SDCN: Sparsity and Diversity Driven Correlation Networks for Traffic Demand Forecasting

Wenjie Li[1,†], Xue Yang[2,3,‡], Xiaohu Tang[1], Shutao Xia[2,3]
[1]School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China
[2]Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[3]PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China
still2009@163.com, xhtang@swjtu.edu.cn, {yang.xue, xiast}@sz.tsinghua.edu.cn

***Abstract*—Traffic demand forecasting is essential to intelligent transportation systems and is widely used to support urban planning, traffic management and vehicle dispatching. One challenge of this problem is to model the complex spatial-temporal correlation. Although both factors have been studied, many of the existing works have strong limitations. They rely too heavily on the locality assumption (i.e., the local area is more relevant than the remote area) and only use a distance-based correlation measurement. However, the spatial correlation is also global (i.e., areas far away may also be relevant) and sparse. And it's insufficient to measure the spatial correlation using only the distance measurement. In this paper, a sparsity and diversity driven correlation network is proposed to tackle these issues. Firstly, Multiple sparse correlation graphs are carefully generated to encode sparsity and diversity. Then a newly designed *hybrid graph filtering module* (HGFM) leverages them to learn a more expressive node representation. Finally, the HGFM-based *recurrent filtering module* (RFM) is introduced to handle the spatial-temporal correlation. Extensive experiments conducted on real-world datasets demonstrate the competitiveness of our model while showing the significance of sparsity and diversity.**

***Index Terms*—Traffic Demand Forecasting, Graph Convolutional Network, Recurrent Neural Network, Graph Filtering**

I. INTRODUCTION

Traffic demand prediction plays an important role in modern intelligent transportation systems. For example, accurate prediction result can help on-line ride-hailing systems like DiDi and Uber to pre-allocate vehicles to alleviate the imbalance between supply and demand. Besides, the prediction result can give guide information for transportation departments to manage and control the traffic more efficiently and meanwhile avoid congestion. Traditional time series prediction methods like autoregressive integrated moving average (ARIMA) and Kalman filtering have been widely applied to traffic prediction problems [1]–[5]. However, they did not pay attention to the spatial dependency between locations. Deep learning is also widely used in traffic forecasting due to its powerful non-linear modeling capability. From a spatial perspective, [6]–[8] transformed city-wide traffic to a heatmap image and employed the convolutional neural network (CNN) to model the spatial dependency. As to the temporal perspective, [9], [10] leveraged recurrent neural networks to capture the temporal dependency. Besides, [11] further proposed a method to jointly model both spatial and temporal dependencies by integrating CNN and long short-term memory [12].

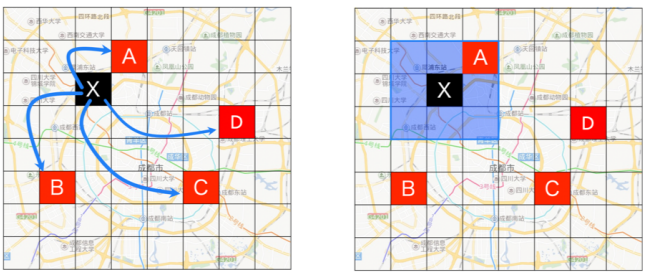

Fig. 1. modeling spatial dependency more efficiently by directly aggregating demands from highly correlated regions (left), rather than aggregating demands from a local adjacent area which contains many low correlated regions (right)

However, These methods only consider single kind of spatial correlation (i.e., spatial closeness) which assumes adjacent regions are more correlated than remote ones. In fact, the traffic demand of one region is also influenced by far apart regions. As an example shown in Fig 1, assuming traffic flow of residence area "X" (marked by black) is caused by office areas "A", "B", "C" and "D" (marked by red) because of daily commuting to work. It's obvious that one single convolution filter is not enough to model the spatial correlation, instead, we need to stack multiple convolution layers to make all the related regions be included in the receptive field. Besides, those adjacent but not correlated regions included in the receptive filed would certainly make it harder to learn the ground truth relationship.

To overcome these disadvantages, we propose the sparsity and diversity driven correlation network that models the spatial correlation more carefully and meanwhile captures the temporal correlation. Specifically, The network models spatial dependency by applying a hybrid graph filtering module on

*This research is supported in part by the Major Frontier Project of Sichuan Province (2015JY0282), the Guangdong Basic and Applied Basic Research Foundation (2019A1515110644), the National Natural Science Foundation of China (61771273), the R&D Program of Shenzhen (JCYJ20180508152204044), and the project "PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019)"

†This work is partially done when the first author is an intern of tsinghua shenzhen international graduate school.

‡Corresponding author.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

diverse sparse graphs. It uses HGFM-enhanced recurrent cell to capture temporal dependency and employs the Encoder-Forecaster architecture to learn the sequence mapping function. Our contribution can be summarized as follow:

- We propose a neighborhood selection method named symmetric top-k to generate a sparse graph based on the original complete correlation matrix. This method can delete lots of weak connections while keeping the node degree non-uniform, which is essential to distinguish hot and cold regions.
- We propose a hybrid graph filtering module (HGFM) to filter a graph signal via multiple filters generated from different graphs and aggregate the filtered signals appropriately. It can capture diverse spatial correlation efficiently.
- Based on HGFM, we propose a recurrent filtering module (RFM) that can model diverse spatial correlation and temporal correlation simultaneously. Then we combine RFM with Encoder-Forecaster architecture [13] to make final predictions.
- We evaluate our model on three real-world datasets. The proposed model shows its strong competitiveness by achieving the lower prediction error compared with two advanced deep learning methods. Besides, extensive experiments on several variants of SDCN reveals the effect of sparsity and diversity in terms of prediction accuracy.

II. RELATED WORK

Although there are various traffic demand forecasting problems (e.g., taxi demand forecast, traffic flow forecast, and arrival time forecast), most of them are similar: predicting a traffic-related value of a location at a certain moment, which is a typical spatial-temporal prediction problem.

Gated recurrent units such as LSTM [12] and GRU [14] are very efficient on sequence modeling and have been successfully applied to short term traffic flow prediction [15]. But they cannot capture spatial dependencies. CNN [16] achieves great success in computer vision and is good at extracting spatial features. But it is helpless in modeling temporal correlation. However, our RFM can model the spatial and temporal correlation simultaneously.

Many studies try to leverage the advantages of both CNN and RNN. ConvLSTM [13] achieves the goal of spatial-temporal modeling by extending the LSTM to have a convolutional structure in state transitions, and it has been successfully applied to transportation problems [17]. ST-ResNet [8] uses deep residual networks [18] to capture spatial dependency and utilizes prior knowledge to describe temporal correlations. DMVST [11] builds a multi-view network with a spatial view, a temporal view, and a semantic view. it explicitly combines CNN and LSTM to predict taxi demand. However, these studies rely on CNN to learn spatial dependency and may neglect far apart regions with similar demand patterns. Besides, they do not explicitly model irregular spatial correlations between pairwise regions. Our model can encode diverse spatial correlations into graphs and efficiently integrate them into the network via sparsification and hybrid filtering.

Emerging graph convolution neural networks like ChebNet [19], GCN [20], and GraphSAGE [21] have made big progress on graph representation learning but only learn from a single graph. MGCNN [22] succeed in using multiple correlation graphs and GCN to predict bike flows but did not explicitly exploit the sparsity of spatial correlation. Benefiting from the tunability of filtering character and the separation of filtering and non-linear mapping, HGFM is more flexible and powerful than these works. Moreover, the introduction of sparsity brings great benefits in terms of computational effectiveness and prediction accuracy.

In summary, our work emphasizes more on the exploitation of sparsity and diversity and finally produces a more elaborate and flexible spatial-temporal prediction model.

III. APPROACH

In this chapter, we first formalize the problem of demand prediction and then propose the *sparsity and diversity driven correlation network*. As shown in Fig. 2, the framework is hierarchically organized from micro to macro and gradually obtains the ability of spatial correlation modeling from HGFM of the bottom level, the ability of spatial-temporal correlation modeling from RFM of the middle level, and the ability of demand prediction from the forecaster of the top level.

A. Problem formalization

We follow previous researches [8], [11] to divide the city into multiple non-overlapped rectangle regions and use $v \in \mathcal{V}$ to denote one of them. Although more sophisticated partition schemas like hexagonal partitioning or partitioning by road network [23] can also be employed, it is not the focus of our work, and our method is still suitable. The time axis is split into fixed time periods (e.g., 30 minutes) and indexed by $t \in \mathbf{N}$. Since traffic services are varied, in this paper, we only focus on traffic forecasting problems that just care about the origin and destination region (e.g., bike rending, taxi calling). Similar to prior work [8], we use *inflow* and *outflow* to describe the traffic demand. The "inflow" here represents the demands for entering a region, and the "outflow" represents the demands for leaving that region. For example, these two kinds of traffic flow can be calculated by counting all the taxi requests going to or departing from a region respectively. We describe the traffic demand of region v during specific period t as a tuple of inflow and outflow $d_{v,t} = (d^{in}_{v,t}, d^{out}_{v,t})$. Then the demand map of the whole city during period t can be denoted as a tuple of inflow set and outflow set:

$$X_t = (X^{in}_t, X^{out}_t) = (\{d^{in}_{v,t} | v \in \mathcal{V}\}, \{d^{out}_{v,t} | v \in \mathcal{V}\}) \in \mathbf{R}^{2\times|\mathcal{V}|}.$$

Thus, we define traffic demand forecasting as a problem of learning a function $f : \mathbf{R}^{m\times 2\times|\mathcal{V}|} \to \mathbf{R}^{2\times|\mathcal{V}|}$ which maps the historical m demand maps to the demand map of next period:

$$f(X_{t-m+1}, X_{t-m+2}, \ldots, X_t) = \widehat{X}_{t+1} = \widehat{Y}$$

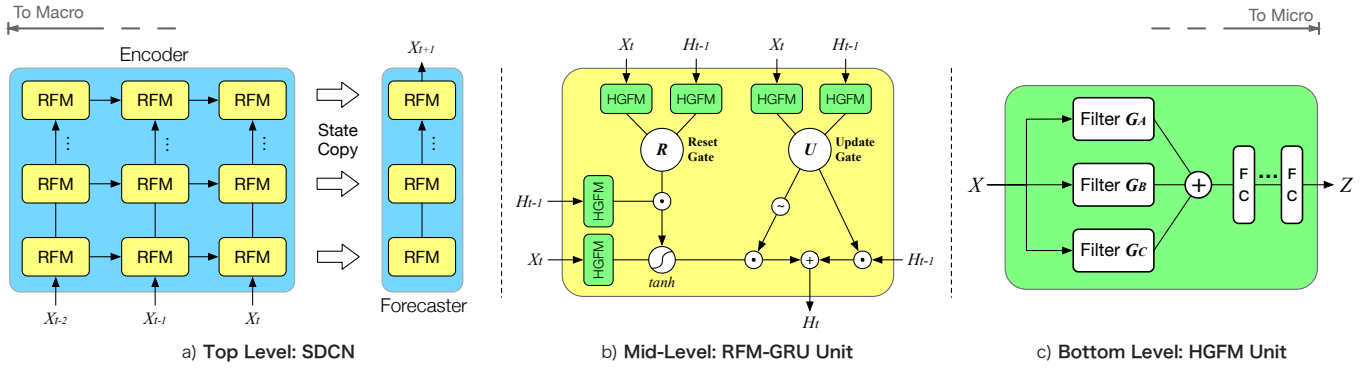

Fig. 2. The architecture of SDCN. (a). The framework contains an encoder and a forecaster each consists of multiple stacked recurrent filtering modules (RFM). (b). An RFM could employ a variety of gate mechanisms like GRU [14] (denoted by RFM-GRU) or LSTM [12] (denoted by RFM-LSTM). RFM can capture the spatial and temporal correlation simultaneously. But in detail, its capability of spatial correlation modeling comes from the underlying hybrid graph filtering modules (HGFM). (c). An HGFM applies multiple graph filters generated with diverse pre-designed sparse spatial correlation graphs (e.g., G_B is a filter generated with demand interaction graph) to the input to adaptively learn spatial-sensitive feature representations.

where $\widehat{Y}$ is the prediction result of ground truth Y. The proposed model SDCN is an efficient implementation of prediction function f and capable to capture the complex spatial-temporal mapping relationship between demand maps.

B. Spatial correlation modeling

In this section, we show how to encode diverse correlations among regions into sparse graphs and how to leverage them in a neural network to learn spatial-sensitive feature representations. Our approach is three-fold: 1) designing correlation graphs based on prior knowledge. 2) sparsifying original complete correlation graphs. 3) utilizing correlation graphs via graph filtering to learn new representations of demand maps. The first two steps belong to the preprocessing stage and the final one belongs to the training stage.

1) Diverse correlation measurements: Inspired by [22], we measure the spatial correlation from three perspectives: 1) spatial closeness 2) demand interaction 3) demand series correlation. These correlation measurements are not constrained in these three types, it is easy to incorporate new types of measurements.

Spatial closeness: As stated in the first law of geography: "near things are more related than distant things" [24]. Demand patterns are more correlated to a certain extent for spatially close regions. Thus we define the spatial closeness as the inverse of distance:

$$A_{ij} = dist(v_i, v_j)^{-1},$$

where A denotes the correlation matrix of spatial closeness, v_i and v_j represents the i-th and the j-th region of the city respectively. $dist$ is a function measures the euclidean distance between two regions' geometric centers.

Demand interaction: Demand interaction refers to the volume of visits between any two regions. Since the outflow of one region would certainly comprise as a part of another regions' inflow, the more frequent the interaction is, the larger the correlation is. Thus it is defined as the number of orders between two regions including inflow and outflow:

$$B_{ij} = |\mathcal{O}_{ij}| + |\mathcal{O}_{ji}|,$$

where B denotes the correlation matrix of demand interaction, $\mathcal{O}_{ij}$ denotes the set of orders whose origin and destination are v_i and v_j, respectively.

Demand series correlation: There are many areas in a modern city that play the same or similar urban functions (e.g., different residential areas and office areas) to meet the needs of people of different places. Although they may be far apart and have little interaction, they do have similar demand patterns. In order to mine these regions with similar demand patterns, we employ the Pearson correlation coefficient [25] to quantify the sequence correlation:

$$r(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

where $r(\mathbf{x}, \mathbf{y})$ calculates the Pearson correlation coefficient between sequence $\mathbf{x} \in \mathbf{R}^n$ and $\mathbf{y} \in \mathbf{R}^n$. $\bar{x}$ and $\bar{y}$ is the mean of $\mathbf{x}$ and $\mathbf{y}$ respectively. Then the demand sequence correlation between two regions is defined as the Pearson coefficient of the averaged traffic demand:

$$C_{ij} = r\left(\frac{\mathbf{d}_{in}^{v_i} + \mathbf{d}_{out}^{v_i}}{2}, \frac{\mathbf{d}_{in}^{v_j} + \mathbf{d}_{out}^{v_j}}{2}\right),$$

where C denotes the correlation matrix of demand correlation, $\mathbf{d}_{in}^{v_i}$ and $\mathbf{d}_{out}^{v_i}$ is the inflow and outflow sequence of region v_i respectively. For simplicity, here we just use the average of inflow and outflow sequence to calculate the correlation coefficient. However, the choices of demand series are flexible and abundant in real applications (e.g., daily 24h demand series, weekly demand series).

2) The sparsification of correlation graphs: The spatial correlation of traffic demand is usually sparse in reality, i.e., the traffic demand of a region is only related to a limited number of regions. This opinion is very natural from the perspective

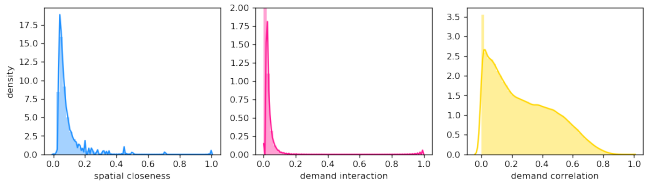

Fig. 3. Kernel density estimation of three correlation graphs' adjacency matrix. We use the dataset "DiDiCD 32×32 T30" introduced in IV-A for analysis. The matrix values have been normalized beforehand.

of urban planning and supported by real-world observations. As shown in Fig. 3, the density of edge weight concentrates more on small values. It indicates that the correlation between most of the regions is very low and the highly related regions only make up a minority.

Thus, the original correlation graph contains a lot of useless (at least less valuable) information and is too dense to use ($O(n^2)$ complexity). To tackle this issue, we propose a node selection method named *symmetric top-k* (STOPK) to generate a sparse correlation graph by choosing a certain number of highly correlated regions as neighbor nodes. The operation of STOPK can be described as the following equations:

$$\hat{A}_{i,:} = \begin{cases} A_{ij}, j \in \mathcal{K}_i \\ 0, j \notin \mathcal{K}_i \end{cases}$$
$$A_s = \hat{A} + \hat{A}^\top$$
$$\tilde{A}_s = D_s^{-1/2} A_s D_s^{-1/2}$$

where $\mathcal{K}_i$ denotes the index set of the largest k elements in the i-th row of A. First, we keep the largest k elements in each row of the adjacency matrix and let the others be zero. Then we add the matrix with its transpose to get a symmetric matrix A_s. Finally, we normalize A_s using its degree matrix D_s. The necessity of the symmetry is reflected in two aspects: (1) make the degrees of nodes different to distinguish hot/cold regions, (2) make the graph un-directed to be compatible with HGFM. We apply this sparsification trick on all the correlation graphs introduced in Section III-B1.

3) Correlation-aware feature extraction: In order to leverage the spatial correlation prior encoded by sparse correlation graphs, we propose the hybrid graph filtering module (HGFM) to learn correlation-sensitive region features which will be used by our recurrent filtering module (RFM). As shown in Fig. 2, HGFM is comprised of multiple parameterized graph filters and alternative multiple (at least one) fully connected layers. the filters are used to apply the influence of spatial correlation to a demand map and the FC layers are employed to enhance the non-linear learning capability.

Graph filtering is based on spectral graph theory [26], it extends the concept of spectral signal filtering to signals defined on graphs, and the filtering process fully utilized the underlying graph's structural information. The spectral of an undirected graph represented by adjacency matrix A is defined by the eigendecomposition of its symmetric graph laplacian:

$$L_s = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} = \Phi \Lambda \Phi,$$

where $D = diag([\sum_{j=1}^{n} A_{1,j}, \ldots, \sum_{j=1}^{n} A_{n,j}])$ is the degree matrix of A, Φ is the orthogonal matrix consists of eigenvectors, and Λ is the diagonal Eigen matrix composed of eigenvalues. Then the eigenvalues and eigenvectors are defined as the frequency and corresponding Fourier basis of the graph. Following [27], we further define a linear spectral graph convolutional filter G by applying a filtering function to the symmetric graph laplacian:

$$G = p(L_s) = \Phi p(\Lambda) \Phi^\top,$$

where $p(\Lambda) = diag(p(\lambda_a), p(\lambda_2), \ldots, p(\lambda_n))$ is called the frequency response function, it totally decides the filtering character (e.g., low-pass, high-pass, etc.) by amplifying or restraining some of the spectral components. For example, a power function is a high-pass filter that restrains the low-frequency components and amplifies the high-frequency ones. its filtering degree depends on the exponent k and the filter degrades to a pass-through filter if k equals 1. Tab I shows more detail about three alternative filter types. Benefiting from the build-up of graph spectral space, the graph filtering (i.e., graph convolution) operation is simplified as the matrix multiplication of spectral filter and input signal: $\bar{X} = GX$, where filter $G \in R^{n \times n}$ is based on a graph with n nodes and signal $X \in R^{n \times m}$ has m features on each node.

Based on the above definitions, we design our hybrid graph filtering as the sum of parameterized graph filtering based on multiple sparse correlation graphs:

$$\widehat{X} = \sum_{i=1}^{|\mathcal{A}|} G_i \odot \Theta_i X,$$

where $\mathcal{A}$ denotes the set of correlation matrices, $\odot$ is the element-wise product and G_i is the graph filter corresponds to the i-th correlation matrix. Different from [20] and [27], we parameterized the filters with a new weight matrix Θ_i to make it learnable and more task-oriented. The character of the filter would be more adaptive to the task instead of being constrained to pre-designed frequency response functions and correlation measurements. This gives the model a chance to update or revise the prior knowledge by back-propagation which largely improves flexibility. It seems that the complexity of Θ seems to be $O(n^2)$, however, G_i and Θ_i are both sparse due to the sparsity of A, so we can implement these computations efficiently using sparse tensor multiplication.

After graph filtering, we apply non-linear mappings to the filtered signal to get a new graph representation. It is meant to apply a multi-layer perceptron to the filtered graph signal $\widehat{X}$, thus we get the final output Z of the HGFM:

$$Z = MLP(\widehat{X}, \mathcal{W})$$

where MLP is an L-layer feedforward network with any activation functions, and $\mathcal{W} = \{W_l | l \in [1, L]\}$ is the set of the weight matrices of all layers in which l denotes the index of a layer.

TABLE I
ALTERNATIVE GRAPH FILTERS.

Filter name	Character	Filter formula
rnm [27]	low-pass	$p(\lambda) = (1-\lambda)^k, G = \tilde{W}^k$
ar [27]	low-pass	$p(\lambda) = (1+k\lambda)^{-1}, G = (I+kL)^{-1}$
pow	high-pass	$p(\lambda) = \lambda^k, G = L^k$

Notes: *rnm*, *ar* and *pow* indicate renormalization, auto regressive filter and the abbreviation of power filter, respectively.

TABLE II
SHARING MODES OF GRAPH FILTER WEIGHT Θ

Share mode	Description
all	all the RFMs share the same Θ
none	every RFM has it's own Θ
e-f	encoder and forecaster each have one Θ
layer-wise	RFMs at the same layer share one Θ

C. Spatial-Temporal correlation modeling

Recurrent neural networks like LSTM [12] and GRU [14] achieve a big success in sequence modeling problems, but they can not process multi-dimensional inputs (e.g., graph signals) and leverage their structural information. To tackle these issues, we propose the recurrent filtering module (RFM) by extending recurrent units using HGFM to make them spatial-sensitive in the state transition process. Picking GRU as an example, we replace the matrix multiplication of original GRU with HGFM in both the input-to-state and state-to-state transition process while extending the input and hidden state to be 2-dimensional. Thus we get the transition functions of RFM-GRU:

$$
\begin{aligned}
R_t &= \sigma\left(g(X_t, \Theta, \mathcal{W}_r) + g(H_{t-1}, \Theta, \mathcal{W}'_r) + b_r\right) \\
U_t &= \sigma\left(g(X_t, \Theta, \mathcal{W}_u) + g(H_{t-1}, \Theta, \mathcal{W}'_u) + b_u\right) \\
\tilde{H}_t &= \tanh\left(g(X_t, \Theta, \mathcal{W}_n) + R_t \odot g(H_{t-1}, \Theta, \mathcal{W}'_n)\right) \\
H_t &= (1 - U_t) \odot \tilde{H}_t + U_t \odot H_{t-1}
\end{aligned}
$$

where $g(\cdot)$ is a function that represents HGFM. Note that the graph filter weight parameter Θ is shared among all gates by default while $\mathcal{W}$ of each gate differs. Since the parameter amount of Θ accounts for a large proportion in RFM, we provide four alternative weight sharing schemas in Tab. II to make a balance between model capacity and efficiency. Fig. 2(b) shows the inner structure of an RFM-GRU unit.

So far, we have obtained RFM with the ability to capture spatial-temporal correlation. To get the final prediction result, we employ the encoder-forecaster framework [13] and stack multiple RFMs as an encoder and a forecaster, as shown in Fig. 2(a). The encoder extracts context information from the historical demand map sequence and the forecaster uses it to make predictions.

D. Loss and metrics

To avoid the training being dominated by regions with high demand magnitude, we follow [11] to use a combined loss of absolute error and relative error. For each sample, the prediction loss is defined as:

$$
\mathcal{L}(Y, \widehat{Y}) = \frac{1}{2|\mathcal{V}|} \sum_{i}^{2} \sum_{j}^{|\mathcal{V}|} \left[(y_{ij} - \hat{y}_{ij})^2 + \gamma \left(\frac{y_{ij} - \hat{y}_{ij}}{y_{ij}} \right)^2 \right]
$$

where $\hat{y}_{ij}$ is the i-th row and j-th column of predicted demand map $\widehat{Y} \in \mathbf{R}^{2 \times |\mathcal{V}|}$ and γ is a hyper-parameter controlling the importance of relative error and absolute error. Following prior studies [8] [11] , we use the root mean square error (RMSE) and mean average percentage error (MAPE) to evaluate the prediction performance:

$$
\begin{aligned}
RMSE &= \sqrt{\frac{1}{2N|\mathcal{V}|} \sum_{k}^{N} \sum_{i}^{2} \sum_{j}^{|\mathcal{V}|} \left(y_{ij}^k - \hat{y}_{ij}^k\right)^2}, \\
MAPE &= \frac{1}{2N|\mathcal{V}|} \sum_{k}^{N} \sum_{i}^{2} \sum_{j}^{|\mathcal{V}|} \frac{|y_{ij}^k - \hat{y}_{ij}^k|}{y_{ij}^k}.
\end{aligned}
$$

where y_{ij}^k is an element of the k-th demand map in a batch.

IV. EXPERIMENT AND DISCUSSION

A. Datasets

We evaluate our model on a real-world traffic demand data archive[1]. It contains 6.1 million taxi orders from 2016/11/01 to 2016/11/30 in the city Chengdu of China. We generate three sub-datasets with following settings:

1) **DiDiCD 32×32 T30**: The city is divided into 32×32 grids of $500m \times 500m$ each. A period is 30 minutes long.
2) **DiDiCD 32×32 T60**: The city division is the same as the previous one while the length of a period is 60 minutes.
3) **DiDiCD 16×16 T30**: The city is divided into 16×16 grids of $1km \times 1km$ each. A period is 30 minutes long.

For the sake of consistency and clarity, we follow prior work [11] to employ a fixed 9:1 dataset split proportion. To all datasets, the first 27 days are used for training and the rest 3 days are used for testing. Furthermore, the last 3 days of the training set are split as the validation set to do early-stop. The final input samples and labels are generated via the sliding window approach.

B. Model settings

We compare SDCN with two representative deep learning models applied to handle spatial-temporal data: ConvLSTM [13] and ST-ResNet [8]. The hyper-parameters of models are denoted as follow:

1) **ConvLSTM S(4-2) 3×3**: "S" refers to the structure. "S(4-2)" means the encoder and forecaster are both comprised of two ConvLSTM layers where the first and second layer has 4 and 2 filters respectively. "3×3" indicates the kernel size.
2) **ST-ResNet c8.p1.t1 R4 3×3**: "c8.p1.t1" means the number of input frames in the closeness component, period component, and trend component is 8, 1, and 1,

[1]thanks `https://gaia.didichuxing.com` for providing dataset

TABLE III
PERFORMANCE OF DIFFERENT MODELS

Model	DiDiCD 32x32_T30			DiDiCD 32x32_T60			DiDiCD 16x16_T30		
	*n_param	RMSE	MAPE	*n_param	RMSE	MAPE	*n_param	RMSE	MAPE
ConvLSTM S(4-2) 3x3	0.3 x 10k	3.2151	0.5404	-	5.8033	0.645	-	6.9928	0.6018
ConvLSTM S(8-4-2) 3x3	1 x 10k	2.9817	0.5176	-	**5.2187**	0.6824	-	6.4029	0.6086
ST-ResNet c8.p1.t1 R4 3x3	90.9 x 10k	2.8665	0.4631	-	5.8266	0.5553	-	**6.1697**	0.5141
ST-ResNet c8.p1.t1 R8 3x3	179.5 x 10k	3.0701	0.4309	-	5.5607	**0.5307**	-	31.8133	0.8343
SDCN R(LSTM) F(e-f:power-1) S(4-2)	8.6 x 10k	2.901	0.429	8.5 x 10k	6.1356	0.6089	2.1 x 10k	6.5011	**0.5033**
SDCN R(LSTM) F(e-f:power-1) S(8-4-2)	8.7 x 10k	**2.824**	**0.4286**	8.6 x 10k	**5.0476**	**0.4805**	2.2 x 10k	6.5193	0.4939
SDCN R(GRU) F(none:power-1) S(4-2)	25.7 x 10k	3.0523	0.4694	25.6 x 10k	5.5104	0.5641	6.3 x 10k	6.5083	0.5085
SDCN R(GRU) F(none:power-1) S(8-4-2)	25.7 x 10k	**2.7617**	**0.4206**	25.6 x 10k	5.8031	0.6143	6.4 x 10k	**6.2479**	**0.473**

The first place of prediction performance is dyed by black and the second place is dyed by grey.
*parameters who are never been used and updated (because of sparse tensor multiplication) are ignored in counting, these "invalid" parameters do not consume any computing power and only require little additional storage space.

respectively. "R4" means the number of residual blocks is 4, and the kernel size is 3×3.

3) **SDCN R(LSTM) F(e-f:power-1) S(8-4-2)**: "R" refers to the recurrent structure. "F" refers to the graph filter. "e-f" indicates the sharing mode of graph filter, as explained in Tab. II . "power-1" indicates the name and filter degree of used graph filter, as shown in I. "S(8-4-2)" has a similar meaning as explained in item 1.

All models are implemented with Keras [28], optimized with Adam [29], validated on the same dataset split proportion. The batch size is 32 for both training and testing. Besides, we fix the global random seed to 123 in all experiments to eliminate the difference of multiple runs. Some traffic flow prediction applications may model the weekday data and the weekend data separately in order to achieve higher accuracy by exchanging a bit of computation efficiency. But in fact, the difference between weekday and weekends is not obvious in our datasets (the urban level weekend and weekday demand series are both bimodal distribution and quite similar). Besides, it is only a secondary factor in model comparison, so we just treat them equally in our experiments.

C. Prediction performance

Tab. III shows the performance of SDCN and the baseline models. We observe that variants of SDCN win first place on RMSE in all datasets and win first place on MAPE in 2 of 3 datasets, with relatively small model size. Although ST-ResNet achieves the lowest RMSE in dataset "DiDiCD 16×16_T30", RFM-GRU achieves the secondary lowest RMSE and the lowest MAPE. More specifically, taking the results on dataset "DiDiCD 32 × 32_T60" as an example, our model gets a 3.3% RMSE decrease and a 9.5% MAPE decrease compared with the second-ranked model. In fact, the importance of every region is not the same, some demand-heavy regions like super malls and transportation hubs have a significant lip in demand magnitude compared to others. RMSE would like to be dominated by these hot regions but MAPE wouldn't. based on this observation, we can see that our model minimizes prediction errors for both hot regions and cold regions.

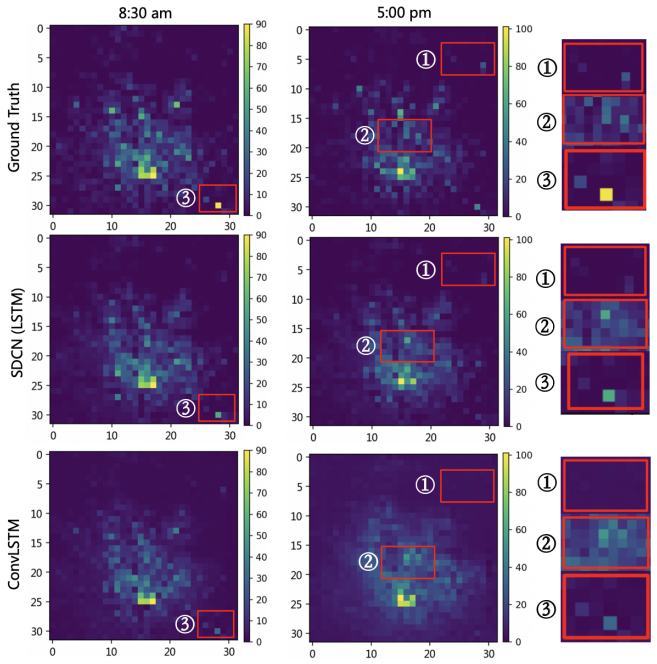

Fig. 4. Texture and sharpness comparison between prediction images generated by ConvLSTM and SDCN. Benefiting from the introduction of sparsity and diversity, our RFM-LSTM generates more clear images than ConvLSTM and achieves lower error dealing with isolated demand-heavy regions.

To validate the effectiveness of HGFM, we compare SDCN-LSTM with ConvLSTM (they have the same recurrent gate structure and both adopt Encoder-Forecaster framework). Fig. 4 visualizes prediction results of two samples from dataset "DIDICD 32×32 T30". The period of two images is 8:30am-9:00am and 5:00pm-5:30pm respectively. We pick three image blocks as a comparison control group to show their differences in detail. Taking a macro view scope, both SDCN(LSTM) and ConvLSTM perform well, the prediction images are very similar to the ground truth. But when we pay more attention to the image texture and sharpness, it can be found that SDCN generates more clear prediction images than ConvLSTM. As shown in Fig. 4, image blocks marked by red rectangles gen-

erated by SDCN(LSTM) is much brighter than ones generated by ConvLSTM. This indicates that HGFM is more effective than convolution for capturing the sparse spatial correlation. From the perspective of global average prediction error, The RMSE of these two samples gets 1.6%(from 2.588 to 2.546) and 4.2%(from 4.481 to 4.294) decreased respectively by using SDCN. From the perspective of an individual region's prediction error, SDCN makes fewer mistakes facing hot regions, which is a meaningful character in commercial applications.

D. Effect of Sparsity

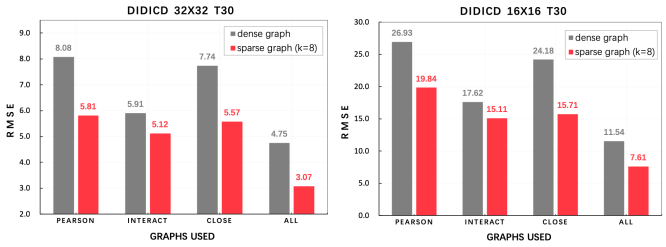

Fig. 5. The significance of sparsity and diversity on improving prediction performance

To investigate the impact of sparsity and diversity on prediction performance, we compare the performance of SDCN on dense graphs (i.e., all nodes are connected) and sparse graphs respectively. The experiment is conducted on two datasets and the evaluated model is "SDCN R(LSTM) F(none:power-1) S(4-2)". As shown in Fig. 5, the sparse graphs always perform better than dense graphs, which strongly shows the effectiveness of sparsity. Besides, we can observe that the interact graph is more efficient than the closeness graph and demand correlation graph while the combination of all graphs achieves the lowest prediction error, which indicates the importance of diversity.

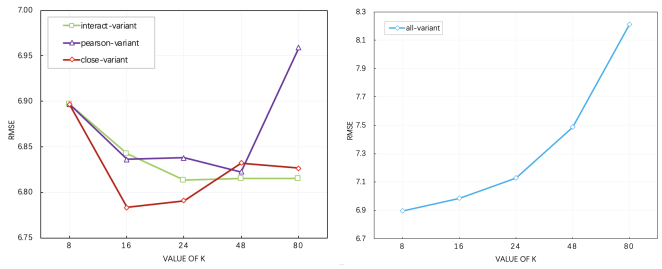

Fig. 6. Impact of sparse level on prediction performance. "close-variant" means only the k of closeness graph is changing while other graphs' k is fixed to 8

But how sparse should the spatial graphs be or how to tune the sparsity level? From the left subfigure in Fig. 6, we can observe that although the changing magnitude of the curves between prediction error and sparsity level is different for different graphs. As the sparsity decreases, the prediction error always decreases at first and then increases. This indicates that there exists an optimal sparsity level setting for combined graphs to best approximate the spatial correlation when tuning the sparsity level of different graphs separately. Besides, it is shown in the right subfigure of Fig. 6 that decreasing the sparsity level of all graphs simultaneously will significantly increase the prediction error, for the non-zero elements of the combined graph increases much faster than a single graph.

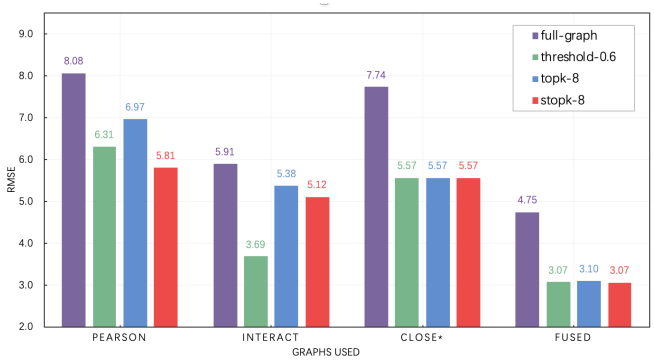

Fig. 7. The performance of different node selection methods on different variants of SDCN. Note that all the variants of SDCN here use a "rnm" filter with strength 1. in the case of asymmetric adjacency matrix, we can treat the filter as a simple normalized adjacency matrix, the multiplication of such adjacency matrix and an input signal is just calculating the linear combination of the value of one-hop neighbors.

To evaluate the impact of different sparsity level tuning method, we conducted additional experiments by using a special type of SDCN that is compatible with asymmetric adjacency matrix (i.e., directed graph). Specifically, we evaluate three methods: 1) threshold-based selection, 2) top-k selection and 3) STOPK selection, and the corresponding three variants on the SDCN, each of which only uses a single type of correlation graph. As shown in Fig. 7, we can obtain that: The performance of STOPK slightly outperforms the top-k selection method in all cases and is better than that of the threshold-based selection method in the Pearson graph case. However, in the interaction graph case, the performance of STOPK is worse than that of the threshold-based selection method. The main reason is the different setting of hyper-parameters (i.e., k or the threshold value). It is worth noting that the hyper-parameter dominates the graph's sparsity. That is, in the interaction graph case, the sparsity of STopK(k=8) is too large, resulting in the loss of some key information. Hence, from the perspective of sparsity control and model compatibility, the STOPK is the first choice when we need to generate a sparse graph.

E. Effect of diversity

Fig. 5 also reveals the following features about the diversity:

- The prediction performance with fused graph is always superior to the single graph due to the advantage of diversity. The reason is that different graphs measure the correlation between regions from different perspectives, diverse graphs can represent the ground truth correlation more completely (see Fig. 8).
- An appropriate sparsity ensures the performance of a single graph. The better the performance of a single graph is, the better the performance of the fused graphs is.

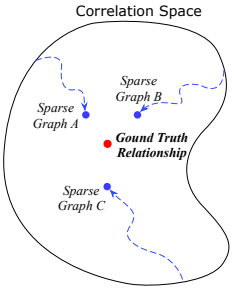

Fig. 8. Ensemble diversity to estimate ground truth relationship

- The divergence of graphs also significantly affects the model performance. The reason is that each graph encodes the spatial correlation from a different perspective, so even if the performance of some graphs is not good enough, the integration of these different graphs still improves the overall performance. For example, the Pearson graph performs worse than both the closeness graph and interact graph, but removing the Pearson graph would decrease the prediction performance.

As a summary, the fusion should pursue the goal of "good enough but different", which implies that only when the graph and the model can fully express the "good divergence" can we obtain satisfactory prediction results.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose the sparsity and diversity driven correlation network for traffic demand prediction. We use STOPK to construct diverse sparse graphs and then employ RFM with Encoder-Forecaster to model the spatial and temporal correlation simultaneously. Experiments on real-world datasets validate the effectiveness of our approach. In the future, we plan to investigate more kinds of spatial correlations and explore better methods to control sparsity.

REFERENCES

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[2] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871–882, 2013.

[3] X. Li, G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang, and Z. Wang, "Prediction of urban human mobility using large-scale taxi traces and its applications," *Frontiers of Computer Science*, vol. 6, no. 1, pp. 111–121, 2012.

[4] S. Shekhar and B. M. Williams, "Adaptive seasonal time series models for forecasting short-term traffic flow," *Transportation Research Record*, vol. 2024, no. 1, pp. 116–125, 2007.

[5] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393–1402, 2013.

[6] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "Dnn-based prediction model for spatio-temporal data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2016, p. 92.

[7] 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," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[8] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[9] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 777–785.

[10] Z. Cui, R. Ke, Y. Wang *et al.*, "Deep stacked bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," in *6th International Workshop on Urban Computing (UrbComp 2017)*, 2016.

[11] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] 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 *Advances in neural information processing systems*, 2015, pp. 802–810.

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *International Conference on Machine Learning*, 2015, pp. 2067–2075.

[15] D. Kang, Y. Lv, and Y.-y. Chen, "Short-term traffic flow prediction with lstm recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

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

[17] Y. Liu, H. Zheng, X. Feng, and Z. Chen, "Short-term traffic flow prediction with conv-lstm," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2017, pp. 1–6.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[19] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[21] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[22] D. Chai, L. Wang, and Q. Yang, "Bike flow prediction with multi-graph convolutional networks," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2018, pp. 397–400.

[23] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1525–1534.

[24] W. R. Tobler, "A computer movie simulating urban growth in the detroit region," *Economic geography*, vol. 46, no. sup1, pp. 234–240, 1970.

[25] K. Pearson, "Notes on regression and inheritance in the case of two parents proceedings of the royal society of london, 58, 240-242," 1895.

[26] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[27] Q. Li, X.-M. Wu, H. Liu, X. Zhang, and Z. Guan, "Label efficient semi-supervised learning via graph filtering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9582–9591.

[28] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.