer as follows:

$$a_i^{(l)} = (w_d)_i^{(l)}(o_d)^{(l-1)} + \left(w_f\right)_i^{(l)}\left(o_f\right)^{(l-1)} + b_i^{(l)} \qquad (9)$$

$$o_i^{(l)} = tanh\left(a_i^{(l)}\right) \qquad (10)$$

where o_d and o_f of denoting the neural representation part and the output of fuzzy logic representation part respectively. w_d and w_f are corresponding learnable parameters. For the fusion layer, its number of nodes is the sum of the nodes' number on the last layer of the fuzzy part and deep part. Note the outputs of the fusion layer which integrates fuzzy degrees and the deep neural representation can be regarded as the general representation of input data.

As for fusion, although there are many other feature extraction methods available, fuzzy learning is chosen here for the following reasons: firstly, fuzzy learning can deal with the uncertainties among data effectively which is very difficult to be supplanted by other learning; secondly, fuzzy learning can produce soft logic values (fuzzy degrees) in a natural manner which is flexible to be fused with the outputs of deep learning; thirdly, fuzzy learning supports intelligent parameter learning through backpropagation, which can avoid the tedious manual tuning process.

Part 4 General learning. The fusion layer is followed by several fully connected layers which are used to perform the general learning process. As shown in Fig 3, we conduct two fully-connected layers in the actual experiment, and the last fully connected layer's output can be regarded as predicted $\boldsymbol{X}_t$.

B. *Structure of external factors*

As shown in the left part of Fig. 2, we model the impact of external factors by two fully-connected layers: the first layer can be regarded as an embedding layer, and the second layer is used to map low dimension to high dimension, whose number of nodes is $2 \times I \times J$ so that the output of the layer has the same shape with $\boldsymbol{X}_t$.

C. *Fusion*

We first combine the outputs of the *close* trend, *daily* trend, and *weekly* trend. Since the impact of these three parts varies with different regions, as in Eq. (11), a parametric-matrix-based fusion is chosen.

$$\boldsymbol{X}_{FConv} = \boldsymbol{W}_c \circ \boldsymbol{X}_c + \boldsymbol{W}_d \circ \boldsymbol{X}_d + \boldsymbol{W}_w \circ \boldsymbol{X}_w \qquad (11)$$

where $\boldsymbol{X}_c$, $\boldsymbol{X}_d$, $\boldsymbol{X}_w$ denotes the output of *close*, *daily,* and *weekly* trends respectively. And $\boldsymbol{W}_c, \boldsymbol{W}_d, \boldsymbol{W}_w$ are corresponding learnable parameters, denoting the effect degrees of $\boldsymbol{X}_c$, $\boldsymbol{X}_d$, $\boldsymbol{X}_w$ respectively.

As shown in Fig 2, $\boldsymbol{X}_{FConv}$ is directly integrated with $\boldsymbol{X}_{Ext}$, and the *tanh* function is chosen as the activation function because of its faster convergence than the *sigmoid* function. So, the predicted flow at t^{th} time interval can be defined as in Eq. (12):

$$\widehat{\boldsymbol{X}}_t = tanh(\boldsymbol{X}_{FConv} + \boldsymbol{X}_{Ext}) \qquad (12)$$

where the $\widehat{\boldsymbol{X}}_t$ denotes the predicted flow at t^{th} time interval.

In terms of Eq. (12), the predicted value is obtained and we can train our FConvLSTM model by minimizing the mean-squared error between the predicted values and ground truth values. Its cost function is defined as in Eq. (13):

$$L(\theta) = \left\|\boldsymbol{X}_t - \widehat{\boldsymbol{X}}_t\right\|_2^2 + \lambda L_{reg} \qquad (13)$$

where θ denotes all learnable parameters in FConvLSTM, L_{reg} is the L2 regularization term that helps to avoid the overfitting problem, and λ is the coefficient.

D. *Learning algorithm*

Backpropagation through time[17] and Adam[18] algorithm is used to train FConvLSTM. However, for deep neural networks, over-fitting is easy to happen during the training process. To alleviate over-fitting, we apply the dropout technique [19]. The core idea of dropout is that randomly drop out or ignore some nodes as well as their connections from the network during training. Here we choose to randomly drop out $p\%$ nodes in the last second dense layer of *close*, *daily* and *weekly* trend structure as shown in Fig 2. The training process of FConvLSTM with the dropout technique is summarized as Algorithm 1. And through Algorithm 1, we can get the well-trained FConvLSTM model that can be used to predict a single-step or multi-step traffic flow in the future. The prediction process is shown in Algorithm 2, during which the pre-predicted value can be regarded as input data to continue to predict the next step. It should be pointed out that $\boldsymbol{E}_t$ represents future weather data, and the real weather data is used here. In real applications, forecasting weather can be used as $\boldsymbol{E}_t$.

Algorithm 1 The training of FConvLSTM

```
Input: historical observations $\{\boldsymbol{X}_t | t = 0,1,\ldots,n-1\}$
External features $\{\boldsymbol{E}_t | t = 0,1,\ldots,n-1\}$
$d, w$ represents daily, weekly trend spans respectively
$l_c, l_d, l_w$ denotes the length of close, daily, weekly trend correspondingly, $r$ is the sub-fragment length
Output: FConvLSTM model $\mathcal{M}$
$D = \emptyset$
for $t$ in range $(1, n)$:
$\boldsymbol{S}_c = \{\boldsymbol{X}_{t-l_c}, \boldsymbol{X}_{t-(l_c-1)}, \cdots, \boldsymbol{X}_{t-1}\}$;
$\boldsymbol{S}_d = \{\boldsymbol{X}_{t-l_d\times d-r}, \cdots, \boldsymbol{X}_{t-l_d\times d}, \cdots, \boldsymbol{X}_{t-d-r} \cdots, \boldsymbol{X}_{t-d}\}$;
$\boldsymbol{S}_w = \{\boldsymbol{X}_{t-l_w\times w-r}, \cdots \boldsymbol{X}_{t-l_w\times w}, \cdots, \boldsymbol{X}_{t-w-r} \cdots \boldsymbol{X}_{t-w}\}$;
    push $(\{\boldsymbol{S}_c, \boldsymbol{S}_d, \boldsymbol{S}_w, \boldsymbol{E}_t\}, \boldsymbol{X}_t)$ into $D$
end for
construct the FConvLSTM model as shown in Fig 3
initialize parameters
repeat
    randomly select a batch of instances $D_b$ from $D$;
    randomly drop out p% nodes of a certain layer, getting $\text{FConvLSTM}_{\text{remain}}$ and the dropout nodes are labeled as $\text{FConvLSTM}_{\text{dropout}}$;
    feedforward the $\text{FConvLSTM}_{\text{remain}}$, getting the fitting error by Eq. (11);
    back-propagate the error and update parameters in $\text{FConvLSTM}_{\text{remain}}$
until stopping criteria is met
return the well-trained FConvLSTM model $\mathcal{M}$
```

Algorithm 2 single/multi-step prediction of FConvLSTM

```
Input: well-trained FConvLSTM model $\mathcal{M}$
number of looking ahead steps: $k$ ($k$=1 means a single-step prediction)
External features $\{\boldsymbol{E}_t | t = n, n+1, \ldots, n+k-1\}$
Historical observations $\{\boldsymbol{X}_t | t = 0,1,\ldots,n-1\}$
$d, w, l_c, l_d, l_w, r$ same as in Algorithm 1
Output: $\{\boldsymbol{X}_t | t = n, n+1, \ldots, n+k-1\}$
for $t$ in range $(n, n + k)$:
$\boldsymbol{S}_c = \{\boldsymbol{X}_{t-l_c}, \boldsymbol{X}_{t-(l_c-1)}, \cdots, \boldsymbol{X}_{t-1}\}$;
$\boldsymbol{S}_d = \{\boldsymbol{X}_{t-l_d\times d-r}, \cdots, \boldsymbol{X}_{t-l_d\times d}, \cdots, \boldsymbol{X}_{t-d-r} \cdots, \boldsymbol{X}_{t-d}\}$;
$\boldsymbol{S}_w = \{\boldsymbol{X}_{t-l_w\times w-r}, \cdots \boldsymbol{X}_{t-l_w\times w}, \cdots, \boldsymbol{X}_{t-w-r} \cdots \boldsymbol{X}_{t-w}\}$;
$\boldsymbol{X}_t = \mathcal{M}(\boldsymbol{S}_c, \boldsymbol{S}_d, \boldsymbol{S}_w, \boldsymbol{E}_t)$
end for
return $\{\boldsymbol{X}_t | t = n, n+1, \ldots, n+k-1\}$
```

IV. Empirical Study

A. Experiment Settings

Environment. The proposed method is implemented using python language with Keras and TensorFlow. Our experiments mainly run on Windows 10, Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz with 128GB Memory, and NVIDIA GeForce GTX 1080Ti.

Dataset. We evaluated our experiments on a real dataset—TaxiBJ[2], which consists of two components: traffic data and meteorological data. Traffic data in the dataset is from a set of GPS trajectories recording by 34000+ taxies in Beijing during 7/1/2013 – 10/30/2013, 3/1/2014 – 6/30/2014, 3/1/2015 – 6/30/2015, 11/1/2015 – 4/10/2016. The meteorological data is the corresponding weather conditions including weather, temperature, wind speed, etc., which is mainly used to explore the external factors. During the experiment, we have chosen the four weeks' data (3/13/2016--4/10/2016) as testing data and the rest is training data, where the training set and testing set contain 12294 and 1344 instances respectively.

Baselines. We compared our approach with other approaches. The descriptions of these approaches are shown as follows.

- **HA:** HA refers to Historical Average which is quite plain. As its name suggests, it directly treats the average of the past values as predictions.
- **ARIMA[20]:** ARIMA referring to the autoregressive integrated moving average, and is often used as a baseline for traffic flow prediction.
- **SARIMA[21]:** Seasonal ARIMA, compared with ARIMA, seasonal terms are considered.
- **LSTM[6]:** LSTM is a special version of RNN, which overcomes the limitation that RNN cannot learn long-term dependencies. In the experiment, the input sequence length of LSTM is one of $\{3,6,12\}$.
- **STGCN[22]:** STGCN uses GCN and CNN to model the spatial and temporal features and combines them into a spatial-temporal block. We define the neighbor of each grid as the grid that is directly adjacent to it.
- **T-GCN[23]:** T-GCN employs GCN to extract spatial hidden features on each input time slice and then captures the temporal correlation in a GRU layer.
- **ST-ANN:** ST-ANN refers to an artificial neural network that selects the values of eight nearby regions and eight historical intervals as spatial and temporal features, respectively.
- **FDCN[24]:** Fuzzy deep convolutional network (FDCN), combines a convolutional residual network module with a fuzzy module to predict traffic flow.
- **DeepST[25]:** A Deep Neural Network based method for citywide traffic flow prediction.
- **ST-ResNet[2]:** A deep residual network model that considers spatial, temporal dependencies, and external factors comprehensively.
- **ConvLSTM[15]:** ConvLSTM refers to a convolutional LSTM network, which has the ability to learn spatial and temporal dependencies jointly.
- **STAR[3]:** STAR refers to a single fully-convolutional residual network.

B. Implementation Details

Pre-Processing: In the experiment dataset, Beijing was divided into a 32×32 grid-map and crowds in each region were recorded every 30 minutes, indicating the time interval is 30 minutes. So, a complete day has 48 records, days with records less than 48 are considered incomplete days and should be removed. In FConvLSTM, *tanh* function is chosen as the final activation function due to its faster convergence speed than the standard *sigmoid* function. Since the range of *tanh* is [-1, 1], the min-max normalization approach is adopted for scaling the flows of each region into [-1, 1]. And one-hot coding is used for external factors. When evaluating the result, the data is re-scaled back into normal values to compare with the ground truth.

Hyperparameters: The learnable parameters of the deep neural network are initialized with the default value in Keras, and the learnable parameters of FN are initialized using the normal distribution. All convolutions of ConvLSTM except the last layer use 64 filters and the last layer use 2 filters. The kernel size of these convolutions is set to $3 * 3$. In the experiment, we use two ConvLSTMs. There are 6 hyperparameters that are l_c, l_d, l_w, r, $n_{neurons}$(the number of nodes in rule layer), dropout rate and learning rate. In the experiment, we set $l_c \in \{0,1,2,3,4,5,6,7,8\}$, $l_d \in \{0,1,2,3,4\}$, $l_w \in \{0,1,2,3\}$, $r \in \{0,1\}$ and $n_{neurons} \in \{0, \cdots, 100\}$ .The learning rate is set to 0.0002 and the batch size is 32. We randomly selected 10% of the training set data as the verification set used for early stopping. If the RMSE on the verification set remains unchanged within four epochs, we carried out early stopping. Afterwards, the entire training data is used to continue to train our model for fixed epochs.

Evaluating: As shown in Eq. (14-16), we measure the performance of our method using three metrics: mean absolute error (MAE), mean absolute percentage error (MAPE) and root-mean-squared-error (RMSE):

$$MAE = \frac{1}{z}\sum_i |x_i - \hat{x}_i| \tag{14}$$

$$MAPE = \frac{1}{z}\sum_i \frac{|x_i - \hat{x}_i|}{x_i} \tag{15}$$

$$RMSE = \sqrt{\frac{1}{z}\sum_i (x_i - \hat{x}_i)^2} \tag{16}$$

where x_i represents the ground truth, $\hat{x}_i$ is the predicted one, $\bar{x}$ is the average, and z represents the total number of predicted values.

Design of experiments: we conduct our experiments on the TaxiBJ dataset from the following three parts.

Comparison with baseline methods. Firstly, we investigate the overall performance under single-step prediction and multi-step prediction between FConvLSTM and other baselines.

Comparison with variants of FConvLSTM. We verify the effectiveness of the FConvLSTM modeling method by comparing it with several FConvLSTM variants.

Efficiency Analysis. We further discuss the efficiency of different methods.

C. Comparison with Baseline Methods

1) Evaluation of Single-step Ahead Prediction

Firstly, we compare the overall performance under single-step prediction between FConvLSTM and other baselines, and single-step prediction refers to the crowds' prediction of the next time interval based on the historical observations. The time interval (step length) is set to 30 minutes. As shown in Table 1, the proposed FConvLSTM method achieves the lowest RMSE among all methods, which is relatively 2.9% better than ST-ResNet and 0.92% better than STAR. More specifically, we can see that HA, ARIMA, SARIMA, and LSTM do not perform well. The main reason lies in that they only consider the temporal dependencies. The effects of DeepST, ST-ANN, FDCN, ST-ResNet, and STAR are better as they further consider spatial dependencies. However, these models do not consider the spatio-temporal dependencies jointly. Taking STAR as an example, it uses a single fully-convolutional residual network to learn the spatio-temporal correlations of traffic data where the convolution kernels take all frames as channel dimensions and ignore their temporal order. FConvLSTM employs ConvLSTM to consider the temporal order of different frames and captures the influence between frames. Two GCN-based models capture spatial and temporal correlations through different deep neural networks, while they perform worse than FConvLSTM, which demonstrates that the method of graph convolution model acting on Euclidean data needs to be further studied. Moreover, these methods (except FDCN) also have not considered the uncertainties among input data and only use deep neural representations to characterize data features. The good performance of FConvLSTM (lowest MAE, MAPE, RMSE) indicates that considering the data uncertainty and temporal order of frames are beneficial to improve the prediction accuracy.

TABLE I. PERFORMANCE COMPARISON WITH BASELINES

Model	MAE	MAPE	RMSE
HA	-	-	57.69*
ARIMA	-	-	22.78*
SARIMA	-	-	26.88*
LSTM	13.20	30.14%	22.33
STGCN	11.52	26.38%	19.34
T-GCN	9.63	23.22%	16.68
ST-ANN	13.77	49.41%	21.53
FDCN	13.53	135.99%[1]	18.87
DeepST	10.46	25.09%	17.48
ConvLSTM	10.28	28.88%	17.15
ST-ResNet	9.61	23.77%	16.48
STAR	9.51	23.67%	16.14
FConvLSTM	**9.34**	**22.42%**	**15.99**

(The BASELINE results with * are cited from [25],

[1] The unusual MAPE of FDCN is due to its poor performance in predicting small values (<10, which account for 20.49% of the test set).

2) Evaluation of Multi-step Ahead Prediction

We also evaluate the performance under multi-step prediction between FConvLSTM and other baselines. Multi-step prediction refers to forecasting traffic flow in multiple successive time intervals. We conduct the multi-step prediction according to Algorithm 2 with the iterative strategy and set the look-ahead steps from 1 to 6.

As shown in Fig.5, FConvLSTM presents the best results among all. T-GCN, ConvLSTM and FConvLSTM perform quite well in multi-step ahead prediction, because they all explicitly model the temporal dependencies in time frames by using RNNs. This shows that paying attention to the temporal order of different frames is beneficial. Compared with T-GCN and ConvLSTM, FConvLSTM is relatively less affected by iteration error in multi-step ahead prediction due to the addition of fuzzy logic representation, which demonstrates the effectiveness of fuzzy representation.

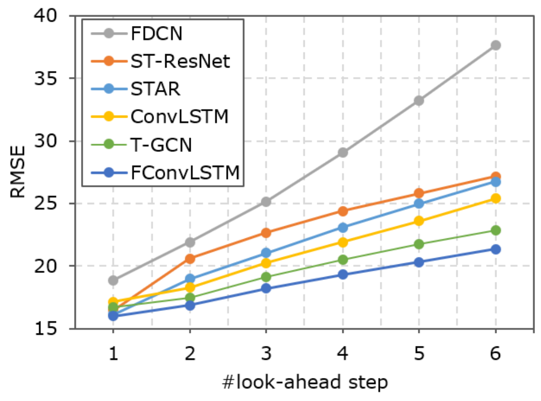

Fig. 5. Multi-step ahead prediction

D. Comparison with variants of FConvLSTM

In FConvLSTM, we propose to combine FL and ConvLSTM to explore data characteristics. To verify the effectiveness of the model, we conduct experiments by using partial components or replacing partial components. The models and variants are described as follows:

ConvLSTM: For this variant, we only use the ConvLSTM part of FConvLSTM and ignore the fuzzy learning part. This variant is mainly for comparison with the proposed method FConvLSTM to verify that combining fuzzy learning is effective, and to provide a baseline for other variants.

ANN+ConvLSTM: In this variant, the fuzzy learning part is replaced by the ANN. In detail, the fuzzification layer and the rule layer of the fuzzy learning part are replaced by a dense layer, where the number of nodes in the dense layer is twice that of the rule layer for a fair comparison. This variant is mainly to evaluate whether FConvLSTM achieves better results because of the higher model complexity.

FConvLSTM-ST311: In this variant, the length of the sub-fragment is set to zero. $[X_{t-3}, X_{t-2}, X_{t-1}]$, $[X_{t-48}]$, $[X_{t-336}]$ frames are selected while the inputs of FConvLSTM are $[X_{t-3}, X_{t-2}, X_{t-1}]$,$[X_{t-49}, X_{t-48}]$and $[X_{t-337}, X_{t-336}]$.

FConvLSTM – External: This variant does not consider the influence of external factors.

Fuzzy + ConvLSTM (FConvLSTM): the proposed model, which combines fuzzy learning and ConvLSTM and takes the influence of external factors into account.

Table 2 shows the experimental results of FConvLSTM and its several variants. For a fair comparison, the inputs of ConvLSTM, ANN + ConvLSTM, and FConvLSTM are the same, and they share similar parameter settings. Note that the parameters of ConvLSTM in this part are different from other parts (eg. single step and multi_step prediction) where the ConvLSTM takes the best parameters. Compared with ConvLSTM, FConvLSTM can achieve lower RMSE, indicating that combining fuzzy learning is effective for improving prediction accuracy. And the training time of FConvLSTM is very close to that of ConvLSTM, indicating that adding the fuzzy neural network module will not increase the training time significantly. Furthermore, compared with ANN + ConvLSTM (sharing similar model size with FConvLSTM), FConvLSTM still achieves better performance, which shows that the good performance of FConvLSTM is achieved without increasing model complexity. It can be seen that FConvLSTM usually gives more accurate predictions, which indicates the great representation ability of FConvLSTM (the two models share the same inputs and model size). Specifically, ANN + FConvLSTM has a poor performance in high-flow areas. It generates higher values than ground truth. The reason why FConvLSTM-ST311 is relatively poor is that the keyframes selected are not sufficiently representative. The effect of FConvLSTM is better than FConvLSTM–External, indicating that the external features from auxiliary information are helpful for prediction.

TABLE 2 COMPARISON WITH FCONVLSTM VARIANTS

Method	RMSE	#parameters(k)	Time (min)
ConvLSTM	17.15	500	157
ANN+ConvLSTM	21.03	38,755	140
FConvLSTM-ST311	16.22	38,656	145
FConvLSTM–External	16.37	38,715	152
FConvLSTM	15.99	38,738	160

E. Efficiency Analysis

Table 3 shows the performance index of the proposed model FConvLSTM compared to several baselines (i.e. T-GCN, ConvLSTM, ST-ResNet, FDCN, and STAR). It can be seen that the RMSE of the proposed method is better than that of other methods. Although its training time (160 minutes) is not the shortest among those methods, it is still practically acceptable. Since the entire training process is carried out offline, the training time overhead can be decreased by today's ever-increasing high-speed computing resources. When the trained model is applied in prediction, FConvLSTM can predict citywide traffic flow in seconds according to our experiments.

TABLE 3 COMPARISON OF RMSE AND TRAINING TIME

Method	RMSE	Training time(min)
FDCN	18.97	148
ConvLSTM	17.15	253
T-GCN	16.68	103
ST-ResNet	16.48	180
STAR	16.14	**46**
FConvLSTM	**15.99**	160

V. Conclusion

In this study, we learn to deal with the citywide traffic flow prediction problem from the massive trajectory data. The key challenge of this problem is that traffic flow is affected by multiple factors such as spatial dependencies, temporal dependencies, weather and holiday, etc. Moreover, when data quantity gets large, data uncertainty cannot be neglected. To address the above issues, we proposed a hybrid method called FConvLSTM which combines fuzzy learning and convolutional LSTM(ConvLSTM) altogether, where a fuzzy neural network based on fuzzy logic is used to reduce the uncertainty of data, and ConvLSTM is to jointly capture spatial dependencies and temporal dependencies of traffic data. Compared with CNN methods, ConvLSTM pays attention to the temporal order of input frames and can learn the influence between frames. The proposed model outperforms several state-of-the-art models in a real dataset TaxiBJ. Besides, in transportation management, the proposed model can also deal with regression-type problems naturally through an end-to-end framework. Despite the good RMSE performance, the training time of the proposed methods still needs to be improved. The proposed model is for the grid-based citywide traffic flow prediction problem. It is particularly efficient to be applied to traffic prediction on regions of a city with quite evenly distributed traffic flow of grids of the same size. How to divide the regions of the city into grids with variant sizes in terms of real traffic is one of our concerns in the future research.

References

[1] Y. Zheng, L. Capra, O. Wolfson, H. Yang, "Urban computing: concepts, methodologies, and applications", *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 38, pp.1-55, Oct. 2014.

[2] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction", in *Proc. 31st AAAI Conf. Artif. Intell.,* 2017, pp. 1655–1661.

[3] H. Wang and H. Su, "STAR: A Concise Deep Learning Framework for Citywide Human Mobility Prediction", in *Proc. IEEE 20th Int Conf. Mobile Data Management (MDM'19),* Aug.2019, pp. 304-309.

[4] Z. Xu, Y. Wang, M. Long, J. Wang, and M. O. KLiss, "PredCNN: Predictive learning with cascade convolutions", in *Proc. IJCAI, Jul.* 2018, pp. 2940-2947.

[5] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting", *in Proc. SIAM Int. Conf. Data Mining. Soc. Ind. Appl. Math*., Jun. 2017, pp. 777-785.

[6] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data", *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[7] Y. Wu, H.Tan, L. Qin, B. Ran, and Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding", *Transp. Res. Part C: Emerg. Tech.,* vol. 90, pp. 166-180, 2018.

[8] C.-T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system", *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.

[9] J. Tang, F. Liu, Y. Zou, W. Zhang, and Y. Wang, "An improved fuzzy neural network for traffic speed prediction considering periodic characteristic", *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2340–2350, Sep. 2017.

[10] H. K. Kwan and Y. Cai, "A fuzzy neural network and its application to pattern recognition", *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 3, pp. 185–193, Aug. 1994.

[11] F. Wong, P. Wang, T. Goh, and B. Quek, "Fuzzy neural systems for stock selection", *Financ. Anal. J.*, vol. 48, no. 1, pp. 47–52, 1992.

[12] M. K. Mehlawat and P. Gupta, "Fuzzy Chance-Constrained Multiobjective Portfolio Selection Model", *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 3, pp. 653-671, June 2014.

[13] F.-J. Lin, C.-H. Lin, and P.-H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive", *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 5, pp. 751–759, Oct. 2001.

[14] Y. Deng , Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A Hierarchical Fused Fuzzy Deep Neural Network for Data Classification" , *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 4, pp. 1006-1012, 2017.

[15] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting", in *Proc. Adv. Neural Inf. Process. Syst.*, Mar. 2015, pp. 802–810.

[16] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning", in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 689–696.

[17] O. De Jeses and M. T. Hagan, "Backpropagation through time for a general class of recurrent network", *Int. Joint Conf. Neural Networks. Proc.* (Cat. No.01CH37222), 2001, vol.4, pp. 2638-2643.

[18] D. P. Kingma, and J. Ba, "Adam: A Method for Stochastic Optimization", in *3rd International Conference on Learning Representations*, 2015

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014

[20] D. Billings and J. Yang, "Application of the ARIMA Models to Urban Roadway Travel Time Prediction - A Case Study", in *Proc. IEEE Int. Conf. Syst., Man Cyber.*, May, 2006, pp. 2529-2534.

[21] B.L. Smith, B.M. Williams, and R.K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting", *Transp. Res., Part C, Emerg. Technol.* vol 10, no.04, pp.303–321, 2002

[22] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, *IJCAI-18*, July 2018, pp. 3634–3640.

[23] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, 2020.

[24] W. Chen, J. An, R. Li, L. Fu, G. Xie, Md. Z. A. Bhuiyan, and K. Li, "A novel fuzzy deep-learning approach to traffic flow prediction with uncertain spatial-temporal data features", *Future Gener. Comput. Syst.*, vol. 89, pp. 78-88, Dec. 2018.

[25] J. Zhang, Y.Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks", *Artificial Intelligence*, vol. 259, pp. 147-166, 2018.