SUB-GRAPH LEARNING FOR SPATIOTEMPORAL FORECASTING VIA KNOWLEDGE DISTILLATION

Mehrtash Mehrabi and Yingxue Zhang
Huawei Noah's Ark Lab

ABSTRACT
One of the challenges in studying the interactions in large graphs is to learn their diverse pattern and various interaction types. Hence, considering only one distribution and model to study all nodes and ignoring their diversity and local features in their neighborhoods, might severely affect the overall performance. Based on the structural information of the nodes in the graph and the interactions between them, the main graph can be divided into multiple sub-graphs. This graph partitioning can tremendously affect the learning process, however the overall performance is highly dependent on the clustering method to avoid misleading the model. In this work, we present a new framework called KD-SGL to effectively learn the sub-graphs, where we define one global model to learn the overall structure of the graph and multiple local models for each sub-graph. We assess the performance of the proposed framework and evaluate it on public datasets. Based on the achieved results, it can improve the performance of the state-of-the-arts spatiotemporal models with comparable results compared to ensemble of models with less complexity.

1. INTRODUCTION
Recently there have been many studies on graph learning and specially spatiotemporal forecasting [1–4]. As the graph expands and includes more nodes, the interactions between nodes becomes more complicated and highly challenging to learn the spatiotemporal dependencies between all nodes. Hence, the prediction model should be capable of precisely learning the various interaction types between nodes and find a policy to relate them to the underlying time series. However, all the existing studies consider only one single model to learn the node interactions across the graph where all the parameters are shared among all nodes equally which makes it unable to learn the node behaviour in local neighborhoods and distinguish different interaction types.

One simple solution to learn the node-specific features and the local interactions is partitioning the main large graph into several sub-graphs and study them independently. However, the information loss caused by isolating the sub-graphs and ignoring the interactions between them can degrade the overall performance. The learning procedure is highly dependent on the differences and relationships between the sub-graphs which should be considered during the learning process. The main two questions in the problem of sub-graph learning, are 1) how to divide the main graph into sub-graphs and investigate them? and 2) how to use the interactions between the sub-graphs to enrich the model?

In this work, we investigate the problem of sub-graph learning which has not been explicitly studied in the literature. We propose a new framework where the main large graph is partitioned into several sub-graphs based on their temporal and spatial proximity and learn an individual model for each graph separately. In order to improve the learning capability and incorporating the interactions between graphs, we introduce knowledge distillation (KD)-based sub-graph learning, called KD-SGL. The KD-based approach can improve the learning capability by incorporating several sub-graph models and the interactions between them.

In the following sections, first in Sec.2 we present some of related work. We formulate the sub-graph learning problem in Sec.3 and propose our model structure. We evaluate KD-SGL in Sec.4 by some of the existing datasets on traffic data. Finally, we conclude the paper in Sec.5.

2. RELATED WORK
In this section, first we present some of the recent works on spatiotemporal forecasting and then briefly summarize the recent studies on multi-task learning which is the closest topic in the literature to the sub-graph learning.

2.1. Spatiotemporal Forecasting
Our work is adapted for multivariate and spatiotemporal forecasting using deep learning and graph neural networks (GNNs). Spatiotemporal forecasting has been extensively studied in the literature and neural network-based techniques have started to offer the best predictive performance for multivariate time-series prediction [5–12]. Numerous algorithms have been proposed that combine GNNs with temporal neural network architectures [13–19].

The authors in [13] propose a model that instead of applying regular convolutional and recurrent units, formulates the problem on graphs and build the model with complete convolutional structures, which enable much faster training speed with
fewer parameters. In order to enhance the traditional graph convolutional network, the authors in [17] propose to use node adaptive parameter learning and data-adaptive graph generation modules for learning node-specific patterns and discovering spatial correlations from data, separately to capture node-specific spatial and temporal correlations in time-series data automatically without a pre-defined graph. In [18], authors present a novel learning architecture called FC-GAGA, that can achieve a good performance without requiring knowledge of the graph. It combines a fully connected time series model with temporal and graph gating mechanisms, that are both generally applicable and computationally efficient. In [19], the authors propose the diffusion convolutional recurrent neural network that captures the spatiotemporal dependencies by using bidirectional graph random walk and recurrent neural network. They integrate the encoder-decoder architecture and the scheduled sampling technique to improve the long-term forecasting performance.

It should be noted that all of the above works consider only one common model between the nodes in the graph where all the nodes equally contribute to the model. However, in sub-graph learning problem, our goal is to divide the main graph into several sub-graphs to better study the local interactions in each neighborhoods of the graph.

2.2. Multi-task Learning

In conventional multi-task learning, the goal is to learn multiple tasks simultaneously, while in our problem the goal is to learn multiple sub-graphs of each single task. Moreover, in multi-task learning models, the task boundaries are decided in advance, however in sub-graph learning the main graph should be split for training. In the following, we present some of the recent works on multi-task learning.

The authors in [20], propose PathNet which is a neural network algorithm that uses embedded agents to discover which parts of the network to re-use for new tasks. Agents are pathways through the network which determine the subset of parameters that are used and updated by the forwards and backwards passes of the backpropogation algorithm. In [21], authors have presented a method to pack multiple tasks into a single network with minimal loss of performance on prior tasks to obtain levels of accuracy comparable to the individually trained networks for each task. In [22], the authors have presented an architecture based on combining a pre-trained encoder, an ensemble of simple classifiers, and a particular activation / loss function pairing. It has shown that, these features independently helps to mitigate catastrophic forgetting in this setting, but they are most potent when working together.

3. KD-SGL FRAMEWORK

Let us consider a graph where \( N \) nodes are distributed across the graph and record the data of different time instances \( t \in [1, T] \), where \( T \) is the length of the window we investigate the graph. At each time window, the data is measured by each node \( n \in [1, N] \) and denoted as \( x_n \in \mathbb{R}^T \) resulting in \( X = [x_1, \cdots, x_N]^T \in \mathbb{R}^{N \times T} \) as the network data. The final objective is to predict the next \( H \) time instants status of nodes as \( \hat{Y} \in \mathbb{R}^{N \times H} \).

The properties of the nodes across the graph can be investigated both locally within local neighborhoods and globally by considering all the node interaction types in the graph. As the graph expands, it becomes more crucial to consider local neighborhoods since the nodes become farther away from each other and include more interaction types. KD-SGL can enhance the overall performance by investigating the node interactions both locally and globally through introducing global and local models, called teacher and student. The overall structure of KD-SGL is illustrated in Fig.1 consisting of three main modules, including, a teacher model, node clustering, and several student models. In the following, we present each component of KD-SGL in detail.

3.1. Teacher Model

Given a large graph formed by several nodes with highly diverse features and interaction types, KD-SGL first utilize all the nodes across the graph equally to train a global model called teacher. The teacher model learns the nodes embedding matrix \( E \in \mathbb{R}^{N \times d} \), where \( d \) is the dimension of the embedding and make a prediction for the next \( i \)-th state of node \( n \) as \( \hat{y}_n, T+i \), where \( i \in [1, H] \). Moreover, the teacher model is later used to supervise the local models (students) for each sub-graph. In order to train the teacher model we minimize the following loss term defined as

\[
\mathcal{L}_{teacher} = \frac{1}{HN} \sum_{j=1}^{H} \sum_{i=1}^{N} |y_{i, T+i} - \hat{y}_{i, T+i}|. \quad (1)
\]

3.2. Node-Clustering

The goal of node-clustering is to define the sub-graphs and determine the contribution of nodes to each of them. The sub-graphs are denied based on the spatial and temporal information of nodes investigated through the learnt nodes embedding by the teacher model and the adjacency matrix of the main graph. Inspired by the clustering method introduced in [23], we introduce our node-clustering approach in KD-SGL, where instead of hard-clustering, and fully isolating the sub-graphs, we perform soft-clustering, and determine how much each node should contribute to each sub-graph. The node-clustering module is the integration of an autoencoder (AE) and GNN layers. The AE is fed by the learned nodes embedding matrix \( E \) and the output of its layer \( l \geq 1 \) is

\[
A^{(l)} = \phi(U^{(l)}A^{(l-1)} + b^{(l)}), \quad (2)
\]

where \( U^{(l)} \) and \( b^{(l)} \) are the weights and biases of layer \( l \), respectively and \( A^{(0)} = E \). By choosing the proper weight matrices,
the output of the encoder is generated as $A^{(L)} \in \mathbb{R}^{N \times K}$ to match the number of $K$ sub-graphs. The output of the decoder part is the reconstruction of the raw data $E$ leading to the following loss function
\[
L_{ae} = \frac{1}{N} \sum_{i=1}^{N} \| e_i - \hat{e}_i \|^2_2. \tag{3}
\]

For each layer of the encoder, there is a corresponding GNN layer fed by the encoded data of last layer, i.e. $A^{(l-1)}$, and the adjacency matrix $W$, as
\[
Z^{(l)} = \phi(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}Z^{(l-1)}U^{(l-1)}), \tag{4}
\]
where $\bar{W} = W + I$, $D$ is diagonal matrix and $D_{ii} = \sum_j W_{ij}$, and $U^{(l-1)}$ is the weight matrix of layer $l - 1$. Function $\phi(\cdot)$ is a non-linear function such as $\mathrm{tanh}$ and $Z^{(l-1)}$ is generated as
\[
\tilde{Z}^{(l-1)} = (1 - \psi)Z^{(l-1)} + \psi A^{(l-1)}, \tag{5}
\]
where $\psi$ is the balance coefficient and here is set to 0.5. It should be noted that $Z^{(0)} = E$ and hence $\tilde{Z}^{(0)} = E$. The last GNN layer is the input of a multiple classification layer with a softmax function to map $N$ nodes to $K$ sub-graphs as
\[
Z = \text{softmax}(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}Z^{(L)}U^{(L)}) \in \mathbb{R}^{N \times K}. \tag{6}
\]

We use the Student’s t-distribution as a kernel to measure the similarity between the data representation $a_i$ and the cluster center vector $\mu_j$ as follows
\[
q_{ij} = \frac{1 + ||a_i - \mu_j||^2/v}{\sum_{j'}(1 + ||a_i - \mu_{j'}||^2/v)}^{-\frac{1}{2}}, \tag{7}
\]
where $a_i$ is the $i$-th row of $A^{(L)} \in \mathbb{R}^{N \times K}$, $v$ is the degree of freedom of the student’s t-distribution, and $\mu_j$ is initialized by K-means on the representations learned by pre-train AE. In order to make data representation closer to cluster centers and improving the cluster cohesion we calculate a target distribution $P$ as
\[
p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'}q_{ij'}^2/f_{j'}}, \tag{8}
\]
where $f_j = \sum_i q_{ij}$ are soft cluster frequencies. Then the clustering loss is defined as
\[
L_{clu} = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{9}
\]

The GNN also provide a clustering assignment distribution $Z$ (output of the classification layer) and we can use distribution $P$ to supervise it by defining the following loss function
\[
L_{gnn} = KL(P||Z) = \sum_{i} \sum_j p_{ij} \log \frac{p_{ij}}{z_{ij}}. \tag{10}
\]

### 3.3. Student Models

In order to investigate the generated sub-graphs, we define local models, called students trained for each sub-graph based on the learnt weights $Z$. The student models are supervised by the teacher model by a pre-defined imitation factor that determines how much student models should follow the teacher model. Using the teacher model and the learnt distribution $Z$, $K$ student models are trained to make the final prediction for $N$ nodes. The teacher supervision through the KD structure can also help to learn less complex student models compared to the teacher model and benefit from the features already learnt by the complex teacher. Hence, we can use a simple multiple layer perception (MLP) for student models with less parameters than the teacher model. Each student model $S_k$, where $k \in [1, K]$ is then trained as
\[
L_{s_k} = (1 - \rho) \left( \frac{1}{HN} \sum_{i=1}^{H} \sum_{j=1}^{N} z_{jk} |y_{j,T+i} - s_{kj,T+i}| \right) + \rho \left( \frac{1}{HN} \sum_{i=1}^{H} \sum_{j=1}^{N} z_{jk} |\hat{y}_{j,T+i} - s_{kj,T+i}| \right), \tag{11}
\]
where $\rho$ is the pre-defined KD imitation factor and $z_{ik}$ is the probability of node $i$ assigned to cluster $k$.

Considering all the loss sources, the final loss function of KD-SGL is computed as
\[
L_{tot} = L_{teacher} + L_{ae} + \alpha L_{clu} + \beta L_{gnn} + \sum_{k=1}^{K} L_{s_k}, \tag{12}
\]
where $\alpha > 0$ is a hyperparameter that balances the clustering optimization and local structure preservation of raw data and $\beta > 0$ is a coefficient that controls the disturbance of GNN layers to the embedding space. The final prediction is
\[
\hat{y}_{i,t} = \frac{1}{2} \left( \hat{y}_{i,t} + \sum_{k=1}^{K} z_{i,k} \cdot \hat{y}_{i,t}^{(k)} \right), \tag{13}
\]
where $\hat{y}_{i,t}$ and $\hat{y}_{i,t}^{(k)}$ are the predicted values for node $i$ at time $t$, by the teacher model and the $k$-th student, respectively.
We assess KD-SGL using two spatiotemporal models including FC-GAGA and DCRNN. We use two datasets with different size to study the impact of the size of the graph on the performance and sub-graph learning. In the following first, we present the experimental setup and then evaluate KD-SGL in terms of the achieved performance and its complexity.

4.1. Simulation Setup

Datasets. We evaluate KD-SGL on two widely explored traffic datasets with different size including METR-LA and PEMS07 with $N = 207$ and $N = 883$ nodes, respectively. METR-LA consists of the traffic speed readings collected from loop detectors and aggregated over 5 minute intervals. It contains 34,272 time steps of 207 sensors collected in Los Angeles County over 4 months. PEMS07 is collected from the Caltrans Performance Measurement System (PeMS) and aggregated into 5-minutes windows, resulting in 288 points in the traffic flow for one day. It contains 28224 time steps of 883 sensors constructing a graph with 883 nodes and 866 edges.

Training Process. In order to train KD-SGL, the datasets are split into 70% training, 10% validation, and the final 20% as the test sets. For the following tests, for case of FC-GAGA, we use it with 4 layers as the teacher model in our KD-SGL framework. We set the number of sub-graphs, i.e., $K$, to 4 and for each student model we use the verification set to find the best imitation factor $\rho$ and then test the performance by the test set. The values of $\alpha$ and $\beta$ are set to 0.1.

Hardware Setup. The following experiments are conducted on a system with 16 Gb of GPU and Cuda version 11.1 and tensorflow 2.2.

4.2. Overall Performance and Complexity Analysis

In Fig. 2 we show the locations of the nodes of different sub-graphs generated by the node clustering method for METR-LA dataset, where for each node the sub-graph with the highest score is selected. We also compare the distribution of nodes in different learnt sub-graphs, and as seen the distributions are close to each other but the details are different which are interpreted as global and local information.

Table 1 shows the comparison between the achieved results by KD-SGL framework using FC-GAGA and DCRNN as the teacher model by running several trials and compare 3 horizons $t = 3, 6, 9$ and use three metrics to compare the results. We compare the achieved performance results with the original results of the base models (FC-GAGA and DCRNN) and also their ensembles. As seen, for the larger dataset we can observe a larger performance gap which shows the importance of sub-graph learning in KD-SGL. We also compare the complexity of different models during the inference in Table 2 which verifies the effectiveness of KD-SGL over ensemble approach.

5. CONCLUSION

In this work, we propose KD-SGL, a novel framework for spatiotemporal forecasting that learns the spatial and temporal dependencies across the graph by partitioning it into several sub-graphs. It introduce two types of models called teacher and student to learn the global and local structural information in the graph. Extensive experiments on comprehensive traffic datasets demonstrates significant improvement of KD-SGL over existing models that study the graphs as a single graph. Our proposed model is generic and flexible, and any existing spatiotemporal forecasting model can be used as the teacher model to enhance the overall performance. For future work, we plan to study the impact of loss terms on the final result and further optimize the clustering method.
6. REFERENCES

[1] S. Pal, L. Ma, Y. Zhang, and M. Coates, “Rnn with particle flow for probabilistic spatio-temporal forecasting,” in International Conference on Machine Learning. PMLR, 2021, pp. 8336–8348.

[2] M. Li and Z. Zhu, “Spatial-temporal fusion graph neural networks for traffic flow forecasting,” in Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 5, 2021, pp. 4189–4196.

[3] C. Song, Y. Lin, S. Guo, and H. Wan, “Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, 2020, pp. 914–921.

[4] Z. Fang, Q. Long, G. Song, and K. Xie, “Spatial-temporal graph ode networks for traffic flow forecasting,” in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 364–373.

[5] W. Bao, J. Yue, and Y. Rao, “A deep learning framework for financial time series using stacked autoencoders and long-short term memory,” PloS one, vol. 12, no. 7, p. e0180944, 2017.

[6] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, “A dual-stage attention-based recurrent neural network for time series prediction,” arXiv preprint arXiv:1704.02971, 2017.

[7] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, “Modeling long-and short-term temporal patterns with deep neural networks,” in The 41st international ACM SIGIR conference on research & development in information retrieval, 2018, pp. 95–104.

[8] T. Guo and T. Lin, “Multi-variable lstm neural network for autoregressive exogenous model,” arXiv preprint arXiv:1806.06384, 2018.

[9] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, “A memory-network based solution for multivariate time-series forecasting,” arXiv preprint arXiv:1809.02105, 2018.

[10] R. Sen, H.-F. Yu, and I. S. Dhillon, “Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting,” Advances in neural information processing systems, vol. 32, 2019.

[11] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, “N-beats: Neural basis expansion analysis for interpretable time series forecasting,” arXiv preprint arXiv:1905.10437, 2019.

[12] S. Smyl, “A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting,” International Journal of Forecasting, vol. 36, no. 1, pp. 75–85, 2020.

[13] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” arXiv preprint arXiv:1709.04875, 2017.

[14] R. Huang, C. Huang, Y. Liu, G. Dai, and W. Kong, “Ls-gcn: Long short-term traffic prediction with graph convolutional networks.” in IJCAI, 2020, pp. 2355–2361.

[15] C. Zheng, X. Fan, C. Wang, and J. Qi, “Gman: A graph multi-attention network for traffic prediction,” in Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 01, 2020, pp. 1234–1241.

[16] C. Park, C. Lee, H. Bahng, Y. Tae, S. Jin, K. Kim, S. Ko, and J. Choo, “St-grat: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed,” in Proceedings of the 29th ACM international conference on information & knowledge management, 2020, pp. 1215–1224.

[17] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, “Adaptive graph convolutional recurrent network for traffic forecasting,” Advances in neural information processing systems, vol. 33, pp. 17 804–17 815, 2020.

[18] B. N. Oreshkin, A. Amini, L. Coyle, and M. Coates, “Fc-gaga: Fully connected gated graph architecture for spatio-temporal traffic forecasting,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 10, 2021, pp. 9233–9241.

[19] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” arXiv preprint arXiv:1707.01926, 2017.

[20] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, “Pathnet: Evolution channels gradient descent in super neural networks,” arXiv preprint arXiv:1701.08734, 2017.

[21] A. Mallya and S. Lazebnik, “Packnet: Adding multiple tasks to a single network by iterative pruning,” in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2018, pp. 7765–7773.

[22] M. Shanahan, C. Kaplanis, and J. Mitrović, “Encoders and ensembles for task-free continual learning,” arXiv preprint arXiv:2105.13327, 2021.

[23] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, “Structural deep clustering network,” in Proceedings of The Web Conference 2020, 2020, pp. 1400–1410.